Technische Universität München
Lehrstuhl für Steuerungs- und Regelungstechnik

Action Selection in Cooperative Multi-Robot Systems

Florian Rohrmüller

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Hans-Georg Herzog

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
2. Univ.-Prof. Michael Beetz, Ph.D.

Die Dissertation wurde am 09.06.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 04.10.2011 angenommen.

Acknowledgments

Even though I am the single author of this thesis, a lot of people gave me assistance in many respects. First, I want to thank my doctoral adviser Prof. Martin Buss, who gave me the opportunity to undertake this work and the latitude to fully carry out my ideas, still giving me guidance whenever I needed. Furthermore, I thank him for always providing me with an excellent working environment, which in this form certainly cannot be taken as granted. I thank also my co-advisor, Dr.-Ing. Dirk Wollherr, who always had an open door for my problems and gave me encouragement, as well as, the confidence to take this thesis to its conclusion. I further thank Prof. Michael Beetz for providing me support whenever I have been asking for.

I also want to express my gratitude to all my friends and colleagues at our institute for the great cooperativeness and the pleasant atmosphere. Special thanks go to Prof. Sandra Hirche for her support and the in-depth discussions that broadened my mind and opened new perspectives to me. I further want to thank the entire team of the *ACE* project (Andrea Bauer, Kolja Kühnlenz, Klaas Klasing, Georgios Lidoris, Quirin Mühlbauer, Stefan Sosnowski, Dirk Wollherr, Tingting Xu and Tianguang Zhang) for the great team spirit throughout the entire project. Special thanks go to Georgios Lidoris for the great collaboration during the last years and the many scientific discussions that inspired my work down to the present day.

My gratitude goes also to the entire MuRoLa team (Daniel Althoff, Dražen Brščić, Johannes Bürger, Abhishek Dutta, Azwirman Gusrialdi, José Ramón Medina Hernández, Vasiliki Koropouli, Omiros Kourakos, Martin Lawitzky, Nikos Mitsou, Alexander Mörtl, Matthias Rambow, Xuelian Zang and Wei Wang) for the extraordinary enjoyable time. Special thanks go to Omiros Kourakos for the great and close collaboration beginning from the first day of the project. I also thank all external cooperators in MuJoA and CoTeSys (they are too many to be explicitly mentioned here) for the great collaborations and the pleasant teamwork, e.g. during many hours of demonstration preparations. Furthermore, many thanks go to all my students for their valuable work and support that enabled me to keep concentrated on my research. Many thanks also to Anil, Georg and Raphi for the proofreading and all their valuable feedback, and especially to Tine for her permanent support. Last but not least, many thanks to my family and all my friends who cared for the essential amount of distraction that enabled me to regenerate and to draw enough power for the completion of this dissertation.

Munich, 2011 Florian Rohrmüller

To all my supporters.

Abstract

The ability to work in a team is an integral skill demanded nowadays of most employees. Teamwork enables the combination of complementary strengths and the distribution of the workload, thereby increasing the productivity of the entire team. It is a coherent consequence that this is also a desirable ability of multiple autonomous robots acting simultaneously in the same environment. Typical applications comprise for example search-and-rescue robots, warehouse delivery robots, or aide robots in hospitals. By cooperation these robots can coordinate their actions in order to increase the entire team performance. Such an ability demands a robotic decision-making that yields a cooperative action selection among a set of multiple robots. The hardest challenges in this respect arise mainly from the existing information uncertainty and the rapidly growing problem complexity. Such robotic systems are mostly called to carry out tasks in partially known or even unknown environments where they are constantly faced with sensory noise resulting in uncertain information. In multi-robot systems (MRS) this uncertainty is further intensified by the fact, that the environment is not only modified by the action of a single robot but rather by the simultaneous actions of all robots. This leads to the huge complexity that results from the number of combinatorial possibilities to select these simultaneous actions, which increases exponentially in the number of robots.

This thesis investigates the problem of a reliable and robust action selection in cooperative multi-robot systems and presents novel methods to handle the mentioned challenges in real-world systems. A framework for a multi-robot task allocation is introduced. By splitting the allocation procedure into two phases, a slower increase of the worst-case complexity compared to state-of-the-art approaches is guaranteed. Furthermore, the complexity arising on average is reduced, by taking system-specific characteristics into account. Such characteristics are also used for failure handling to achieve a more robust system operation. In order to cope with the given uncertainty, a new approach is described to understand the influence of the environment on the robotic performance and thereby to achieve a more reliable and situation-aware reward estimation. This method solves a fundamental but often neglected problem in the action selection of real-world robots. All methods provide generic contributions to the field that are excellently suited but not limited to complex service robots. In this respect, this thesis enhances not only the existing theoretical knowledge on robotic action selection, but additionally outlines the practical use of the presented methods by the given experimental results, thereby providing valuable findings for future research.

Zusammenfassung

Die Fähigkeit zur Teamarbeit ist eine wesentliche soziale Kompetenz, welche heutzutage von den meisten Arbeitnehmern erwartet wird. Teamarbeit ermöglicht die Kombination von sich ergänzenden Stärken und die Verteilung der Arbeitslast, wodurch eine Leistungssteigerung des gesamten Teams erreicht wird. Es ist eine schlüssige Folgerung, dass dies auch eine wünschenswerte Fähigkeit von mehreren autonomen Robotern ist, welche gleichzeitig in einer gemeinsamen Umgebung agieren. Typische Anwendungen umfassen beispielsweise Such- und Rettungsroboter, Warenlieferungsroboter, oder Hilfsroboter in Krankenhäusern. Mittels Kooperation können diese Roboter ihre Aktionen koordinieren und dadurch die gesamte Teamleistung erhöhen. Eine derartige Befähigung erfordert ein robotisches Entscheidungsvermögen welches eine kooperative Aktionswahl innerhalb einer Gruppe von mehreren Robotern ermöglicht. Die größten Herausforderungen in diesem Zusammenhang entstehen hauptsächlich durch die existierende Informationsunsicherheit sowie die schnell anwachsende Problemkomplexität. Derartige Robotiksysteme finden meist in teilweise oder gar gänzlich unbekannten Umgebungen Anwendung, wo sie permanent mit Sensorrauschen konfrontiert sind, was wiederum zur Informationsunsicherheit führt. In Multirobotersystemen (MRS) wird diese Unsicherheit weiterhin durch die Tatsache verstärkt, dass die Umgebung nicht alleinig durch die Aktion eines einzelnen Roboters, sondern vielmehr durch die gleichzeitigen Aktionen aller Roboter verändert wird. Dies führt zu der immensen Komplexität, die aus der Anzahl an kombinatorischen Möglichkeiten, diese simultanen Aktionen auszuwählen, resultiert, und exponentiell mit der Zahl der Roboter wächst.

Diese Dissertation untersucht das Problem einer zuverlässigen und robusten Aktionsselektion in kooperativen Multirobotersystemen und beschreibt neue Methoden um die genannten Herausforderungen in realen Systemen zu beherrschen. Ein Framework für eine Multiroboteraufgabenverteilung wird vorgestellt. Indem der Verteilungsvorgang in zwei Phasen unterteilt wird, kann eine langsameres Ansteigen der größtmöglichen Komplexität, im Vergleich zu den bisher bekannten Ansätzen, garantiert werden. Weiterhin werden systemspezifische Eigenschaften in Betracht gezogen, um auch eine Verringerung der durchschnittlichen Komplexität zu erzielen, sowie im Rahmen einer Fehlerbehandlung ein höheres Maß an Zuverlässigkeit zu erreichen. Zur Berücksichtigung der gegebenen Unsicherheit wird ein neuer Ansatz vorgestellt, mit welchem der Einfluss der Umgebung auf die Leistungsfähigkeit des Roboters bestimmt werden kann. Dadurch wird eine zuverlässigere und situationsbewusstere Kostenabschätzung ermöglicht. Diese Methode löst ein grundlegendes aber oft vernachlässigtes Problem bei der Aktionswahl von realen Robotern. Alle Methoden liefern generische Beiträge zum Fachbereich, welche hervorragend aber nicht ausschließlich auf komplexe Serviceroboter anwendbar sind. In dieser Hinsicht, erweitert diese Dissertation nicht nur die bisherigen theoretischen Erkenntnisse im Bereich der robotischen Aktionswahl, sondern weist zudem, angesichts der präsentierten experimentellen Resultate, auch den praktischen Nutzen der vorgestellten Methoden auf, und liefert damit wertvolle Erkenntnisse für zukünftige Forschungsarbeiten.

Contents

1. Introduction **1**
- 1.1. Motivation . . . 1
- 1.2. Practical Applications and Demands . . . 3
- 1.3. Taxonomy of Multi-Robot Systems (MRSs) . . . 6
  - 1.3.1. Interfield Taxonomy and Terminology . . . 6
  - 1.3.2. Intrafield Taxonomy: Multi-Robot Systems at a Glance . . . 8
- 1.4. Open Challenges in the Field . . . 13
- 1.5. Outline and Contributions . . . 14

2. Cooperative Action Selection **17**
- 2.1. The Scope of Action Selection within a General System Architecture . . . 17
- 2.2. Problem Definition . . . 21
- 2.3. Taxonomy for Multi-Robot Task Planning Problems . . . 24
- 2.4. Summary . . . 25

3. A Framework for Action Selection focusing on Task Allocation in MRSs **27**
- 3.1. Introduction . . . 28
- 3.2. Related Work . . . 29
- 3.3. Problem Definition . . . 33
- 3.4. The MuRoCo Framework . . . 35
  - 3.4.1. General Approach . . . 35
  - 3.4.2. Considering Heterogeneity . . . 37
  - 3.4.3. Single-Robot Tasks . . . 40
  - 3.4.4. Multi-Robot Tasks . . . 42
  - 3.4.5. Robustness and Failure Recovery . . . 46
    - 3.4.5.1. Selection of the Auctioneer Role . . . 46
    - 3.4.5.2. Failure-Aware Cost Computation . . . 47
    - 3.4.5.3. Error Recovery . . . 49
- 3.5. Analysis of the Approach . . . 50
  - 3.5.1. Soundness and Completeness . . . 50
  - 3.5.2. Scalability . . . 52
    - 3.5.2.1. Computational Complexity . . . 52
    - 3.5.2.2. Communicational Complexity . . . 54
    - 3.5.2.3. Efficiency of Pruning Strategies . . . 55
  - 3.5.3. Optimality . . . 58

3.6. Experimental Results . . . 59
3.6.1. Experimental Setup . . . 59
3.6.1.1. Description of the Robotic Hardware . . . 59
3.6.1.2. The Service Scenario . . . 60
3.6.2. The Course of Action Selection during a Cooperative Service Task . 61
3.6.3. The Course of Action Selection under Uncertainty . . . 63
3.6.4. Benchmark Evaluation . . . 64
3.7. Summary . . . 65

4. Uncertainty- and Situation-Aware Performance Estimation 67
4.1. Introduction . . . 67
4.2. Related Work . . . 70
4.3. Problem Definition . . . 73
4.4. Learning System Interdependence Models . . . 75
4.4.1. System Interdependence Analysis . . . 75
4.4.1.1. Algorithm Overview . . . 76
4.4.1.2. Component Performance Evaluation . . . 77
4.4.1.3. Learning Bayesian Network Structures . . . 78
4.4.1.4. Information-Theoretic Criteria . . . 80
4.4.2. A Case Study on the *ACE* Robot . . . 80
4.4.2.1. Indicators for Perception Performance . . . 81
4.4.2.2. Indicators for Planning Performance . . . 82
4.4.2.3. Indicators for Execution Performance . . . 83
4.4.2.4. All Performance Indicators at a Glance . . . 83
4.4.3. Experimental Results . . . 84
4.4.4. Discussion . . . 90
4.5. Uncertainty- and Risk-Aware Reward Estimation . . . 91
4.5.1. Quantile-Based Reward Estimation . . . 91
4.5.2. An Application Example . . . 96
4.5.3. Experimental Results . . . 101
4.5.3.1. Situation-Aware Action Selection . . . 102
4.5.3.2. Reliable Reward Estimation . . . 106
4.5.3.3. Forecasting of Poor Performance . . . 109
4.5.4. Discussion . . . 110
4.6. Summary . . . 111

5. Conclusion and Outlook 113
5.1. Conclusion . . . 113
5.2. Outlook . . . 115

A. Distribution of the Coalition Responsibilities in MuRoCo 117

B. The ACE Project: Mobile Robot Navigation in Urban Environments 119
B.1. Motives for the Project . . . 119
B.2. System Description . . . 121

B.3. Processing Layer 121
B.3.1. Simultaneous Localization and Mapping 121
B.3.2. Grid Fusion 123
B.3.3. Path Planning 124
B.4. Control Layer 125
B.4.1. Robot Behavior Description 125
B.4.2. Behavior Selection 126
B.4.3. Behavior Control 127
B.5. Execution Layer 129
B.6. Experimental Results 130
B.7. Conclusion 134

C. Map Overview **135**

Bibliography **139**

List of Illustrations

List of Figures

1.1. Applications and situations where robots can benefit from cooperation . . 4
1.2. Embedding of MRSs into the fields of Distributed Computing and Artificial Intelligence. 6
1.3. Categorization of robot interaction styles 10

2.1. General scheme of a multi-layered multi-robot system architecture 18
2.2. MRS as it is addressed by the different layers of the architecture in Fig. 2.1 19
2.3. The action selection problem in cooperative multi-robot systems 21

3.1. Illustration of the market-based task allocation 36
3.2. Exemplary capability constraint for the task τ_{fetch} 38
3.3. Illustration of the coalition formation 44
3.4. Number of computational and communicational operations for a MR-task assignment with respect to the number of robots 53
3.5. Relations of complexity for the MR-task assignment 54
3.6. Experimental results showing the cooperative serving by two robots 62
3.7. Sequence of trials showing the cooperative serving by two robots in the incidence of external disturbances 63
3.8. Experimental time required for a MR-task assignment 64

4.1. Exemplary scene with various sources of uncertainty 69
4.2. Scheme of the proposed system interdependence analysis. 76
4.3. Course of complexity for an exhaustive and for the K2 search 79
4.4. The relation between mutual information $I(x, y)$ and joint entropy $H(x, y)$. 81
4.5. Two representative situations chosen for the interdependence analysis . . . 85
4.6. Discretized indicator values extracted from experimental data for two different environments 85
4.7. Acceptance ratio versus the number of MCMC steps 86
4.8. Directed Acyclic Graph learned with MCMC and K2 87
4.9. Learned dependency values $\eta(x_i, x_j)$ for all used indicators 88
4.10. The marginal distributions of H^P and $\mathrm{Var}(\varphi_r)$ for assigned values of H^m . 89
4.11. Exemplary quantile function Q_{f_G} (right) for a Gaussian mixture pdf (left) . 92
4.12. Map 1 out of 20 maps used for the experiment 97
4.13. Bayesian graph structures: derived by expert knowledge and by learning. . 100
4.14. Number of errors in relation to the number of maps, shown for all ratios . 103

4.15. Achieved performance shown for each ratio 104
4.16. Average performance of all policies 106
4.17. Reward reliability achieved by the inference policies 107
4.18. The conditional distribution of the cost metric t_r 108
4.19. Relative MSE of the reward estimation 109

B.1. The *ACE* robot and its developer team 120
B.2. The software architecture of the navigation subsystem of *ACE* 122
B.3. Path planning approach 124
B.4. Finite State Machine of the *Behavior Selection* module 127
B.5. Flow chart of the `checkConstraints()`procedure 128
B.6. Flow chart of the `checkConsistency()`procedure 128
B.7. The route of the *ACE* robot through the downtown area of Munich 131
B.8. Three different scenes encountered during the experiment (left side) and the corresponding outputs of the *Control* and *Processing Layer* (right side) . . 132
B.9. Two scenes from the field experiment in the *Deutsches Museum*. 133

List of Tables

1.1. Intrafield taxonomy: a collection of MRSs characteristics 11

3.1. Number of messages required during the phases of the allocation procedure 54
3.2. Comparison of the required number of operations and number of messages for MuRoCo and related work 55
3.3. Exemplary greedy assignment of a MR-task 58
3.4. Capabilities of the MuRoLa robots 60

4.1. Overview of proposed performance indicators. 84
4.2. Parameters used for the experimental runs 102

A.1. List $l_{\mathcal{Z}}$ of coalitions for a multi-robot system of four robots 117
A.2. Sets $\mathcal{Z}_{\text{resp},r}$ of coalition responsibilities for the robots $\{r_1, r_2, r_3, r_4\}$ 117

C.1. Map parameters and respective ranges of values. 135

List of Algorithms

3.1. Update routine of the auctioneer of robot r_a 42
3.2. Update routine of the broker of robot r_b 43
A.1. Distribution of the coalition responsibilities among the robots 118
B.1. The *Behavior Control* of the *ACE* robot 129

Notations

Abbreviations

ACE	Autonomous City Explorer (robot)
ADL	Action Description Language
AI	Artificial Intelligence
BIC	Bayesian Information Criterion
BN	Bayesian Network
cdf	Cumulative distribution function
CPT	Conditional Probability Table
C-Space	Configuration Space
DAG	Directed Acyclic Graph
DAI	Distributed Artificial Intelligence
DC	Distributed Computing
MAS	Multi-Agent System
MCMC	Markov Chain Monte Carlo
MR	Multi-Robot
MRS	Multi-Robot System
MRTA	Multi-Robot Task Allocation
MSE	Mean square error
MT	Multi-Task
PDDL	Planning Domain Definition Language
pdf	Probability distribution function
SAS	Single-Agent System
SR	Single-Robot
SRS	Single-Robot System
SLAM	Simultaneous Localization And Mapping
ST	Single-Task
STRIPS	Stanford Research Institute Problem Solver
W.l.o.g.	Without loss of generality

Conventions

Scalars, Vectors, and Sets

x	Scalar
$\|x\|$	Absolute value of x
$\hat{x}$	Estimated or predicted value of x
x^*	Optimal value of x
$\boldsymbol{x}$	Vector
$\|\boldsymbol{x}\|$	Dimension of the vector x
$\boldsymbol{x}^T$	Transpose of the vector x
$\boldsymbol{X}$	Matrix
$\mathcal{X}$	Set
$\|\mathcal{X}\|$	Cardinality of the set $\mathcal{X}$

Number Sets

$\mathbb{B}$	Set of boolean values
$\mathbb{N}$	Set of natural numbers including zero
$\mathbb{R}$	Set of real-valued numbers
$\mathbb{R}_0^+$	Set of non-negative real-valued numbers

Operators

$\arg\max(f(x))$	Argument x for which $f(x)$ is maximal
$\arg\min(f(x))$	Argument x for which $f(x)$ is minimal
$\angle(x, y)$	Angle $\in [0, 2\pi]$ between the vector $(x, y)^T$ and the x-axis
$\mathrm{Cov}(x, y)$	Covariance of x and y
$\mathcal{X}\Delta\mathcal{Y}$	Symmetric difference of the sets $\mathcal{X}$ and $\mathcal{Y}$
$\mathrm{E}[x]$	Expected value of x
$\emptyset$	Empty set
$f(\cdot)$	Scalar function
$\max(f(\cdot))$	Maximal value of $f(\cdot)$
$\min(f(\cdot))$	Minimal value of $f(\cdot)$
$P(x)$	Probability of a random variable x
$Q_x(\varsigma)$	ς-Quantile of the random variable x
$\mathcal{K}_x \prec \chi(\mathcal{K}_y)$	Set $\mathcal{K}_x$ satisfies the constraint $\chi(\mathcal{K}_y)$ on the set $\mathcal{K}_y$
$\mathcal{K}_x \not\prec \chi(\mathcal{K}_y)$	Set $\mathcal{K}_x$ does not satisfy the constraint $\chi(\mathcal{K}_y)$ on the set $\mathcal{K}_y$
$\mathrm{Var}(x)$	Variance of x

Symbols

General

β	Normalization factor
$\mathcal{F}$	Cumulative distribution function (cdf)
$\delta(\cdot)$	Dirac delta function
$D(\mathsf{f}_x \| \mathsf{f}_y)$	Kullback-Leibler divergence between the probability distributions f_x and f_y
$\mathcal{G}(\mathcal{V}, \mathcal{E})$	Graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{E}$	Set of edges in a graph
μ	Mean
$\mathcal{M}$	Maximum weight matching
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ
f	Probability distribution function (pdf)
$O(\cdot)$	Big O-Notation, also known as Landau-Notation
σ	Standard deviation
$\mathcal{V}$	Set of vertices in a graph

Multi-Robot System

a	Action
$\mathcal{A}$	Set of actions
β_u	Overall obtainable utility
c	Cost
$h(r_i, r_j)$	Heterogeneity between robots r_i and r_j
κ	Capability
$\mathcal{K}$	Set of capabilities
$p(\tau)$	Plan to solve task τ
$p_{\mathcal{A}}(\tau)$	Action plan, i.e. plan composed of actions to solve task τ
$p_{\mathcal{T}}(\tau)$	Task plan, i.e. plan composed of subtasks to solve task τ
$\mathcal{P}$	Set of plans
$\mathcal{P}_{\mathcal{A}}$	Set of action plans
$\mathcal{P}_{\mathcal{T}}$	Set of task plans
ϖ	Policy
Π	Set of all policies
q	Performance
r	Robot
$\mathcal{R}$	Set of robots
ϱ	Reward
s	Symbolic state
$\mathcal{S}$	Set of possible symbolic states
τ	Task
$\mathcal{T}$	Set of tasks
u	Utility

ψ	Assignment of robots to tasks
ψ^{-1}	Inverse assignment of tasks to robots
$\chi^{\wedge}$	Conjunctive constraint
$\chi^{\vee}$	Disjunctive constraint
z	Coalition
$\mathcal{Z}$	Set of coalitions

Interdependence Model

Δc	Cost deviation
$\Delta \varrho$	Reward deviation
$H(x)$	Entropy of random variable x
$H(x, y)$	Joint Entropy of random variables x and y
$I(x, y)$	Mutual Information of random variables x and y
$\eta(x, y)$	Ratio of mutual information $I(x, y)$ and joint entropy $H(x, y)$
INFO$(x\|\|y)$	Relative quantity of information between the probability distributions x and y
$rel(\hat{x})$	Reliability of the estimation $\hat{x}$ about the value of x

Navigation

$\mathfrak{b}$	Robot behavior
$\mathfrak{c}$	Submodule of a robotic system
g	Cell of an occupancy grid
cad	Cumulative sum of angular deviation
$\mathfrak{f}$	Frontier between an unknown and known part of a map $\mathfrak{m}$
$H^{\mathfrak{m}}$	Map uncertainty
H^{P}	Pose uncertainty
l_p	Path length
$\mathfrak{m}$	Map, here commonly an occupancy grid
n_w	Number of waypoints
n_v	Number of Voronoi waypoints
o	Observation
$\mathcal{O}$	Set of observations
ρ	Map resolution
φ	Orientation, yaw angle
t	Time
$\mathfrak{u}$	Odometry measurement
$\mathcal{U}$	Distribution of odometry measurements
v	Velocity
$\mathfrak{x}$	Pose in form of $(x, y, \varphi)^T$
$\mathcal{X}$	Trajectory of poses
$\mathfrak{w}$	Waypoint
$\mathfrak{w}_{gp}$	Goalpoint

1. Introduction

Along with the overall progress in robotics, scenarios where multiple robots act in parallel gained in relevance. Such systems cannot only be improved by enhancing the skills of the individual robots, but also by cooperation of all robots. This thesis focuses on robotic decision making for action selection in cooperative multi-robot systems. The problem is discussed for systems with different characteristics and from diverse perspectives. The subsequent chapters address the substantial challenges of this problem and provide adequate solutions that are not only theoretically grounded, but also are shown to be suitable to solve the problem in practice. This introduction gives the motivation and the outline of this work.

1.1. Motivation

The objective of artificial intelligence (AI) is to create intelligent machines that are able to make smart decisions on their own. This form of intelligence is represented by agents, which perceive their environment, reason about the subsequent actions, and thereupon act respectively. Decision making is referred to as symbolic reasoning in the sense of classical AI, i.e. deliberation based on a symbolic world representation [108]. In the case of embodied agents, which carry out their actions in the physical world, one speaks of robots. This distinctive characteristic makes robots suitable for daily helpers in the real world. Their application areas comprise industrial settings, military fields, and service scenarios.

In [47], the Roadmap for US Robotics is proposed, discussing the future trends and opportunities of robotic markets and the resulting demands to robotic technologies. In a similar study, by the European Robotics Technology Platform (EUROP), the Strategic Research Agenda for robotics in Europe [72] was developed. The major markets of robotics were identified in the areas of manufacturing, logistics, medical robots, healthcare, space, and service robots [47, 72]. Robots are expected to be used as robotic workers and co-workers, for surveillance and intervention, for exploration and inspection, or for edutainment. Service robotics focuses on assisting humans in their daily life and is expected to be an emerging market with high potential. Especially in aging societies, such as in Europe, the US, or Japan, support for the care of elderly is strongly needed [72]. Also the field of manufacturing robotics is seen as a revolutionizing key technology [47].

In larger settings, e.g. hospitals, warehouses, or industrial factories, the needs for assistance rapidly exceed the capabilities of a single robot creating the demand for multiple, simultaneously operating robots. Such a setup is referred to as multi-robot system (MRS). Besides the obvious advantage of tasks being processed in parallel, the productivity of MRSs is further improvable by an efficient cooperation of the robots with each other and

their environment. While setups with competitive agents are in research likewise popular, the practical benefits of cooperative robots are expected to be considerably higher, especially in service or industrial applications.

The authors of [47] expect that it will last until 2020 and beyond before products for a broad range of applications will have established in the market. It is further expected, that robots will get more frequently into physical contact with other robots and humans, e.g. in form of cooperative tasks. This raises the requirements to safety, reliability, and robustness. While today cooperating robots are mostly controlled in a centralized manner and designed for robotic-specific tasks, e.g. the warehouse delivery described in [42], future MRSs will use distributed control, inter-agent communication, and team-specific tasks [72]. On the long-term (beyond 2020) the authors of [72] see systems that are capable of a learning-based automation and online planning in high-dimensional spaces. In order to satisfy these demands from the MRSs perspective, sophisticated solutions for a cooperative action selection in multi-robot systems are required.

From the scientific perspective the major difference of a cooperative MRS to a single-robot system (SRS) arises from the interest to provide the robots with the mutual awareness and the capability to incorporate this information into the robotic decision making. In order to overcome the limits of single robots and exploit the benefits of MRSs, recent research is interested in the cooperation of multiple robots to increase their joint performance. Today, most available solutions are still limited to small scale and/or semi-autonomous systems, provided with pre-scripted knowledge and plans. Instead, future systems need to handle much more complex tasks, act fully autonomously and possibly in highly dynamic environments [47, 72]. The related challenges arise mainly from the uncertain information and the rapidly growing problem complexity. As a consequence, finding the optimal solution, i.e. the one that yields the best team performance, is often intractable for non-trivial problems and thus requires approximative approaches. Accordingly, respective methods need to tradeoff between speed (approximation) and optimality (generality). Generality also provides a greater flexibility to various application domains.

These issues are still insufficiently solved in literature, what raises the potential and demand for future research in the field. One key aspect is the action selection of the individual robots within a team. Therefore, this thesis provides an approach to this key component by introducing a generic task allocation framework. In addition, a novel method is presented to learn the environmental influence on the quality of the action execution and to utilize this knowledge in order to improve the system performance. Moreover, this dissertation will provide experimental work, which verifies these theoretical approaches under realistic practical conditions.

In order to motivate the relevance of the research topic further, Section 1.2 gives a more precise description of potential application domains. Afterwards Section 1.3 presents a detailed taxonomy of the field followed by a discussion of its major challenges in Section 1.4. Finally an outline of this thesis is given in Section 1.5.

1.2. Practical Applications and Demands

The requirements and benefits resulting from a MRS are strongly dependent on its specific application. [92] examines the demands and challenges of multi-agent systems (MASs) in industrial settings. In [28] an overview of commonly used benchmark applications for evaluating MRSs coordination methods is given. The motivation of this section is to show some example application domains in order to illustrate the type of robotic setups considered in this thesis and their respective practical implications.

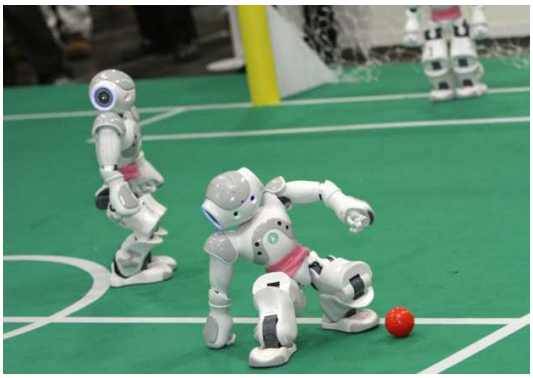
A very popular benchmark domain is *RoboCup Soccer*, see Fig. 1.1(a), where different robotic platforms compete in different leagues against each other. The robots within a team need to find suitable cooperation strategies in order to outplay their opponents. In order to ensure fair play, soccer robots are commonly homogeneous. This means they are constructed equally, provided with the same sensors and actuators.

Another domain where robots benefit from cooperation are planetary exploration tasks. In order to investigate unknown terrain, the robots also need to overcome larger obstacles, such as shown in Fig. 1.1(b) where a mobile robot climbs a cliff with the help of two anchor-bots at the top [52]. In this case the robots are heterogeneous, which means that the robots are equipped with different sensors and/or actuators that provide them with different capabilities. This is also very beneficial in the search and rescue domain, see e.g. Fig. 1.1(c). Robots are supposed to eliminate possible sources of danger or find injured persons in hazardous environments. In such situations, a heterogeneous robot team may operate much faster if some robots are highly mobile and have a very accurate detection system to quickly locate spots of interest, while less flexible but more powerful robots are responsible to bring persons out of the hazardous zone.

Another application domain for MRSs are industrial settings, where the robots are closely integrated into industrial processes. For example, autonomous fork lifters [34, 69] retrieve material from huge scaffolds or storage racks, and other robots may be pure transport vehicles, see Fig. 1.1(e), which carry objects to a specified location. Within these categories, the robots may be further distinguishable, for example with respect to the height they can reach or the maximum payload they can carry. Similarly, the tools and material to be fetched may differ in their attributes, such as size, weight, shape or stiffness. Depending on the capabilities of the robots and the attributes of the objects, the latter may be picked up or carried by a single robot or, in case of heavier and/or larger objects, either by multiple robots, see Fig. 1.1(f), or by humans and robots in a tightly coupled manner.

A further domain containing versatile robotic applications are hospitals. The robots disperse medication and meals and/or make deliveries [27]. In [81], autonomous mobile robots are used for the conveyance of blood samples, see Fig. 1.1(g). Robots also regularly visit and assist patients [26, 88]. They provide an opportunity for doctors to remotely inspect a patient or even to do tele-operated surgery [43]. Robots can move the beds through corridors, carry patients [86, 87] and support the staff.

Even though these scenarios act in different environments they have several characteristics in common. In general the performance of a company is measured by the progress it makes within a certain time. Similarly the assistance by the robots to such a company,



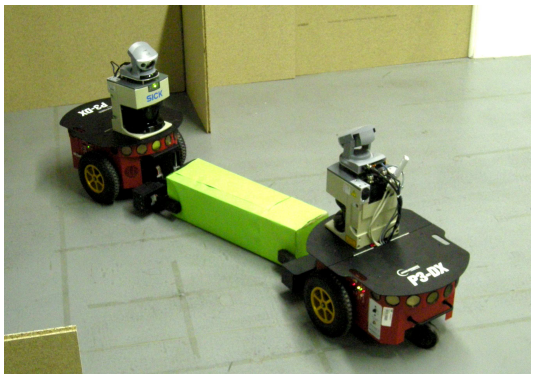
(a) RoboCup Soccer [109].

(b) Planetary exploration [52].

(c) Search and rescue [64].

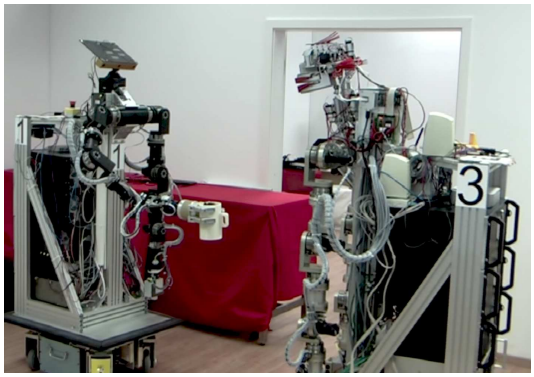
(d) Warehouse with autonomous fork lifter [69].

(e) Warehouse delivery [42].



(f) Cooperative cleanup.

(g) Hospital aide and nursing [81].



(h) Household services.

Fig. 1.1.: Applications and situations where robots can benefit from cooperation.

needs to be determined by the number of relieved tasks within this time. However, the concern management is primarily not interested in the performance of each individual robot – assuming these have overlapping task responsibilities – but rather in the performance of the entire MRS. Accordingly in these cases a MRS is judged by the joint group or team performance of all robots.

An additional characteristic of many applications is that the robots act in a human-populated environment. In these situations, the robots need to adapt to the human work flow, react to respective changes and sometimes even cooperate physically with humans. Consequently they need to cope with the uncertainty given in the physical world and provide a high level of robustness in order to ensure reliability to its operators.

In order to satisfy their operators expectations, the robots need to be reliable but they also need some joint policy how to coordinate their actions with other robots and/or humans in order to increase the joint performance. A simple strategy for the robots to process the dedicated tasks is to randomly wander around and whenever a robot meets a human who instructs it with a task, the robot tries its best to successfully complete the task. This implies no exchange of information among the robots and no inference about the mutual actions. Accordingly, a robot that is for example only able to identify and lift objects, does not inform any other robot, which is only able to carry objects, about the locations of the latter. As a consequence, a major problem of this strategy is that complex tasks can only be accomplished by sophisticated robots. In the current example, the robot needs to be able to detect, load and carry objects. Even in this case, this strategy is most likely highly sub-optimal as the instructing human first needs to find a suitable robot.

It becomes even more problematic when objects need to be passed from one robot to another, see Fig. 1.1(h), or require a synchronized handling by multiple robots. For example a fork-lifter robot needs to load some construction material onto a transport robot or a heavy object requires a joint carrying by multiple robots. Without an explicit policy such a synchronization would occur only by chance.

As a consequence, a cooperation strategy or policy among the robots is required, to improve the overall performance of the robots which enables them to exploit the benefits of the specific MRS. For example in the warehouse system shown in Fig. 1.1(e), which has been already successfully realized in practice [42], the policy is determined by a central control unit that decides which robot should bring which article to which location. However, in this case the robots are all identical, i.e. homogeneous, and are not required to cooperate in a tightly coupled manner with their environment what simplifies the problem considerably. For a MRS composed of heterogeneous robots that also are supposed to execute tasks in a synchronized manner, the challenges with respect to scalability, complexity and robustness are substantially harder and demand for new solutions. On the other hand, such systems are beneficial as they allow for specialization. This results in a larger variety of performable tasks while keeping redundant hardware components and thus also the related costs low. For a more clear understanding of the different types of MRSs and those related benefits and challenges an overview of the determining characteristics of MRSs is given next.

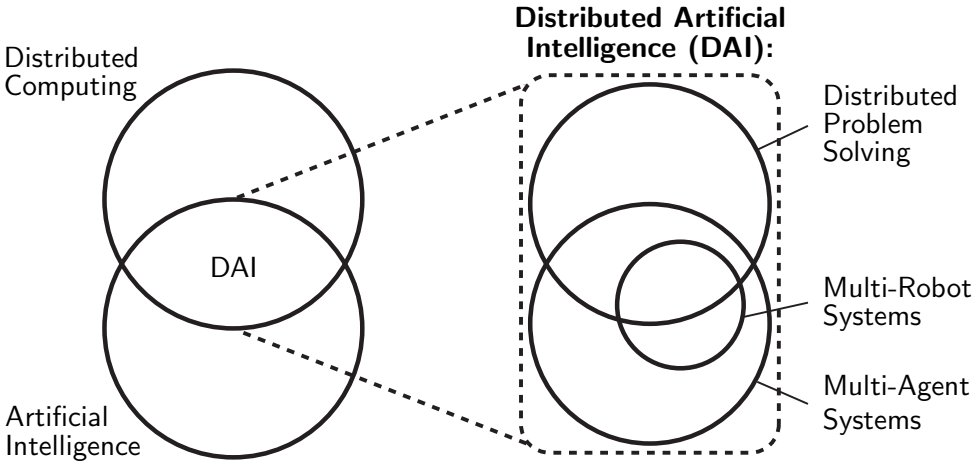


Fig. 1.2.: Embedding of MRSs into the fields of Distributed Computing and Artificial Intelligence.

1.3. Taxonomy of Multi-Robot Systems (MRSs)

This section provides a general overview of multi-robot systems (MRSs) and shows how this thesis is embedded into the related literature. Section 1.3.1 examines how research in MRSs is connected with its most related fields followed by a clarification of the most important terminology. Thereafter, the characteristics of MRSs are described in an intrafield taxonomy given in Section 1.3.2.

1.3.1. Interfield Taxonomy and Terminology

Research in MRSs is a comparably young domain. The first work dates back to the mid 1980s, but the field mainly gained in importance during the last decade. In order to exemplify how research in MRSs is embedded within the domain of Artificial Intelligence (AI), the taxonomy of research fields including Multi-Agent Systems (MASs) – introduced by Stone and Veloso in [114] – is extended by MRSs as shown in Fig. 1.2.

Both fields – MASs and MRSs – are embedded into the domain of Distributed Artificial Intelligence (DAI), which itself is the intersection of Distributed Computing (DC) and AI. DC relates to the study of interconnected programs that share a common goal, but run on separate computers. In this respect, DC became possible owing to the invention of computer networks in the 1960s. AI is the development and study of intelligent machines where the first recognized work was originated in 1943 according to [100].

The domain of DAI splits mainly into the overlapping fields of Distributed Problem Solving and MASs [114]. While Distributed Problem Solving is understood as the management of distributed information, the domain of MASs relates rather to the management of distributed behavior. In this context, MRSs can be categorized as a subfield of the MASs domain that may, but does not necessarily need to, overlap with Distributed Problem Solving.

In order to make this more evident, a clarification of some terminology is required. First of all, it is necessary to get an idea of what is meant by an agent within the AI community.

According to Russell and Norvig:

Definition 1.1 *"an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators" [100, p. 34].*

This statement needs to be understood in a very abstract sense, that is sensors and actuators do not necessarily need to relate to physical objects. For example, a software agent – also called "*softbot*" – may receive its input as data packages or commands in form of byte sequences and similarly send out its own commands or data. Nevertheless, independent of its specific type, an agent is generally expected to act logically leading to the expression of a rational agent.

Definition 1.2 *"For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has" [100, p. 37].*

Accordingly, this provides a clear idea about agents and rationality but it still leaves the question about the difference between an agent and a robot.

Definition 1.3 *"Robots are physical agents that perform tasks by manipulating the physical world" [100, p. 971].*

In contrast to a pure software agent a robot retrieves information from physical sensors such as cameras or laser range finders and sends its commands to motors. However, this does not preclude that a robot may comprise several agents. For example one agent may be responsible for the planning of future actions while another agent handles the execution of these planned actions. W.l.o.g. throughout this thesis the term robot always refers to the rational agent that is responsible for the selection of actions.

In general a system that is constituted by a set of individual agents is referred to as MAS. Accordingly, a MRS is a set of individual robots. With respect to Definition 1.3, this implies that each robot is constituted by at least one robot-specific agent. For example, a distributed set of several physically self-contained robotic hardware that is controlled in a fully centralized manner by one single agent is strictly speaking not a MRS. In fact a MRS differs mainly from a single-robot system by the environment being also influenced by the actions of other robots.

Furthermore, in order to clarify the title of this work still the term of a cooperative MRS needs to be determined. In this respect a definition according to [90] is used:

Definition 1.4 *A cooperative MRS is a set of individual robots, which are aware of each other, share common goals and mutually advance the goals of the others by their own actions.*

In other words, the robots are assumed to know about the existence of each other and their performance is evaluated as a team. From the latter, it also follows that a robot influences the performance of all other robots by its actions. In this respect, Parker sees the objective of the field in the development of MRSs, in which the robots are working together as efficiently as humans, where the actual challenge is to find the best design [90]. The characteristics of cooperative MRSs and existing alternative types are further discussed in Section 1.3.2. The next section describes some exemplary applications which illustrate the practical benefits provided by MRSs.

1.3.2. Intrafield Taxonomy: Multi-Robot Systems at a Glance

During the last decade the domain of MRSs research attracted increasing interest that led to a rising variety of subfields focusing on different aspects with varying assumptions and constraints. This in turn gave reason for several work on the categorization and the survey of the field. In the following the most important work in this respect is summarized and an intrafield taxonomy of MRSs as well as an overview of the benefits and challenges in the field is given.

A very common approach is to categorize the systems based on the type of environment, the type of agents and/or those style of interactions, see e.g. [51, 100, 124]. Some of the taxonomies found in literature go at great length and include for example even the used integration infrastructure or the practice of development and service, e.g. [51]. While such issues also relate to the means of how the specific problems are solved, the subsequent intrafield taxonomy concentrates on the question of what needs to be solved. More specifically, it is considered by which characteristics the MRSs differ physically from each other and by which requirements with respect to their dedicated applications the systems are distinguishable.

Physical Characteristics refer to the physical attributes of MRSs and can be split in those related to the robotic hardware and those related to the system infrastructure [51, 92, 124]. A very essential attribute of the individual robots that strongly determines those applicability is the mobility. While established industrial robots that are integrated within production lines are mostly stationary the majority of the present research is engaged in mobile robotics.

Another very powerful functionality in MRSs is the availability of communication between the robots, as it is a valuable feature for coordination, negotiation or simply information exchange. However, with communication also various problems arise, such as time delay, data reliability or an increased system complexity due to the need for protocols etc. This may lead to a merely limited accessibility of the communication system. In this respect for example [39] defined a value of communication in order to achieve rational communication. Such a value enables the explicit incorporation of the communication act into the action planning and thus to only exchange information when really necessary.

A further mostly used characteristic, e.g. [51, 92, 114, 123], is the uniformity within the system. Depending on the composition of robot types a distinction between homogeneous systems - all robots are identical - and heterogeneous systems - consisting of different robots - can be done. Even though this mostly relates to the robotic hardware in principle robots also can differ solely with respect to their used program. For example for benchmarking reasons a developer might provide some robots with a very limited and other robots with a very sophisticated algorithm. As this provides the robots with a different capability level such a system is of heterogeneous type as well.

Besides the robots also the attributes of the surrounding environment are determining factors of a MRS [51, 123, 124]. Robots may have to act in a static or dynamic environment. Actually an environment with multiple operating robots is already dynamic from a single robot's point of view. However, for locationally very spacious systems a static assumption about the environment might be still acceptable.

Furthermore, a huge part of research focuses on the interpretation of noisy sensor data and the combination of the individual perceptions of the robots. Depending on the current states of the robots, their perceptions may differ temporally, spatially or even semantically, resulting only in a partially observable world state. In this respect a robot without any sensors or communication is not able to observe its environment at all while a fully observable environment does practically not exist, at least for robotic agents. Closely related to the observability is the predictability of the environment, which relates to whether the effects of actions are exactly predictable (deterministic) or if they are subjected to some unknown processes (stochastic). Note that the term stochastic, in contrast to non-deterministic, implies some probabilistic model of the outcome uncertainty [100]. However, w.l.o.g. any process outcome can be approximated by some probabilistic model, the challenge is rather to find a suitable one.

Application Requirements arise from the dedicated application of a MRS and relate to scenario-specific attributes and/or constraints that also allow for a classification of MRSs. In [90] Parker takes a general perspective on applied MRSs and describes three paradigms to cluster systems of distributed intelligence. The first are bio-inspired approaches, which assume a large number of homogeneous robots, i.e. the robots are equally capable to perform the tasks. A huge part of this category is for example covered by swarm robotics. These systems often make use of stigmergy, meaning that the robots sense on their own when their help is needed. This yields a coordinated behavior while keeping communication and complexity low. Nevertheless, the applicability of these approaches is limited to rather simple domains such as foraging or exploration tasks. The other two paradigms described in [90] are the organizational approaches, such as role-based structures, and the knowledge-based approaches, which make use of the mutual awareness of the others capabilities. However, the transition between these two paradigms is anyway rather smooth. All of the work presented in the following can be ascribed to at least one of these two paradigms. While this presents rather a common ground there are several application-specific characteristics that allow for a clear taxonomy of the various system types.

One issue related to the application is the level of cognition required in the system, see e.g. [51, 90]. In some scenarios it is sufficient for the robots to act reactively depending on their current percepts while in others they need to also deliberate about the future effects of their actions. Which approach is appropriate or even mandatory is closely linked to the causal relation of the tasks. In case this is only of episodic character, e.g. reiterative pick-and-place tasks, some reactive behavior may be sufficient. Instead for picking objects that lie upon each other some deliberation about the sequence in which the objects are taken is probably beneficial.

Besides this causal relation tasks may also have an operational relation in the sense that their execution requires some loosely or tightly coupled coordination [28]. For example foraging or mapping are loosely coupled as they require no precise synchronization during execution. In contrast box-pushing or soccer are examples for tightly coupled tasks.

A further application-determined aspect is the level of autonomy. In [51] five classes of autonomy are distinguished. These include for example even the autonomy of the designer in the sense of its independence with respect to interface or construction-related choices.

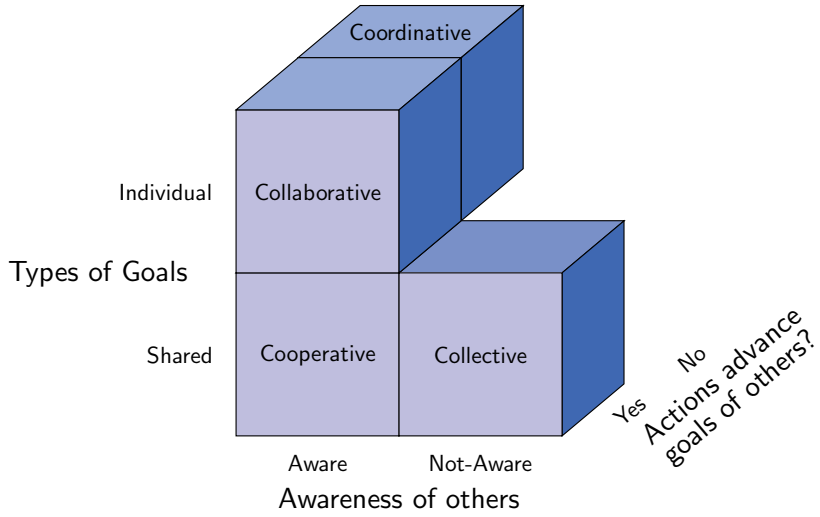


Fig. 1.3.: Categorization of robot interaction styles according to [90].

However, as mentioned at the beginning of this section the taxonomy described here is limited to hardware and application-related attributes. In this respect a distinction of tele-operated systems, fully autonomous ones and mixtures of those is made. Nevertheless, from the MRS perspective it needs to be considered that an increase of interaction with other agents entails already a lower autonomy of the individual agents.

Additionally MRSs are distinguishable according to their type of robot-robot interaction. Therefore commonly the subclasses collective, cooperative, collaborative and coordinative are used, for which in [90] a categorization as shown in Fig. 1.3 is proposed. It is based on the types of goals, which may be individual or shared, the awareness of each other, and whether the own actions advance the goals of others.

Table 1.1 summarizes the list of described characteristics, where it is neither the objective nor the claim of the author that this list is complete. In fact MRSs could be further classified with respect to the amount of knowledge provided by the designer, the capability of short or even long-term learning or whether a centralized or decentralized architecture is used. Nevertheless, as previously mentioned, the list in Table 1.1 has been kept small on purpose as it shall rather provide a reduction to the most essential attributes of multi-robot systems. For an extended overview the interested reader is referred to the above cited literature.

The General Benefits and Challenges of MRSs are of interest in order to decide why, where and how to apply MRSs. In this respect it seems manifest to determine their advantages compared to single-agent systems (SASs) or single-robot systems (SRSs). The primary answers given in the literature, see e.g. [25, 90, 114, 123], comprise:

- Computational and operational speedup due to the ability of a parallel and/or distributed employment.
- Robustness and fault tolerance due to redundancy and no single point of failure.
- Scalability as robots can be easily added or removed as needed.

Characteristic	Subcategories
Robot:	
Mobility	stationary / mobile
Communication	none / limited / unlimited
Uniformity	homogeneous / heterogeneous
Environment:	
Variability	static / dynamic
Observability	none / partial / full
Predictability	deterministic / stochastic
Application:	
Level of cognition	reactive / deliberative
Causal task relation	episodic / sequential
Operational task relation	decoupled / loosely coupled / tightly coupled
Level of autonomy	tele-operated / semi-autonomous / fully autonomous
Style of interaction	collective / cooperative / collaborative / coordinative

Tab. 1.1.: Intrafield taxonomy: a collection of robot-, environment- and application-specific characteristics of MRSs.

- Eased development due to higher modularity and code reusability compared to a single complex system.
- Lower hardware costs as multiple specialized robots allow for more simplicity and low cost units compared to a single complex robot.
- Feasibility of tasks whose requirements exceed the limitations of single robots, e.g. distributed sensing.

However, closely connected to these benefits are also the respective challenges of MRSs. In [92] the scalability, uncertainty, decidability and the sensitivity of the system are identified as the four major operational challenges for manufacturing applications. While environmental uncertainty and system sensitivity likewise apply to SRSs, the scalability and decidability are considerably harder in MRSs. This results mainly from the distributed existence of knowledge and the tremendous combinatorial increase of possible solutions. In this respect Parunak sees the scalability as key factor, which to handle requires either supercomputers or applications that allow for slow response times [92]. However, for a MRS in general both is not given demanding for means to find acceptable solutions based on the given computational resources and within an acceptable amount of time.

An advantage in homogeneous systems is that many solutions are redundant and thus need to be evaluated only once. This reduces the computational effort during planning and can be further beneficially used to achieve more robustness and fault tolerance as malfunctioning robots are easily replaceable. In contrast, the increased functional variety in a heterogeneous system of specialized robots allows for a multifaceted applicability. However, this enhanced variety involves mostly a dramatic increase of the number of alternative solutions and thus of the computational complexity. Furthermore it also demands

a far more elaborate resource allocation and decision making as malfunctioning robots or faulty system components are not that easily exchangeable as in systems with homogeneous robots.

A straightforward approach to solve these problems may be to use proven methods from the MAS domain. However, a direct transfer to MRSs is often not possible as pointed out in [121]. MASs are commonly composed of software agents, underlying no locational or hardware constraints. Consequently software agents can easily exchange or combine their capabilities. Also the input information is mostly reliable and the effects of commands are in general predictable. In contrast the sensor or actuator capabilities of a robot are locationally fixed and the retrieved information or the resulting effects of actions are subject to unknown disturbances. This leads to additional challenges robotic agents operating in real-world settings are faced with. The resulting information unreliability leads to a partial observability of the environment, which in general allows only for a stochastic predictability of the latter. Also the restriction of resources and the exposure to hardware faults need to be explicitly considered. Moreover, Dias mentions in [25] that also the type of tasks in robotic systems may vary significantly to those in softbot systems. Robotic tasks may be constrained by physical requirements what for example can lead to additional latencies due to distant resources.

Accordingly all of these issues need to be taken into account in integrated robotic systems. As the transfer of MASsapproaches to MRSs is often not applicable or only with modifications a demand for research in MRSs is given. In this respect, in [2] seven research areas in the domain of MRSs have been identified, which meanwhile can be extended by multi-robot learning as eighth category:

- Biological inspirations;
- communication;
- architectures, task allocation, and control;
- localization, mapping, and exploration;
- object transport and manipulation;
- motion coordination;
- reconfigurable robots and
- multi-robot learning.

All of these still leave a lot of space for future investigation. The present work on cooperative action selection in MRSs is embedded into the area of task allocation and thus belongs to the third category. The importance of this field is further grounded in [92], where reliability, resource allocation and coordination are identified as major research issues for real industrial applications. The explanation of what is actually understood as cooperative action selection in MRSs is given next.

1.4. Open Challenges in the Field

The major challenges of an efficient action selection in cooperative MRSs arise mainly from the existing **information uncertainty** and the huge **problem complexity**, as for example also identified in [47, 72].

The uncertainty originates mainly from the partial- or even non-observability of the environment and of the states of the other robots. While single robots are similarly confronted with environmental uncertainty the distributed existence of the information and the resulting uncertainty about the other robots is specific for MRSs. This is even more significant when robot-robot communication is not available as in this case the intentions and states of the team members need to be fully estimated. Accordingly, a communication network is a highly valuable system resource in MRSs. Since nowadays, the usage of such communication networks is neither technologically nor economically problematic but instead strongly increases the versatility of a MRS, robot-robot communication should be always integrated if possible what also is assumed in the following.

The problem complexity arises mainly from the combinatorially explosive number of possible solutions. In this respect the controllable complexity directly determines the feasible scalability of a system. It needs to be considered that in contrast to a SRS, in a MRS the changes of the environment are not only influenced by a single robot but rather by the combination of the actions of all robots. As a consequence the solution space is significantly larger as all possible tuples of team actions and those commonly non-deterministic effects need to be considered. This is even harder for heterogeneous systems in which the robots are capable to perform differing actions. In these the fraction of equivalent solutions is in general a lot lower compared to homogeneous systems where all robots are capable of the same set of actions.

The above described complexity and uncertainty often lead to *NP*-hard problems for which exhaustive search strategies are no longer feasible. This in turn demands for alternative solutions, such as approximative approaches that focus the search on the most promising regions of the solution space. Even though many existing approximative methods perform successfully in specific scenarios and applications they face already problems in case of small changes in the environmental setup. For example the comparison of four extreme task allocation strategies in [74] showed that no strategy performed best in all situations. Consequently, in order to provide a comprehensive applicability respective methods need to be kept general while also providing an easy extendability. This is for example achievable by providing a core solver with the essential functionalities and enabling the incorporation of application-specific heuristics that account for characteristic aspects of the problem in focus.

Besides appropriate means to cope with the vast amount of possible solutions it is of further importance to keep the system complexity low enough to leave it also manageable by humans, e.g. by ensuring a low communication overhead and a modularized architecture. Furthermore the degree of heterogeneity in a MRS needs to be carefully considered during system design. While a higher heterogeneity increases in general the functionality and applicability of a MRS it also raises the challenges with respect to system robustness, optimality and manageability. A robust operation in a dynamic environment for example

requires fast situation-aware reactions and a forecasting as well as a handling of failures. Such skills imply a decision policy that is capable to trade off between the best possible performance and the imminent risk.

This leads to the next major problem namely to get an accurate estimate of the expected performance as well as the respective risk. The performance is determined by the efficiency of the execution, which in turn results from a functional relation of all performed actions and the current environmental state. This functional relation is application- and MRS-specific and commonly not given at all or only a rough approximation of it, at least for non-trivial problems. In order to yield an acceptable degree of consistency between prior planning and the posterior execution a respective accuracy of the underlying estimation model is essential. So far this problem is often neglected in literature and there exists only few work in this direction what demands for future investigation.

In summary, besides on the capability to handle the problem complexity and information uncertainty, the system performance of future MRSs will also be strongly dependent on the ability to dynamically adapt to environment changes. These demands lead to the importance and potential of present and future research in the MRS domain in order to combine the requirements of a dynamic system scalability, a situation-aware action selection, and an efficient failure handling while still keeping a manageable complexity.

1.5. Outline and Contributions

The subsequent chapters focus on the challenges mentioned above and present methods to cope with the complexity as well as the uncertainty in MRSs.

- In Chapter 2 the cooperative action selection problem is discussed. It is shown how respective solutions can be integrated into an entire robotic architecture. Thereafter a formulation of the action-selection problem is given followed by an illustration of a well-known taxonomy on multi-robot task planning problems.

- In Chapter 3 the market-based MuRoCo (Multi-Robot Cooperation) framework is presented, which handles the multi-robot task allocation problem in heterogeneous MRSs. In the course of a negotiation procedure tasks are allocated to the robots, which then complete the tasks by selecting appropriate sequences of actions. MuRoCo yields optimal solutions for the instantaneous assignment of tasks that require a tight cooperation of multiple robots (ST-MR-IA), commonly known as coalition formation. Major contributions of MuRoCo are a two-phase allocation procedure, which also optimizes the subtask assignment among the coalition members and yields a slower increase of the worst-case complexity compared to state-of-the-art approaches. Additionally, in order to also reduce the complexity arising on average, several pruning strategies, which take system-specific characteristics into account, as well as the distributed version MuRoCo-D, where the computational load is shared among the robots, are proposed. In order to cope with the omnipresent uncertainty, means for failure prevention, forecasting, handling and removal are integrated. Altogether, MuRoCo presents an exhaustive framework that is based on a generic problem formulation. This allows for a high versatility to diverse applications and domains.

The suitability of the presented framework for the robust operation of a complex MRS even under various sources of uncertainty, is verified in a service scenario with a MRS composed of four heterogeneous robots.

- In Chapter 4 a novel approach is described to understand the influence of the environment on the robotic performance and to utilize this knowledge to yield a risk- and situation-aware reward and cost estimation. More specifically, a bipartite approach is proposed to first identify the crucial factors that influence the cost metric most and to learn how the latter is influenced, and second to use this knowledge for a better decision making. This is of high importance as the quality of the cost estimation strongly determines the validity of the planned solutions. In this respect, the presented method solves a fundamental but in literature often neglected problem, in regard to the action selection of real-world robots.
  - More specifically, in the first part a method for a system interdependence analysis is introduced. It aims at learning a Bayesian network that models the interdependencies between selected performance indicators of different system components of autonomous robots, as well as of the influence of environmental parameters on the system. In a subsequent information-theoretic analysis the strength of the coherence among the parameters is quantitatively identified. The analysis enables to determine whether the current situation has an effect on the robotic system, what also is shown by experimental results. The method provides an alternative in comparison of deriving the deterministic system model, what may be quite hard for complex systems, or it also can be used to verify the latter. The respectively gained knowledge may help the designer to guide future research but also can be directly incorporated during system operation.
  - This is considered in the second part of Chapter 4, where an approach is described that makes use of this knowledge to yield a situation-aware and risk-concerted reward estimation by usage of a quantile-based reward computation. The quantile-based reward estimation enables the consideration of large variances and even of asymmetries in the respective indicator distributions. Benefits of the presented method are that nonlinear effects of the environment on the reward are taken into account. Furthermore, a single model of the environment suffices to serve multiple types of inference requests. This keeps the demand for training data comparably low. The approach is suitable to learn a model from scratch but also allows the incorporation of full or only partial prior knowledge. The experimental results show the applicability also for environments with large uncertainty. In the end an outlook on how to avoid an upcoming performance deterioration by an adequate failure-forecasting is given.
- In Chapter 5 the conclusion of this thesis is given, which discusses its major contributions and presents an outlook on directions for future work. References containing previously published parts of this thesis are listed at the end of the Bibliography.

2. Cooperative Action Selection

Along with the overall advancement in robotics, scenarios where multiple robots act simultaneously within the same environment become more important. In these scenarios, the robots can increase their team performance by a coordinated selection of actions. In this respect, the cooperative action selection problem refers to the joint decision making about which action each individual robot should choose in order to maximize the common performance of the entire group. Amongst the crucial questions is the choice of organizational structure. An overview of organizational paradigms for multi-robot systems is given in [48].

One of the earliest approaches to achieve multi-robot coordination is the so-called "emergent coordination". In such systems, robots coordinate their behavior based solely on their perceptions and local interactions. Very little or no communication is used and no specific allocation of tasks occurs [37]. These systems benefit from easy scalability and high parallelism, but the difficulty in emergent coordination is the hard predictability of the resulting behavior. Therefore, in scenarios where humans want the robots to perform specific tasks [37], and in situations that require a more direct type of cooperation than in swarm systems, the class of intentional coordination is far more appropriate.

The amount of related literature is huge and the most relevant one is discussed in the respective parts of the subsequent chapters. This chapter illustrates how respective solutions can be embedded into an overall robotic system architecture. This will help to understand the operation of a MRS on the whole and will facilitate to embed, use, and/or transfer the methods presented in the remainder of this thesis in/to other systems. Furthermore, a formal definition of the action selection problem is given, which provides the basis for the next chapters and also will enable future work in this field to relate to the definitions presented here. Finally, a general and well-established taxonomy of problem types is presented, which allows the categorization of this and related work.

2.1. The Scope of Action Selection within a General Architecture for Complex MRSs

The present work considers the selection of actions in a set of cooperative robots. However, in order to operate a MRS in domains such as those described in Section 1.2 further components are required. An overall system structure that defines how these components are interconnected is commonly referred to as architecture.

Various types of robotic architectures have been proposed in literature. However, here it certainly makes no sense to start a discussion on robotic architectures as this topic could easily fill an entire thesis on its own. The interested reader is referred to respective litera-

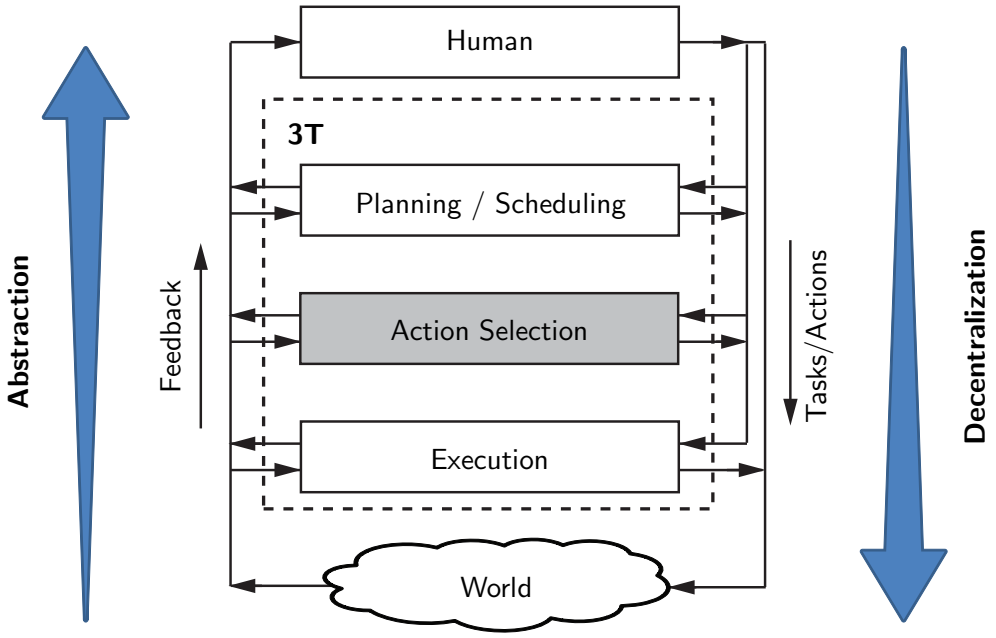


Fig. 2.1.: General scheme of a multi-layered multi-robot system architecture.

ture, such as [66] where an exhaustive discussion about research on cognitive architectures can be found.

Instead of a survey of architectures, in the following the structure of a 3T architecture is used to exemplify how action selection components can be integrated into the entire operational system of a MRS. 3T architectures are composed of three interacting layers or tiers and have been already investigated in the 80's, see [35] for an overview. Meanwhile, many applications and implementations of 3T-style architectures can be found in the literature, see e.g. [8, 70], which can be seen as indicator of its generality and easy transferability. This should not be understood as claim that a 3T architecture is the best or only possible solution, but it is a very suitable one when considering complex heterogeneous MRSs as will be shown in the following.

Figure 2.1 shows a general scheme of the 3T architecture for MRSs and its interrelation with the human and the world. The human represents the layer above and the physical world the layer below the MRS. The MRS architecture itself is similar as proposed in [9]. It is partitioned into a planning and scheduling layer, an action selection layer and an execution layer. Task and action commands are passed from higher to lower levels while the corresponding flow of feedback information runs reversely. Tasks can be seen as kind of high-level goals that can be achieved by performing an appropriate sequence of actions. A more precise definition follows in Section 2.2.

The human is the superordinate operator of the system who can issue tasks at any time to any of the other layers including the physical world, i.e. the human can also actively participate in the physical action. In this respect the human is assumed to instruct either the overall system as an entity or a specific individual robot, as illustrated in 2.2, but it does not attend to create detailed plans about when which robot should perform which action. This is left to the MRS itself.

In this respect, the tasks issued by the human may be very abstract circumscriptions of

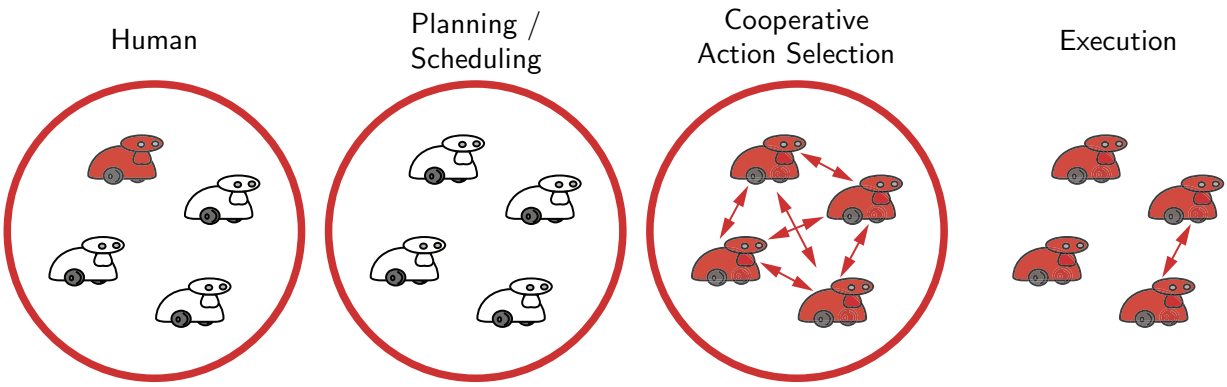


Fig. 2.2.: A MRS as it is addressed by the different layers of the architecture in Fig. 2.1. The highlighted parts are in focus of the respective layer, where the circle represents the overall system as an entity.

complex processes or work flows, which are not directly executable by a single robot. Thus it is the responsibility of the planning and scheduling layer to transfer and decompose these instructions into deliberative plans composed of robot-executable task and action commands. Appropriate methods can be found in the field of scheduling algorithms, see e.g. [11] for an overview, or in the planning domain, e.g. [79]. It should be mentioned that it is common practice in the planning domain to talk of goals instead of tasks, where a goal refers to a desired goal state of the world. A generic formalism to represent the effects of actions is for example ADL[1] [93], which is an advancement of STRIPS[2] [31]. ADL is seen more appropriate for robotic problems compared to STRIPS. Amongst others due to its modified world representation that allows for example to model also context-dependent effects of actions in contrast to STRIPS. A further advancement and an attempt to find a standard based on ADL and its competitors is the nowadays widely-used PDDL[3] [38], which is a kind of byproduct of the International Planning Competition[4]. The latter is also a promising starting point to search for state-of-the-art planning algorithms. These provide the means for the actual reasoning based on the formalisms and provide the tools to generate task and action plans while considering timing constraints and respective causal dependencies.

In principle, these planning formalisms could be used to describe the relevant part of the world including the robots as well. So detailed action plans could be generated that exactly specify which robot is supposed to perform which action at which time. However, depending on the complexity of the processes and work flows the respective planning problems are often *NP*-hard and thus require already approximations to be solved at the system level, i.e. by considering the group of robots only as a single system entity. Nevertheless, even more problematic is the fact that robots act in a physical world where they are prone to perceptual noise and uncertainty. As pointed out in Section 1.3.2, this presents a major difference to multi-agent systems, where single agents may correspond to pure software agents that may perform a deterministic sequence of computational opera-

[1] ADL: Action Description Language
[2] STRIPS: Stanford Research Institute Problem Solver
[3] PDDL: Planning Domain Definition Language
[4] http://ipc.icaps-conference.org/

tions. The uncertainty in a robotic system leads permanently to deviations of the actual outcome of executed actions and accordingly of the respective plan. Actions may even fail or unforeseen events may occur. An accurate prediction of the future process becomes even harder in case the human takes part in the physical action. Accordingly the original plans would be permanently violated demanding for replanning in order to react to changes in the dynamic environment.

As a consequence, it is beneficial to split the entire planning process into two layers as proposed by the 3T architecture in Fig. 2.1. The planning and scheduling layer is responsible to generate a system schedule in order to fulfill the human instructions on a system level. Based on this schedule the currently pending tasks are issued to the action selection layer, which is responsible to allocate the tasks among the robots and select the appropriate actions with respect to the current situation. For this purpose the action selection layer is also provided with task and action plans from the scheduler. These plans present solutions how to decompose tasks into sequences of actions or subtasks. However, these provide only general information of possibly alternative solutions how to perform specific tasks but they are not taking the current environmental situation into account.

Situation-awareness is primarily the responsibility of the action selection layer. In single-agent implementations of 3T architectures the middle layer is often referred to as sequencing layer [8, 70], which is supposed to activate actions dependent on the current situation in order to react to dynamic changes in the world. Even though here the actual function is similar, the term sequencing may be misleading since the main purpose of this layer in MRSs is to exploit the ability of parallel execution. Accordingly it is referred to as action selection layer throughout this work. Its responsibility is to allocate the tasks issued from the higher layers to the individual robots and select the appropriate actions in order to optimize the performance of the entire system by explicitly considering the current situation and the present uncertainty. Respective methods are discussed in the subsequent parts of this work. One essential assumption related to this layer is that of task independence [37], i.e. all tasks issued from higher layers can be performed independently without considering causal relations etc. This independence is assumed to be assured by the planning layer, which only issues tasks that are either independent or in case of dependent tasks combined in a respective task plan as a kind of parent task.

The selected actions are finally forwarded to the execution layer where they are transformed into hardware-specific motor commands. This transformation occurs often by respective control methods, where in the control domain actions are commonly referred to as symbolic skills. In this context each robot focuses on the optimal execution of its own actions. Explicit cooperation occurs only in case of coupled tasks, i.e. tasks that require a synchronized execution. The latter is for example achievable by trigger messages or the usage of decentralized control methods such as in [76].

With the planning layer above and the execution layer below, the embedding of the action selection into the system architecture is clarified, leaving the demand for an explicit formulation of the problem itself.

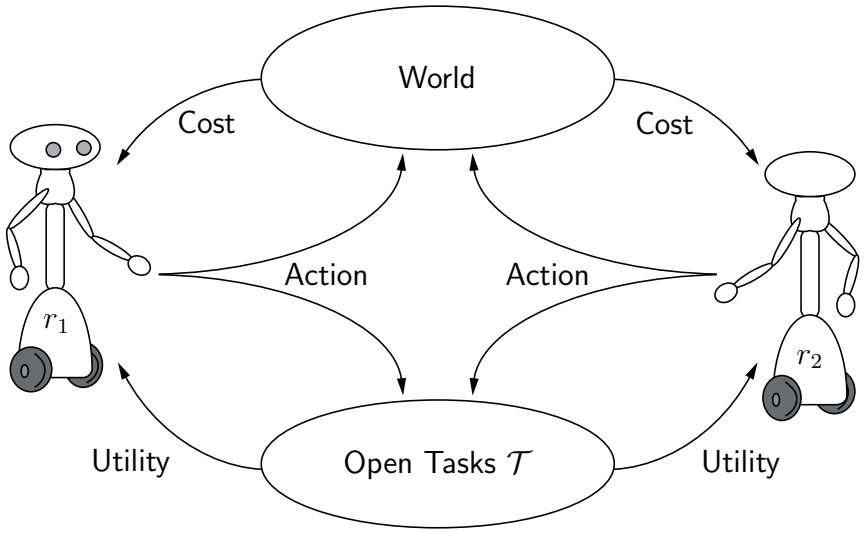


Fig. 2.3.: The action selection problem in cooperative multi-robot systems.

2.2. Problem Definition

The interrelations in a MRS that lead to the demand for cooperation among the robots are illustrated in Fig. 2.3 exemplary for two robots. In general, a MRS is composed of a set

$$\mathcal{R} = \{r_1, r_2, \ldots, r_n\} \tag{2.1}$$

of n robots r. Furthermore there is a set

$$\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\} \tag{2.2}$$

of m unaccomplished tasks τ currently issued to the action selection layer of the MRS. For cooperative MRSs holds, as previously shown in Fig. 1.3, that the robots share their goals. In this respect, whenever a task τ is completed by the robots, the entire group of robots obtains an utility $u(\tau) \in \mathbb{R}_0^+$ that reflects the profit gained by the completed task. Every robot r_i is able to perform a set

$$\mathcal{A}_i = \{a_{i1}, a_{i2}, \ldots, a_{ik}\} \tag{2.3}$$

of actions a. In order to complete a task τ_j and obtain the respective $u(\tau_j)$, the robots need to execute the correct sequence of actions. Such a sequence is referred to as action plan

$$p_{\mathcal{A}}(\tau_j) := \langle a_{j1}, a_{j2}, \ldots, a_{jl} \rangle \,, \tag{2.4}$$

which defines a decomposition of a task τ_j into a tuple of actions. $\mathcal{P}_{\mathcal{A}}$ is the set of all action plans, which is assumed to be retrieved from the planning layer. The plans present generic instructions independent of the specific situation. Consequently it is reasonable to once compute all feasible solutions and store these into a knowledge base. Considering for example a warehouse scenario, a task might be to bring an article to the package station and a possible action plan is to drive to the shelve the article lies in, grasp the article, drive

to the package station, and release the article on the conveyor belt. Only after performing all of these actions in the correct sequence the task is accomplished and the utility is obtained. The utilities are e.g. specified by the application, predefined by the designer or calculated by the planning layer and are assumed to be fixed for a specific action plan. In other words, all executions of a plan $p_{\mathcal{A}}(\tau)$ will always generate the same $u(p_{\mathcal{A}}(\tau))$.

However, each action a executed by the robots results in some cost $c(a, \boldsymbol{s}) \in \mathbb{R}_0^+$. This cost is not only dependent on a but also on the current state of the world $\boldsymbol{s} \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. A state $\boldsymbol{s}$ is a symbolic representation of the status of all robots and the environment. It includes static knowledge, e.g. a map of a building, as well as transient information, such as which robots are currently busy in performing tasks and which are not. The generation of such a state representation is certainly a non-trivial task and in full generality still considered as unsolved [108]. In practice, robots are only provided with partial representations that describe at least the part of the environment that is relevant for the completion of the dedicated tasks. Respective information is either given by the designer, or gathered by the robot itself by using its perception and learning capabilities. As this problem is not in focus of this thesis, in the following a respective state representation is assumed to be given.

In order to reason about the current $\boldsymbol{s}$ and deliberatively decide which action to take next, a robot also needs a model about the influence of the environmental state $\boldsymbol{s}$ on the arising cost $c(a, \boldsymbol{s})$. Similarly, this may be either provided by the designer and/or learned by the robot itself, as discussed in Chapter 4. During the execution of an action plan $p_{\mathcal{A}}(\tau_j)$, the action costs of all $a \in p_{\mathcal{A}}(\tau_j)$ contribute to the overall cost

$$c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) := f\left(c(a_{j1}, \boldsymbol{s}_{j1}), \ldots, c(a_{jl}, \boldsymbol{s}_{jl})\right) \in \mathbb{R}_0^+, \tag{2.5}$$

of the task τ_j, where $f(\cdot)$ is a MRS-specific function and $\boldsymbol{s}_j = \langle \boldsymbol{s}_{j1}, \ldots, \boldsymbol{s}_{jl} \rangle$ is the tuple of action-related world states. The net gain received for the completion of a task τ_j is given by the reward

$$\varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) := u(p_{\mathcal{A}}(\tau_j)) - c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \in \mathbb{R}, \tag{2.6}$$

which is the difference between the obtained utility and the overall cost of all actions required to complete τ_j.

As discussed in the previous section the premise of task independence is assumed, i.e. tasks can be performed independently without considering causal relations. Accordingly follows that independence holds for the related rewards as well. In this respect, the reward function is assumed to be an additive function, i.e.

$$\varrho((p_{\mathcal{A}}(\tau_1), \boldsymbol{s}_1) \wedge (p_{\mathcal{A}}(\tau_2), \boldsymbol{s}_2)) = \varrho(p_{\mathcal{A}}(\tau_1), \boldsymbol{s}_1) + \varrho(p_{\mathcal{A}}(\tau_2), \boldsymbol{s}_2).$$

Accordingly are potential interdependencies among the costs, occurring during the parallel execution of tasks, assumed to be taken into account by the functional relation $f(\cdot)$ of the action costs. Means to obtain $f(\cdot)$ for a specific MRS are discussed in Chapter 4.

The overall objective of a cooperative MRS is that the robots select future actions such that the reward obtained during system operation is optimized. In this context the

performance

$$q := \frac{1}{\beta_u} \sum_{j=1}^{m_c} \left(u(p_{\mathcal{A}}(\tau_j)) - c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \right) = \frac{1}{\beta_u} \sum_{j=1}^{m_c} \varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \in \left]-\infty, 1\right], \quad \text{where } \beta_u = \sum_{k=1}^{m_a} u(\tau_k), \tag{2.7}$$

is measured by the ratio of the obtained reward relative to the obtainable reward. m_c is the number of completed tasks and m_a the number of all tasks issued to the system with $m_c \leq m_a$ and $\bigcup \tau_j \subseteq \bigcup \tau_k$. For an instantaneous action selection, i.e. without consideration of the long-term performance, the normalization by the overall obtainable utility β_u can be neglected corresponding to a maximization of the overall obtained reward. However, in order to enable a performance comparison between different MRSs or of a MRS within different time periods or varying environmental conditions, a comparative value is needed. Accordingly, with respect to the physical definition of power the achieved reward could be set into relation to the operation time of the MRS. However, in this case the performance would be strongly affected by the frequency and type of upcoming tasks. Therefore the overall obtainable utility was chosen, which provides a measure independent of the number and type of tasks.

Since the performance $q \in \left]-\infty, 1\right]$ results from the actions of all robots, the optimization of q requires a cooperative selection of actions by the robots. This is achievable by a respective common policy ϖ that is followed by all robots. A policy ϖ is a mapping from states to actions, $\varpi : \mathcal{S} \rightarrow \mathcal{A}$, where Π is the set of all possible policies. So in case a robot r knows the current state $\boldsymbol{s}$, $\varpi(\boldsymbol{s}, r)$ tells it which a to perform next. As this action a is always part of some action plan $p_{\mathcal{A}}(\tau)$ for some task τ, a policy ϖ consequently always also implies the prior selection or assignment of tasks.

The suitability of ϖ is dependent on the specific system setup and the respectively arising challenges, and is evaluated by its corresponding performance $q(\varpi)$. A policy ϖ_x is superior to ϖ_y if ϖ_x results in a higher performance, i.e. $q(\varpi_x) > q(\varpi_y)$. So the optimal solution for the cooperative action selection problem in a MRS is that the robots follow the optimal policy

$$\varpi^* := \arg\max_{\varpi \in \Pi} q(\varpi), \tag{2.8}$$

which yields the best possible performance

$$q^* := q(\varpi^*). \tag{2.9}$$

In other words, solving (2.8) optimally requires to determine, among the set of all possible action-state-robot triples, the subset of action-state-robot triples that yields the highest performance. This presents a combinatorial optimization problem, which is often – due to the vast amount of possible triples – *NP*-hard. In this respect, algorithms to solve this problem try to efficiently reduce the space of possible solutions.

In case of a problem setup, where finding a solution to (2.8) is intractable, e.g. due to

insufficient computational resources, the alleviated form

$$\varpi' := \arg\min_{\varpi \in \Pi'} (q^* - q(\varpi)) \tag{2.10}$$

is used, i.e. to find a sub-optimal policy ϖ' whose performance

$$q' := q(\varpi'). \tag{2.11}$$

is as close as possible to the optimal solution. Hereby $\Pi' \subseteq \Pi$ is the set of all known or evaluated policies. For practical problems finding a solution to (2.8) is indeed often intractable, wherefore approximative methods are used to find best possible solutions to (2.10).

Before approaches to tackle (2.8) or (2.10) are discussed in subsequent chapters, the remainder of this chapter gives a generic classification of problem types and an identification of the most important challenges in the field.

2.3. Taxonomy for Multi-Robot Task Planning Problems

The vast amount of research in the field during the last decade resulted in a large variety of MRS approaches for different kinds of problems. However, many of these approaches focus on similar problems and often can be seen as instances of each other. In order to be able to benchmark different approaches or to identify requirements to solve a specific problem, a general classification of MRSs has been proposed in [37], which became widely accepted in the literature. Originally the work in [37] was intended to classify instances of the multi-robot task allocation (MRTA) problem. The MRTA problem assumes task independence, i.e. that any task issued to the MRS can be executed independent of all other issued tasks. Nevertheless, the taxonomy also allows for a more generic categorization of MRS task planning problems regardless of task independence. The classification of [37] is performed according to the following characteristics:

- Single-Task robots (**ST**) vs. Multi-Task robots (**MT**),
  i.e. the capability of the robots to perform only one single task or multiple tasks at the same time.

- Single-Robot tasks (**SR**) vs. Multi-Robot tasks (**MR**),
  i.e. tasks that can be executed by a single robot on its own compared to multi-robot tasks that require the cooperation of multiple robots.

- Instantaneous Assignment (**IA**) vs. Time-extended Assignment (**TA**),
  where **IA** means the available information allows only to plan the assignment of tasks which are executed immediately. In contrast, **TA** implies the availability of knowledge about the task arrival process or the set of all future tasks and thus the possibility to also consider future task assignments during the planning.

This taxonomy provides a general distinction of eight classes of MRS task planning problems and helps to identify the demands and challenges associated with a specific problem.

Actually actions are not explicitly considered by this taxonomy. However, the above characteristics determine the set of feasible action plans and thus also the typical constraints and challenges arising for the action selection.

With respect to the architecture in Section 2.1, it is assumed that the planning layer takes care of the time-extended (TA) assignment problem. Respectively all tasks issued to the action selection layer are supposed for instantaneous assignment (IA). Still the action selection problem remains often very hard to solve due to the challenges arising in practical applications.

2.4. Summary

In this chapter the problem of cooperative action selection, which refers to the joint decision making about which action each individual robot should choose in order to maximize the common performance of the entire group, has been introduced. It has been shown how solvers for the action selection problem can be incorporated into an overall MRS architecture. This helps to understand how solutions for the action selection need to be connected with high level planning/scheduling components and low level execution modules, and will facilitate transfer of the methods presented in the following chapters to other systems. Furthermore, a clear formulation of the problem itself was given, which also enables future work in this field to relate to the definitions given here. Finally, a widely accepted taxonomy of task planning problems has been described, which enables the classification of specific problem instances and facilitates the identification and embedding of present and future research in the field.

Concerning this research, adequate solvers to the action selection problem need to cope mainly with the related problem complexity and the high information uncertainty, as already discussed in Section 1.4. Furthermore, they need to be capable of dynamically adapting to environmental changes. In this respect, Chapter 3 presents a framework for an efficient and robust action selection by focusing on multi-robot task allocation.

3. A Framework for Action Selection focusing on Task Allocation in MRSs

This thesis focuses on scenarios where multiple robots, which act simultaneously in the same environment, can benefit from cooperation. In the previous chapter, the problem of cooperative action selection in multi-robot systems (MRSs) has been introduced. In this respect, tasks are issued by humans to the MRS. The robots are provided with a set of action plans, which determine alternative sequences of actions to be executed in order to complete a specific task. Consequently, the action selection problem can be split into two problems. The first of which is to select the best possible action plan for a specific task, and while the other is to decide which robot should execute this specific task. While the selection of the best possible action plan can be determined in linear time, the more complex problem is to find a group agreement on the distribution of tasks amongst the robots, commonly known as multi-robot task allocation (MRTA) problem. Even under the availability of robot-robot communication, the MRTA problem may still remain very hard to solve. In the literature, various approaches for different instances of the MRTA problem can be found. However, so far the problem complexity is still insufficiently handled. Furthermore, only few approaches integrate appropriate means to cope with the environmental uncertainty, such as failure handling.

In this respect, in the following the MuRoCo (Multi-Robot Cooperation) framework is introduced, which is an exhaustive framework to handle the MRTA problem for complex MRS operating in dynamic environments. MuRoCo enables tight cooperation amongst multiple heterogeneous robots by solving the MRTA problem for cooperative and communicative MRSs. This requires not only the assignment of tasks to single robots, but also to teams of robots. In the literature, this assignment of multi-robot tasks is commonly referred to as "coalition formation" and known to be a *NP*-complete problem. The major contribution of MuRoCo is a lower increase of the worst-case complexity compared to previous solutions, while guaranteeing optimality for sequential multi-robot task assignments. Furthermore, in order to ensure a robust operation in dynamic environments, MuRoCo takes potential disturbances and the environmental uncertainty explicitly into account by providing capability- and situation-aware solutions for real world systems. The framework is theoretically analyzed and is validated in a cooperative service scenario, showing its suitability to complex applications, its robustness to environmental changes and its ability to recover from failures.

3.1. Introduction

The availability of a communication network provides a huge advantage with respect to a cooperative action selection in MRSs, as it enables the robots to mutually exchange information in order to reach a consensus. Nowadays, communication networks are easy to install and maintain for comparably low costs. Modern wireless technologies allow for seamless data exchange at high bandwidth among many participants while covering huge areas. Consequently, it is strongly beneficial to make use of robot-robot communication if possible, what commonly applies for complex modern MRSs.

With respect to the architecture described in Section 2.1 (p. 17), the action plans, which are the sequence of actions required to solve a task, are assumed to be provided by the planning layer. So when a task is assigned to a robot, it simply needs to select the next action according to the given plan, thereby making the actual selection of actions in such a setting more straightforward. However, in order to decide which action plan to follow, first a group agreement on which robot should perform a specific task is required.

This autonomous distribution of tasks among the robots is known as the MRTA problem. In contrast to the posterior action selection, yet with the availability of a communication network it may remain very hard or even impossible to find an optimal solution to the MRTA problem, at least within a reasonable amount of time. This is primarily due to the resulting combinatorial complexity and the given environmental uncertainty. This requires approximative solutions and situation-aware decisions in order to achieve an efficient, but also robust system operation in real-world settings. The arising complexity depends on the requirements to solve the tasks and the respective capabilities of the robots. In [37], the classification of MRTA problems, already described in Section 2.3 (p. 24), is proposed based on the characteristics of robots and tasks and whether the available information allows foresighted planning. For example, tasks may require a tight cooperation to be feasible, such as carrying a table together. In another example it needs to be taken into account that some robots might be better qualified for certain tasks than others.

In this chapter, problems of the ST-MR-IA class are considered, i.e. the instantaneous assignment (IA) of tasks that possibly require a tight cooperation amongst multiple robots (MR) to a set of robots, where each robot is only capable of performing a single task (ST) at the same time. In the literature, the assignment of MR-tasks is also known as coalition formation. Although numerous approaches are able to form coalitions (teams) of robots, e.g. [14, 36, 54, 56, 91, 103, 122], so far only few [91, 122] describe procedures that are able to find the best solution among a set of alternatives.

In line with the latter, the MuRoCo framework is presented, which adopts a market-based approach to determine the optimal coalition for the tight cooperation amongst multiple heterogeneous robots. Thereby, MuRoCo provides an optimal subtask assignment which is so far only considered in [91]. The main contribution is a two-step optimization, which leads to a reduced time complexity from previously $O(|\mathcal{R}|!)$ [91] to $O(2^{|\mathcal{R}|})$ for the MR-task assignment, where $|\mathcal{R}|$ is the number of robots. MuRoCo further integrates pruning strategies that aim at the elimination of infeasible solutions in order to reduce the space of possible solutions and thus to lower the time complexity. Also of importance are the integrated means for detection, prevention and handling of failures, which take

disturbances and the perception uncertainty into account to ensure a robust operation in dynamic environments. The framework is validated in a cooperative service scenario, showing its suitability to complex applications, its robustness to environmental changes and its ability to recover from failures.

The remainder of the chapter starts with an overview of the related literature in the field of MRTA. In Section 3.3, a formulation of the problem in focus is given. Section 3.4 presents the MuRoCo framework, followed by its theoretical analysis in Section 3.5 and experimental results from its practical application to a heterogeneous MRS in Section 3.6.

3.2. Related Work

Originating from the multi-agent domain, the multi-robot task allocation has been extensively studied during the last decade making it an inherent part of robotics. In [37], an overview of the types of MRTA problems and its related complexity classes is given. The simplest type (ST-SR-IA) is an instance of the optimal assignment problem (OAP) and solvable in polynomial time. In more complex settings, the task allocation problem is known to be *NP*-hard [37], that is, it scales badly in terms of finding optimal solutions for larger problems. Therefore, approximate solutions are required to solve these problems in practical applications what led to a vast amount of respective MRTA approaches.

General Overview on Methods for Cooperative MRSs: A comparably large field relates to behavior-based approaches, where robots switch among a finite set of behaviors depending on the environmental condition. One representative is *ALLIANCE* [89], which aims at increasing fault-tolerance in multi-robot systems through mutual observation and respective behavior activation by the robots. A major advantage of *ALLIANCE* is its independence of communication, making it applicable for fully distributed systems and thus very robust. Another behavior-based system is presented in [30], where the focus is primarily on the behavior design in order to achieve a coordinated motion of a group of robots. Nevertheless, behavior-based approaches are closely linked to the bio-inspired ones, at least with respect to their suitability for more complex systems. Behavior-based systems mostly use a distributed decision making, providing the benefit that communication and/or the system complexity can be kept low. Therefore behavior-based systems are used in swarm robotics, for exploration, or simple clean-up scenarios. However, for more complex tasks that possibly require a tight coordination between multiple heterogeneous robots, they are usually less suited as the distributed decision making often yields highly suboptimal solutions.

Up to now, in most cases where MRTA approaches are applied to real robots, the robotic capabilities are rather limited. First approaches mainly focus on single-task robots and single-robot tasks (ST-SR-IA). One example is the *broadcast of local eligibility* (*BLE*) [125] approach, where the robot with the highest utility blocks the other robots from executing the respective task. However, the constant progress in robotics leads to robots with increased capabilities. Modern robots, for example [4, 57, 126], are equipped with multiple sensors exploiting different modalities, such as cameras, lasers, and/or ultrasonic sensors, and possess various capabilities, such as people tracking, object recognition, manipulation

with multiple degrees of freedom and/or varying end-effectors. These enable such robots to perform increasingly sophisticated tasks and thus allow for more complex MRS scenarios. A recent example is cooperative table cleaning of three humanoid robots shown in [73]. One consequence of this increasing variety of performable tasks and robotic capabilities is that the respective task assignment becomes less trivial.

In [29], the problem is addressed by using tokens to assign tasks to the robots. Each task is represented by a token and only the robot having that token can execute the corresponding task. However, tasks are allocated asynchronously and in a distributed manner, thus that optimality cannot be guaranteed even for simpler problems. Another possibility is to model task selection as a game [10, 84]. Refer to [107] for an elaborate introduction into cooperative and non-cooperative games for multi-agent systems. Although game theoretical approaches seem promising, they assume the robots to be self-interested. This contravenes with the type of MRS considered here, which aim to maximize the joint group performance according to (2.7).

A further class of methods models the team decision problem as a multi-agent partially observable Markov decision process (MA-POMDPs), e.g. [95, 105]. The primary focus is to incorporate a model of the environmental uncertainty into the decision process, in order to infer about the current environmental state and the probable consequences of future team actions. MA-POMDPs provide a powerful tool to take the given uncertainty into account, but the downside is the related time complexity. Time grows exponentially with respect to the number of robots, as well as, the dimensions of the environment model. In other words, it may already be hard to solve even for a system of only two robots. In theory, given free communication and full observability which are assumed in the current chapter, the complexity is reduced to polynomial time [105]. But under these assumptions the actual strengths of MA-POMDPs, that is the modeling of the unobservable part of the environment, become mostly needless.

Market-Based Approaches: The most important class of methods which solve the MRTA problem in MRSs with communication, at least when considering the amount of respective literature, are market-based approaches. These follow the concept of economical markets: A set of robots bids for the assignment of tasks, which are then assigned through an auction such that a common group objective is optimized. An exhaustive survey is given in [55], where various auction types are described followed by a comparison of their respective time and communication complexities. Literature searches indicate that market-based approaches currently dominate the field. In [48], they are identified as belonging to the most popular organization structures in the related literature, which is an indicator for their suitability in many practical applications. Their success is primarily attributed to the good tradeoff between a fully decentralized system design, which guarantees only suboptimal solutions, and a fully centralized design, which suffers complexity problems and a single point of failure.

In [25], a free market architecture is introduced for the distributed control of a MRS. The concept is to define revenues and costs for performing specific tasks. The idea is that robots can profit from exchanging goods or services. By using a self-interested robot model, robots negotiate to minimize their costs and maximize their profits. Thus these

systems do not correspond to the type of cooperative MRS considered here. In [25], a robot has an incentive to bid when it leads to more profit. Another characteristic of the free market approach is the assumption that the overall system costs and revenues result from the sum of the costs and profits of the single robots. Accordingly, self-serving robots do not only maximize their own profit but also increase the overall profit. Although it cannot be guaranteed that the optimal team solution is reached since the robots do not act cooperatively in a systematic manner.

In [36], the market-based system *MURDOCH* is presented, where tasks are allocated to the most profitable robot in a greedy fashion. To keep the system simple and simultaneously capable of handling a huge diversity of tasks, *MURDOCH* is based on a hierarchical task structure. A task tree is proposed, where each node is a task which in turn can obtain other subtasks. A parent task is responsible for the allocation of resources required by its child tasks. Using a broadcast-based publish/subscribe communication enables the management of negotiations in systems with multiple participants and dynamic robot compositions. By addressing tasks to specific resources, only robots capable of these resources will subscribe to and receive their respective tasks.

Coalition Formation: While these approaches focus on ST-SR-IA problems, the MRTA problem becomes more challenging when tasks require joint execution of multiple robots. This may occur either loosely coupled, for example a fork lifter that loads material on the cargo area of a transport robot, or tightly coupled, such as two robots jointly carrying an object. Such multi-robot (MR) tasks, and accordingly instances of the ST-MR-IA problem, are especially in MRS that are composed of robots with increased complexity and functionality of practical relevance.

Among the first ST-MR-IA approaches is *Hoplites* [56], where self-interested robots are provided with an active and a passive coordination strategy. These enable the robots to initiate coalition requests in cases where certain levels of profitability may be achieved. This yields an emergent coalition formation, but does not integrate an explicit search for the global optimal solution.

In [121], the framework RACHNA for multi robot coalition formation is described, where a specific service agent exists for each kind of service a robot can provide. This service agent is responsible for communication with all robots that are able to provide the respective service. In RACHNA, the usual bidding process is reversed, in that a task is bidding for the assignment of robots. This occurs through negotiation of a task agent with the responsible service agents. In [121] also the task switching of the robots is considered. When a robot receives a better offer for another task, it requests a new offer from the service agents and thereupon decides for the task with highest utility. In order to avoid an unreliable overall system because of frequent task switching, a heuristic penalty is charged for revoking an uncompleted task. Fault detection is not explicitly considered, however in case of a sensor failure for example, the respective robot is simply deleted from the list of the corresponding service agents.

A modified version of Shehory and Kraus' algorithm [106], which provides the transfer from the multi-agent to the multi-robot domain, is presented in [122]. Each robot is described by a capability vector. In the original version, the capabilities are treated

interchangeably within coalitions, that is the coalition capabilities are the sum of the single robots' capabilities. However, due to locational capability constraints, sensors or actuators cannot be easily exchanged between the robots, this is not possible in MRSs. Therefore, [122] uses a constraint graph to verify that potential coalitions meet all capability constraints. Some additional constraints are used, such as the restriction of the maximum coalition size and the demand for disjoint coalitions. This means robots are not allowed to be in multiple coalitions as they are assumed to be only capable to perform one task at the same time (ST). The complexity to choose the largest valued coalition and assign the task is the same as for the multi-agent approach [106], and is given with $O(|\mathcal{R}|^k)$ for the distributed computation and with $O(|\mathcal{R}|^{k+1})$ for the centralized case, where k is the maximal coalition size [122].

An approach combining the market concept with a multi-robot planning framework is *ASyMTRe* [91] and its distributed version *ASyMTRe-D*. Complex task plans are generated by adequately interconnecting low-level actions. The execution responsibilities of the corresponding actions are assigned to the robots based on their capabilities. *ASyMTRe* is capable of allocating tasks that require the tight coordination of a robot team, that is coalition formation. However, [91] focuses on the reasoning capabilities of *ASyMTRe* to autonomously configure the interconnections of low-level modules – so-called schemas – to generate coalescent task solutions. Although the complexity is reduced by removing all duplicate solutions from the original configuration space and allowing only certain schema transitions, the time complexity of *ASyMTRe* is still $O(|\mathcal{R}|!)$ resulting from an exhaustive search. *ASyMTRe* provides anytime characteristics by sequentially trying different robot orderings, through finding a first suboptimal solution in quadratic time which is then continuously improved, as long as time allows. *ASyMTRe* so far also has been the only approach that explicitly handles the subtask assignments among the coalition members. Furthermore heuristics are used to reduce the size of the solution space, such as timeouts, an upper bound of the coalition size or orderings of the robots by increasing capabilities or the number of potential coalition partners.

In [14], a modeling of robotic resources and a leader-follower coalition formation is proposed. Instead of searching for the global optimum, the approach aims at a fast determination of a feasible, but most probably suboptimal, solution in a greedy depth-first manner. To focus the search, various heuristics are proposed, such as assigning the most capable robots as leaders that are responsible to form the coalitions, or to include only robots within a specified vicinity into the coalitions.

This overview shows that various approaches to the MRTA and also to the coalition formation problem can be found in literature, but only few approaches aim at finding the optimal solution for the ST-MR-IA problem and for those the respective complexity is still huge. Note that "optimal" in this context relates to the assignment of a single MR-task. For the optimal simultaneous assignment of multiple MR-tasks, so far no efficient solution is known. Moreover, although means for fault tolerance and failure detection are incorporated in several approaches, for example [25, 36, 59, 122], the consideration of environmental uncertainty during decision making is still insufficient. MuRoCo yield at combining both aspects into a highly integrated framework for complex heterogeneous robots, in order to robustly handle instances of the ST-MR-IA problem.

3.3. Problem Definition

Before a formal definition of the MRTA problem is given, the underlying assumptions are summarized:

(i) Robot-robot communication:
A core assumption of market-based approaches is that all robots are able to communicate with all other robots at anytime. So systems are considered in which communication is cheap, reliable and always available.

(ii) Task independence:
An essential assumption of most MRTA approaches is that of task independence [37], that is all tasks issued from higher layers can be performed independently without considering causal relations.

(iii) Instantaneous assignment:
As mentioned in Section 2.1 (p. 17), it is assumed that the planning layer is taking care of any timing constraints so that only tasks that need to be instantaneously assigned (IA) are issued to the action selection layer.

(iv) Time delay generates cost:
A further assumption is that time delay generates cost, that is a delayed completion of a task has to lead to higher cost. If this was not the case, the best solution for a task assignment would simply be that each task is executed by the robot or coalition with the lowest cost. In an extreme case, all tasks might be sequentially performed by the same robot. In other words, a reasonable MRTA needs to benefit from parallelization.

For these types of MRSs, a formulation of the MRTA problem is given as follows. According to Section 2.2 (p. 21) a MRS is composed of a set $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ of n robots r. Furthermore there is a set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ of m unaccomplished tasks τ, which need to be assigned to the robots.

Definition 3.1 *Let an assignment $\psi : \mathcal{R} \to \mathcal{T}$ be a mapping of robots to tasks, i.e. $\tau_r = \psi(r)$ is the task $\tau_r \in \mathcal{T}$ assigned to robot $r \in \mathcal{R}$. Let further be $|\mathcal{R}| = |\mathcal{T}|$, then $\psi^{-1} : \mathcal{T} \to \mathcal{R}$ is the inverse mapping, where $r = \psi^{-1}(\tau_r)$.*

The constraint $|\mathcal{R}| = |\mathcal{T}|$ can be w.l.o.g satisfied by extending the smaller set with placeholders ι. Any mapping $\iota = \psi(r)$ or $\iota = \psi^{-1}(\tau)$ corresponds to an unassigned robot or task respectively.

Definition 3.2 *An assignment ψ, for which holds $\psi(r) \neq \iota$, $\forall r \in \mathcal{R}$, and further $\psi^{-1}(\tau) \neq \iota$, $\forall \tau \in \mathcal{T}$, i.e. no task and no robot remains unassigned, is said to be a perfect assignment.*

A valid policy ϖ returns the respective action $a_r = \varpi(r, \boldsymbol{s}, \psi, \mathcal{P}_\mathcal{A})$ to be executed by r, according to the current state $\boldsymbol{s} \in \mathcal{S}$, the mapping ψ and an action plan $p_\mathcal{A}(\tau_r) \in \mathcal{P}_\mathcal{A}$ for τ_r, where $a_r \in p_\mathcal{A}(\tau_r)$. $\mathcal{S}$ is the set of all possible states $\boldsymbol{s}$, which provide a symbolic

representation of the status of all robots and the environment. Recalling (2.4) (p. 21), an action plan

$$p_{\mathcal{A}}(\tau_j) = \langle a_{j1}, a_{j2}, \ldots, a_{jl} \rangle$$

refers to a specific sequence of actions to be executed in order to complete a task τ_j. $\mathcal{P}_{\mathcal{A}}$ is the set of all known action plans. In case of the latter, the reward

$$\varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) = u(p_{\mathcal{A}}(\tau_j)) - c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \in \mathbb{R}$$

is received, which is, according to (2.6) (p. 22), given by the difference between the predetermined utility $u(p_{\mathcal{A}}(\tau_j)) \in \mathbb{R}_0^+$ and the cost $c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \in \mathbb{R}_0^+$ arising during the execution of $p_{\mathcal{A}}(\tau_j)$. In this respect, the performance of a MRS is according to (2.7) (p. 23) measured by the ratio of the obtained reward relative to the obtainable reward:

$$q = \frac{1}{\beta_u} \sum_{j=1}^{m_c} \varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \leq 1, \quad \text{where } \beta_u = \sum_{k=1}^{m_a} u(\tau_k).$$

m_c is the number of completed tasks and m_a the number of all tasks issued to the system with $m_c \leq m_a$ and $\bigcup \tau_j \subseteq \bigcup \tau_k$. Assuming that the reward $\varrho(p_{\mathcal{A}}(\tau_r))$ obtained for the completion of $p_{\mathcal{A}}(\tau_r)$ is independent of any other simultaneously executed task, then the determination of a solution that satisfies (2.8) can be split into two steps:

(i) For each $r \in \mathcal{R}$ and each $\tau \in \mathcal{T}$ find the optimal plan $p^*_{\mathcal{A},r}(\tau)$ and add it to the set

$$\mathcal{P}^*_{\mathcal{A}} \overset{\cup}{\leftarrow} p^*_{\mathcal{A},r}(\tau) = \underset{p_{\mathcal{A}}(\tau) \in \mathcal{P}_{\mathcal{A}}}{\arg\max}\ \varrho(r, p_{\mathcal{A}}(\tau)), \quad \forall r \in \mathcal{R},\ \forall \tau \in \mathcal{T}. \tag{3.1}$$

(ii) Find the optimal assignment

$$\psi^* = \underset{\psi}{\arg\max}\ q(\psi, \mathcal{P}^*_{\mathcal{A}}) = \underset{\psi}{\arg\max}\ \left(\frac{1}{\beta_u} \sum_{\tau \in \mathcal{T}} \varrho(r, p^*_{\mathcal{A},r}(\tau)) \right), \tag{3.2}$$

where $r = \psi^{-1}(\tau)$. The time complexity to solve (3.1) in a centralized manner is $O(|\mathcal{R}||\mathcal{P}_{\mathcal{A}}(\mathcal{T})|)$, where $\mathcal{P}_{\mathcal{A}}(\mathcal{T})$ is the set of all plans for all $\tau \in \mathcal{T}$. Note that an action or task plan specifies only one possible way to execute a task meaning that there may be multiple alternative actions and/or task plans for the same task. Under the assumption that $\mathcal{P}_{\mathcal{A}}(\mathcal{T})$ is given $\mathcal{P}^*_{\mathcal{A}}$ is determinable in linear time.

Thus, the actual challenge is to find the optimal assignment ψ^* within a reasonable amount of time. This becomes even harder when a task τ_j is, additionally or alternatively to action plans, see (2.4), also decomposable into task plans.

Definition 3.3 *A task plan*

$$p_{\mathcal{T}}(\tau_j) := \langle \tau_{j1}, \tau_{j2}, \ldots, \tau_{jk} \rangle,$$

defines a decomposition of τ_j into a tuple of other tasks, where the $\tau_{jl}, 1 \leq l \leq k$, are referred to as subtasks of τ_j. $\mathcal{P}_{\mathcal{T}}$ is the set of all task plans. Any task τ for that $\exists\ p_{\mathcal{T}}(\tau) \in \mathcal{P}_{\mathcal{T}}$, s.t.

$|p_{\mathcal{T}}(\tau)| \geq 2$ is referred to as MR-task, since its subtasks are executable by different robots. All other tasks are referred to as SR-tasks.

Since in the case of the action plans, the task plans are assumed to be pre-computed by the planning layer and derivable from a common knowledge base. Note that an action or task plan specifies only one possible way to execute a task meaning that there may be multiple alternative action plans and/or task plans for the same task. In case a plan relates to a MR-task, a coalition of robots is required.

Definition 3.4 *Given a set of robots $\mathcal{R}$, then any subset $z \subseteq \mathcal{R}$ that forms a team in order to jointly execute a MR-task is referred to as coalition. $\mathcal{Z}$ is the set of all possible coalitions within a set of robots $\mathcal{R}$.*

Thus, MR-tasks complicate the search for the optimal assignment since additionally all possible coalitions need to be examined to find the optimal coalition.

3.4. The MuRoCo Framework

As stated in Section 3.1, handling the multi-robot task allocation (MRTA) problem can essentially improve the cooperation efficiency in a MRS, but respective solutions are often not trivial. MuRoCo is a framework for handling the MRTA problem for cooperative MRSs, which are composed of heterogeneous robots that are capable of full communication. In other words, the robots have a common group objective, possess different capabilities, and every robot is able to communicate with any other robot within the system.

As already explained in Section 2.1 (p. 17), here only the instantaneous assignment (IA) of tasks is considered since the planning for future allocations is assumed to be handled by the higher level planning. As the planning layer may provide multiple alternative plans for the execution of a specific task, the MRTA is also responsible to select the most suitable plan according to the current situation. The tasks to be assigned may require either a single robot (SR) or a tight cooperation of multiple robots (MR) to be completed.

The general objective of a MRTA framework is to exploit the advantages of MRSs mentioned in Section 1.3.2 (p. 8), for example to achieve a performance improvement through parallelization or by increasing robustness. In order to meet these demands, MuRoCo implements a market-based approach for which an overview is given in Section 3.4.1. Section 3.4.2 describes how the heterogeneity of the system is taken into account, followed by the explanation of the SR-task assignment procedure in Section 3.4.3. Section 3.4.4 describes the assignment of MR-tasks to coalitions. In order to increase the robustness and performance of the system, means for the prevention, detection and handling of failures are presented in Section 3.4.5

3.4.1. General Approach

Market-based approaches provide a weakly centralized compromise by distributing the computational workload to keep the complexity scalable, and by making use of the system redundancy to increase fault tolerance since the system may still be able to operate in case

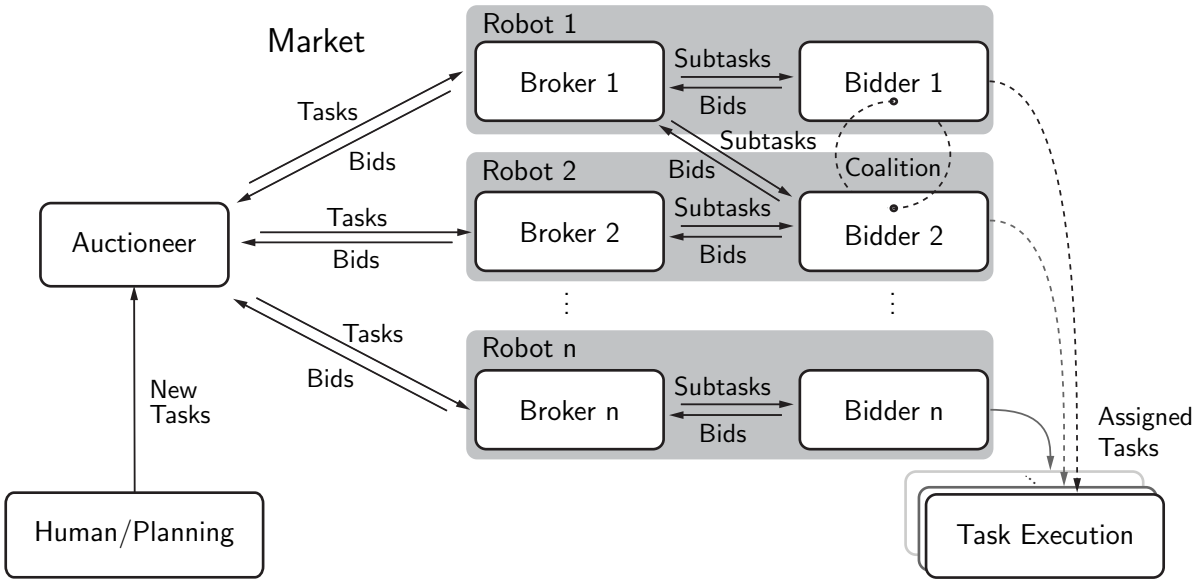


Fig. 3.1.: Illustration of the market-based task allocation: The auctioneer forwards perceived tasks to the brokers, which request respective bids from their bidders. Then these bids are returned to the auctioneer, which assigns the task to the most efficient broker or coalition (team). In the example case here, Robots 1 and 2 form a coalition for one task while Robot n executes a further task alone.

a single component breaks down. During operation, an auctioneer retrieves the new tasks from its commanding modules, for example from its human-machine interfaces or higher level planning modules. However, the auctioneer that retrieved the task does not necessarily need to be the one auctioning the task. In the following a single active auctioneer is assumed, i.e. all other auctioneers forward their tasks to the active one. Alternative strategies are discussed in Section 3.4.5.

In MuRoCo, each robot is capable to take over the auctioneer role while simultaneously operating as broker and bidder. The bidder of robot r provides the bid for a task τ in form of the reward estimate

$$\hat{\varrho}(r,\tau) = \max_{p_{\mathcal{A}}(\tau)\in\mathcal{P}_{\mathcal{A}}} \hat{\varrho}(r, p_{\mathcal{A}}(\tau)), \tag{3.3}$$

where $\hat{\varrho}(r, p_{\mathcal{A}}(\tau))$ is the estimated reward that robot r expects to receive after successful completion of $p_{\mathcal{A}}(\tau)$. Accordingly, (3.1) is solved in a distributed manner by the bidders of all $r \in \mathcal{R}$ and thus its time complexity reduces to $O(|\mathcal{P}_{\mathcal{A}}(\mathcal{T})|)$. The bids are forwarded to the brokers that negotiate with the auctioneer as shown in Fig. 3.1. Thereby a broker acts as an intermediary that bids for a single robot and/or a specific set of coalitions of robots. For each new task, there is exactly one auctioneer that is responsible for the assignment and also for the subsequent surveillance of the task. This means it also has to take care that the task is successfully executed or reallocated in case of a failure. The allocation is optimized by the active auctioneer according to (3.2), that is by maximizing the relative obtainable reward. This corresponds to the MiniAve group objective described in [117], where the formulation is given for the minimization of costs.

The number $|\mathcal{R}|$ of robots within $\mathcal{R}$ does not necessarily remain constant during system

operation. Some robots may be unavailable due to recharging, malfunctioning or because of shut-off by the human operator. In order to know when to start the search for the assignment, the auctioneer needs to be aware of how many bids are expected. In other words how many robots, i.e. brokers, are currently available. This knowledge is retrieved from alive messages that all robots broadcast periodically. In addition, the alive messages contain information about the capabilities of each robot, which define whether a robot is capable of performing a specific task or not as described in the next section.

3.4.2. Considering Heterogeneity

The assumption that the robots in a MRS may be heterogeneous needs to be explicitly taken into account within a task allocation procedure, since the robots may not be equally capable of performing the different tasks. In general, these capabilities may be time-varying as hardware components may fail or robots may improve their skills during the course of learning processes. However, while the task costs may rapidly change in a dynamic environment, the capabilities of the robots are commonly subject to a far slower variation. This condition can be effectively used in the early phase of a task allocation procedure, to prune infeasible constellations from the solution space as described below. This reduces the complexity of the problem to solve, as the pruned constellations are simply neglected during the subsequent assignment procedures.

In this respect, to formulate the functionality of the robots, those capabilities are defined as follows:

Definition 3.5 *Let $\kappa \in [0,1]$ be a capability and $\mathcal{K}$ the set of all capabilities known to the system. Then holds*

$$\kappa := \frac{n_c(\kappa)}{n_c(\kappa) + n_f(\kappa)} \in [0,1]\,,$$

where $n_c(\kappa) \in \mathbb{N}$ is the number of successfully completed actions for which κ was used and $n_f(\kappa) \in \mathbb{N}$ is the number of actions, which failed due to a malfunction of κ.

Note that actions which failed due to a fault of another capability $\kappa' \neq \kappa$ are excluded as they allow no clear conclusion on the reliability of κ. In principle, a capability may relate likewise to a hardware resource and/or to a software component, even though mostly the hardware resources are the major limiting factor in the practical application. To determine the functionality of a robot, let each robot $r \in \mathcal{R}$ be equipped with a set

$$\mathcal{K}_r = \{\kappa_{r1}, \kappa_{r2}, \ldots, \kappa_{rl}\} \subseteq \mathcal{K}. \tag{3.4}$$

The actual value of κ is a measure of its reliability, that is the probability that no failure will occur during its use, which is applied later in Section 3.4.5. For the following considerations, a binary interpretation, in the sense of existence or non-existence, is sufficient.

Definition 3.6 *For a capability $\kappa \in [0,1]$ and a robot r with capability set $\mathcal{K}_r$ holds:*

$$\kappa \in \mathcal{K}_r := \begin{cases} \text{true}, & \text{if } \kappa(r) > 0 \\ \text{false}, & \text{else.} \end{cases}$$

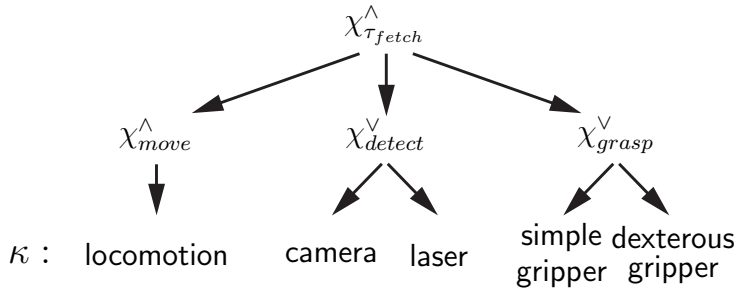


Fig. 3.2.: Exemplary capability constraint for the task τ_{fetch} to fetch a cup of coffee from a table.

In other words, any non-zero value of $\kappa(r)$ is interpreted as r being equipped with $\kappa(r)$.

In this respect, given the sets of capabilities for two robots, their mutual heterogeneity is defined by the ratio of redundant capabilities.

Definition 3.7 *The heterogeneity of two robots r_i and r_j with capability sets $\mathcal{K}_{r_i}$ and $\mathcal{K}_{r_j}$ is given by*

$$h(r_i, r_j) := \frac{|\mathcal{K}_{r_i} \Delta \mathcal{K}_{r_j}|}{|\mathcal{K}_{r_i}| + |\mathcal{K}_{r_j}|} \in [0, 1]\,,$$

where

$$\mathcal{K}_{r_i} \Delta \mathcal{K}_{r_j} = (\mathcal{K}_{r_i} \cup \mathcal{K}_{r_j}) \setminus (\mathcal{K}_{r_i} \cap \mathcal{K}_{r_j})$$

is the symmetric difference of $\mathcal{K}_{r_i}$ and $\mathcal{K}_{r_j}$ which is defined as the union of both sets excluding their intersection.

Consequently, for a set of robots $\mathcal{R}$ the heterogeneity is determined by

$$h(\mathcal{R}) = \frac{2}{|\mathcal{R}|(|\mathcal{R}| - 1)} \sum_{i=1}^{|\mathcal{R}|-1} \sum_{j=i+1}^{|\mathcal{R}|} \frac{|\mathcal{K}_{r_i} \Delta \mathcal{K}_{r_j}|}{|\mathcal{K}_{r_i}| + |\mathcal{K}_{r_j}|} \in [0, 1]\,. \tag{3.5}$$

For a homogeneous MRS holds $h = 0$ and for a MRS without any redundant capability $h = 1$. The heterogeneity provides a mean to measure the degree of functional variety within the system, where the capabilities express the available functionalities of the robots.

Instead, when considering functionalities with respect to actions or tasks, these rather present constraints that have to be satisfied in order to perform the respective action or task. Consider for example a task τ_{fetch}, that is to fetch a cup of coffee from a table. In order to execute τ_{fetch} a robot needs to possess various capabilities as illustrated in Fig. 3.2. First of all it needs to be capable of locomotion in order to drive to the table and return to its original location. Additionally, it requires some capability, such as a laser- or camera-based detection, to locate the exact position of the cup, and of course some further capability to grasp the cup. As the cup is assumed to be easy to grasp, a simple gripper is sufficient. However, a dexterous one would work as well.

Apparently, some of these capabilities are alternatives in the sense that they provide a redundant functionality for this specific task. So in the example of τ_{fetch} it is sufficient if a robot is equipped with either a laser-based or a camera-based detection. Similarly it only needs either a simple or a dexterous gripper. Nevertheless, a certain combination of

capabilities is mandatory. Thus τ_{fetch} requires at least one locomotion, detection and grasp capability. In order to enable a fast check on whether a robot is capable of performing a task, a formalism of constraints is defined that allows a hierarchical and nested description of mandatory and alternative capability constraints.

Definition 3.8 *Let $\chi(\mathcal{K}_y)$ be a constraint on the capability set $\mathcal{K}_y$, where $\chi \in \mathbb{B}$ is a boolean variable with $\mathbb{B} = \{\text{true}, \text{false}\}$. Then $\mathcal{K}_x \prec \chi(\mathcal{K}_y)$ is defined as set $\mathcal{K}_x$ satisfying the constraint $\chi(\mathcal{K}_y)$ on set $\mathcal{K}_y$. Respectively, $\mathcal{K}_x \not\prec \chi(\mathcal{K}_y)$ is defined as set $\mathcal{K}_x$ not satisfying $\chi(\mathcal{K}_y)$ on set $\mathcal{K}_y$.*

Definition 3.9 *Let $\chi^{\wedge}(\mathcal{K}_y)$ be a conjunctive constraint on the capability set $\mathcal{K}_y$. Then, a set $\mathcal{K}_x$ is defined to satisfy $\chi^{\wedge}(\mathcal{K}_y)$ as follows:*

$$\mathcal{K}_x \prec \chi^{\wedge}(\mathcal{K}_y) := \begin{cases} \text{true}, & \text{if } \mathcal{K}_y \subseteq \mathcal{K}_x \\ \text{false}, & \text{else.} \end{cases}$$

Definition 3.10 *Let further be $\chi^{\vee}(\mathcal{K}_y)$ a disjunctive constraint on the capability set $\mathcal{K}_y$. Then a set $\mathcal{K}_x$ is defined to satisfy $\chi^{\vee}(\mathcal{K}_y)$ as follows:*

$$\mathcal{K}_x \prec \chi^{\vee}(\mathcal{K}_y) := \begin{cases} \text{true}, & \text{if } \mathcal{K}_y \cap \mathcal{K}_x \neq \emptyset \\ \text{false}, & \text{else.} \end{cases}$$

A conjunctive constraint $\chi^{\wedge}(\mathcal{K}_m)$ demands the availability of all $\kappa \in \mathcal{K}_m$, where $\mathcal{K}_m$ presents a set of mandatory capabilities. Instead, a disjunctive constraint $\chi^{\vee}(\mathcal{K}_a)$ demands at least one $\kappa \in \mathcal{K}_a$. Accordingly, $\mathcal{K}_a$ presents a set of alternative capabilities.

This enables the determination of whether a robot is capable of performing an action.

Definition 3.11 *Let any action $a \in \mathcal{A}$ be bound to a constraint χ_a. Then r is said to be capable of performing a if*

$$\mathcal{K}_r \prec \chi_a.$$

Thereby, χ_a may be either a conjunctive, disjunctive or any functional combination of those. For the constraints χ apply the operations and laws of Boolean algebra. Thus, any valid boolean function $f : \mathbb{B}^n \to \mathbb{B}$ of constraints, where $n \in \mathbb{N}$ is the number of arguments of f, is itself a valid constraint. Accordingly, given the constraints χ_a for any a, the constraint $\chi_{p_{\mathcal{A}}(\tau)}$ on a task τ to be executed according to an action plan $p_{\mathcal{A}}(\tau)$ is derivable. Since a robot needs to be capable to execute all $a \in p_{\mathcal{A}}(\tau)$ in order to complete τ, $\chi_{p_{\mathcal{A}}(\tau)}$ is determined by the conjunction of constraints χ_a of all $a \in p_{\mathcal{A}}(\tau)$. Consequently, a robot r is capable of performing $p_{\mathcal{A}}(\tau)$ if

$$\mathcal{K}_r \prec \chi_{p_{\mathcal{A}}(\tau)} = \bigwedge_{\forall a \in p_{\mathcal{A}}(\tau)} \chi_a. \tag{3.6}$$

Generally r is said to be capable to perform τ if $\exists p_{\mathcal{A}}(\tau) \in \mathcal{P}_{\mathcal{T}}$ for which $\mathcal{K}_r \prec \chi_{p_{\mathcal{A}}(\tau)}$. This description enables the logical combination of various constraints to more complex constraints. For example, the constraint $\chi^{\wedge}(\tau_{fetch})$ for τ_{fetch} is w.r.t. Fig. 3.2 given by

$$\chi^{\wedge}(\tau_{fetch}) = f(\chi^{\wedge}_{move}, \chi^{\vee}_{detect}, \chi^{\vee}_{grasp}) = \chi^{\wedge}_{move} \wedge \chi^{\vee}_{detect} \wedge \chi^{\vee}_{grasp}$$

and is satisfied by r according to

$$\mathcal{K}_r \prec \chi^{\wedge}(\tau_{fetch}) = \begin{cases} \text{true}, & \text{if } (\mathcal{K}_r \prec \chi^{\wedge}_{move}) \wedge (\mathcal{K}_r \prec \chi^{\vee}_{detect}) \wedge (\mathcal{K}_r \prec \chi^{\vee}_{grasp}) \\ \text{false}, & \text{else.} \end{cases}$$

With (3.6), the capability check required for SR-tasks is given. However, for MR-tasks, not only the capabilities of individual robots, but the combined capabilities of coalitions have to be considered.

Definition 3.12 *The approximated set of capable coalitions*

$$\mathcal{Z}_{cap}(p_{\mathcal{T}}(\tau)) := \left\{ z \in \mathcal{Z} \,\middle|\, |z| = |p_{\mathcal{T}}(\tau)| \wedge \bigcup_{\forall r \in z} \mathcal{K}_r \prec \chi_{p_{\mathcal{T}}(\tau)} = \bigwedge_{\forall \tau_j \in p_{\mathcal{T}}(\tau)} \chi_{\tau_j} \right\}$$

is the subset of all coalitions $z \in \mathcal{Z}$, for which the union of capabilities satisfies the constraint on the task plan $p_{\mathcal{T}}(\tau)$.

The examination of the robotic capabilities by (3.6) and Definition 3.12 facilitates an early determination of infeasible solutions and thereby enables the reduction of the computational complexity during the assignment procedure.

3.4.3. Assignment of Single-Robot Tasks

As a consequence of Definition 3.3, tasks are distinguishable into SR-tasks and MR-tasks. Since SR-tasks are executable by a single robot and MR-tasks by a coalition of robots task-specific assignment procedures are required. This section describes the optimal assignment of SR-tasks to ST-robots and thus solves according to Section 2.3 (p. 24) the ST-SR-IA assignment problem. This problem is already well-studied in literature and shown to be solvable in polynomial time. In principle, there is no need to go further into detail here, but as it is also part of the subsequently explained MR-task assignment, it is described for the reader who is not familiar with market-based approaches.

The ST-SR-IA assignment problem corresponds to a classical problem in combinatorial optimization that is commonly known as the *Maximum Weight Matching Problem in Bipartite Graphs*[1], see e.g. [62].

Definition 3.13 *Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a bipartite graph with bipartition $\mathcal{V}(\mathcal{G}) = \mathcal{R} \cup \mathcal{T}$ and edge rewards $\hat{\varrho} : \mathcal{E}(\mathcal{G}) \to \mathbb{R}$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}(\mathcal{G})$ the set of edges of $\mathcal{G}$. Each edge $e_{r\tau} \in \mathcal{E}$ is a weighted link between $r \in \mathcal{R}$ and $\tau \in \mathcal{T}$ with weight $\hat{\varrho}(r, \tau)$. Let $|\mathcal{R}| = |\mathcal{T}|$, then a maximum weight matching is a subset of edges $\mathcal{M} \subseteq \mathcal{E}$ that satisfies*

$$\max \left\{ \frac{1}{\beta_u} \sum_{e \in \mathcal{M}} \hat{\varrho}(e) \,\middle|\, |\mathcal{M}(v)| = 1, \forall v \in \mathcal{V}(\mathcal{G}) \right\},$$

[1] Note that this is equivalent to the *Minimum Weight Perfect Matching Problem in Bipartite Graphs* used for the minimization of costs, as positive weights can be simply transformed into negative weights by negation.

where $\mathcal{M}(v) \subseteq \mathcal{M}$ is the subset of edges that are incident on vertex v.

The constraint on the right ensures that for each robot $r \in \mathcal{R}$ there is exactly one assignment to a task $\tau \in \mathcal{T}$. The assumption $|\mathcal{R}| = |\mathcal{T}|$ can be satisfied w.l.o.g. because the asymmetric assignment problem, i.e. $|\mathcal{R}| \neq |\mathcal{T}|$, is reducible to the symmetric one [7] by equally shifting all edge rewards to $\mathbb{R}^+$ and extending the smaller set with placeholders for which all respective edge rewards are set to $\hat{\varrho}_e = 0$. The optimal matching of the asymmetric problem is then derived by discarding all edges in the symmetric solution that are incident on a placeholder. Since a solution to Definition 3.13 also satisfies (3.2), a maximum weight matching provides an optimal assignment $\psi^* : \mathcal{R} \to \mathcal{T}$.

The assignment procedure is derivable from Alg. 3.1 and 3.2, showing the update routine of the auctioneer and the broker respectively. In order to find a solution to the SR-task assignment problem in a MRS setup, firstly all bids $\hat{\varrho}(r, \tau)$, $\forall r \in \mathcal{R} \wedge \forall \tau \in \mathcal{T}$, have to be collected by the auctioneer. For this, the auctioneer retrieves all of its open tasks $\mathcal{T}_o$, which includes all previously failed or unassigned tasks in addition to new tasks. In case the set $\mathcal{T}_o$ contains no MR-task, it is announced to the broker modules (Alg. 3.1, l. 5). The latter request a bid from their corresponding bidder for each SR-task $\tau_{SR} \in \mathcal{T}_{SR} \subseteq \mathcal{T}_o$, where $\mathcal{T}_{SR}$ is the subset of SR-tasks. The handling of MR-tasks is described in Section 3.4.4. The bidder computes the bid according to (3.3), by first applying the capability check (3.6). In case $\mathcal{K}_r \not\prec \chi_{p_\mathcal{A}(\tau)}$ follows $\hat{\varrho}(r, \tau) = -\infty$, else a reward estimate $\hat{\varrho}(r, \tau)$ based on the current state, i.e. the reward robot r would currently achieve when it would perform τ, is retrieved. A possibility of how this estimation may occur is presented in Chapter 4 (p. 67). Each broker collects all bids for all announced $\tau_{SR} \in \mathcal{T}_{SR}$ and then returns these to the auctioneer.

After receiving all bids from all brokers, the auctioneer calculates the optimal assignment ψ^* w.r.t. (3.2) by deriving a solution to the *Maximum Weight Matching Problem in Bipartite Graphs* according to Definition 3.13. Therefore, various algorithms can be found in literature, such as the Hungarian method [63] for which implementations are available that find a solution in $O(n^3)$ time, where $n = \max(|\mathcal{R}|, |\mathcal{T}|)$ is the number of vertices. An alternative approach is presented in [32], where a combination of Fibonacci heaps and the Dijkstra algorithm is used, yielding $O(n\,(m + n\log(n)))$ where m is the number of edges. Note that in the most complex case, when every robot is suited for every task, holds $m = n^2$. A further alternative using a scaling of the cost values is proposed in [33]. However, the solutions are only approximate optimal and furthermore, the reward values need to be small integers. The time complexity is $O(\sqrt{n}m\log(n\varrho_{max}))$, where ϱ_{max} is the largest possible reward value. A good overview of alternative algorithms and respective worst-case complexity bounds is given in [19]. In MuRoCo, an implementation based on the Hungarian method with $O(n^3)$, is used to find the optimal assignment of SR-tasks ψ^*_{SR}. It is capable of handling asymmetric assignments and bids of $\hat{\varrho} = -\infty$. The latter can be similarly handled by setting these to zero and equally shifting all other bids, that is $\hat{\varrho} \neq -\infty$, to $\mathbb{R}^+$. Any invalid assignments are then discarded to yield the final solution ψ^*_{SR}, which is then broadcasted to all brokers.

On the broker side, a resource re-check is performed for all assignments which are received by the robot. This is supposed to ensure that the availability of the required resources is still guaranteed. The result of the resource re-check is sent back to the respective

Algorithm 3.1: Update routine of the auctioneer of robot r_a.

Input: new tasks $\mathcal{T}_n = \mathcal{T}_{SR} \cup \mathcal{T}_{MR}$, operating robots $\mathcal{R}$, status messages

```
/* Announce tasks */
```
open tasks $\mathcal{T}_o = \mathcal{T}_o \cup \mathcal{T}_n$;
if $\exists \tau_{MR} \in \mathcal{T}_o$ **then**
  select $\tau_j \in \mathcal{T}_{MR}$ according to (3.7);
end
announce $\mathcal{T}_a$ according to (3.8);

```
/* Assign tasks */
```
retrieve $bids(\mathcal{T})$ from all $r \in \mathcal{R}$ (Alg. 3.2, l. 15);
compute the optimal assignment $\psi^*(\mathcal{R}, \mathcal{T}_o)$ according to (3.2);
send ψ^* to the brokers of $\mathcal{R}$;
retrieve confirmations (Alg. 3.2, l. 25);
forall $\tau \in \mathcal{T}$ **do**
  if *confirmation(r) is positive,* $\forall r \in z = \psi^{-1}(\tau)$ **then**
    remove τ from $\mathcal{T}_o$;
    move τ from $\mathcal{T}_o$ to the set of executed tasks $\mathcal{T}_e$;
  end
end

```
/* Handle executed tasks */
```
forall *received status messages* **do**
  if $\exists status(r, \tau) = failed$ **then**
    move τ from $\mathcal{T}_e$ to $\mathcal{T}_o$;
  else if $status(r, \tau) = completed, \forall r \in z = \psi^{-1}(\tau)$ **then**
    remove τ from $\mathcal{T}_a$;
  end
end

auctioneer in order to confirm or reject the task assignment. In case of acceptance, the task is forwarded to the task execution components. From then on, the robot is bound to the task until it is either completed or a failure occurs. That the latter does not happen is the responsibility of the task execution components. These also provide respective feedback information, such as failure or completion, to the broker which forwards it to the responsible auctioneer. The feedback contains the necessary status information required for failure recovery as explained in Section 3.4.5.

3.4.4. Assignment of Multi-Robot Tasks

In the previous section, the assignment of SR-tasks is described. In the case of a MR-task that is issued to the system, an extended assignment procedure is required to solve the ST-MR-IA problem. This includes a solution to the coalition-formation problem, which has been shown in [37] to be an instance of the *Set Partitioning Problem*.

Definition 3.14 *Given a set of robots $\mathcal{R}$, a set $\mathcal{Z}$ of all possible coalitions of $\mathcal{R}$, a set $\mathcal{T}_{MR}$ of MR-tasks, and a reward function $\hat{\varrho} : \mathcal{Z} \times \mathcal{T}_{MR} \to \mathbb{R}$. Then a maximum-reward*

Algorithm 3.2: Update routine of the broker of robot r_b.

Input: new task announcements $\mathcal{T}$, operating robots $\mathcal{R}$ and those capability sets $\mathcal{K}_r, \forall r \in \mathcal{R}$, new assignments ψ, status messages

`/* Handle announcements */`

forall $\tau_j \in \mathcal{T}$ **do**
  if $\exists p_\mathcal{T}(\tau_j) \in \mathcal{P}_\mathcal{T}(\tau_j)$ *s.t.* $|p_\mathcal{T}(\tau_j)| \geq 2$ **then** `// MR-tasks`
    forall $p_\mathcal{T}(\tau_j) \in \mathcal{P}_\mathcal{T}(\tau_j)$ **do**
      derive $\mathcal{Z}_{r_b,cap}(p_\mathcal{T}(\tau_j))$;
      forall $z \in \mathcal{Z}_{r_b,cap}(p_\mathcal{T}(\tau_j))$ **do**
        $\hat{\varrho}(z, p_\mathcal{T}(\tau_j)) = \text{deriveReward}(z, p_\mathcal{T}(\tau_j))$;
      end
    end
    $z' = \arg\max_{z \in \mathcal{Z}_{r_b}} \left(\max_{p_\mathcal{T}(\tau_j) \in \mathcal{P}_\mathcal{T}(\tau_j)} \left(\hat{\varrho}\left(z, p_\mathcal{T}(\tau_j)\right)\right)\right)$;
    $bids(\mathcal{T}) \overset{\cup}{\leftarrow} \hat{\varrho}(z', \tau_j) = \max_{p_\mathcal{T}(\tau_j) \in \mathcal{P}_\mathcal{T}(\tau_j)} \left(\hat{\varrho}\left(z', p_\mathcal{T}(\tau_j)\right)\right)$;
  else `// SR-tasks`
    $bids(\mathcal{T}) \overset{\cup}{\leftarrow} \hat{\varrho}(r_b, \tau_j) = \max_{p_\mathcal{A}(\tau_j) \in \mathcal{P}_{\mathcal{A},\tau_j}} \hat{\varrho}(r_b, p_\mathcal{A}(\tau_j)) \leftarrow$ from bidder of r_b;
  end
end
forward $bids(\mathcal{T})$ to auctioneer of $\mathcal{T}$;

`/* Handle assignments */`

wait for assignment ψ^* (Alg. 3.1, l. 8);
if $\exists \tau_j = \psi^*(z_b)$ *s.t.* $r_b \in z_b$ **then** `// MR-tasks`
  retrieve subtask assignment $\tau_{jl} = \psi^*(r_b, \tau_j)$;
  request confirmation for τ_{jl} from bidder;
  forward confirmation to all $r \in z_b \setminus \{r_b\}$;
  wait for confirmations of all $r \in z_b \setminus \{r_b\}$;
else if $\exists \tau_j = \psi^*(r_b)$ **then** `// SR-tasks`
  request confirmation for τ_j from bidder;
end
forward confirmation to auctioneer;
if *all confirmations are positive* **then**
  start execution;
end

`/* Handle feedback */`

forall *received status messages* **do**
  forward $status(r_b, \tau_j)$ to auctioneer of τ_j;
  if $|z| > 1$ *where* $z = \psi^{*,-1}(\tau_j)$ **then** `// MR-tasks`
    forward $status(r_b, \tau_j)$ to all $r \in z$;
  end
  if $status(r, \tau_j) = failed$ *for* $r \neq r_b$ **then**
    stop execution of $\tau_{jl} = \psi^*(r_b, \tau_j)$;
  end
end

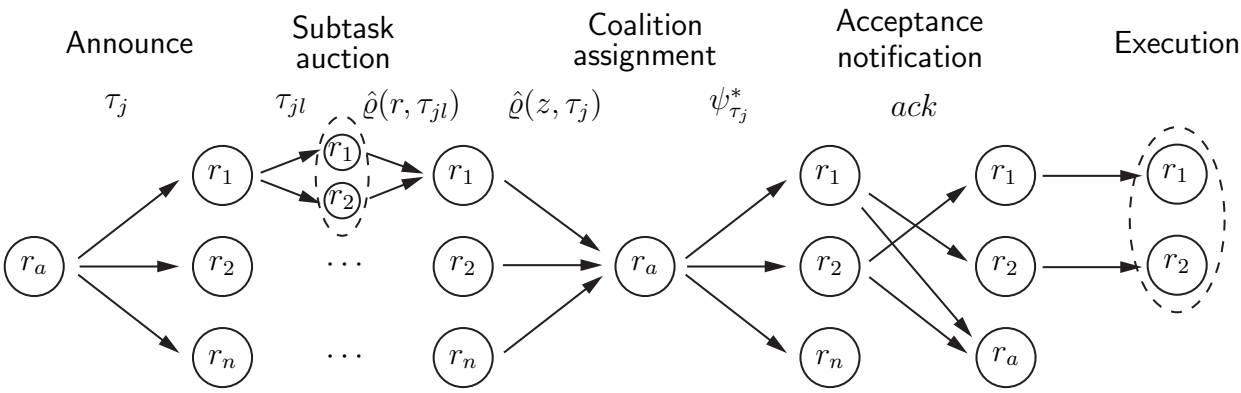


Fig. 3.3.: Illustration of the coalition formation: $\tau_j \in \mathcal{T}_{MR}$ is the MR-task to be assigned, r_a is the auctioning robot, τ_{jl} is a SR-subtask of τ_j.

partition of $\mathcal{R}$ *is a subset of coalitions* $\mathcal{Z}^* \subset \mathcal{Z}$ *that satisfies*

$$\max \left\{ \sum_{z \in \mathcal{Z}^*} \hat{\varrho}(z, \tau_z) \middle| \bigcap_{z \in \mathcal{Z}^*} = \emptyset \wedge \bigcup_{z \in \mathcal{Z}^*} = \mathcal{R} \wedge \bigcap_{z \in \mathcal{Z}^*} \{\tau_z\} = \emptyset \right\}.$$

However, the assignment of MR-tasks requires not only the formation of the robot groups but also the assignment of the subtask responsibilities within the groups. Finding an optimal solution to this problem has been shown to be *NP*-complete [102], i.e. it is amongst the hardest problems in *NP*. In the literature, the subtask optimization is often neglected. Either the subtasks are assumed to be identical and/or the coalition rewards are assumed to be independent of the specific subtask assignment. As already mentioned in Section 3.2, until this point only [91] takes the subtask assignment explicitly into account, but due to an exhaustive search, the respective time complexity is in $O(|\mathcal{R}|!)$.

In this respect, the proposed approach illustrated in Fig. 3.3 finds a solution to this problem in less time, while still guaranteeing the same optimality. Thereby, all $\tau_j \in \mathcal{T}_{MR}$ are assigned in successive rounds, where the assignment of each round for itself is shown to be optimal. The optimization to assign a MR-task τ_j is split into two steps:

(i) For each $z \in Z_{cap}$ and for each $p_{\mathcal{T}}(\tau_j) \in \mathcal{P}_{\mathcal{T}}(\tau_j)$ find an optimal subtask assignment $\psi^*_{z,p_{\mathcal{T}}(\tau_j)} : \mathcal{R}_z \to \mathcal{T}_j$, where $\mathcal{R}_z = \bigcup_{r \in z} r$ is the set of all robots in z and $\mathcal{T}_j = \bigcup_{\tau_{jl} \in p_{\mathcal{T}}(\tau_j)} \tau_{jl}$ is the set of all subtasks in the task plan $p_{\mathcal{T}}(\tau_j)$ for τ_j.

(ii) Find the coalition assignment $\psi^*_{\tau_j} : \mathcal{Z} \to \tau_j$ with highest coalition reward among all subtask assignments $\psi^*_{z,p_{\mathcal{T}}(\tau_j)}$.

In principle, (i) could be solved locally at the auctioning robot r_a, but it is beneficial to distribute the responsibilities for the subtask assignments amongst the robots as it reduces the time complexity in average by $\frac{1}{|\mathcal{R}|}$. In this respect, coalition responsibilities $\mathcal{Z}_{r_b} \subseteq \mathcal{Z}$, i.e. the set of coalitions for whose subtask assignments broker r_b is responsible for, are equally distributed among the brokers. The respective algorithm is described in Appendix A. The assignment based on this distributed optimization is referred to as MuRoCo-D.

The auctioneer retrieves the set, $\mathcal{T}_{MR}$, of new MR-tasks similar to the set, $\mathcal{T}_{SR}$, of SR-tasks along with the set of new tasks, $\mathcal{T}_n$. As the assignment of the MR-tasks occurs successively, a greedy prioritisation according to the following heuristic is used:

$$\tau_j = \underset{\forall \tau \in \mathcal{T}_{MR}}{\arg\max} \frac{u(\tau)}{\overline{|z(\tau)|}} = \underset{\forall \tau \in \mathcal{T}_{MR}}{\arg\max} \frac{u(\tau)|\mathcal{P}_{\mathcal{T}}(\tau)|}{\sum\limits_{p_{\mathcal{T}}(\tau) \in \mathcal{P}_{\mathcal{T}}(\tau)} |p_{\mathcal{T}}(\tau)|}, \tag{3.7}$$

where $\overline{|z(\tau)|} = \frac{\sum_{p_{\mathcal{T}}(\tau) \in \mathcal{P}_{\mathcal{T}}(\tau)} |p_{\mathcal{T}}(\tau)|}{|\mathcal{P}_{\mathcal{T}}(\tau)|}$ is the average expected coalition size with respect to the set $\mathcal{P}_{\mathcal{T}}(\tau)$ of available task plans for τ. In other words, the task τ_j, for which the ratio between retrieved utility and average expected coalition size is largest, is chosen.

Similarly, the decision of the auctioneer on whether to announce τ_j or the entire set $\mathcal{T}_{SR}$, is based on

$$\mathcal{T}_a = \begin{cases} \{\tau_j\} & \text{if } \frac{u(\tau_j)}{\overline{|z(\tau_j)|}} > \frac{\sum\limits_{\tau_{SR} \in \mathcal{T}_{SR}} u(\tau_{SR})}{|\mathcal{T}_{SR}|} \\ \mathcal{T}_{SR} & \text{else,} \end{cases} \tag{3.8}$$

i.e. the set with a higher relative reward expectation is selected and announced to all brokers in the system.

When receiving a MR-task τ_j, a broker r_b has to derive a bid $\hat{\varrho}(z, \tau_j)$ for each coalition $z \in \mathcal{Z}_{r_b}$ it is responsible for (Alg. 3.2, ll. 1 - 15). First the subset of capable coalitions

$$\mathcal{Z}_{r_b,cap} = \bigcup_{\forall p_{\mathcal{T}}(\tau_j) \in \mathcal{P}_{\mathcal{T}}(\tau)} \mathcal{Z}_{r_b,cap}(p_{\mathcal{T}}(\tau_j)) \ \subseteq \mathcal{Z}_{r_b}$$

is derived with respect to Definition 3.12. The required knowledge of the capability sets $\mathcal{K}_r$, $\forall r \in z$, is retrieved through the alive messages as previously described in Section 3.4.1. At this step, the capability check yields its substantial computational savings as the estimated reward is simply set to $\hat{\varrho}(z, \tau_j) = -\infty$ for all $z \in \mathcal{Z}_{r_b} \setminus \mathcal{Z}_{r_b,cap}$. Instead, for all $z \in \mathcal{Z}_{r_b,cap}$ a subtask auction is initiated by the broker in order to derive the coalition reward $\hat{\varrho}(z, \tau_j)$.

The subtask auctions occur similar to the procedure described in Section 3.4.3, but now the initiating broker acts in parallel as auctioneer and the subtasks $\tau_{jl} \in p_{\mathcal{T}}(\tau_j)$ are only announced to the brokers of all $r \in \mathcal{R}$ for which $\exists z \in \mathcal{Z}_{r_b,cap}(p_{\mathcal{T}}(\tau_j))$ such that $r \in z$. These request bids $\hat{\varrho}(r, \tau_{jl})$, $\forall \tau_{jl} \in p_{\mathcal{T}}(\tau_j)$, from their corresponding bidders and reply these to the broker r_b.

After receiving all subtask bids, the broker r_b derives the optimal subtask assignment $\psi^*_{z,p_{\mathcal{T}}(\tau_j)}$ for z and $p_{\mathcal{T}}(\tau_j)$. Based on $\psi^*_{z,p_{\mathcal{T}}(\tau_j)}$ it then calculates the reward

$$\hat{\varrho}(z, p_{\mathcal{T}}(\tau_j)) = f\left(\bigcup_{\tau_{jl} \in p_{\mathcal{T}}(\tau_j)} \hat{\varrho}(\psi^{*-1}_{z,p_{\mathcal{T}}(\tau_j)}(\tau_{jl}), \tau_{jl}) \right)$$

for each $z \in \mathcal{Z}_{r_b}$ and for each $p_{\mathcal{T}}(\tau_j) \in \mathcal{P}_{\mathcal{T}}(\tau_j)$ by a functional relation $f(\cdot)$ of the respective subtask bids. An approach to derive $f(\cdot)$ is discussed in more detail in Chapter 4 (p. 67). Furthermore,

$$\hat{\varrho}(z, p_{\mathcal{T}}(\tau_j)) = -\infty \tag{3.9}$$

holds in case any subtask $\tau_{jl} \in p_{\mathcal{T}}(\tau_j)$ could not be assigned, i.e. in case the assignment is not perfect. The actual coalition bids are derived by

$$\hat{\varrho}(z', \tau_j) = \max_{p_{\mathcal{T}}(\tau_j) \in \mathcal{P}_{\mathcal{T}}(\tau_j)} \hat{\varrho}(z', p_{\mathcal{T}}(\tau_j)),$$

where

$$z' = \arg\max_{z \in \mathcal{Z}_{r_b}} \left(\max_{p_{\mathcal{T}}(\tau_j) \in \mathcal{P}_{\mathcal{T}}(\tau_j)} \hat{\varrho}\left(z, p_{\mathcal{T}}(\tau_j)\right) \right)$$

is the potential coalition candidate for τ_j, and sent back to the auctioneer r_a. The latter finally searches a solution to step (ii) by computing the optimal coalition

$$z^*(\tau_j) = \psi_{\tau_j}^{*^{-1}} = \arg\max_{z \in \mathcal{Z}} \hat{\varrho}(z, \tau_j) \tag{3.10}$$

for task τ_j. Thereafter, the coalition assignment $\psi_{\tau_j}^*$ is send to all brokers, where $\hat{\varrho}(z^*(\tau_j), \tau_j) = -\infty$ means that no capable coalition is found.

In order to confirm that all resources and capabilities are still available, a resource re-check is performed. Therefore all $r \in z^*(\tau_j)$ confirm the acceptance of τ_j by sending an acknowledgment mutually to each other and also to the auctioneer. In case of a successful re-check, the respective subtasks are issued to the execution components, otherwise the auctioneer initiates a reallocation of τ_j. The robots are bound to their tasks until these are either completed or the latter becomes infeasible due to a failure.

This approach for coalition formation extends the SR-task assignment of Section 3.4.3 and enables MuRoCo to also assign MR-tasks. In order to complete the tasks in principle the task execution components of the robots only have to execute the appropriate actions according to the respective action plans. Nevertheless, robots acting in the real world are always prone to uncertainty and failure. Furthermore, situations where no solution is found need to be handled. All this demands for means that enable a robust system operation.

3.4.5. Robustness and Failure Recovery

Robotic systems are faced with noisy sensor data leading to uncertainty in the build world model, which again may lead to failed actions. Besides this permanent disturbance, robots are further confronted with the occasional occurrence of hardware or software failures. Even though the primary objective in a robotic system is to anticipate and avoid such incidences, this cannot be always guaranteed and consequently, this means that the detection and recovery of system faults are required in order to yield a robust system operation.

In this context, four main issues which yield a dependable computing system are identified in [5]. These are the prevention, the tolerant handling, the removal and the forecasting of faults. In the following, various means are proposed to address these issues within MuRoCo.

3.4.5.1. Selection of the Auctioneer Role

A single auctioneer in the entire system is in general problematic as it presents a single point of failure. Using instead multiple distributed auctioneers that decide purely on their local

information leads to suboptimal solutions and thus demands for a mutual synchronization as for example in [128]. In this respect, three strategies for the selection of the auctioneer role are determined:

(i) *A single permanent auctioneer*:
A specific robot or an external device acts as unique auctioneer in the system. The advantage is that optimal solutions are possible due to a centralized optimization but the uniqueness of the auctioneer represents a single point of failure.

(ii) *The receiving auctioneer is always the auctioning one*:
Thereby, robustness is increased as, in case an auctioneer fails, only the tasks the auctioneer was responsible for are affected. However, due to the distributed optimization, the assignment of an auctioneer is only guaranteed to be locally optimal.

(iii) *A priority queue of auctioneers*:
With this strategy, the auctioneers assign priorities amongst each other, e.g. during an initial auction or according to their temporal activation. The auctioneer with highest priority acts as active auctioneer, while all other auctioneers stay inactive and forward their tasks to the active one. Each inactive auctioneer observes its directly higher-ranked counterpart. In case the latter fails, the inactive auctioneer takes over those role. This strategy allows for optimal solutions while avoiding a single point of failure. The disadvantages are increased communicational load and a more complex negotiation procedure.

So in general, strategy (iii) is the preferable one as it improves the prevention of failures while still allowing for optimal solutions. From the practical point of view, in a MRS with a limited operating area, a reliable network and approved software components strategy (i) may be sufficient for many applications as well.

3.4.5.2. Failure-Aware Cost Computation

A further aspect to yield a dependable computing system is the forecasting of failures [5]. Accordingly, even though robotic systems will remain always prone to errors, those effects on the system performance are reducible by taking the respective risk of failure already during the action selection into account.

When an action fails, the consequence is a deviation from the intended plan which often results in an increase of the costs. A simple approach to handle such an error is to repeat the respective action. However, depending on the task this may not be always possible. Certain action plans may require the complete repetition of the entire plan in case a single action fails, even if it is the last action in the plan. Considering the exemplary task τ_{fetch} introduced in Section 3.4.2, where a robot is supposed to fetch a cup of coffee from a table. In case the robot spills the coffee while handing it over to the person, it would have to repeat at least the entire plan $p_{\mathcal{A}}(\tau_{fetch})$, if not announce the demand for new cleaning tasks. A way to take this risk already during the assignment phase into account is by incorporating it into the cost estimation.

In the following, an approach is described that provides this estimation under the assumption of cost additivity, i.e. with respect to (2.5) that $c(p_{\mathcal{A}}(\tau_j))$ is assumed to result from the sum of all related action costs. In case the n-th sequential action of $p_{\mathcal{A}}(\tau)$ fails under such a condition, the respective additional cost can be approximated by the average number of prior actions that have to be repeated:

$$\begin{aligned} c_{fail}(a_n) &= \frac{1}{n}\left(c(a_1) + c(a_2) + \ldots + c(a_n)\right) + \ldots + \frac{1}{n}\left(c(a_{n-1}) + c(a_n)\right) + \frac{1}{n}c(a_n) \\ &= \frac{1}{n}\left(c(a_1) + 2c(a_2) + \ldots + nc(a_n)\right) \\ &= \sum_{i=1}^{n} \frac{i}{n} c(a_i). \end{aligned}$$

Since $c_{fail}(a_n)$ occurs only in case a_n fails, it needs to be weighted with the probability that a_n will fail that is derivable as follows.

According to Definition 3.5, a capability κ is defined as the ratio of its successful employments compared to the number of total employments, corresponding to the probability that κ will not fail during its usage. As a result, the probability that an error of κ will occur is $1 - \kappa$.

Nevertheless, to estimate the failure risk of an action, not only single but in fact all capabilities required in the course of its execution need to be considered. Let $P(\chi_a)$ be the probability that all capabilities used for the execution of a will not fail. $P(\chi_a)$ is derivable from the logical function f_{χ_a}. For a conjunctive constraint of two capabilities κ_x and κ_y holds

$$P(\chi^{\wedge}(\{\kappa_x, \kappa_y\})) = \kappa_x \kappa_y \tag{3.11}$$

as both are in use and the action fails as soon as one of them fails. It appears to be not that simple for a disjunctive constraint, since in this case, only one of the capabilities is in use and thus it depends upon which one the robot chooses for the execution of a. In this respect the following approximation is proposed to select a $\kappa \in \mathcal{K}_x$ for a disjunctive constraint $\chi^{\vee}(\mathcal{K}_x)$:

$$\kappa_{\chi^{\vee}(\mathcal{K}_x)} = \underset{\forall \kappa \in \mathcal{K}_x}{\arg\min} \; \left(c(a, \kappa) + (1 - \kappa) c_{fail}(a)\right)$$

where $c(a, \kappa)$ is the cost for performing a with κ. In other words the capability is chosen for which the failure-aware cost estimate is lowest. Accordingly follows for the success probability of a disjunctive constraint $\chi^{\vee}(\mathcal{K}_x)$

$$P(\chi^{\vee}(\{\kappa_x, \kappa_y\})) = \kappa_{\chi^{\vee}(\mathcal{K}_x)} \tag{3.12}$$

With (3.11) and (3.12), the basis is provided to derive the success probability $P(r, \chi_a)$ of a executed by robot r based on the constraint χ_a.

As a consequence, the cost estimation for the n-th action a_n can be extended by taking

the potential failure cost into account:

$$\begin{aligned}\hat{c}(r, a_n) &= P(r, \chi_{a_n})c(a_n) + (1 - P(r, \chi_{a_n}))(c(a_n) + c_{fail}(a_n)) \\ &= c(a_n) + (1 - P(r, \chi_{a_n})) \sum_{l=1}^{n} \frac{l}{n} c(a_l)\end{aligned}$$

So by considering (2.6) and (3.3) follows, under the assumption of cost-additivity, for the failure-aware bid calculation for task τ by the bidder of r:

$$\begin{aligned}\hat{\varrho}(r, p_{\mathcal{A}}(\tau)) &= u(\tau) - \sum_{a_n \in p_{\mathcal{A}}(\tau)} \hat{c}_g(r, a_n) \\ &= u(\tau) - \sum_{a_n \in p_{\mathcal{A}}(\tau)} \left(c(a_n) + (1 - P(r, \chi_{a_n})) \sum_{l=1}^{n} \frac{l}{n} c(a_l) \right)\end{aligned} \tag{3.13}$$

A task assignment based on this failure-aware cost estimation takes the unreliability of fault-prone robots into account and thereby improves the forecasting of failures in favoring those robots that provide a safe and reliable operation. The drawback of (3.13) is that it is only applicable in cost-additive setups.

3.4.5.3. Error Recovery

While the means described above focus on the prevention and forecasting of failures, a robotic system also needs to cope with situations where a failure did already occur. In [5], a set of elementary fault classes for computing systems is defined, such as internal vs. external or hardware vs. software faults. These are further clustered into the three major classes of design, physical and interaction faults. Similar to this concept MuRoCo implements three general categories into which an error is classified as follows:

$$\text{error} = \begin{cases} \textit{recoverable}, & \text{if } \hat{\varrho}'(r, \tau) > \varrho_{min}(\tau) \\ \textit{semi-recoverable}, & \text{if error} \neq \textit{recoverable} \wedge \exists z \in \mathcal{Z} : \mathcal{K}_z \prec \chi_\tau \\ \textit{non-recoverable}, & \text{else.} \end{cases} \tag{3.14}$$

Note that the second constraint for *semi-recoverable* errors also implies single robots since $\{r\} \in \mathcal{Z}, \forall r \in \mathcal{R}$.

Recoverable errors are those that can be ascribed to uncertainty problems, e.g. due to temporary bad sensor data, an unexpected change of the world state or interaction faults. Whenever an action terminated unsuccessfully the bidder calculates an updated reward $\hat{\varrho}'(r, \tau)$ based on the current state and verifies that this is still larger than a minimum bound $\varrho_{min}(\tau)$. The bound $\varrho_{min}(\tau)$ is set by the auctioneer along with the task assignment according to

$$\varrho_{min}(\tau) = \max\left(\alpha \hat{\varrho}^*(\tau), \hat{\varrho}'(\tau)\right), \tag{3.15}$$

where $\hat{\varrho}^*(\tau)$ is the best bid for τ, $\hat{\varrho}'(\tau)$ is the second best bid and $\alpha > 1$ is some weighting factor. With (3.15), any robot that accepts a task assignment takes the pledge to achieve at least $\varrho_{min}(\tau)$. This is utilized in a twofold manner. First, when some failure occurs

but $\hat{\varrho}'(r,\tau) > \varrho_{min}(\tau)$ is still satisfied, the error is classified as *recoverable* and solved by repeating the respective action or returning to a prior action in the currently executed action plan $p_{\mathcal{A}}(\tau)$.

Secondly, $\varrho(\tau)$ is permanently re-estimated during system operation. As soon as the robot observes that it will no longer exceed $\varrho_{min}(\tau)$, e.g. due to some dynamic change in the environment, it stops the execution of τ and reports a failure to the auctioneer. This enables to successfully detect and resolve deadlocks and to deal with miscalculated bids. Such failures, which are not solvable by a pure retry, are classified as *semi-recoverable*. A further example is a broken hardware module that partially or even completely affects the serviceability of the robot, so that it is no longer able to complete its dedicated tasks. In such a case, the updated reward results in $\hat{\varrho}'(r,\tau) = -\infty$ since $K_r \not\propto \chi_\tau$. Consequently, the failure type is sent to the auctioneer responsible for the allocation handling of the corresponding task. Additionally, all coalition partners are informed about the fault and released from depending tasks. Thereafter, the auctioneer evaluates the failure cause and decides whether a re-announcement of the task is possible or not. For example, in case one robot has a hardware problem, there is still the chance that another robot or coalition is capable to take over the task.

Situations where the auctioneer is not able to find a new valid assignment for the task or where the same error has occurred already too often in prior trials, are classified as instances of the third category of *non-recoverable* failures. Likewise, design faults belong to this category as well. Respective tasks have no or only a very unlikely chance of success in a further attempt. This may occur, for example, when a robot failed to manipulate an object because the latter could not be found in several previous trials or there remain simply not enough robots to form a capable coalition. In such a situation, the auctioneer module desists from a re-allocation and instead reports the failure cause to the higher level planning components or to a human operator.

To summarize, with the proposed auctioneer strategies, the failure-aware bid computation and the error classification, means are proposed to cover the prevention, the forecasting, as well as the handling and removal from potential failures. In this respect, the assignment of SR and MR-tasks of MuRoCo, as well as, its means to cope with the environmental uncertainty have been described throughout this section. The remainder of this chapter gives a theoretical and experimental evaluation of the framework.

3.5. Analysis of the Approach

Throughout this section is examined how well MuRoCo copes with the type of problems in focus. For these hold the underlying core assumptions as stated in Section 3.3. In the following parts, the soundness, completeness, scalability and optimality of MuRoCo are analyzed.

3.5.1. Soundness and Completeness

Soundness is the proof that the generated solutions are correct with respect to the environmental setting. Completeness is the guarantee that if a solution exists, it will be

found.

Proposition 3.15 *The task allocation of MuRoCo is sound.*

Proof: Assume the opposite would be the case, i.e. the solution is not correct. This may only happen in case

(i) a task is assigned to a robot which is not capable of performing it.
This is precluded by the capability check w.r.t. (3.6), since if this fails the robot is excluded from the assignment. The capability check follows the rules of Boolean algebra and thus is correct assuming that the task constraints χ_τ are correctly provided by the planning layer and/or designer.

(ii) the SR-assignment is not correct,
i.e. multiple tasks are assigned to the same robot or one SR-task is assigned to multiple robots. This may only occur in case the Hungarian method [63] is incorrect, which has been shown to be not the case.

(iii) the subtask-assignment of the coalition formation is not correct.
This is performed similar to the SR-assignment with the additional constraint that it needs to be a perfect assignment, i.e. neither a subtask nor a robot are allowed to be unassigned. The correctness of the assignment is given by (ii) and its perfectness is explicitly verified by (3.9).

(iv) a robot is part of multiple coalitions.
This is also not possible due to the greedy MR-assignment. As robots which have already an assignment reply a reward of $-\infty$ in the next round, all coalitions that contain these robots are classified as non-capable.

Consequently the presented task allocation algorithm is correct. ■

Proposition 3.16 *The task allocation of MuRoCo is complete with respect to the assignment of single MR-tasks or multiple SR-tasks given enough time.*

Proof: Assume again the opposite would be the case, i.e. a solution exists but it is not found. This may only happen in case

(i) the SR-assignment method is not complete, which is known to be not the case for the Hungarian method.

(ii) no Coalition is found for a MR-task even though one exists. Since during a MR-task assignment, given enough time, all possible coalitions and for these all possible subtask assignments are examined, any existing solution will be tested and the one with highest reward will be chosen.

Accordingly the presented task allocation algorithm is also complete with respect to the assignment of single MR-tasks or multiple SR-tasks. ■

However, with respect to the assignment of multiple MR-tasks or of one MR-task and multiple SR-tasks, completeness can not be guaranteed due to the sequential MR-assignment. For example a robot, which is indispensable for a second task, may be part of a previously formed coalition, even though there would have been alternatives for the first assignment. Adequate approaches to also tackle this very hard problem remain part of future research.

The completeness of MuRoCo has been shown under the assumption that enough time is provided to the algorithm. How this time scales with the number of robots and tasks is analyzed next.

3.5.2. Scalability

Analyzing the asymptotic complexity reveals how well a method scales with the problem size. For MRSs, this is primarily of interest with respect to the number of robots $|\mathcal{R}|$ and the number of tasks $|\mathcal{T}|$. In the following, the computational and communicational complexity of MuRoCo are analyzed, as well as the respective influence of the capability check on these.

As described in Section 3.4.4 the computational load for the coalition formation may be distributed among the robots, which is referred to as MuRoCo-D. In order to analyze the respective benefits, it is compared to the centralized version MuRoCo where all coalitions are evaluated locally by one single robot.

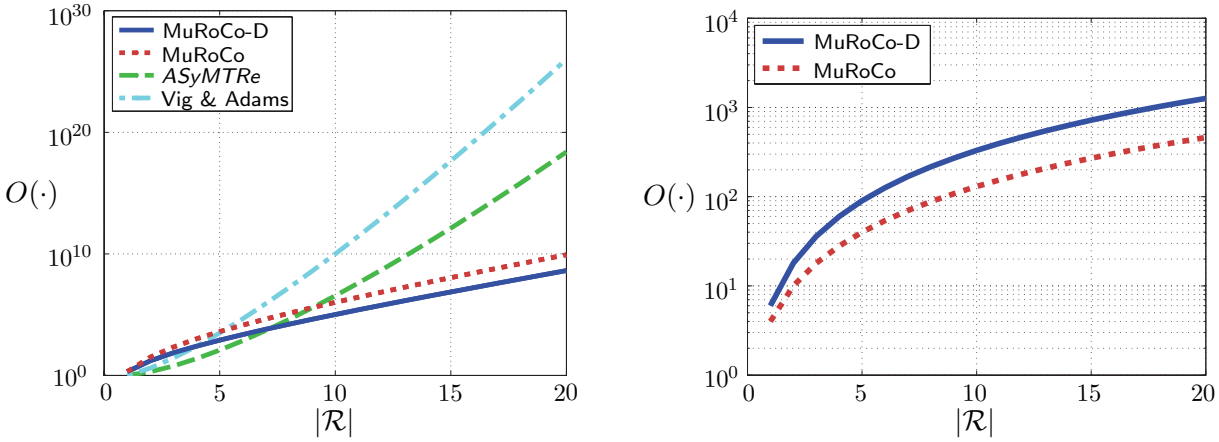
3.5.2.1. Computational Complexity

The computational complexity is here understood as time complexity, i.e. it refers to the number of computational operations required to solve instances of a problem depending on the problem size.

The assignment of SR-tasks based on the Hungarian method [63] has been shown to be solvable in $O(n^3)$ time, where $n = \max(|\mathcal{R}|, |\mathcal{T}|)$. An overview of alternatives to the Hungarian algorithm and respective worst-case bounds is given in [19]. All are solvable in loglinear or polynomial time.

So while SR-assignments are easily solvable for large instances, the actual challenge is opposed by MR-task assignments. The size of the set $\mathcal{Z}$ of all possible coalitions is given by $|\mathcal{Z}| = \sum_{k=0}^{|\mathcal{R}|} \binom{|\mathcal{R}|}{k} = 2^{|\mathcal{R}|}$. Taking also all possible subtask constellations into account, the number of possible robot orderings results in $\sum_{k=0}^{|\mathcal{R}|} \binom{|\mathcal{R}|}{k} k! = |\mathcal{R}|! \sum_{k=0}^{|\mathcal{R}|} \frac{1}{k!}$. Since $1 \leq \sum_{k=0}^{|\mathcal{R}|} \frac{1}{k!} \leq e$, where e is Euler's constant, an exhaustive search through all orderings lies in $O(|\mathcal{R}|!)$.

In order to specify the time complexity of the MR-assignment described in Section 3.4.4 the complexity of the subtask assignment and of the coalition formation need to be determined. The time complexity of the subtask assignment is similarly to the SR-assignment in $O(|\mathcal{R}|^3)$. Accordingly, assuming there exist exactly m task plans for every possible coalition size, then the computational complexity of the centralized MR-task assignment, MuRoCo, is given by $O(2^{|\mathcal{R}|}|\mathcal{R}|^3 m)$. In the distributed version of the algorithm, MuRoCo-D, the evaluation of the coalitions is shared among all robots. Assuming an equal balancing of the computational load among the robots as e.g. provided by Algorithm A.1, the effective number of coalitions is reduced to $2^{|\mathcal{R}|}/|\mathcal{R}|$ leading to $O(2^{|\mathcal{R}|}|\mathcal{R}|^2 m)$. Considering that



(a) Computational Operations

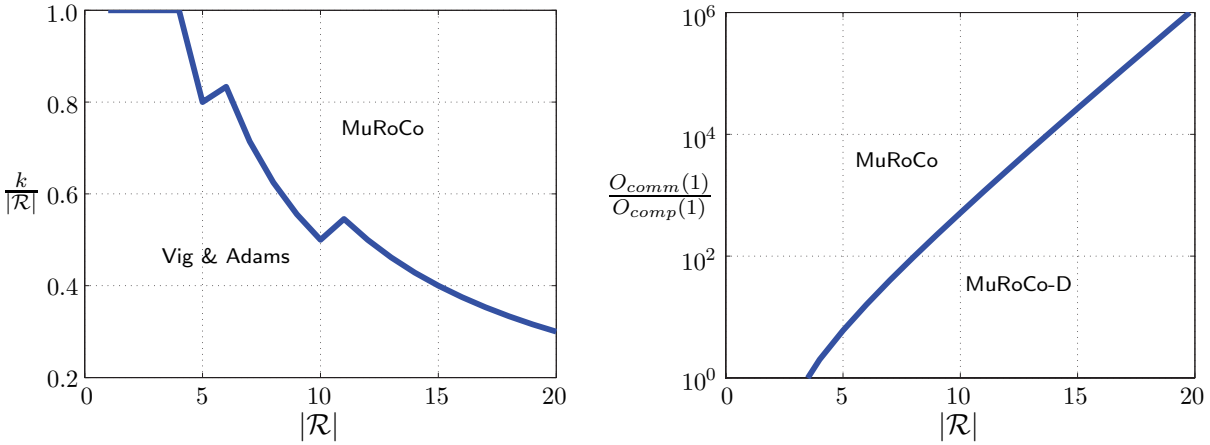
(b) Communicational Operations

Fig. 3.4.: Number of computational and communicational operations for a MR-task assignment with respect to the number of robots. The number of subtasks is set to the average coalition size $|\mathcal{R}|/2$.

there is still the exponential growth by $|\mathcal{R}|$ this improvement is admittedly rather marginal. Nevertheless it yields a lower complexity what gave reason to mention it here.

Fig. 3.4(a) shows the course of the computational complexity with the number of robots $|\mathcal{R}|$ for the centralized and the distributed version. The number of subtasks is set to $|\mathcal{R}|/2$ what corresponds to the average number of robots per coalition. It further shows the respective curves for the two known approaches that are likewise capable to find the optimal MR-assignment among a set of alternative solutions. These are *ASyMTRe* [91] and the approach by Vig and Adams [122]. The focus of *ASyMTRe* is actually on the plan generation, i.e. the automatic decomposition of the MR-tasks into feasible subtasks, while for the actual optimization they integrate an exhaustive search in an anytime manner. Consequently *ASyMTRe* scales with $O(|\mathcal{R}|!)$. Admittedly *ASyMTRe* aims to solve a different problem but as it is to our knowledge the so far most efficient approach in literature, which likewise takes the subtask assignment into account, it is used as benchmark here. In Fig. 3.4(a) is observable that the factorial growth is in the beginning actually lower than the exponential ones. Accordingly, only for MRSs with more than ten robots (MuRoCo) or more than eight robots (MuRoCo-D), a reduced worst-case complexity is achieved.

Vig and Adams [122] modified the Multi-Agent algorithm of Shehory and Kraus [106] to be applicable also for MRSs. Nevertheless, they maintained the complexity of $O(|\mathcal{R}|^k)$, where k is a constraint on the maximal coalition size. In this respect, for a fixed k their approach yields solutions in polynomial time but for the sacrifice of suboptimality. In order to ensure optimality, k needs to be set equal to $|\mathcal{R}|$, for what [122] would scale theoretically even worse than the exhaustive search. Fig. 3.5(a) shows the minimal ratio of $k/|\mathcal{R}|$ for which $O(2^{|\mathcal{R}|}|\mathcal{R}|^3) \leq O(|\mathcal{R}|^k)$. The curve shows that [122] is beneficial as long as k is kept low compared to $|\mathcal{R}|$, but the discrepancy is growing with an increase of $|\mathcal{R}|$. In other words, given the same number of computational operations, the fraction of orderings evaluated by [122] is decreasing with increasing $|\mathcal{R}|$. Accordingly the solution when using a constraint k with MuRoCo is likely to be closer to the optimal one as with [122].



(a) (b) Communicational vs. computational cost

Fig. 3.5.: Relations of complexity: (a) minimal ratio of $k/|\mathcal{R}|$ for which $O(2^{|\mathcal{R}|}|\mathcal{R}|^3) \leq O(|\mathcal{R}|^k)$; (b) ratio $O_{comm}(1)/O_{comp}(1)$ of the costs occurring for one communicational operation compared to one computational operation. For ratios above the curve the centralized optimization performs better, for ratios below the curve the distributed optimization performs better.

	SR-Task	MR-Task	
		MuRoCo	MuRoCo-D
Announce:	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$
Request:	-	-	$\|\mathcal{R}\|^2$
Reply:	-	-	$\|\mathcal{R}\|^2$
Bid:	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$
Assign:	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$	$\|\mathcal{R}\|$
Ack:	$\|\mathcal{R}\|$	$\|\mathcal{R}\|^2$	$\|\mathcal{R}\|^2$
Total:	$4\|\mathcal{R}\|$	$\|\mathcal{R}\|^2 + 3\|\mathcal{R}\|$	$3\|\mathcal{R}\|^2 + 3\|\mathcal{R}\|$

Tab. 3.1.: Number of messages required during the phases of the allocation procedure.

3.5.2.2. Communicational Complexity

In distributed applications, besides the computational complexity also the communicational complexity is of interest as it quantifies the amount of communicational events required. The communicational complexity is in the following expressed by the number of messages that need to be exchanged during the allocation procedure.

Thereby it is assumed that all communication events occur over a perfect network channel, i.e. without any package loss or time delay. Moreover, any communication among processes on the same computer, e.g. among auctioneer and broker of the same robot, are treated similarly to the communication among different computers. Further are mutual waiting times neglected, so it is assumed that all distributed computation occurs simultaneously. Table 3.1 shows the respective quantities of messages for the various allocation phases. The SR-assignment is split into the four phases of task announcement, bidding,

	Computation	Communication
SR-tasks	$O(\|\mathcal{R}\|\|\mathcal{T}\|^2)$	$O(4\|\mathcal{R}\|)$
MuRoCo	$O(2^{\|\mathcal{R}\|}\|\mathcal{R}\|^3 m)$	$O(3\|\mathcal{R}\| + \|\mathcal{R}\|^2)$
MuRoCo-D	$O(2^{\|\mathcal{R}\|}\|\mathcal{R}\|^2 m)$	$O(3\|\mathcal{R}\|^2 + 3\|\mathcal{R}\|)$
ASyMTRe [91]	$O(\|\mathcal{R}\|!)$	?
V&A [122] / S&K [106]	$O(\|\mathcal{R}\|^k)$	?

Tab. 3.2.: Number of operations and number of messages required, where k is the maximal coalition size and m is the number of taskplans $p_{\mathcal{T}}$ available for each coalition size.

task assignment and the final acceptance notification. In the worst case, i.e. in case of a perfect assignment where all robots receive a task, each phase requires $|\mathcal{R}|$ messages as the auctioneer needs to communicate with all brokers and vice versa. Accordingly the total number of messages results in $4|\mathcal{R}|$.

From the communicational point of view, the centralized version of MuRoCo is similar to the SR-task assignment, except for the acceptance notification. During the coalition formation not only the auctioneer but also all coalition partners need to be notified. In the worst case a coalition of all robots $r \in \mathcal{R}$ is formed requiring $|\mathcal{R}|^2$ acknowledgment messages.

The required communication of the distributed version MuRoCo-D is again similar to the centralized one, but due to the distributed computation of the coalition bids an additional effort to obtain the subtask bids is necessary. Accordingly two further phases are required to request and reply the subtask bids from all potential coalition members. In the worst case all brokers require subtask bids from all bidders resulting in $|\mathcal{R}|^2$ for the request and once more for the reply. Note that it is assumed that the bids for all subtasks are bundled in a single message. The course of the communicational complexity of MuRoCo and MuRoCo-D is shown in Fig. 3.4(b).

The overall numbers of required communicational and computational operations are summarized in Table 3.2. It is observable that MuRoCo-D requires less computational operations but therefore more communicational ones. Fig. 3.5(b) shows for which ratios $O_{comm}(1)/O_{comp}(1)$ of the cost $O_{comm}(1)$ for one communicational operation compared to the cost $O_{comp}(1)$ for one computational operation the distributed version is more beneficial and vice versa.

3.5.2.3. Efficiency of Pruning Strategies

In Section 3.4.2 the capability check was introduced to prune incapable coalitions in order to reduce the solution space and thus the complexity in an early stage of the allocation. Therefore the set of capable coalitions is approximated according to Definition 3.12 as the subset of all coalitions $z \in \mathcal{Z}$, for which the union of capabilities satisfies the constraint on

the task plan $p_{\mathcal{T}}(\tau)$. This is in the following referred to as

$$\mathcal{Z}_{cap1}(p_{\mathcal{T}}(\tau)) = \mathcal{Z}_{cap}(p_{\mathcal{T}}(\tau)) = \left\{ z \in \mathcal{Z} \,\middle|\, |z| = |p_{\mathcal{T}}(\tau)| \wedge \bigcup_{\forall r \in z} \mathcal{K}_r \prec \chi_{p_{\mathcal{T}}(\tau)} = \bigwedge_{\forall \tau_j \in p_{\mathcal{T}}(\tau)} \chi_{\tau_j} \right\}.$$

The time complexity to check one $z \in \mathcal{Z}$ according to this Definition is $O(2|z|)$, where $|z| = |p_{\mathcal{T}}(\tau)|$, so $\mathcal{Z}_{cap1}(p_{\mathcal{T}}(\tau))$ is derivable in linear time. However, it provides only a rough approximation instead of the true set of capable coalitions, i.e. depending on the task constraints and the robot capabilities also non-capable coalitions may be part of $\mathcal{Z}_{cap1}(p_{\mathcal{T}}(\tau))$. In this respect an alternative more precise definition may be used.

Definition 3.17 *The approximated set of capable coalitions*

$$\mathcal{Z}_{cap2}(p_{\mathcal{T}}(\tau)) := \left\{ z \in \mathcal{Z} \,\middle|\, |z| = |p_{\mathcal{T}}(\tau)| \wedge \exists r \in z : \mathcal{K}_r \prec \chi_{\tau_j}, \forall \tau_j \in p_{\mathcal{T}}(\tau) \right\}$$

is the subset of all coalitions $z \in \mathcal{Z}$, in which for each task in the task plan $p_{\mathcal{T}}(\tau)$ there is at least one robot that is capable to execute the task.

Definition 3.17 considers not only the union of all capabilities, but verifies that these are also physically linked to the robots in a suitable manner. The time complexity of Definition 3.17 is $O(|z|^2)$, thus it provides a more accurate approximation of the set of capable coalitions but it scales quadratic with $|z|$. A further alternative with $O(|z|^3)$ is given by the exhaustive capability check.

Definition 3.18 *Let $\psi_{\mathcal{K}} : z \rightarrow \bigcup_{\forall \tau_j \in p_{\mathcal{T}}(\tau)} \tau_j$ be a capability-based assignment in the sense that for the bids holds:*

$$\hat{\varrho}(r, \tau_j) = \begin{cases} 1 & \text{if } \mathcal{K}_r \prec \chi_{\tau_j} \\ 0 & \text{else} \end{cases} \quad \forall r \in z, \forall \tau_j \in p_{\mathcal{T}}(\tau).$$

Then the set of capable coalitions

$$\mathcal{Z}_{cap3}(p_{\mathcal{T}}(\tau)) := \{ z \in \mathcal{Z} \,|\exists \psi_{\mathcal{K}}(z, p_{\mathcal{T}}(\tau)),\ s.t.\ \psi_{\mathcal{K}} \text{ is perfect} \}$$

is the subset of all coalitions $z \in \mathcal{Z}$, for which a perfect capability-based assignment $\psi_{\mathcal{K}}$ exists.

Definition 3.18 finds the true set of capable coalitions, but results in a larger time complexity of the capability check. Since the capability check is supposed to reduce the complexity of the subsequent allocation, it is of interest how this additional effort compares to the subsequent savings. In the following the alternative Definitions 3.12, 3.17 and 3.18 are compared with each other for the centralized MuRoCo. Let n_{CC} be the average number of operations required for the capability check of the $z \in \mathcal{Z}$ and n_{SA} the average number of operations for a subtask assignment within the $z \in \mathcal{Z}_{capx}$. Then the overall number of computational operations required for the coalition formation is approximated by

$$n_{CF} = |\mathcal{Z}| n_{CC} + |\mathcal{Z}_{capx}| n_{SA} \tag{3.16}$$

In order to benefit from the capability check, n_{CF} needs to be smaller than $|\mathcal{Z}|n_{SA}$ corresponding to a coalition formation without any prior pruning. This leads to

$$P(z \in \mathcal{Z}_{capx}) = \frac{|\mathcal{Z}_{capx}|}{|\mathcal{Z}|} \overset{!}{\leq} 1 - \frac{n_{CC}}{n_{SA}}, \tag{3.17}$$

i.e. the upper bound of the probability that any $z \in \mathcal{Z}$ is in $\mathcal{Z}_{capx}$ needs to be less than $1 - \frac{n_{CC}}{n_{SA}}$ in order to yield a computational benefit by the pruning. As n_{CC} and n_{SA} depend both on the actual coalition sizes of the specific z contained in the sets, only a rough approximation of the bounds is possible. Assuming $\overline{|z|}$ is the average coalition size which is the same for $\mathcal{Z}$ and all $\mathcal{Z}_{capx}$, $x \in \{1, 2, 3\}$, then holds

$$\begin{aligned} P(z \in \mathcal{Z}_{cap1}) &\approx 1 - \frac{1}{\overline{|z|}^2} \\ P(z \in \mathcal{Z}_{cap2}) &\approx 1 - \frac{1}{\overline{|z|}} \\ P(z \in \mathcal{Z}_{cap3}) &\approx 0. \end{aligned}$$

The latter is intuitive, as from the computational perspective a pruning with respect to Definition 3.18 is similar to a full coalition formation without pruning. Accordingly this is only beneficial in case none of the coalitions is capable to perform the task. As already mentioned, these bounds present rather a rough estimation and are primarily supposed as practical guideline which pruning strategy to choose for a specific MRS.

Further possibilities to reduce the complexity are to limit the search based on heuristics, which for example exploit the physical constraints of the MRS. Thereby not the entire solution space is examined but the search is rather focused on the most likely candidates, but for the sacrifice of suboptimality. In [14] for example, only coalitions are evaluated whose members are located within a specified mutual operating distance, or [122] set a hard constraint on the maximum coalition size. Especially for tightly coupled tasks the number of potentially cooperating robots is often physically constrained making the evaluation of huge coalitions in these situations obsolete. Nevertheless, even though respective heuristics may be beneficial in many practical situations, their actual effectiveness is determined by the specific application. In case all robots are anyway located within a limited area, there is not much to gain by constraining the operating distance. Likewise, if there are tasks that are substantially better performed by larger coalitions, e.g. the surveillance of huge areas, a constraint on the maximum team size may rather negatively affect the performance. In this respect, MuRoCo aims at providing a general framework that is independent of specific applications. Nevertheless the incorporation of heuristics may be favorable in the practical case, but besides those influence on the complexity also the respective effects on the solution quality have to be considered.

	τ_1	τ_2	τ_3
$\hat{\varrho}_{z1}$	**10**	~~10~~	~~10~~
$\hat{\varrho}_{z2}$	10	**1**	~~10~~
$\hat{\varrho}_{z3}$	10	1	**1**

Tab. 3.3.: Exemplary greedy assignment of a MR-task for $\varrho_{max}/\varrho_{min} = 10$ and $n_{MR} = 3$. Instead of 30 the greedy cost is 12. The circles indicate the best possible assignment. The striked out numbers indicate the a priori cost, i.e. without prior assignment.

3.5.3. Optimality

The optimality analysis examines how close a found solution is to the optimal one. In this respect let

$$\varrho_{max} = \max_{z \in \mathcal{Z}, \tau \in \mathcal{T}} (\varrho(z, \tau)) \tag{3.18}$$

be the largest and

$$\varrho_{min} = \min_{z \in \mathcal{Z}, \tau \in \mathcal{T}} (\varrho(z, \tau)) \tag{3.19}$$

the lowest possible reward for any task $\tau \in \mathcal{T}$ executed by any coalition $z \in \mathcal{Z}$. Let further be

$$\frac{\varrho_{max}}{\varrho_{min}} \in [1, \infty[\tag{3.20}$$

the respective ratio.

As described in Section 3.4.4, MR-tasks are assigned successively. The assignment of a single MR-task is thereby optimal, as the algorithm examines all feasible solutions given enough time. In case multiple MR-tasks are issued to the system, this may lead to suboptimal solutions as coalitions that might be more suitable for a later task may have been already assigned in a previous round.

A worst-case situation is exemplary illustrated in Table 3.3 where three tasks are issued to a MRS with three exemplary coalitions. The task τ_1 is assigned first. As all bids for τ_1 are equal it is assigned to z_1, e.g. by random choice. In the second round τ_2 is assigned for which z_1 would achieve the best reward $\hat{\varrho}(z_1, \tau_1) = \varrho_{max} = 10$. However, as z_1 is already bound to τ_1 it is excluded from the assignment. As a consequence z_2 receives τ_2 even though its reward $\hat{\varrho}(z_2, \tau_2) = \varrho_{min} = 1$ is a lot lower as the one z_1 could have achieved. A similar situation is given in the third round where the only free coalition z_3 is the one with the poorest reward.

In this respect, assume n_{MR} is the number of simultaneously arriving MR-tasks and further that $|\mathcal{Z}_{cap}| \geq n_{MR}$, i.e. there are always enough capable coalitions. Further assume that there is at least one coalition with $\hat{\varrho}(z, \tau) = \varrho_{max}$ and at least one with $\hat{\varrho}(z, \tau) = \varrho_{min}$ for each τ. In the first round, the coalition z with highest $\hat{\varrho}(z, \tau_1)$ is chosen, as all coalitions are still free. For the $(n_{MR} - 1)$ subsequent rounds the respectively best coalition might be already dedicated to a task. Thus the performance after n_{MR} rounds is given according

to (2.7) by

$$q'(n_{MR}) = \frac{1}{\beta_u}(\varrho_{max} + (n_{MR} - 1)\varrho_{min})$$

in the worst-case and by

$$q^*(n_{MR}) = \frac{1}{\beta_u}(n_{MR}\ \varrho_{max})$$

in the optimal one. From this follows

$$\epsilon_{MR} = \lim_{n_{MR}\to\infty} \frac{q'(n_{MR})}{q^*(n_{MR})} = \lim_{n_{MR}\to\infty} \frac{\varrho_{max} + (n_{MR} - 1)\varrho_{min}}{n_{MR}\ \varrho_{max}} = \frac{\varrho_{min}}{\varrho_{max}},$$

i.e. the iterative assignment of MR-tasks is guaranteed to be at least $\epsilon_{MR}-$optimal.

3.6. Experimental Results

This section shows experimental results for the presented cooperation framework, where the distributed version MuRoCo-D has been applied to four heterogeneous robots in a service scenario. First a description of the multi-robot system and the service scenario is given. Thereafter, the operational sequence of MuRoCo-D is described, where first a faultless run is presented and then a run during which various external disturbances have been forced to the system. Finally a benchmark evaluation is given that shows the practical limitations of MuRoCo with respect to realizable problem sizes.

3.6.1. Experimental Setup

In the following the used robotic hardware is described from which the capability sets of the robots are deduced. Thereafter the service scenario is presented and respective task plans and constraints are formulated.

3.6.1.1. Description of the Robotic Hardware

The used multi-robot system is composed of the four heterogeneous MuRoLa[2] robots. The basis of each robot is a four-wheeled omnidirectional mobile platform [45]. For safe navigation each robot has two Sick LMS300 laser range finders which allow a circumferential obstacle detection in the scanning plane. The platform and the lasers provide the robots with the κ_{move} capability.

Two identical anthropomorphic 7-degrees-of-freedom (DoF) arms in a mirrored configuration are front-mounted on the top to provide a human-like working space [113]. The arms are able to carry up to 7 kg of static load each and are equipped with JR3 6-DoF force-torque sensors positioned before the end-effectors.

While all four robots have the same platform and arms, they are provided with different end-effectors and heads. Two robots (Eddie and Jimmy) have a Schunk PG70 two-finger gripper on each arm providing them with the $\kappa_{2f-grasp}$ capability. One robot (Poppy) is equipped with a three-fingered BarrettHand^{TM} ($\kappa_{3f-grasp}$), while the fourth robot (Tommy)

[2]The Multi-Robot Lab (`www.murola.de`)

Robot	Capabilities					
	κ_{move}	κ_{detect}	$\kappa_{2f-grasp}$	$\kappa_{3f-grasp}$	$\kappa_{cue-hold}$	κ_{dialog}
Eddie	✓	✓	✓			✓
Jimmy	✓	✓	✓			
Poppy	✓	✓		✓		
Tommy	✓	✓			✓	

Tab. 3.4.: Capabilities of the MuRoLa robots.

has custom-made end-effectors to hold a billiard cue ($\kappa_{cue-grasp}$) but these are not suitable to grasp any other object.

Furthermore the emotion display head EDDIE [110] is mounted on top of the body of Eddie, while the other robots are equipped with a pan-tilt stereo camera head instead. Eddie also possesses a dialog module which is capable of speech recognition and synthesis (κ_{dialog}).

For robot-robot communication Wireless-LAN is used. To enable the precise tracking of objects, robots and humans, the area is covered by a Visualeyez^{TM} VZ4000 motion measurement and tracking system from PhoeniX Technologies Inc.[3] that is based on active infrared markers. The tracking data is accessible by all robots (κ_{detect}). The capabilities of the robots are summarized in Table 3.4.

3.6.1.2. The Service Scenario

The robots are deployed in a service scenario, where they are supposed to take orders from the present persons and serve cups with the favored drinks to them. The persons as well as the cups are tagged with tracker markers based on which their position and orientation is perceived by the robots.

First the task τ_{order} is needed to retrieve orders from a person p. For τ_{order} no task plans are available, i.e. $\mathcal{P}_{\mathcal{T}}(\tau_{order}) = \emptyset$, but the action-plan set

$$\mathcal{P}_{\mathcal{A}}(\tau_{order}(p)) = \{\langle a_{move}(p), a_{dialog}(p)\rangle\}$$

is known.

For the serve task τ_{serve} a set of three task plans to bring an object o to a recipient p is known:

$$\mathcal{P}_{\mathcal{T}}(\tau_{serve}(o,p)) = \{\langle \tau_{fetch}(o,p)\rangle, \langle \tau_{bring}(o,p)\rangle, \langle \tau_{fetch}(o, p(\tau_{bring})), \tau_{bring}(o(\tau_{fetch}), p)\rangle\}.$$

For τ_{fetch} the set of action plans

$$\mathcal{P}_{\mathcal{A}}(\tau_{fetch}(o,p)) = \{\langle a_{move}(o), a_{grasp}(o), a_{move}(p), a_{handover}(o,p)\rangle\}$$

[3] `www.ptiphoenix.com`

and for τ_{bring} the set

$$\mathcal{P}_{\mathcal{A}}(\tau_{bring}(o,p)) = \{\langle a_{move}(o), a_{grasp}(o), a_{move}(p), a_{handover}(o,p), a_{dialog}(p)\rangle\}$$

is known. So for both tasks only one action plan $p_{\mathcal{A}}$ is known and the only difference between $p_{\mathcal{A}}(\tau_{fetch})$ and $p_{\mathcal{A}}(\tau_{bring})$ is that the latter has an additional dialog action in the end. The availability of the dialog is also the motivation for the cooperative plan $\langle \tau_{fetch}(o, p(\tau_{bring})), \tau_{bring}(o,p)\rangle$ as will be shown in Section 3.6.2.

The respective constraints for each task follow from the sets of action plans:

$$\begin{aligned}
\chi(\tau_{order}) &= \chi_{move} \wedge \chi_{detect} \wedge \chi_{dialog} \\
\chi(\tau_{fetch}) &= \chi_{move} \wedge \chi_{detect} \wedge \chi_{grasp} \\
\chi(\tau_{bring}) &= \chi_{move} \wedge \chi_{detect} \wedge \chi_{grasp} \wedge \chi_{dialog} \\
\chi_{grasp} &= \chi_{2f-grasp} \vee \chi_{3f-grasp}
\end{aligned}$$

It is assumed that the task plans and the respective constraints are known to all robots.

With respect to the reward calculations, the utilities are in general set equal for all tasks. However, as the application presents a service scenario, a high value is set on service quality. In this respect, for any task or action plan that involves a dialog action the utility is set twice as high, i.e. $u(\tau_{order}) = u(\tau_{bring}) = 2u(\tau_{fetch})$. The action costs are derived by estimating the respective execution times. These estimations are derived by distance-based calculations or experimentally. Furthermore, the functional relation

$$\hat{\varrho}(z, p_{\mathcal{T}}(\tau_j)) = f\left(\bigcup_{\tau_{jl}\in p_{\mathcal{T}}(\tau_j)} \hat{\varrho}(\psi^{*-1}_{z,\tau_j}(\tau_{jl}), \tau_{jl})\right) = \frac{1}{|z|}\sum_{\tau_{jl}\in p_{\mathcal{T}}(\tau_j)} \hat{\varrho}(\psi^{*-1}_{z,\tau_j}(\tau_{jl}), \tau_{jl}) \tag{3.21}$$

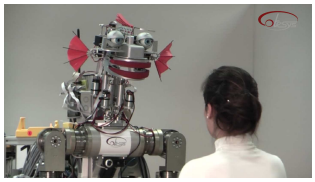
is used to derive the coalition bids w.r.t. (3.4.4). That means the individual robot costs are averaged, which in turn means the used resources are disregarded in favor of a faster execution time. This may be in general not the most economical solution but it is conform with the service-oriented policy.

Due to the slow task arrival in the present application the auctioneer policy is less relevant. The strategy that *the receiving auctioneer is always the auctioning one* has been chosen due to its higher robustness. Even though theoretically this policy guarantees only locally optimal solutions these are commonly globally optimal in the current setup due to the comparably rare task arrival.

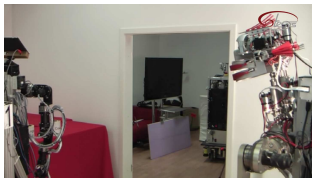
So along with the robot capabilities, the task formulations and the reward computations, all essential specifications for the MRS are determined in order to enable a successful operation.

3.6.2. The Course of Action Selection during a Cooperative Service Task

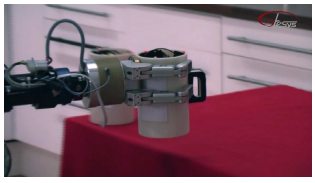
In this part the cooperative serving of a drink by two robots is described. The course of action is shown in Fig. 3.6. At the beginning all robots are idle. An observation module



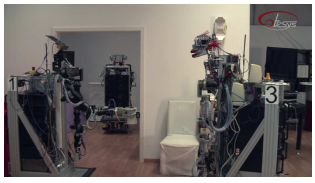
(a) Eddie takes an order from the person.



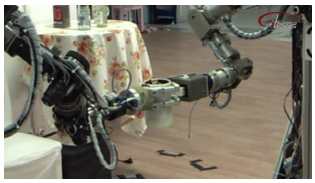
(b) A coalition of Eddie and Poppy has been formed.



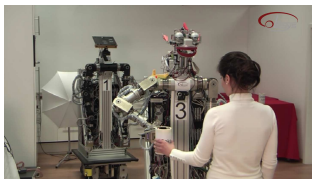
(c) Poppy uses a three-finger grasp.



(d) Poppy is done with grasping and the handover to Eddie starts.



(e) the robot-robot handover takes place.



(f) Eddie hands the drink with a kind comment to the person.

Fig. 3.6.: Scenes showing the execution of the plan $p_{\mathcal{T}}(\tau_{serve}) = \langle \tau_{fetch}, \tau_{bring} \rangle$ where two robots serve a drink in a cooperative manner.

(κ_{detect}) running on each robot notifies the local auctioneer whenever a new person enters the area. In case it has not been already done by any other robot, the auctioneer announces the task τ_{order}. As no task plan for τ_{order} is available, i.e. $\nexists p_{\mathcal{T}}(\tau_{order})$ s.t. $|p_{\mathcal{T}}(\tau_{order})| \geq 2$, a SR-auction is initiated.

The brokers of all robots compute their rewards $\hat{\varrho}(r, \tau_{order})$. Since Eddie is the only robot which possesses the κ_{dialog} capability, its performance estimate is the only finite one and it is assigned the task. So Eddie starts the execution of τ_{order} and takes the order as shown in Fig. 3.6(a).

The order of the person results in a new task τ_{serve} which is passed on to the auctioneer of Eddie. As for τ_{serve} there is a $p_{\mathcal{T}}(\tau_{serve})$ s.t. $|p_{\mathcal{T}}(\tau_{serve})| \geq 2$, a MR-auction is initiated. As the distributed version MuRoCo-D is used, τ_{serve} is announced to all brokers that start to derive the coalition bids for the coalitions they are responsible for. See Table A.2 for an exemplary responsibility distribution. In the current example, all robots except Tommy are capable to perform τ_{fetch} but only Eddie is able to perform τ_{bring} due to $\kappa_{dialog} \in \mathcal{K}_{Eddie}$.

Accordingly all coalitions except $\{Tommy\}$ pass the capability check. So the brokers announce the respective subtask auctions to all $z \in \mathcal{Z}_{cap}$. In the exemplary run shown in Fig. 3.6 the handle of the cup is not reachable, which is why the cup can only be grasped with the BarrettHand ($\kappa_{3f-grasp}$). Consequently the only coalitions with a finite cost are $\{Poppy\}$ and $\{Eddie, Poppy\}$, where Poppy calculates a finite cost for τ_{fetch} and Eddie a

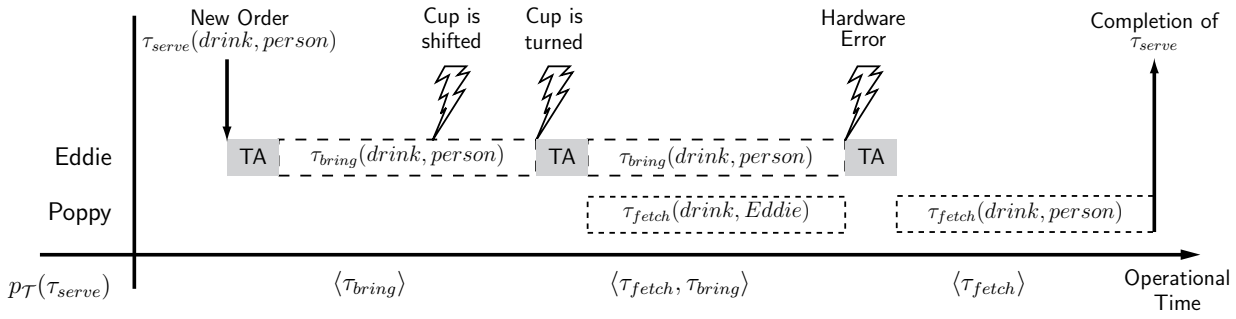


Fig. 3.7.: Sequence of trials to complete the task τ_{serve} in the incidence of external disturbances. The illustration is not true to scale w.r.t. to the operational time, as the task (re-)allocation phases (TA) are substantially shorter than the execution times.

finite cost for τ_{bring}.

The auctioneer receives the coalition bids from the brokers. As in the current example according to (3.21) the coalition rewards are divided by the coalition size, the two coalitions with a $\hat{\varrho}(z, \tau_{serve}) \neq -\infty$ do not vary a lot with respect to their costs. However, as $\{Eddie, Poppy\}$ yield the double utility due to the dialog action they estimate the highest $\hat{\varrho}(z, \tau_{serve})$ and thus obtain the task assignment $\langle \tau_{fetch}, \tau_{bring} \rangle \rightarrow \langle Poppy, Eddie \rangle$. Thereafter the cooperative execution starts, Fig. 3.6(b). Poppy grasps the cup, Fig. 3.6(c), and then turns towards Eddie while at the same time notifying that it is done with grasping, Fig. 3.6(d). Once the robots face each other the handover of the cup takes place, Fig. 3.6(e). Subsequently, Eddie approaches the person a second time and delivers the order as seen in Fig. 3.6(f).

3.6.3. The Course of Action Selection under Uncertainty

In the following, the course of actions is described, when unanticipated situations occur or external disturbances are induced to the MRS. All the disturbances have been forced to the system and occur after τ_{order} is completed. So it is assumed that the auctioneer of Eddie has already received τ_{serve} from a person. The course of tasks is shown in Fig. 3.7.

In this exemplary run, at the beginning the cup is in a position where the handle is reachable. In contrast to the previous case it is now also graspable with the Schunk PG70 two-finger gripper ($\kappa_{2f-grasp}$) such that $z = \{Eddie\}$ yields a finite cost. In the current case $\hat{\varrho}(\{Eddie\}, \tau_{serve}) > \hat{\varrho}(\{Poppy, Eddie\}, \tau_{serve})$ so Eddie wins the auction and starts the execution of $\tau_{bring}(drink, person)$.

In order to induce a recoverable error, the cup is shifted sideways while Eddie is already in the final grasp phase. Eddie observes the displacement and recalculates $\hat{\varrho}(Eddie, \tau_{bring})$ which is lower than the initial estimate due to the increased execution time. Nevertheless, as the cup was only shifted locally at the table $\hat{\varrho}(Eddie, \tau_{bring}) > \varrho_{min}$ is still satisfied, classifying this error according to (3.14) as *recoverable*. This might probably not be the case when the cup is for example moved to another room.

So while Eddie pursues the execution of τ_{bring} and starts a second time to grasp the cup, the latter is turned around such that the handle is no longer reachable. This leads to a situation that is similar to the one previously described in Section 3.6.2. So in

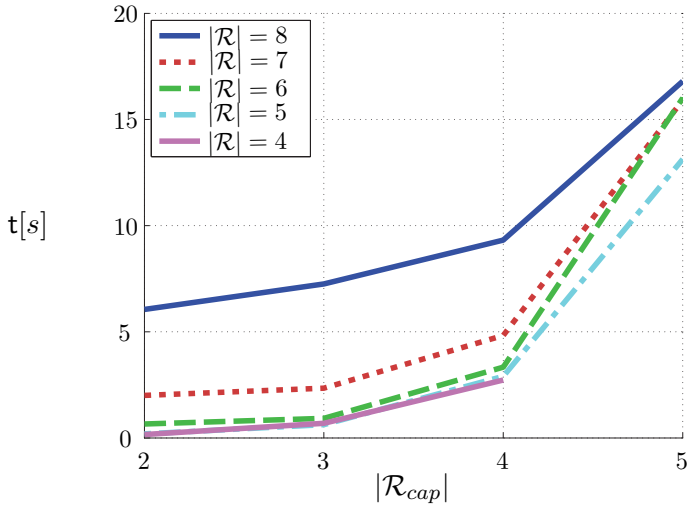


Fig. 3.8.: Experimental time t in seconds required for a MR-task assignment: $|\mathcal{R}|$ is the number of robots in the MRS and $|\mathcal{R}_{cap}|$ is the number of robots capable to perform a subtask. In this benchmark setup, all subtasks had identical capability constraints.

principle $\mathcal{K}_{Eddie} \prec \chi_{\tau_{bring}}$ is satisfied but $\hat{\varrho}(Eddie, \tau_{bring}) = -\infty \not> \varrho_{min}$. Accordingly Eddie stops τ_{bring} and reports a failure to the responsible auctioneer. The auctioneer re-evaluates $\mathcal{Z}_{cap}(\tau_{serve})$ and since $|\mathcal{Z}_{cap}(\tau_{serve})| \neq 0$, i.e. there are still capable coalitions available, it classifies the failure as *semi-recoverable* and re-announces τ_{serve}. Similarly to the run described in Section 3.6.2 this auction results in the task assignment $\langle \tau_{fetch}, \tau_{bring} \rangle \rightarrow \langle Poppy, Eddie \rangle$.

While Poppy and Eddie perform the cooperative task plan, a complete hardware failure of Eddie is forced by pushing its emergency stop button just before the handover occurs. Eddie observes the malfunction of its hardware and updates its capability set to $\mathcal{K}_{Eddie} = \{\kappa_{detect}\}$. Since $\mathcal{K}_{Eddie} \not\prec \chi_{\tau_{bring}}$ Eddie reports a failure to the auctioneer.

The auctioneer re-announces the task a third time as Poppy and Jimmy are still capable to perform τ_{fetch}, i.e. still $|\mathcal{Z}_{cap}(\tau_{serve})| \neq 0$. Since Poppy already holds the cup its reward is highest and it is assigned $\langle \tau_{fetch} \rangle \rightarrow \langle Poppy \rangle$. So Poppy performs τ_{fetch} and finally successfully completes the serve task.

3.6.4. Benchmark Evaluation

In order to illustrate the applicability of MuRoCo-D in practice, a benchmark evaluation, of the time required to determine a MR-assignment, is given. During the evaluation for each robot a separate computer[4] was used. All computers were connected via LAN. The implementation is in C++, but it needs to be noted that the code was not carefully optimized with respect to its computational efficiency.

Fig. 3.8 shows the time required for a MR-task assignment. Each curve corresponds to a MRS composed of a specific number of robots $|\mathcal{R}|$. The horizontal axis gives the number $|\mathcal{R}_{cap}|$ of robots capable to perform a subtask of the MR-task, where $\mathcal{R}_{cap} \subseteq \mathcal{R}$.

[4]AMD Phenom II X4 945, 3GHz, 4GB RAM.

Furthermore, for each possible coalition size a respective task plan was available.

Such a setup could for example correspond to a cooperative object carrying task, where each subtask corresponds to picking the object and then moving it jointly to its designated location. Accordingly only those robots are part of $\mathcal{R}_{cap}$, that are able to move as well as to pick the object. Respective task plans indicate for example how the robots should arrange in order to carry the object with two, three, four or more robots in a joint manner.

The curves show that the computational effort can be considerably reduced by heterogeneous and specialized robots. For homogeneous systems, where at least one robot is capable to perform a subtask, holds $|\mathcal{R}_{cap}| = |\mathcal{R}|$. In the current evaluation such a homogeneous system of $|\mathcal{R}| = 6$ needed already around 80 seconds what in most cases is too much for real applications. In these rather maximal respond times below ten seconds are desired.

In this regard, Fig. 3.8 clearly shows that the computational times were considerably lower the less capable robots were available. This results from the early pruning of infeasible candidates. In MRSs with higher heterogeneity commonly the ratio of capable robots is lower. Fig. 3.8 shows further that for a MRS of eight robots, a solution was found within ten seconds when only half of the robots was able to perform a subtask. In case all eight robots were capable the computations took several minutes.

In general, the curves show the practical limitations arising on customary hardware as it is used in present systems. Of course, the actual computational times are reducible by optimizing the code and/or more powerful hardware. Actually this is also to be expected to happen in the future. Moreover, in practical applications the number of task plans are usually limited. Consequently is it rather unusual that feasible solutions exist for all sizes of coalitions. In this respect, this evaluation is primarily supposed to be understood qualitatively in order to get a practical estimation of the limitations with respect to realizable problem sizes.

3.7. Summary

In order to solve the action selection problem, a robot needs to decide which sequence of actions to choose in order to complete a specific task. In order to yield a cooperative decision making in a multi-robot system (MRS), this implies a prior group agreement on which robot should execute this specific task. This is known as multi-robot task allocation problem (MRTA).

In this chapter the market-based MuRoCo framework has been introduced, which handles the MRTA problem in heterogeneous MRSs that are capable of full communication. MuRoCo yields optimal solutions for the instantaneous assignment of tasks that require a single robot to be completed (ST-SR-IA) and also for the sequential assignment of tasks that require a tight cooperation of multiple robots (ST-MR-IA), commonly known as "coalition formation". Thereby MuRoCo is, to the author's best knowledge, so far the second approach which explicitly optimizes also the subtask assignment among the coalition members. Major contributions of MuRoCo are the computational complexity of $O(2^{|\mathcal{R}|}|\mathcal{R}|^3)$ for optimal multi-robot task assignments. Moreover, the distributed version, MuRoCo-D, where the computational load is shared among the robots yielding $O(2^{|\mathcal{R}|}|\mathcal{R}|^2)$,

has been proposed. At this point in time, both versions guarantee the lowest increase of the worst-case complexity compared to previously existing work. Additionally, in order to also reduce the complexity arising on average, several pruning strategies that take the system-specific characteristics into account have been presented. Altogether MuRoCo presents an exhaustive framework that provides a generic problem formulation. This allows for a high versatility to new applications and domains. In order to cope with the omnipresent uncertainty, means for failure prevention, forecasting, handling and removal are incorporated. The suitability of the presented framework for the robust operation of a complex MRS, even under various sources of uncertainty, has been verified in a service scenario with a MRS of four heterogeneous robots. Finally, a benchmark evaluation has been given, which shows the practical limitations of MuRoCo with respect to realizable problem sizes. This is, again to the author's best knowledge, the first quantitative practical examination of such a framework.

Despite the promising advancements presented here, further improvement may be possible, by not re-calculating everything from scratch in case of a failure, but instead, re-using the result of the prior optimization. For example, in [75], a dynamic version of the Hungarian method is proposed in this respect. Also very common is to specify heuristics based on which a greedy selection is performed to find an approximative solution. In this respect, learning methods could be used to learn suitable heuristics for the specific system setup as proposed in [12]. Still part of future research remain adequate approaches to tackle the very hard problem of simultaneous assignments of multiple multi-robot tasks. Similarly, so far no approach to efficiently handle the assignment to multi-task robots (MT-MR-IA) is known. However, the relevance of multi-tasking in real applications may be limited and thus should be explicitly considered in advance. Moreover, in the MRTA literature, the cost functions are commonly assumed to be given, which in general does not hold for real world applications. An additional complicating factor, especially in MRSs, is that due to physically constrained resources, the execution of one task or action may affect the performance of a parallel-running action. This interdependence may affect the accuracy of the cost estimation, especially during tightly cooperative actions and for multi-task robots. As the validity of the solutions generated by frameworks such as MuRoCo is only given in the case of accurate cost estimates, these interdependencies need to be explicitly taken into account. This demands for further investigation of these interdependencies, which is given in the following chapter.

4. Improved Action Selection by an Uncertainty- and Situation-Aware Performance Estimation

In the preceding chapter, the MuRoCo framework has been described, which solves the action selection problem in cooperative MRSs. MuRoCo assigns tasks to the robots, which then execute actions according to specified plans. In this respect, MuRoCo determines the task assignment and selects the action plans, for which it expects the best possible performance of the MRS. However, whether this estimated performance is actually achieved during the subsequent action execution, relies on the validity of the reward estimates and thus also of the respectively estimated task and action costs. In other words, if the actual cost that arises during the action execution differs from its previously estimated value, the calculated performance of the planned solution can no longer be guaranteed. Admittedly, it is the fate, but also the challenge of robotics, that this is almost always the case. Real world systems are faced with environmental uncertainty leading to a more or less strong deviation between the outcome of planning and execution. Even though this is a quite fundamental problem of robotics, it is still often neglected in the literature. Accordingly, methods are needed that lessen the impact of this deviation on the system performance, by explicitly taking the system sensitivity on the environmental uncertainty into account.

This chapter contributes in this respect by introducing a novel bipartite approach that provides the generic means to learn the discrepancy of planning and execution for a robotic system, and thereafter, improve the action selection by usage of the gained knowledge. More specifically, a system interdependence model is learned from experimental data, which enables the qualitative, as well as, quantitative inference of the influence that environmental factors have on the system behavior. This knowledge is then used during the system operation for a situation- and uncertainty-aware cost estimation which takes potential risks already during the planning phase into account. Initial estimates are further improved by information gathered during the execution. This allows for possible counteractions in order to achieve a better consistency of planning and execution, and thus an overall improvement of the system reliability. This is also verified by experimental evaluations of the proposed approach.

4.1. Introduction

Autonomous robotic systems typically act in partially known or unknown environments where they are constantly faced with situations that require decision making capabilities under perceptual uncertainty. This uncertainty can lead to undesired system behavior.

However, the respective consequential series of internal reactions that cause the observed behavior is often unclear, since it results from the interaction of various system components. As a consequence, in order to ensure robustness and reliability of such autonomous robots, it is of high interest to identify the crucial environmental and system component indicators that reflect the overall system behavior. The identification of their mutual interdependencies allows to draw conclusions about the influence of these indicators on the system behavior. Such knowledge is for example valuable for design choices, since the factors that contribute most to the variability of the system behavior can be identified and it can be determined if these require additional research to strengthen system robustness. Additionally, this information can be used to enable autonomous systems to avoid failures by predicting the effects of actions and to correctly adjust their behavior.

With respect to action selection, the effects caused by the existing uncertainty are observable in the difference between the estimated and the arising action cost. In case of failure they are additionally identifiable in the non-obtainment of the expected utility. The causes of these effects are ascribable to various influencing factors, which can be classified according to their extrinsic or intrinsic cause. Extrinsic factors are related to the surrounding environment and are not directly controllable by the robot itself. Intrinsic factors instead are primarily induced by the internal robotic system or the effects of its own actions.

A major result of the extrinsic factors, occurring in most robotic systems, is perceptual noise. Possible causes of environmental uncertainty are manifold and comprise perceptual noise, unexpected changes by actions of other robots or humans, and lack of knowledge for example. The respective intensity is determined by the specific environmental conditions. For example the scenario shown in Fig. 4.1 implicates various sources of perceptual uncertainty. The robot in the lower left corner aims to estimate its cost in order to fetch the toy helicopter from the living room on the right side. Assume the robot retrieves its current position estimate from a localization module that fuses odometry data with landmark positions retrieved from a vision-based detection such as in [15]. In a well-lit corridor the localization module is able to precisely detect the landmarks in the environment and to generate accurate position estimates. In a dimly lit corridor instead, the localization module often erroneously detects landmarks or even finds no landmark at all. Additionally, in a corridor with rough concrete floor the odometry data is less accurate than in a corridor covered with smooth laminate. This results in erratic position estimates which may lead to the robot loosing its way and thus to higher cost.

Another type of extrinsic factors results from partial knowledge about the environment. In principle, partial knowledge may be also classified as intrinsic type since the robot can increase or even obtain full knowledge by exploration of the environment. However, this holds only for static environments. In a dynamic environment the permanent changes force the robot to always question whether its knowledge is still up-to-date. Other robots or humans may have locked a door, switched the light, moved furniture or are currently blocking a corridor. All these factors may influence the cost of a specific task. For example, in case the robot has incomplete knowledge about the map of the environment, as indicated in the bottom right corner of Fig. 4.1, it may discover a more efficient path while its navigating along the initial one. Similarly, it might discover that someone locked a door and

Fig. 4.1.: Exemplary scene with various sources of uncertainty.

the planned path is blocked. These cases would lead to a lower or higher cost respectively, both resulting in a deviation from the prior estimate.

Besides such extrinsic factors there are further causes of uncertainty that can be ascribed to the robotic system itself. A major aspect of robots acting in the physical world is the achieved level of robustness. Even though, the primary causes of errors are usually of extrinsic type, such as perceptual noise, a robotic system can be characterized by its ability to cope with these extrinsic factors, i.e. its proneness to failures. Consider for example the previously mentioned implicit dependency of an efficient goal-oriented navigation on the given illumination and floor covering. A robot that has an internal representation of this interrelation and furthermore has some information about the current lighting conditions and/or floor covering, can incorporate this knowledge already during planning to yield a more accurate cost estimate and thus a better decision. Another practical example are failure-prone actions that are likely to require multiple trials, such as dexterous manipulation tasks. In case the robot knows that grasping the fragile helicopter will take most likely several attempts, it can already incorporate this into the prior cost estimation, e.g. in form of an expected success rate.

Especially in the case of a cooperative multi-robot system (MRS), where the robots aim to find a jointly optimal action policy, cost discrepancies may affect the action choices of all robots and thus lead to sub-optimal decisions. For example assuming that fetching the helicopter requires a joint manipulation by two robots and one of these robots needs to

move a far longer distance as the other one. In this case the latter may decide whether to wait or whether to execute another task in between. However, both decisions bare the risk of being sub-optimal in case the first robot is significantly slower or faster as expected respectively.

As a consequence, in order to take these instances into account and to obtain a reliable and robust action selection, respective means for a more accurate cost estimation and a situation-aware failure forecasting are required. These means need to first identify the relevant influencing factors, second also determine the respective strength of those influence, and thereafter incorporate such knowledge into the robotic decision making.

The remainder of this chapter presents solutions to these problems. First an overview of related work is given in Section 4.2, followed by the problem formulation in Section 4.3. Thereafter Section 4.4 describes an approach to identify the interdependency among a set of selected factors and in Section 4.5 this knowledge is used for a more reliable and risk-aware cost estimation. Finally the chapter is summarized in Section 4.6.

4.2. Related Work

A finding of [36] is the cost or utility metric to be one of the crucial parameters in an efficient multi-robot task allocation. Actually this can be generalized to any type of robotic decision making, as the metric specifies whether a solution is better as some alternative or not and in this respect strongly determines the resulting behavior of the system.

The existence of literature focusing on the performance evaluation of autonomous mobile robots indicates the importance of the topic. Benchmark scenarios, such as the DARPA Grand Challenge [24], RoboCupSoccer [109] or RoboCup@Home [97], are a way of directly and objectively comparing the performance of robotic systems. A similar approach which additionally gives reproducible performance results is using standardized testbeds in both simulation and reality [53]. A drawback of benchmark scenarios is the scenario-dependence, which does not allow to compare robotic systems applied in different scenarios. For example, it is not possible to compare a robot which was built to operate in a home environment [111] with autonomous vehicles, which are supposed to navigate through an urban environment [119]. Actually, the scenario-dependence is so strong, that the winning vehicle of the first DARPA Grand Challenge [116] would not be able to take part in the second challenge, since the scenario changed from the desert to an urban environment. Standardized benchmark scenarios also bear the risk of developing robotic systems and algorithms specialized towards these scenarios at the cost of generality.

Scenario-independent criteria for the qualitative evaluation of robotic systems have been proposed in [22]. These approaches focus on task objective and social measures to identify both, the efficiency of robot and human. So performance may be also determined by a functional mixture of multiple metrics. For example in [50] an evaluation framework for characterizing the autonomy of unmanned vehicles by considering mission complexity, environmental difficulty and HRI is presented.

Nevertheless, it is not only crucial which specific metric is used but also how accurate it can be determined. Its value needs to be calculated based on the current state which in turn is estimated from the robot's uncertain sensor data. Consequently this sensor uncertainty

directly influences the accuracy of the computed cost and thus the quality of the made decisions. In [67] the authors even argue that robot performance is not measurable in terms of optimality due to the environmental uncertainty that allows no testing under identical circumstances. Indeed, in current robotics research this cost variability is an often neglected problem. Most of the work either assumes some cost function to be given or handles those variability by frequent replanning. However, for more complex real world applications a cost function that also takes the current environmental situation into account is in general not given and frequent replanning is only possible in a limited scope.

In this respect, other approaches introduce quantitative metrics that also reflect the influence of the environment on the robot performance. For example in [80] several metrics are proposed to characterize the path quality during navigation missions. The entropy and the compressibility of the environmental information are used in [1] to estimate the complexity of an environment. The latter is in [65] combined with task performance in order to evaluate the degree of autonomy of a robot. In [71] the emergent behavior of a homogeneous MRS is modeled with a probability distribution function (pdf) over all states of the entire MRS including the task states. The motivation is to derive a probabilistic model that describes how the behavior of a robot changes over time. The authors of [71] state that from such a pdf the average behavior as well as its variation can be derived. Knowledge about this variation enables to assess the probabilistic influence of the robot's observations and actions on the global performance of the system.

In this context, the authors of [16] identified the importance of self-diagnostics in algorithms, which enables the consuming algorithm within a cognitive system to reason about the quality of the retrieved information. This raises the general question in which way the variation in the model input influences the variation in the model output. Exactly this question is addressed by sensitivity analysis, which aims not to identify the cause for the current output but rather what leads to its variation. A good introduction into the field, which can be classified into local and global analysis, is given in [101]. While the local analysis evaluates the output variation resulting from the small change of a specific input variable, the global analysis examines the output variation when all input factors are changed simultaneously. In this respect the global analysis enables also to identify which input elements are the most influential in inducing the uncertainty in the output [85].

In consideration of robotic systems, the global analysis is of major importance as in real world settings it is in general impossible to keep all parameters identical expect a single factor of interest. Also, partial studies of various sets of factors are not applicable, since e.g. a system may be insensitive to each specific set on its own but sensitive to the simultaneous changes of all.

In some simple cases, the global system sensitivity can be solved analytically. For example differential analysis can give useful results at least as long as the variations are small. However, stochastic models are usually too complex for an exact analytic treatment [71, 85, 101]. In these cases, probabilistic methods provide powerful alternatives such as the multivariate analysis of variance (MANOVA) or least-square estimators such as regression models. However, the MANOVA is often not applicable as it assumes normal distributions and homoscedasticity, i.e. a homogeneous variance in the input variables. Regression methods assume a linear contribution of independent input variables to the

output. In principle there are also means to handle non-linear dependencies, but this requires a prior model identification of all main effects and interactions, which is commonly not straightforward and thus impractical for complex problems.

In [85] a Bayesian approach for probabilistic sensitivity analysis is proposed. Its major advantages are that it is computationally highly efficient as it requires only a small number of runs and further that all measures of interest are derivable from a single set of runs. Bayesian methods are also applied in [46], where the relation between the environment and the performance of a robotic system is learned by using a Dynamic Bayesian Network. This way the coherence among the metrics and also the environment is identified.

Such models allow for a situation-aware estimation of the metric of interest. However, as mentioned in the beginning of this section, besides the actual value of an estimate also its related accuracy is of huge importance, as it determines the risk of a large deviation that may lead to highly sub-optimal decisions. In [13] sensitivity analysis is applied to probabilistic belief systems. It is analyzed how local changes in the parameters of a belief model influence the global belief. This can be for example used to identify the sensitivity of decisions with respect to specific model parameters or to determine how input parameters need to be changed in order to ensure a desired outcome. Here instead, the global belief change in general needs to be investigated, as in robotic setups all parameters may be subject to simultaneous variations. Moreover, the (sensor) input is commonly not directly controllable. In this respect, most of the previously described methods use the expected value for the cost estimation as it provides the best prediction in terms of minimizing the squared error. However, this may be problematic in case of asymmetric or very flat distributions as it provides only a very approximative representation of a pdf.

An alternative to the expected value that also takes these criteria into account is the quantile function. In [60] quantile regression functions are described, which are an estimation of conditional quantile functions that are expressed as an additive mixture of the observed covariates. Similar to the quantiles providing a better representation of a pdf compared to its expected value, the quantile regression provides a better representation as its mean-based version. In [58] an approach is presented to estimate a quantile mixture based on L-moments, which are – in analogy to the moments of a pdf – representations of the model parameters. The major advantage of a quantile-based cost estimation, in comparison to a mean-based one, is that it also takes one-sided deviations into account. In some situations a cost underestimation may have a much stronger impact on the system performance as an overestimation. In these cases quantiles enable to derive a metric estimation that explicitly considers such a system-specific risk. This leads to the concept of risk aversion, i.e. the attitude to prefer a more reliable option even though it is expected to be worse than a risky alternative [3]. This may seem illogical in the first place. For example an expected-value optimizer will act risk neutral. However, in economics it has been shown that humans often act in a risk averse way, meaning they choose the more reliable option. Consequently, as robots are supposed to be helpers and assistants for humans, risk aversion may also be a desirable behavior for robots. The tradeoff between risk and reward is also an important topic in the financial optimization of asset portfolios, see e.g. [61]. However, the methods for portfolio optimization are not directly transferable as their core problem is to optimize the composition of the individual assets contained in the

portfolio. In robotic action selection instead, the composition is already predetermined in form of the dedicated tasks and the respective action plans. Instead in robotics the problem is to find reliable risk and reward estimates and an efficient tradeoff among these two.

An additional robotic-specific challenge is the relatively huge input uncertainty. This necessitates tools that offer the possibility to make use of as much prior knowledge as possible, while also remaining computationally tractable in case no information is given at all. Furthermore, they need to get along with few data, as this is often hard to gather in robotic systems, while still taking possible non-linearities into account. In the following a probabilistic approach is presented that unifies all these demands and provides a mean for a situation and risk-aware cost estimation. Before a formulation of the actual problem in focus is given. An additional robotic-specific challenge is the comparably huge input uncertainty. This demands for tools that offer the possibility to make use of as much of prior knowledge as possible, while also remaining computationally tractable in case no information is given at all. Furthermore they need to get along with few data, as this is often hard to gather in robotic systems, while still taking possible non-linearities into account. In the following a probabilistic approach is presented that unifies all these demands and provides a mean for a situation and risk-aware cost estimation. Next a formulation of the actual problem in focus is given.

4.3. Problem Definition

In the preceding chapter procedures are described, which aim to find the action policy, in form of a task assignment, that yields the best possible performance. According to (2.7) (p. 23) the performance q of a cooperative MRS is determined by

$$q = \frac{1}{\beta_u} \sum_{j=1}^{m_c} \left(u(p_{\mathcal{A}}(\tau_j)) - c(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \right) = \frac{1}{\beta_u} \sum_{j=1}^{m_c} \varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \leq 1, \tag{4.1}$$

i.e. the ratio of the rewards $\varrho(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \in \mathbb{R}$ obtained for the completed tasks τ_j, $j = \{1, \ldots, m_c\}$, relative to the sum of all obtainable utilities β_u. The reward for executing a task τ by following an action plan $p_{\mathcal{A}}(\tau)$, see (2.4) (p. 21), results w.r.t. (2.6) from the difference between obtained utility $u(p_{\mathcal{A}}(\tau)) \in \mathbb{R}_0^+$ and arising cost $c(p_{\mathcal{A}}(\tau), \boldsymbol{s}) \in \mathbb{R}_0^+$. As stated in Section 2.2 (p. 21), the utility $u(p_{\mathcal{A}}(\tau))$ is assumed to be determined by the application, the designer or the planning layer and is obtained for the completion of the task τ according to the specific plan $p_{\mathcal{A}}(\tau)$, i.e. independent of the actual execution of task τ. In contrast, the related cost is also dependent on the environment state $\boldsymbol{s} \in \mathcal{S}$ and thus is influenced by various situation-dependent factors. In this respect $c(p_{\mathcal{A}}(\tau), \boldsymbol{s})$ reflects the quality of the specific execution of the plan $p_{\mathcal{A}}(\tau)$. The environment state $\boldsymbol{s}$ is a symbolic representation of the world status and $\mathcal{S}$ is the set of all known states.

As in general the process of action selection takes place prior to the action execution, the actual occurring cost $c(p_{\mathcal{A}}(\tau), \boldsymbol{s})$ is at the moment of planning still unknown and therefore needs to be estimated. From (4.1) follows that the accuracy of a cost estimate $\hat{c}(p_{\mathcal{A}}(\tau), \boldsymbol{s})$

directly determines the accuracy of the respective reward estimate

$$\hat{\varrho}(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) = u(p_{\mathcal{A}}(\tau_j)) - \hat{c}(p_{\mathcal{A}}(\tau_j), \boldsymbol{s}_j) \tag{4.2}$$

and performance estimate $\hat{q}$ and thus also determines the validity of an action policy ϖ, which corresponds to a mapping from states to actions, $\varpi : \mathcal{S} \to \mathcal{A}$, where Π is the set of all possible policies. $\mathcal{A}$ is the set of all actions a. As a consequence of (4.2) can the optimal policy, according to (2.8) (p. 23), only be estimated:

$$\hat{\varpi}^* := \arg\max_{\varpi \in \Pi} \hat{q}(\varpi). \tag{4.3}$$

This corresponds to a solution of the alleviated form given in (2.10) (p. 24). Accordingly, in order to get as close as possible to the optimal solution, an accurate model of the underlying cost function, which takes the influencing situation-dependent factors into account, may be as important for an efficient cooperation as the planning framework described in the previous chapter.

In other words, a functional relationship $\hat{c}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ is needed, which enables to infer a cost estimate for an action plan $p_{\mathcal{A}}(\tau)$ based on the current environment state $\boldsymbol{s}$. In case a deterministic model is available, $\hat{c}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ can be derived analytically. However, especially in robotics the given uncertainty is usually the result of a complex interplay of various, often unknown, causes and thus commonly hard to predict analytically. An alternative is to model the relationships probabilistically. More specifically, if the conditional pdf $\mathsf{f}_c(c(a) \mid \boldsymbol{s})$ about the action costs $c(a)$ is known, it provides a basis to obtain a probabilistic estimate $\hat{c}(a \mid \boldsymbol{s})$ of the cost $c(a \mid \boldsymbol{s})$ for action a given $\boldsymbol{s}$. However, in general the knowledge about the pdf is not given a priori and even in case it is available, it needs to be taken into account that respectively derived estimates are subject to a stochastic process and thus may deviate to a greater or lesser extent from the actual outcome. This difference is defined as cost deviation

$$\Delta c(a \mid \boldsymbol{s}) := \hat{c}(a \mid \boldsymbol{s}) - c(a \mid \boldsymbol{s}). \tag{4.4}$$

and

$$\Delta \varrho(p_{\mathcal{A}}(\tau), \boldsymbol{s}) := \hat{\varrho}(p_{\mathcal{A}}(\tau), \boldsymbol{s}) - \varrho(p_{\mathcal{A}}(\tau), \boldsymbol{s}) \tag{4.5}$$

is the reward deviation respectively. In order to find a persistent solution to (4.3), i.e. a solution that has a high chance to keep its validity throughout and even after its execution, the following aspects need to be adequately considered for an efficient action selection:

(i) Determination of the factors that constitute $\boldsymbol{s}$.

(ii) Derivation of the cost pdf $\mathsf{f}_c(c(a) \mid \boldsymbol{s})$.

(iii) A risk-aware estimation of $\hat{c}(a \mid \boldsymbol{s}')$ that takes the characteristics of $\mathsf{f}_c(c \mid \boldsymbol{s}')$ into account, where $\boldsymbol{s}'$ is the known and/or observable part of $\boldsymbol{s}$.

(iv) A failure-forecasting by observation of the cost deviation $\Delta c(a \mid \boldsymbol{s})$ and the ability of adequate counteractions.

A bipartite approach that handles these aspects for a robotic system is presented in the subsequent parts of this chapter. Section 4.4 describes a method that solves (i) and (ii) by learning a functional relationship of the costs and the environment based on the past experience. This learned model knowledge is then used in Section 4.5 to derive an uncertainty- and risk-aware cost estimation (iii) and obtain its subsequent reevaluation (iv), in order to achieve an increase in performance and reliability.

4.4. Learning System Interdependence Models

When predicting the performance of systems that are prone to environmental uncertainty, the latter needs to be explicitly taken into account in order to obtain a reliable estimate. In this respect it is first essential to identify the situation-dependent factors that are crucial for the performance of a given task. Second, since the latter is determined by the interplay of all these factors and the system components it is of further interest to what extent they are affected by each other. Such knowledge is not only valuable for developers in order to improve their systems, but can also be directly utilized by the systems themselves, e.g. to detect failures and thereby correctly adjusting the system behavior.

In the following a method for a system interdependence analysis is presented. The basic idea is to learn and quantitatively evaluate the coherence between performance indicators of different system components, as well as the influence of environmental parameters on the system. The presented analysis is exemplified by the application of the presented method on the navigation system of the Autonomous City Explorer (*ACE*) robot. In Section 4.4.2 its navigational methods are presented and suitable indicators are derived. Results of the method, which is performed based on experimental data from an extended field experiment, are given in Section 4.4.3.

4.4.1. System Interdependence Analysis

Most of the current autonomous robots are complex systems designed for specific applications. They usually consist of various components, which commonly can be separated into three categories according to their purpose. Perceptual components are responsible for building an environment representation, e.g. in form of a map, and also for localizing the robot. This representation is consequently used by planning and action selection components to calculate a plan of actions such as the trajectory of the robot. In the following planning is used as collective term for the scheduling and action selection layer of the architecture in Fig. 2.1 (p. 18). Finally, the chosen plan is executed and the progress is monitored by task execution components. It is obvious that sensing, planning and execution are interconnected. Although performance indicators have been proposed for each of these domains, it is still hard to assess the effect that environmental parameters, or variations of the performance of specific system components, have on the rest of the system.

The system interdependence analysis described next provides a generic method to tackle this problem. A probabilistic model of the interdependencies between system components, such as perception, planning and execution, is learned, which provides a mathematical

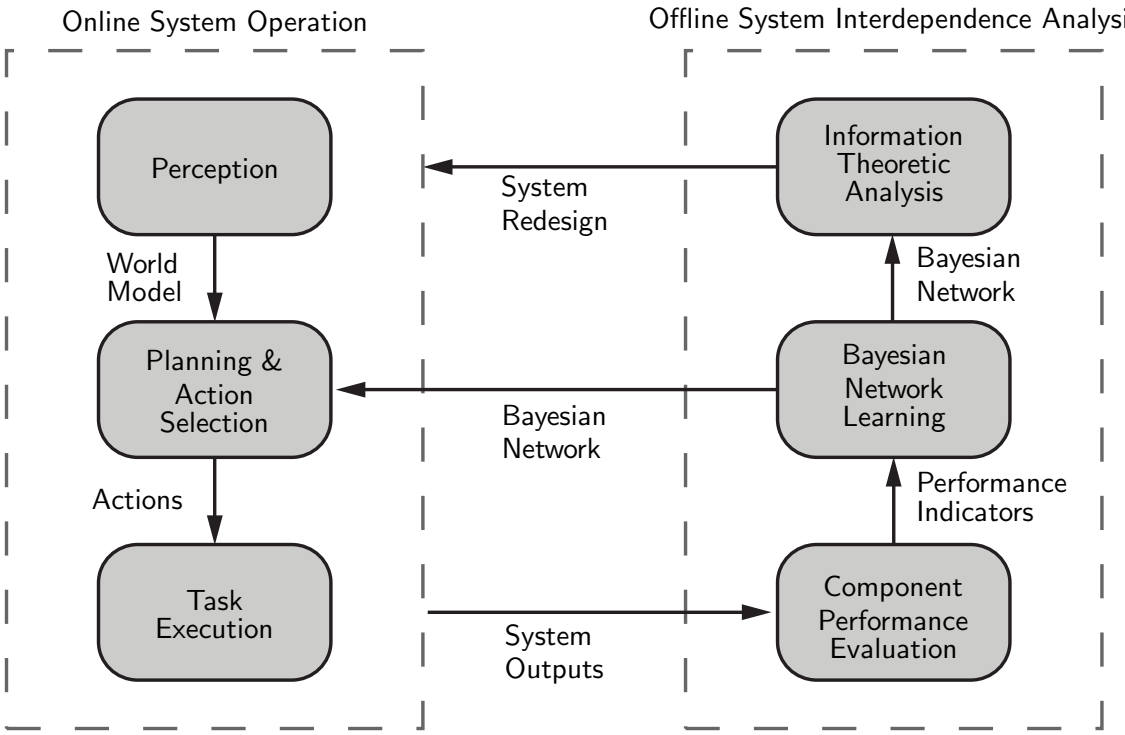


Fig. 4.2.: Scheme of the proposed system interdependence analysis.

system model that enables to determine the crucial components with respect to robustness within a system. In principle such a model can be also derived by examining the deterministic interdependencies within a system. However, with increasing system complexity, i.e. more components and higher interconnectivity among those, the derivation of a respective deterministic model becomes harder. In this regard, a learned model facilitates design choices without the need of fully examining the deterministic interrelations in the system. It also provides a mean to verify existing deterministic system models and to identify relations which have not been modeled. Furthermore the gained knowledge can be integrated into the online reasoning process of the system itself to enhance its autonomy. The presented analysis is applicable to any robotic system for which the components of interest are observable. For these it identifies the interdependence model that explains the data, retrieved from the system, best.

4.4.1.1. Algorithm Overview

The proposed approach is illustrated in Fig. 4.2. As the robot operates, system outputs are monitored and performance indicators for the system components are calculated. The indicator values are used offline to learn the structure of a Bayesian Network (BN), which reflects the gathered system data best, and train its parameters. Bayesian Networks are a network-based framework for representing and analyzing models involving uncertainty. They find application in several fields ranging from intelligent decision support aids, to data fusion, 3D feature recognition, intelligent diagnostic aids, automated free text understanding and data mining.

The learned BN structure identifies the coherence between the performance indicators

computed from the system output, i.e. to which extent the indicators are associated with each other. In order to quantitatively evaluate this coherence between indicators from different system parts, information-theoretic analysis is performed on the parameters of the learned BN. The acquired quantitative relation supports the developer to focus further redesign efforts and the learned model can be used to adjust the online functionality of the robot to the current situation, as discussed in Section 4.5. In the following the determination of indicators, the acquisition of respective data, the structure learning and the information-theoretic analysis are described.

4.4.1.2. Component Performance Evaluation in Autonomous Robotic Systems

Before a respective model can be learned a set of component indicators needs to be specified, which reflect the state of the components of the autonomous robot as well as its environment. Therefore, for each component of interest at least one representative indicator needs to be chosen. In principle, these indicators can be arbitrarily chosen by the system designer, for example based on design knowledge or experience, thus there is no fixed set of universal indicators. The choice should be rather based on the properties of the system in question. In general, a good starting point is to select indicators that reflect the performance of the components.

In literature, suitable performance measures have been established for various methods and problems. For example, planning algorithms are commonly measured with respect to solution quality or required time, such as [68, 98, 120]. For perception algorithms instead typically statistical criteria are used, e.g. [16, 99].

Nevertheless, the indicator selection is not crucial for the performance of the analysis, since inappropriate indicators are identified as they have no or only weak interdependencies to the other indicators. This means that these indicators provide no information about the part of the system represented by the other indicators, but of course they might be still relevant for the analysis of the respective component itself.

When the set of indicators is defined, the respective system output data, which is required to compute the indicator values, needs to be gathered. This is simply done through various experimental runs under respective environmental conditions. During these runs, all system data of interest is recorded by sampling with a fixed rate. The latter is assumed to be the same for all data of interest and should be chosen with respect to the dynamics of the system and the environment. Thereafter the gathered data is processed offline and the indicator values, which provide the basis for the model learning, are derived.

Before the model can be learned, the indicator data needs to be discretized. The number and size of the intervals used for the discretization of the indicator values should not be too small, in order to maintain the contained information. However, in case they are set too large, the respective probability distributions become too flat what makes the determination of mutual interdependencies hard as well. This similarly applies to the rate of the sampling as it also presents a discretization, just in the temporal domain. A possible solution to select the intervals is for example to use an entropy-based approach, such as [17].

4.4.1.3. Learning Bayesian Network Structures

As discussed previously, in order to find out whether and to what extent performance indicators of system components interact with each other, a Bayesian Network (BN) is learned from system outputs and respective performance indicator values. In general the topology of the network is unknown beforehand, but it is assumed that the system is fully observable by the data. In order to find the network structure that models the data best, a search through the space of possible structures is performed using a likelihood heuristic.

BNs are well-established tools for representing uncertain relations between several random variables [100]. They demonstrate several advantages over other knowledge representation and probabilistic analysis tools by providing a simple way to visualize the structure of a probabilistic model. This structure can be used to design and motivate new models. Also insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph. Uncertainty is handled in a mathematically rigorous yet efficient and simple way, by using Bayesian statistics. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

A BN is an annotated directed acyclic graph, that encodes a joint probability distribution over the set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of random variables. Formally it is a tuple $\langle \mathcal{G}, \mathcal{E} \rangle$, where $\mathcal{G}$ is a Directed Acyclic Graph (DAG) whose vertices correspond to the random variables. A DAG implies conditional independence of each variable x_i and its non-descendants, given its set of parents $\mathcal{X}_{Pa}(x_i)$. $\mathcal{E}$ represents the set of parameters that define the transition between nodes. It contains a value $e_{i,j,k} = P(x_i = k_i | \mathcal{X}_{Pa}(x_i) = j_i)$ for each possible value k_i of x_i and each possible set of values j_i of $\mathcal{X}_{Pa}(x_i)$. The conditional probability distribution of each node is represented in a Conditional Probability Table (CPT).

In case there is no a priori transition information available, the space of possible DAGs is super-exponential in n, the number of variables described, and is given according to [96] by

$$|\mathcal{G}(n)| = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} |\mathcal{G}(n-k)|. \tag{4.6}$$

Fig. 4.3 shows the rapid growth of $|\mathcal{G}(n)|$ with increasing n, what illustrates that an exhaustive search gets already problematic for less than ten indicators. Therefore tools are required to reduce the amount of examined graph structures to a tractable number while still allowing for good solutions. A widely used possibility are sampling-based methods, such as the Markov Chain Monte Carlo (MCMC) [82] search. MCMC takes randomly sample structures from the space of possible DAGs and evaluates them. The number of samples is chosen large enough for the search to converge. As convergence indicator for the search the acceptance ratio is used. This is the fraction of proposed samples with likelihood, which is accepted by the approximation algorithm, divided by the samples that are rejected. Even though there is no guarantee that MCMC will find the optimal solution, its major benefits are the controllable complexity and that it does not get stuck in local optima.

As scoring function, to evaluate the structures, the Bayesian Information Criterion

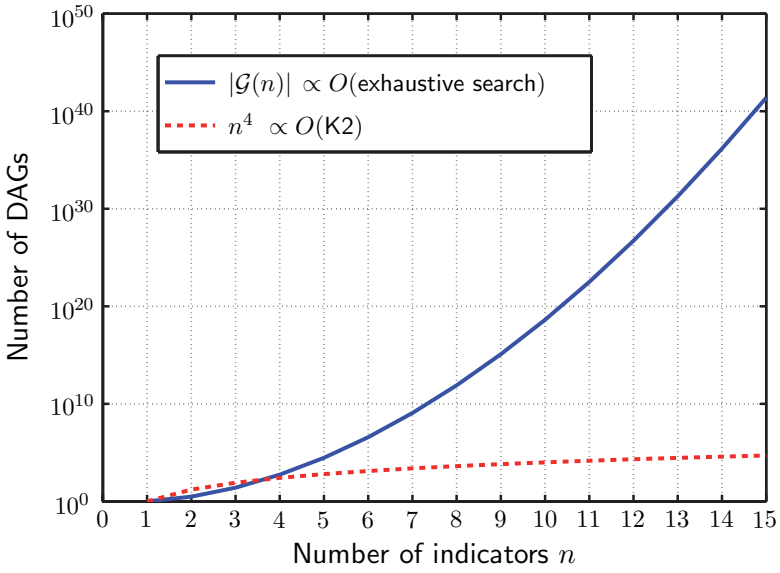


Fig. 4.3.: Course of complexity with an increasing number n of indicators: for an exhaustive and for the K2 search.

(BIC) [104] is used, which is a function of the log likelihood of the structure according to the training data penalized by the complexity of the structure.

In case a sequential prioritisation of the indicators is available, an alternative search algorithm is applicable: the well-established local greedy search algorithm K2 [20]. The prioritisation implies available information that subsequent nodes are supposed to be dependent on its preceding nodes and is used to initialize the K2 search. The search starts from an empty set of nodes. Parents are added incrementally – according to the prioritisation – and the one whose addition increases the score of the resulting structure most, is kept. The maximum number of parents can be constrained by an upperbound. The algorithm stops adding parents to the node, when it is no longer possible to increase the BIC score of the structure. In the worst case, i.e. without upperbound, the complexity of K2 is $O(n^4)$. K2 is beneficial in the sense of its comparably low complexity, as shown in Fig. 4.3. The major drawback is the requirement of an initial node prioritisation, which is not available in general and also biases the solution. For a robotic system, the prioritisation could be retrieved from information about the sequential structure or causal relation of the indicators, e.g. deduced from the system design. Alternatively the node prioritisation can be also acquired from the solution of the MCMC search – by placing parents first and children subsequently – to further improve the result of the latter.

The presented search algorithms have been chosen due to their good tradeoff between complexity and solution quality. Furthermore, in combination they require no initial information. Of course, a very large amount of search methods that might be applicable is available in the literature. The possible benefits of using other methods remain to be investigated.

Resulting from the search, the structure with highest BIC score is used for further analysis. It provides a first qualitative view on the mutual interactions among the indicators.

In the following, information-theoretic criteria are used to evaluate the coherence between the indicators within the learned network.

4.4.1.4. Information-Theoretic Criteria

The BN structure itself does not provide a quantitative measure of the indicator interdependence. In order to derive the latter information-theoretic criteria [21] are applied. Once the structure of the net is learned, the CPTs can be computed by using the experimental data. For each pair of indicators x_i, x_j, the mutual information

$$I(x_i, x_j) = \sum_j P(j) \sum_{k_i} P(k_i|j) \log \frac{P(k_i|j)}{P(k_i)} \tag{4.7}$$

is derived. Intuitively, mutual information measures the information that x_i and x_j share, i.e. to what extent knowledge about the one of these variables reduces the uncertainty about the other. For instance, if two variables are independent then knowledge about one of them does not give any information about the other. Consequently their mutual information is zero.

In order to make comparisons between different pairs of variables a distance metric is required. In this respect the joint entropy

$$H(x_i, x_j) = H(x_i|x_j) + H(x_j) \tag{4.8}$$

is additionally calculated, where $H(x) = -\sum_{k \in x} P(x) \log P(x)$ is the entropy of the random variable x. The joint entropy measures the overall uncertainty about the two variables. The final distance metric is then derived by

$$0 \leq \eta(x_i, x_j) = \frac{I(x_i, x_j)}{H(x_i, x_j)} \leq 1, \tag{4.9}$$

which corresponds to the ratio of mutual information and joint entropy. The relation of $I(x, y)$ and $H(x, y)$ is illustrated in Fig. 4.4. It can be proven [21] that η satisfies all properties of a metric such as the triangle inequality, non-negativity and symmetry. If two variables are independent then $\eta(x_i, x_j) = 0$, whereas when the variables are fully dependent and knowledge about the one completely reduces the uncertainty about the other $\eta(x_i, x_j) = 1$.

By computation of η the interdependence between any pair within a set of indicators is determinable, no matter whether there exists a direct connection in the BN or not.

The usage of the method is demonstrated by its exemplary application to the *ACE* robot, for which a performance indicator set is determined in the next section.

4.4.2. A Case Study on the ACE Robot

We highlight the performance of the system interdependence analysis by its exemplary application to the navigation system of the *ACE* mobile robot, which is described in detail in Appendix B (p. 119). During the field experiments the robot was faced with

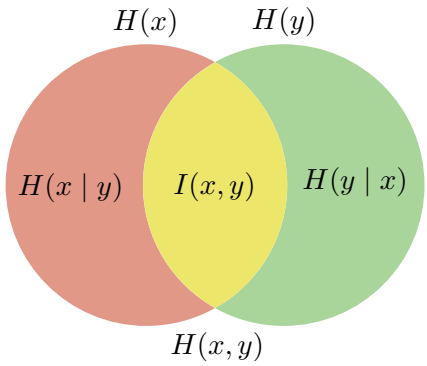


Fig. 4.4.: The relation between mutual information $I(x, y)$ and joint entropy $H(x, y)$.

various difficult situations, where it had to navigate through crowded and narrow places. These environmental conditions had an observable influence on the robotic motion, such as more frequent turns or stops for example. By applying the method of Section 4.4.1 the influence on the navigation system of the robot can be identified. In the following adequate indicators for its navigation system are determined. Thereby a system structure typical for most autonomous robots is used. System components are separated in perception, planning and execution components.

4.4.2.1. Indicators for Perception Performance

In order to navigate safely to a certain goal, the *ACE* robot must be capable of localizing itself, generating a representation of the environment and finding a drivable path through it. This section describes the approaches used for Simultaneous Localization and Mapping (SLAM).

Within the *ACE* project a grid-based approach that makes use of particle filters has been chosen, in order to approach the SLAM problem. Particle filters allow the approximation of arbitrary probability distributions, making them more robust to unpredicted events such as small collisions, which often occur especially in outdoor environments and cannot be modeled. Furthermore, grid-based SLAM does not rely on predefined feature extractors, which are dependent on the assumption that the environment exhibits a known structure. Therefore, in cluttered outdoor environments the grid-based approach provides a more robust and accurate mapping. More details on the SLAM implementation, which was deployed on the robot, can be found in Appendix B (p. 119).

The most likely occupancy grid map m^b of the environment is acquired by the SLAM module. In order to integrate traversability information, such as detected curbs, the grid m^b is fused with the grid m^n retrieved from the traversability assessment [130], to obtain the combined 2.5D grid m^c. The resulting 2.5D grid m^c is sent to the path planning module, which this way considers the obstacles from the SLAM module and non-traversable regions detected by the inclined laser range finder for path planning.

Perceptual indicators describe the uncertainty of the robot about its position and its environment model. In case of a mobile robot such a model is commonly represented by a map. Map uncertainty can be measured by the entropy H^m of the map. For the case of

an occupancy grid m this is given as in [112] by

$$H^m = -\rho^2 \sum_{g \in m} -P(g)\log P(g) + (1 - P(g))\log P(1 - P(g)), \tag{4.10}$$

where g is a cell, $P(g)$ the occupancy probability of g and ρ the resolution of m.

Pose uncertainty

$$H^P = H(P(\mathcal{X}_t|\mathcal{O}_t, \mathcal{U}_t)) \approx \frac{1}{t} \sum_{j=1}^{t} H(P(\mathcal{X}_t|\mathcal{O}_t, \mathcal{U}_t)), \tag{4.11}$$

is given as an average over the uncertainty of the different poses along the path as proposed in [99].

Finally, map information $\mathrm{INFO}(m_{t-1} \| m_t)$ has been proposed in [46] as a measure of the local complexity of a map. It is defined as the relative entropy of m_{t-1} with respect to m_t, where m_t is the local map at time t. The local map m_t is extracted from the occupancy grid m, by taking an area $10\,\mathrm{m} \times 10\,\mathrm{m}$ around the robot. m_{t-1} is the spatially corresponding part of the respective map at the previous time step. The relative entropy

$$D_g(P_{t-1}(g) \| P_t(g)) = P_{t-1}(g) \times \log \frac{P_{t-1}(g)}{P_t(g)}, \tag{4.12}$$

for cell g is also known as Kullback-Leibler divergence. By taking the sum of the symmetric form

$$\mathrm{info}_g(m_{t-1} \| m_t) = \frac{D_g(P_{t-1}(g) \| P_t(g)) + D_g(P_t(g) \| P_{t-1}(g))}{2}, \tag{4.13}$$

the relative quantity of information around the robot

$$\mathrm{INFO}(m_{t-1} \| m_t) = \frac{1}{\beta} \sum_{g \in m_t} \mathrm{info}_g(m_{t-1} \| m_t) \tag{4.14}$$

is derived similar to [46], where β is a normalization factor.

With (4.10), (4.11) and (4.14) three indicators for the perception modules of *ACE* are determined. The next part describes the method and respective indicators for the planning module.

4.4.2.2. Indicators for Planning Performance

The planning and action selection components of a robotic system are responsible for reasoning about the appropriate actions to be taken next. In case of a mobile robot, a path planning module is needed, which generates safe paths to a specified goal location. The path planner of the *ACE* robot is described in Appendix B.3.3.

In order to assess the quality of a path planning module, its generated paths are examined with regard to several quantitative indicators. Below a set of indicators is proposed, which are applicable to most path planning approaches.

The probably most intuitive indicator is the path length l_p, since it is usually supposed to be minimized. Indicators that characterize the complexity of the planned path are the

number n_w of waypoints w in the path relative to the Euclidean distance to the goal, the variance $\mathrm{Var}(\angle(w^1, \varphi_r))$ of the angular deviation

$$\angle(w^1, \varphi_r) = \left|\arctan(w_y^1, w_x^1) - \varphi_r\right| \quad \in [0, \pi] \tag{4.15}$$

between next waypoint w^1 and robot orientation φ_r, and the cumulative sum of the angular deviation

$$\mathrm{cad} = \sum_{i=1}^{n_w} \angle(w^i, w^{i-1}) \tag{4.16}$$

between consecutive w in the path, where $\arctan(w^0) = \varphi_r$. Finally, the number of waypoints n_v which satisfy a maximum clearance constraint is considered. This can be acquired by using for example distance transformation algorithms [23].

These metrics can be applied to any global planner which generates paths consisting of a sequence of waypoints. The planning approach used by *ACE* performs an A* search on a hybrid graph composed of nodes extracted from a bounding box structure and a Voronoi graph. Since Voronoi graphs belong to the family of distance transformation algorithms, the corresponding waypoints satisfy the maximum clearance constraint. Next the execution module is considered, which is responsible for a safe drive along the computed path.

4.4.2.3. Indicators for Execution Performance

Once the next action of the robot has been chosen it needs to be transformed to motion commands which must be carried out by the robot actuators in a coordinated manner. The responsible components present the task execution. For example the planner of a mobile robot chooses an intermediate target in the form of a waypoint. The execution components of the robot are responsible for generating controls for the wheel motors such that the target is safely reached.

The execution components of the *ACE* robot obtain global waypoints from the path planning module which has been described in the previous section. These are given as input to the obstacle avoidance module, which generates motor commands for the mobile platform. This module takes into account dynamic obstacles in the vicinity of the robot and ensures safe local navigation. A method similar to [94] is used to generate smooth and safe robot trajectories.

The execution efficiency of a performed navigation task can be evaluated by observing the execution time and the smoothness of the path. More specifically, the robot speed v_r and the variance of the robot orientation $\mathrm{Var}(\varphi_r)$.

4.4.2.4. All Performance Indicators at a Glance

All aforementioned indicators are summarized in Table 4.4.2.4. The indicators are grouped into the three categories according to the system part they characterize. Even though the indicators have been chosen to represent the internal system state of the navigational components of the *ACE* robot, part of them is retrieved from literature and they all are directly applicable to any mobile robot, which performs SLAM and plans a path that it finally drives along.

Category			Indicators
Perception	H^m	=	map uncertainty (4.10)
	H^P	=	pose uncertainty (4.11)
	$\mathrm{INFO}(m_{t-1} \| m_t)$	=	relative quantity of information (4.14)
Planning	l_p	=	path length
	n_w	=	number of waypoints
	$\mathrm{Var}(\angle(w^1, \varphi_r))$	=	variance of angular deviation
	cad	=	cumulative sum of angular deviation (4.16)
	n_v	=	number of maximum clearence waypoints
Execution	v_r	=	robot speed
	$\mathrm{Var}(\varphi_r)$	=	variance of robot orientation

Tab. 4.1.: Overview of proposed performance indicators.

The performance indicators discussed in this section are now evaluated on experimental data gathered by the *ACE* robot.

4.4.3. Experimental Results

In order to validate the proposed method, the system interdependence analysis has been performed on the *ACE* robot, based on data gathered during the outdoor experiment described in Appendix B.6 (p. 130). Data chunks from two representative situations were used for the system interdependence analysis. Fig. 4.5 shows two typical scenes encountered during the situations. The first situation, which is referred to as `Sidewalk`, demonstrates navigation on a sidewalk in a less populated area. The corresponding occupancy grid map acquired by the robot is illustrated in Fig. B.7(a) (p. 131). The second situation, referred to as `Pedestrian zone`, is a typical example of navigation in a densely populated pedestrian zone. The respective acquired occupancy grid map is illustrated in Fig. B.7(b).

Several considerable differences exist between these two settings. In the `Sidewalk` the moving ability of the robot is constrained by the narrow sidewalk but the dynamic characteristics of the environment are low. In the `Pedestrian zone` the environment is extensive but primarily characterized from high dynamics and local complexity. This is already observed from the indicator values, introduced in Section 4.4.2, which have been sampled in both scenes at 2 Hz. Part of them is shown in Fig. 4.6. For example in the `Pedestrian zone` the map uncertainty H^m and robot orientation variance $\mathrm{Var}(\varphi_r)$ have mean values that are 43% and 45% higher, respectively. The same applies to their variance which is 6.3 and 6.5 times higher in the `Pedestrian zone`. Intuitively this can be explained by the lower dynamics in the `Sidewalk`. In contrast, the speed of the robot v_r is on average the same. This is due to the fact that the robot speed was limited by design for safety reasons. Before the structure of the BN is learned, the data must be discretized and transformed into a predefined number of states. For the following results a discretization of three steps was used for all indicators.

(a) (b)

Fig. 4.5.: Two representative situations which were chosen for the interdependence analysis. (a) Navigation on a sidewalk in a less populated district shown in Fig. B.7(a). (b) Navigation in a densely populated pedestrian zone illustrated in Fig. B.7(b)

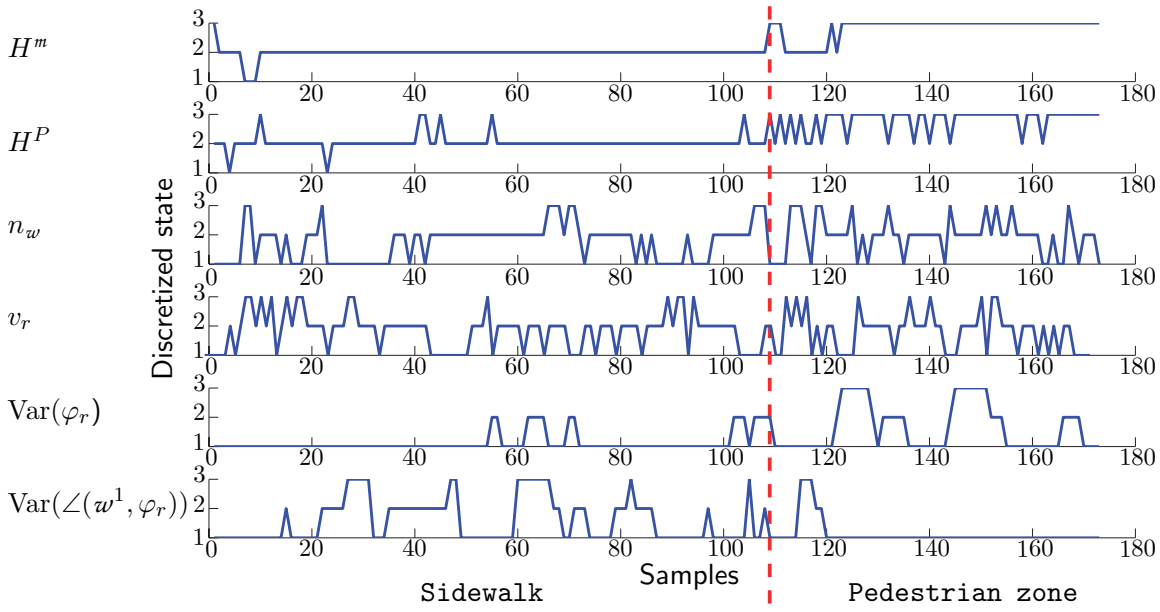


Fig. 4.6.: Discretized indicator (vertical axis) values extracted from experimental data, for two different environments. The dashed line indicates the transition between the environments. The horizontal axis shows the consecutive sample number.

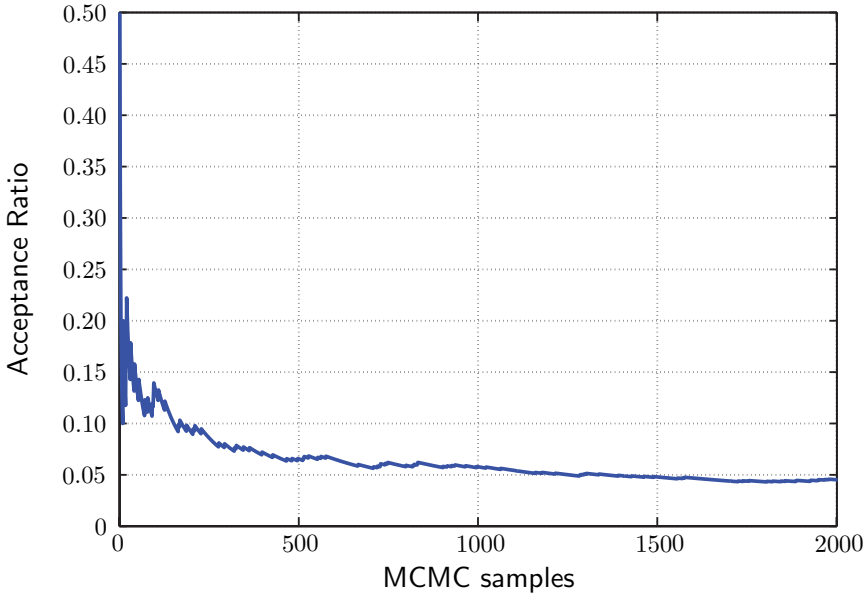


Fig. 4.7.: Acceptance ratio versus the number of MCMC steps. The MCMC search converges since the acceptance ratio does not change after 2000 steps.

As described in Section 4.4.1.3 in order to learn the structure of the BN, which describes the interaction between indicators, the space of possible DAGs needs to be searched and by choosing a scoring function the most likely structure needs to be identified. However the space of possible DAGs is super-exponential with the number of variables described. In the presented case study ten indicators have been identified, leading to a search space with 4.2×10^{18} graphs, which cannot be searched exhaustively.

Therefore a MCMC search was performed on the preprocessed data to acquire the node order for the BN. In Fig. 4.7 the acceptance ratio versus the number of MCMC steps is illustrated. In order to converge to the most likely graph 2000 steps are needed. The resulting prioritisation is $\langle H^P, n_w, H^m, l_p, \mathrm{Var}(\varphi_r), n_v, \mathrm{cad}, \mathrm{Var}(\angle(\boldsymbol{w}^1, \varphi_r)), \mathrm{INFO}, v_r \rangle$.

Using this prioritisation, the K2 algorithm generated the final BN, which has an improved BIC score of about approximately 6% and is shown in Fig. 4.8. The resulting BN indicates a lot of interdependencies between the indicators but cannot express the intensity of these relations. For that reason information-theoretic criteria are applied, as described in Section 4.4.1.4.

The learned structure was utilized to train a BN with all the data. Sequential Bayesian parameter updating was performed and the respective CPTs were acquired for the network. As prior a Bayesian Dirichlet distribution with a sample size of one was chosen. An implementation based on the Bayes Net Toolbox for Matlab [83] was used. The distance metric given by (4.9) is calculated for each possible pair of indicators. The results are illustrated in Fig. 4.9 by the solid line.

A strong interdependence of H^m on H^P, cad, $\mathrm{Var}(\angle(\boldsymbol{w}^1, \varphi_r))$ and $\mathrm{Var}(\varphi_r)$ is observed. The relation between H^m and H^P is expected, since without map knowledge it is impossible for the robot to localize itself. Also the influence of H^m on the planning indicators is intuitive, since the path quality is directly dependent on the used map. Map knowledge influences the planned path and therefore the motion of the robot, as reflected by the

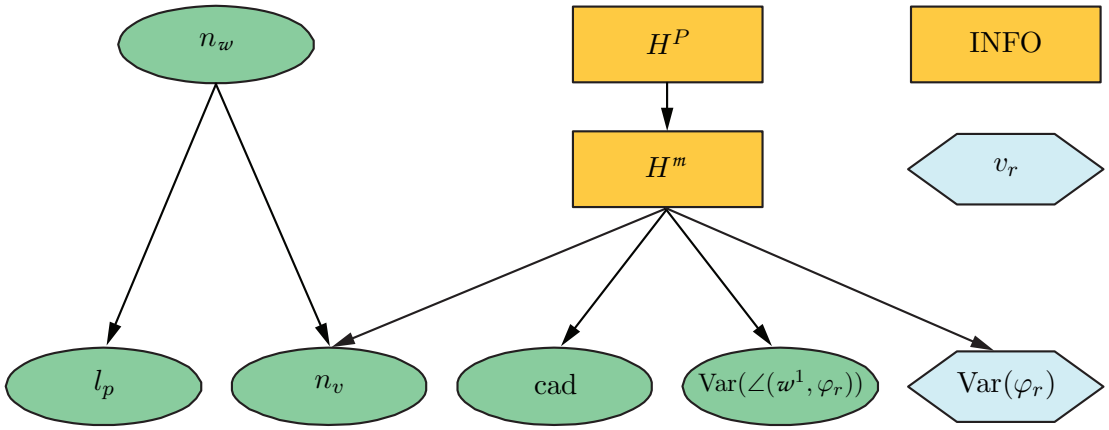


Fig. 4.8.: Directed Acyclic Graph (DAG) learned with MCMC and K2, showing the relations between the perceptual (rectangles), planning (ellipses) and execution (hexagons) indicators.

dependency between H^m and $\mathrm{Var}(\varphi_r)$. Furthermore n_w is strongly interconnected to n_v and l_p, what is ascribable to the fact that all of them are indicators for the complexity of the calculated path.

The indicators INFO and v_r show no influence from and to other indicators. Accordingly these indicators cannot give any information about the internal system state or the influence of the environment on the system. The complexity of the system and the application domain cannot be captured by simple and purely local indicators. More specifically the speed of the robot has been limited by design for security reasons in most situations. It would be dangerous to allow sudden accelerations or fast speeds for the robot, in the proximity of people. Therefore it is logically consistent that the influence from other indicators is found to be insignificant. The proposed analysis identifies in this case, a design choice of the system.

In order to assess the environmental influence on the indicators, two additional BNs are trained using the data from the `Sidewalk` and the `Pedestrian Zone` respectively. A comparison of η, which is also shown in Fig. 4.9, reveals the differences for the two scenes. A stronger influence of H^m on $\mathrm{Var}(\angle(w^1, \varphi_r))$ and $\mathrm{Var}(\varphi_r)$ in the `Pedestrian zone` is identified. The presence of moving people results in higher map uncertainty, less directed, i.e. more variable planned path and consequently more complex robot motion. On the other hand, n_v is stronger related to n_w in the `Sidewalk` scene. In this specific situation the robot has to navigate through narrow passages, where a maximum clearance path is desired. Consequently, the nodes of the Voronoi graph are more frequently used.

More detailed information about the influence of specific indicators can be extracted by the learned BNs by examining the marginal distributions of the indicators of interest while setting other indicators to specific values. This way the behavior of specific system components can be predicted for various environments and the robustness of the system can be evaluated.

This is shown for the influence of H^m on H^P and $\mathrm{Var}(\varphi_r)$. Fig. 4.10 illustrates the marginal distributions which are calculated from the learned BN for all assigned values of H^m by applying Bayesian inference. It can be observed, when map uncertainty increases,

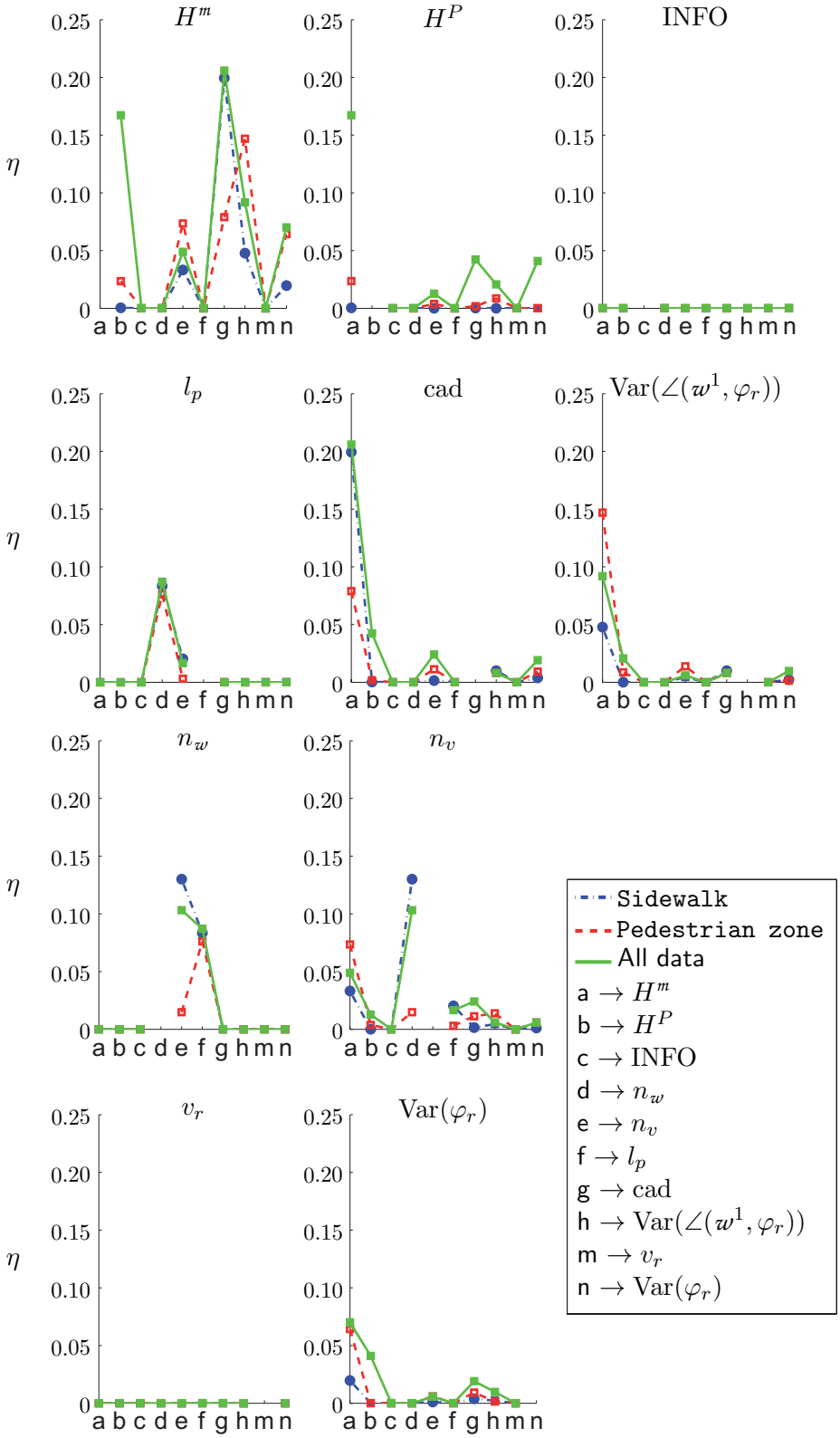


Fig. 4.9.: Learned dependency values $\eta(x_i, x_j)$ (vertical axis) for all indicators, where the i'th graph shows the dependencies of indicator i to all indicators j (horizontal axis). Since $\eta(x_i, x_i) = 1$, these values were skipped for illustrative purposes.

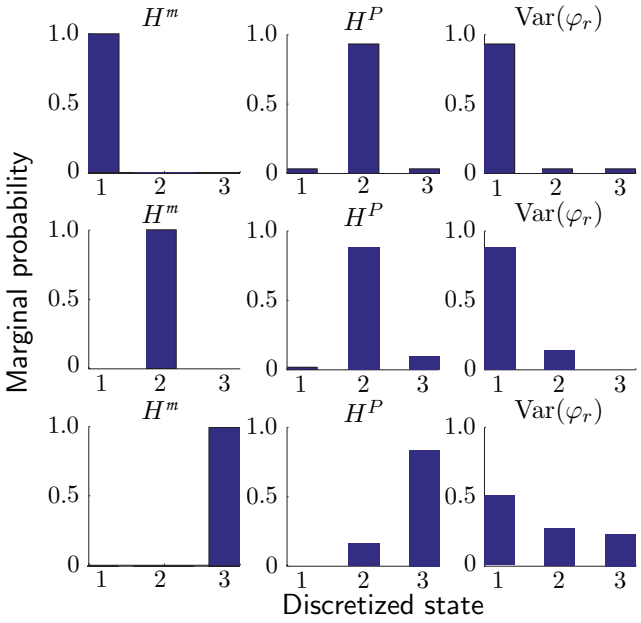


Fig. 4.10.: The marginal distributions of the dependent indicators H^P and Var(φ_r) as calculated from the learned BN, for assigned values of H^m.

H^P increases as well. The learned BN captures the interconnection between localization and mapping which constitutes the SLAM problem. When perceptual uncertainty increases, the motion of the robot becomes more variable as indicated by the uniformly distributed predicted states of Var(φ_r). However it can be seen that even with high uncertainty it is predicted that the variance of the robot motion will not be unacceptably high for most of the situations. Therefore the utilized planning algorithms can be expected to be robust even with uncertain environment models. Such information is very useful for making design choices. For example if it was predicted by the learned BN that even with low perceptual uncertainty the generated path of the robot will be very variable, then the system designer would have to reconsider the planning algorithms used. In the case of *ACE* it has been determined that the performance of all system components is sufficient in both environments.

In summary, the interdependence analysis of system state indicators and the environment, identified map uncertainty H^m as an indicator with very strong influence on the system. Consequently, the intuitive assumption is verified that knowledge of the environment – in this case map knowledge – is a crucial factor for the robustness of an autonomous robotic system. Further is shown that simpler and local complexity indicators such as v_r and INFO cannot characterize the behavior of the *ACE* robot. In general, by using the proposed method for system analysis, several indicators can be tested in respect to their representation ability. By using the learned BN and inference techniques, predictions can be made about the behavior of performance indicators given the values of others as evidence. However, the results of the analysis reflect only the system interdependencies in the examined environments and for the executed tasks. Even though these may provide an indication of the system behavior in different environments or for different tasks, a direct transfer is not coherent in general. Instead, new system data needs to be gathered followed

by a reapplication of the analysis.

4.4.4. Discussion

A method for a system interdependence analysis has been introduced. It aims at learning and quantitatively evaluating the coherence between performance indicators of different system components of autonomous robots, as well as of the influence of environmental parameters on the system. The presented method allows the identification of the limitations of an autonomous robotic system. The complexity of the environment determines the requirements to the robotic hardware and algorithms in order to perform a given task. Conversely, the capabilities of a robotic system define the environments where it can operate and the tasks it can handle. The proposed analysis provides an alternative in comparison of deriving the deterministic system model, what may be quite hard for complex systems, or it can be also used to verify the latter.

To validate the proposed method, component performance indicators for the navigation system of the autonomous mobile robot *ACE* were derived and the system interdependence analysis was performed based on experimental data from an extended field experiment. For this specific system, it has been shown that some of the proposed indicators have very strong representational capabilities, e.g. the map uncertainty. At the same time indicators have been proved to be unsuitable for the mutual performance evaluation of the system components, e.g. the robot speed. Furthermore, the influence of the environment on the performance indicators has been identified. Such gained knowledge is primarily useful for the improvement of the examined system itself but it is also transferable to similar systems, at least qualitatively. A quantitative transfer would be only valid under identical circumstances what is hard to guarantee for different practical systems.

Further steps should concentrate on the generality of indicators, in the sense if some of them are suitable for representing the performance of different purpose systems. This would allow to specify application-independent benchmark tests with respect to system robustness, in order to facilitate system comparability. Concerning the method itself, different algorithms for the BN structure search may be evaluated, e.g. whether they provide a better tradeoff between complexity and solution quality. This would improve the scalability of the method. Additionally, instead of a static network also dynamic Bayesian networks may be learned, which allow to examine also temporal interdependencies of dynamic systems. The discretization leaves also space for future research. For example methods to determine an optimal discretization but also networks with continuous nodes may be considered.

Nevertheless, the knowledge gained by the presented method is useful for system redesign, but also during system operation. It can be used to make predictions and estimations about the current environmental situation based on the observations the robot makes about the internal or external state. This way the robot can improve its decision making by anticipating the influence of the current environmental situation on its actions as described next. Means are presented to yield a situation- and risk-aware cost estimation and furthermore a better forecasting of failures enabling respective reactions.

4.5. Uncertainty- and Risk-Aware Reward Estimation

Recalling Section 4.3, four aspects were identified that are essential for a persistent action selection. The approach presented in the preceding section determines the relevant factors $\boldsymbol{s}$ and provides the respective pdf $\mathsf{f}_c(c \mid \boldsymbol{s})$ in form of a learned Bayesian Network. This leaves the demand for a risk-aware estimation of $\hat{c}(a \mid \boldsymbol{s})$ and a failure-forecasting by observation of the cost deviation $\Delta c(a \mid \boldsymbol{s})$, which is presented next.

4.5.1. Quantile-Based Reward Estimation

The conditional pdf $\mathsf{f}_c(c(a) \mid \boldsymbol{s})$ provides the basis to obtain a probabilistic estimate $\hat{c}(a \mid \boldsymbol{s})$ of the cost $c(a)$ for action a given $\boldsymbol{s}$. A frequently used approach is to set

$$\hat{c}(a \mid \boldsymbol{s}) = \mathrm{E}\left[c(a) \mid \boldsymbol{s}\right],$$

where $\mathrm{E}\left[c(a) \mid \boldsymbol{s}\right]$ is the first moment of $\mathsf{f}_c(c(a) \mid \boldsymbol{s})$, also referred to as expected value or mean. Since the expected value is linear, i.e. additivity and homogeneity of degree one apply even for statistically dependent variables, the respective estimates for reward and performance are given by

$$\hat{\varrho}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) = \mathrm{E}\left[\varrho(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}\right] = u(p_{\mathcal{A}}(\tau)) - \sum_{a_l \in p_{\mathcal{A}}(\tau)} \mathrm{E}\left[c(a_l) \mid \boldsymbol{s}\right] \tag{4.17}$$

and

$$\hat{q}(\boldsymbol{s}) = \mathrm{E}\left[q \mid \boldsymbol{s}\right] = \frac{1}{\beta_u} \sum_{j=1}^{m_c} \left(u(p_{\mathcal{A}}(\tau_j)) - \sum_{a_{jl} \in p_{\mathcal{A}}(\tau_j)} \mathrm{E}\left[c(a_{jl}) \mid \boldsymbol{s}\right] \right). \tag{4.18}$$

The expected value provides the best prediction in terms of minimizing the squared error. However, it gives only a very limited representation of a pdf which is for example problematic in case of asymmetric or very flat distributions. The shape of a pdf is further characterized by its higher order moments such as variance, skewness and kurtosis. In order to also take these criteria into account, the quantile function provides, in contrast to the expected value, a more comprehensive representation of a distribution. For example, the quantile function enables to also take potential asymmetries of a pdf into account.

The quantile function Q is the inverse $\mathcal{F}^{-1}$ of the cumulative distribution function (cdf) $\mathcal{F}$. The cdf

$$\mathcal{F}_c(x) := P(c \leq x) = \varsigma \tag{4.19}$$

of c gives the probability $\varsigma \in [0, 1]$ that c will be at most x. The quantile

$$Q_c(\varsigma) := \mathcal{F}_c^{-1}(\varsigma) := \inf\left\{x \in \mathbb{R} : \varsigma \leq \mathcal{F}_c(x)\right\} \tag{4.20}$$

is accordingly the smallest value x, for which the probability that c is not greater than x is at least ς. Note that in some literature the alternative definition

$$Q'_c(\varsigma) = \inf\left\{x \in R : \varsigma < \mathcal{F}_c(x)\right\} = \sup\left\{x \in R : \varsigma \geq \mathcal{F}_c(x)\right\} \tag{4.21}$$

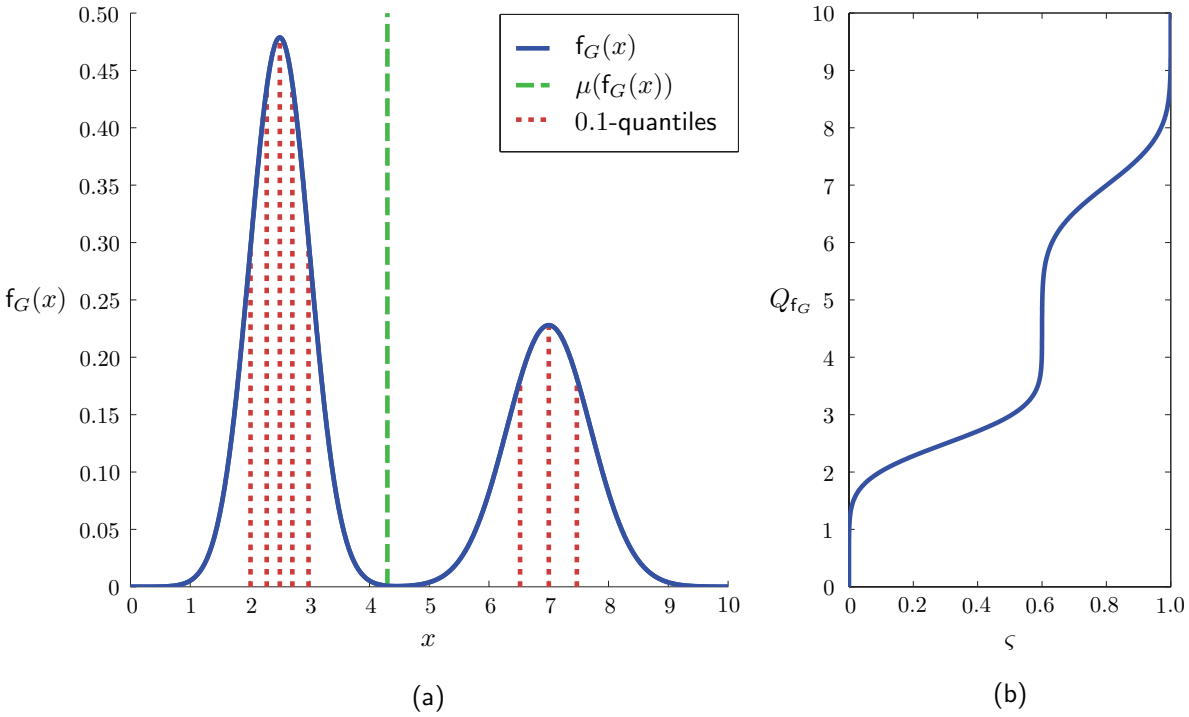


Fig. 4.11.: The quantile function Q_{f_G} (right) for the pdf $\mathsf{f}_G(x) = 0.6\ \mathcal{N}(2.5, 0.5) + 0.4\ \mathcal{N}(7, 0.7)$ (left), where $\mathcal{N}(\mu, \sigma)$ is a normal distribution with mean μ and standard deviation σ. The left side shows further the mean $\mu = \mathrm{E}[x]$ and the 0.1-quantiles (deciles) of $\mathsf{f}_G(x)$.

is used, which may return a different result than (4.20), e.g. in case the cdf has discontinuities. However, in [49] – where the relation of both definitions is exhaustively examined – is shown that

$$Q_c(\varsigma) \leq Q'_c(\varsigma) \leq Q_c(\varsigma + \epsilon), \quad \epsilon \in \mathbb{R}_0^+, \tag{4.22}$$

where in most cases holds $\epsilon \to 0$. The quantiles for a specific ς are also referred to as ς-quantiles. For example the 0.25-quantile (quartile) of a pdf is the value, for which the probability is 0.25 that a random variable of this pdf will be equal or smaller.

Fig. 4.11(a) shows the 0.1-quantiles, also referred to as deciles, for a Gaussian mixture pdf f_G. Its quantile function Q_{f_G} is shown in Fig. 4.11(b). Thereby the k-th 0.1-quantile equals the $k/10$-quantile. Fig. 4.11(a) further shows the mean for the pdf. This example illustrates how poorly the mean may approximate a pdf. While the mean in this example is 4.3, the median, i.e. the 0.5-quantile, is 2.98. In other words, 50% of the values are expected to be at least 30% lower than the mean. A similar calculation can be made for larger values. As the 0.7-quantile is given with 6.52, more than 30% of the values are expected to be more than 51% larger than the mean. Thus, in this example a mean-based cost estimate entails a high risk to be far off the real value.

By enabling the consideration of a large variance and even of asymmetries in the pdf, the quantile function allows for risk averse policies, i.e. reducing the risk that the estimated rewards are not reached. How this may be beneficially used in a cooperative action selection is illustrated by the following example.

Assuming a task τ, which demands the cooperative execution of the actions a_1 and a_2 by two robots, has the cost function

$$c(\tau) = f(c(a_1), c(a_2)) = \max(c(a_1), c(a_2)). \tag{4.23}$$

This function may for example correspond to the overall time required to complete the task. Assuming further there are three cost estimates $\hat{c}_{r_1}(a_1)$, $\hat{c}_{r_2}(a_2)$, and $\hat{c}_{r_3}(a_2)$ from robots r_i, $i \in \{1, 2, 3\}$, where $\hat{c}_{r_1}(a_1) > \hat{c}_{r_2}(a_2)$ and $\hat{c}_{r_1}(a_1) > \hat{c}_{r_3}(a_2)$. Accordingly follows $\hat{c}(\tau) = \hat{c}_{r_1}(a_1)$ w.r.t. (4.23). However, this leaves the question which of the possible coalitions $z_1 = \langle r_1, r_2 \rangle$ and $z_2 = \langle r_1, r_3 \rangle$ should execute τ, resulting in a multi-objective optimization problem. In this regard one needs to bear in mind that $\hat{c}(\tau) = \hat{c}_{r_1}(a_1)$ is only true as long as the inequality constraints $\hat{c}_{r_1}(a_1) > \hat{c}_{r_2}(a_2)$ and $\hat{c}_{r_1}(a_1) > \hat{c}_{r_3}(a_2)$ are valid. Even though this can not be guaranteed at least the risk of violation may be reduced. A feasible solution to the multi-objective optimization is to choose the coalition that implies a lower risk that the respective constraint is violated. This gives reason for the following definitions.

Definition 4.1 *The reliability*

$$rel(\hat{c}) := \frac{\left| \bigcup_{\Delta c \geq 0} \hat{c} \right|}{\left| \bigcup_{\Delta c} \hat{c} \right|}$$

of a cost estimation $\hat{c}$ is defined as the ratio of the number of overestimations, i.e. estimates $\hat{c}$ for which the respective cost deviation $\Delta c = \hat{c} - c$ according to (4.4) is non-negative, compared to the number of total estimations.

Definition 4.2 *Similarly the reliability*

$$rel(\hat{\varrho}) := \frac{\left| \bigcup_{\Delta \varrho \leq 0} \hat{\varrho} \right|}{\left| \bigcup_{\Delta \varrho} \hat{\varrho} \right|}$$

of a reward estimation $\hat{\varrho}$ is defined as the ratio of the number of underestimations, i.e. estimates $\hat{\varrho}$ for which the respective reward deviation $\Delta \varrho = \hat{\varrho} - \varrho$ is non-positive, compared to the number of total estimations.

Following from its definition in 4.20, the probability that $Q_c(\varsigma)$ turns out to be an overestimation is exactly ς. However, the quantile function is in general not linear and thus the quantile for the outcome of an additive mixture of quantiles is not derivable without knowing the pdf of the outcome variable. This can be exemplified by considering the second moment, the variance, of a mixture of pdfs. The variance of the outcome depends not only on the individual variances of the input variables but also on the respective covariances $\mathrm{Cov}(\cdot, \cdot)$, which reflect the interrelation among the input variables. A reliable identification

of the $\mathrm{Cov}(\cdot,\cdot)$ without the knowledge of a functional relationship is only possible by observation of the outcome. This similarly applies to the derivation of higher order moments. These also have to be considered for a reliable Q-estimation as the interrelation among the input variables may not only affect the mean or the variance of the output pdf, but also lead to asymmetric distortions of the distribution.

As a consequence, in the following the non-general but essential assumption is made, that the output is observable and thus can be incorporated in the learning process described in Section 4.4. Even though this presents a strong assumption only non-observable environments are excluded by it. In partially or even fully observable environments, which actually comprise the vast majority of robotic applications, a robot is able to perceive some effect of its actions in the environment. Still, these observations may be noisy and/or incomplete but they provide the basis for a robot to learn from its experience. In this respect, the quantile of the outcome distribution is directly derived from the observations. As a consequence, the quantile $Q_{c(p_{\mathcal{A}}(\tau))}$ of the cost of the action plan $p_{\mathcal{A}}(\tau)$ not directly derivable from the quantile $Q_{c(a)}$ of the cost for action a. Similarly is the quantile Q_q of the performance q is not directly derivable from the quantile Q_ϱ of the reward ϱ. Nevertheless, a derivation of Q_ϱ based on $Q_{c(p_{\mathcal{A}}(\tau))}$ is feasible:

Proposition 4.3 *Assume that $c_{min}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ is the lowest possible and $c_{max}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ the largest possible cost for executing $p_{\mathcal{A}}(\tau)$ given $\boldsymbol{s}$. If the pdf $\mathsf{f}_c(c(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s})$ of the action plan costs is continuous in $[c_{min}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}), c_{max}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})]$ and further*

$$\mathsf{f}_c(c(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}) \neq 0, \quad \forall c(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) \in [c_{min}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}), c_{max}(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})]\,,$$

then the following statement holds:
A cost estimation based on the ς-quantile, i.e. $\hat{c}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) = Q_{c(p_{\mathcal{A}}(\tau)|\boldsymbol{s})}(\varsigma)$, yields an expected reward reliability of

$$\widehat{rel}\left(\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})\right) = \varsigma.$$

Proof: In order to prove this proposition, the following properties of the quantile function are used:

(i) $Q_x(\varsigma) = -Q'_{-x}(1-\varsigma)$.

(ii) $Q_{f(x)}(\varsigma) = f(Q_x(\varsigma))$, iff $f : \mathbb{R} \to \mathbb{R}$ is continuous and non-decreasing.

(iii) $Q_x(\varsigma) = Q'_x(\varsigma)$, iff $\mathcal{F}(x)$ is strictly increasing.

A verification of these properties is given in [49]. From (i) follows

$$Q'_{-c(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(1-\varsigma) = -Q_{c(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(\varsigma)$$

and since $u(p_{\mathcal{A}}(\tau))$ is constant and thus $f(x) = x + u(p_{\mathcal{A}}(\tau))$ is continuous and non-decreasing, follows for the quantile of the reward according to its definition (2.6) (p. 22) and property (ii):

$$\begin{aligned} Q'_{\varrho(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(1-\varsigma) &= u(p_{\mathcal{A}}(\tau)) + Q'_{-c(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(1-\varsigma) \\ &= u(p_{\mathcal{A}}(\tau)) - Q_{c(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(\varsigma). \end{aligned} \tag{4.24}$$

From the assumption that $\mathsf{f}_c(c(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s})$ is continuous and $\mathsf{f}_c(c(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}) \neq 0$ for all possible values of $c(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ follows that the same properties also apply to $\mathsf{f}_\varrho(\varrho(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s})$ as it results – similar to its quantile function – from the linear transformation

$$\mathsf{f}_\varrho(\varrho(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}) = \mathsf{f}_u(u(p_{\mathcal{A}}(\tau))) - \mathsf{f}_c(c(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}),$$

where $\mathsf{f}_u(u(p_{\mathcal{A}}(\tau))) = \delta(x - u(p_{\mathcal{A}}(\tau)))$ and $\delta(\cdot)$ is the Dirac delta function. Consequently follows that $\mathcal{F}_{\varrho(p_{\mathcal{A}}(\tau)|\boldsymbol{s})}$ is strictly increasing since for $\varrho_2 > \varrho_1$ holds

$$\mathcal{F}_\varrho(\varrho_2(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}) - \mathcal{F}_\varrho(\varrho_1(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s}) = \int_{\varrho_1}^{\varrho_2} \mathsf{f}_\varrho(\varrho(p_{\mathcal{A}}(\tau)) \mid \boldsymbol{s})\, d\varrho > 0.$$

Thus follows according to (iii) that

$$Q_{\varrho(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(1-\varsigma) = Q'_{\varrho(p_{\mathcal{A}}(\tau)|\boldsymbol{s}))}(1-\varsigma).$$

This in turn allows for the following ς-quantile-based reward estimation:

$$\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) = Q_{\varrho(p_{\mathcal{A}}(\tau)|\boldsymbol{s})}(1-\varsigma) = u(p_{\mathcal{A}}(\tau)) - Q_{c(p_{\mathcal{A}}(\tau)|\boldsymbol{s})}(\varsigma) \tag{4.25}$$

With respect to (4.19) and (4.21) follows that a reward underestimation, i.e.

$$\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) < \varrho_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}),$$

will occur in case of a cost overestimation and thus follows:

$$\widehat{rel}\,(\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})) = \widehat{rel}\,(\hat{c}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})) = \varsigma.$$

■

The probability $1-\varsigma$ is referred to as confidence interval. The value of ς may be chosen by the designer and can be used to set the risk aversion of the MRS. The lower ς is chosen the more likely it is that the estimated $\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ will be reached. However, as $\boldsymbol{s}$ may change over time the validity of the prior reward estimation needs to be permanently verified during the posterior execution.

During the latter $\boldsymbol{s}$ may change or new knowledge about it may be obtained. Given a respective new state $\boldsymbol{s}'$ an updated reward estimate $\hat{\varrho}'_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}')$ is retrieved. In case the constraint

$$\Delta\varrho(p_{\mathcal{A}}(\tau)) = \hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}) - \hat{\varrho}'_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s}') \overset{!}{\leq} 0.$$

is violated, replanning may be considered based on the extent of $\Delta\varrho(p_{\mathcal{A}}(\tau))$.

Accordingly, a reward estimation $\hat{\varrho}_\varsigma(p_{\mathcal{A}}(\tau) \mid \boldsymbol{s})$ has been presented that takes the environmental situation in form of the predicted state $\boldsymbol{s}$ into account. Moreover, the parameter ς enables to specify the risk of a reward underestimation. The usage of the approach in practice is demonstrated next.

4.5.2. An Application Example

The presented performance estimation is exemplified in a navigation scenario in which the robot is faced with a dynamic environment that is subject to large uncertainty. A respective scene is shown in Fig. 4.12. The task $\tau_{goto}(w_{gp})$ assigned to the robot is to navigate from its start position to the goalpoint w_{gp}. Only in case the robot reaches w_{gp} a respective utility $u(\tau_{goto}(w_{gp}))$ is obtained. The cost arising during the execution of $\tau_{goto}(w_{gp})$ is the time t_r needed to reach w_{gp}. While the robot should try to get as fast as possible to the goalpoint it also has to ensure that it is getting there safely. In case the robot collides with any part of the environment the execution is considered as failed and the robot does not obtain any utility. Thus, in such a case the reward purely results from the incurred costs. A collision may either occur with the static part of the environment or with dynamic objects. The latter may be for example other robots or humans, and are here modeled by circles with a radius of 0.3 m. The robot has a size of 0.4 m× 0.3 m.

Experimental setup: In order to yield a large environmental uncertainty the robot is faced with 20 different maps. While the general scenario remains always the same the maps differ with respect to various environment parameters. The first map is shown in Fig. 4.12 and has a size of 10.88 m× 6.8 m. The other maps are mirrored and/or rotated versions of map 1 whose start and goal position may be additionally shifted clockwise around the obstacle in the middle. Still the relative distance between start and goal is kept unchanged such that the robot has always the option between a *short* and a *long way* as shown in Fig. 4.12. However, a complicating factor on the *way* are dynamic objects that move back and forth on a path orthogonal to the moving direction of the robot as shown in Fig. 4.12. Even though the objects move with constant speed they present a large collision risk for the robot. Highly relevant for the performance of the robot is the number of dynamic objects in the map. In this respect the parameter $ratio = x/y$ specifies that x dynamic objects are located on the *short* and y dynamic objects on the *long way*. In each map a

$$ratio \in \{1/1,\ 1/3,\ 3/1,\ 1/5,\ 5/1\}$$

is used such that each constellation occurs exactly in four maps. A detailed description of all maps is given in Appendix C.

Moreover, while the robot is always provided with full knowledge about the static part of the map, two different conditions with respect to the dynamic part are used. In the first, in the following referred to as `limited` condition, the robot has no prior knowledge about the spreading of objects. In the second, referred to as `overview` condition, the robot is given the information about the current *ratio* of objects.

Throughout one entire run the robot is faced with all maps in both conditions resulting in 40 maps in total. In case of successful completion the next map is selected while in case of a collision the robot needs to repeat the current one. However, the maximum number of trials for each map is limited by an upper bound α. If this is reached the robot proceeds to the next map even in case of failure.

For the actual execution of $\tau_{goto}(w_{gp})$ an implementation making use of the Player/Stage framework [18] is used. In order to keep the influence of other system components on the

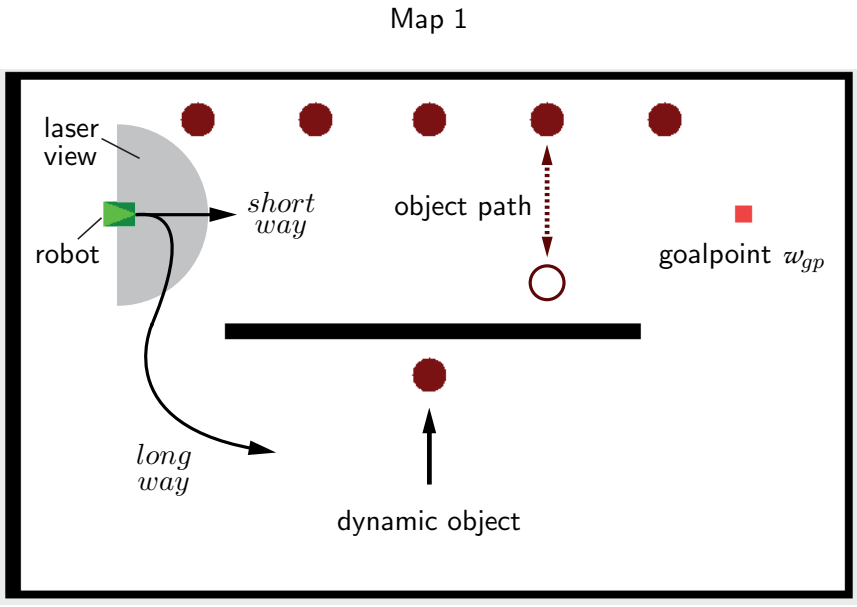


Fig. 4.12.: Map 1 out of 20 maps used for the experiment. The map shows an object *ratio* of 5/1, is not swapped, not rotated and start (current location of robot) and goal positions are not shifted. An overview of all maps is given in Appendix C (p. 135).

robot performance low a global localization, i.e. ground truth, was used. Furthermore, instead of using a path planner the robot moved along paths of fixed waypoints. For the obstacle avoidance the *vfh* driver of Player is used, which implements the Vector Field Histogram Plus method introduced in [118]. As sensor for *vfh*, the robot was equipped with a laser range finder. However, the latter had a very limited view of 1.7 m that allowed a detection of obstacles only in the close front area as indicated in Fig. 4.12. Thus the navigation of the robot is reactive and without taking the object velocity into account. Nevertheless, it yields a comparably uniform moving behavior throughout the various maps and conditions. This allows for a more clear examination of the actual influence that the action selection has on the overall performance.

In order to maximize the latter the robot needs to decide upon taking the *short* or *long way* based on the given information. Consequently it decides for the *way* where it expects a higher reward which is estimated as follows.

Reward estimation: For the current experiment the sole task is $\tau_{goto}(w_{gp})$. The corresponding action-plan set

$$\mathcal{P}_{\mathcal{A}}(\tau_{goto}(w_{gp})) = \{\langle a_{short}(w_{gp})\rangle, \langle a_{long}(w_{gp})\rangle\}$$

contains two action plans. Action $a_{short}(w_{gp})$ corresponds to taking the *short way* to reach w_{gp} and $a_{long}(w_{gp})$ to take the *long* one respectively. In order to decide which one of the two options to choose a reward estimate for both needs to be retrieved.

According to (4.2) the reward results from

$$\hat{\varrho}(p_{\mathcal{A}}(\tau_{goto}), \boldsymbol{s}) = u(p_{\mathcal{A}}(\tau_{goto})) - \hat{c}(p_{\mathcal{A}}(\tau_{goto}), \boldsymbol{s}).$$

For the utility a value of $u(p_{\mathcal{A}}(\tau_{goto})) = 100$ was chosen by the designer. In general, a higher utility also creates a higher interest to complete a task. In contrast, a lower utility leads to more frequent rejections or stops of tasks in case the robot is provided with the respective choice. The latter is discussed below in Section 4.5.3.3. The cost estimate is derived by

$$\hat{c}(p_{\mathcal{A}}(\tau_{goto}), \boldsymbol{s}) = P_s^1 \hat{c}_s^1 + (1 - P_s^1)\hat{c}_f^1 \tag{4.26}$$

where

$$P_s^i = \sum_{\forall ratio} P\left(status^i = success \mid ratio, way^i\right) P(ratio) \tag{4.27}$$

is the probability of $p_{\mathcal{A}}(\tau_{goto})$ being successfully completed in trial i.

$$\hat{c}_s^i = \sum_{\forall ratio} \hat{c}\left(\hat{t}_r^i \mid status^i = success, ratio, way^i\right) P(ratio) \tag{4.28}$$

is the expected cost, i.e. the value of the expected run time $\hat{t}_r$, arising in case of success and

$$\hat{c}_f^i = P_s^{i+1}\hat{c}_s^{i+1} + (1 - P_s^{i+1})\hat{c}_f^{i+1} + \sum_{\forall ratio} \hat{c}(\hat{t}_r^i \mid status^i = failed, ratio, way^i)P(ratio) \tag{4.29}$$

the cost in case of failure respectively. $\hat{c}(\hat{t}_r)$ returns the dimensionless quantity of $\hat{t}_r$. The maximum number of trials is limited by α, i.e. $i \in \{1, \ldots, \alpha\}$ and $\hat{c}_s^{\alpha+1} = \hat{c}_f^{\alpha+1} = u(p_{\mathcal{A}}(\tau_{goto}))$ to account for the non-obtaining of the utility in case τ_{goto} is not accomplished at all.

The results of (4.27), (4.28) and (4.29) depend all on the way^i selected in the respective trial i and further on the given *ratio*. As previously explained the *ratio* may be known or not. While in the `overview` condition the state of *ratio* is simply set to the given value in the `limited` condition no prior knowledge is given to the robot. However, the probability of the specific *ratio* is inferable by the experience gathered through the preceding $i - 1$ failed trials, assuming $i > 1$:

$$P(ratio) = \frac{1}{\beta} \sum_{j=1}^{i-1} P\left(ratio \mid status^j = failed, way^j, l_p^j\right) \tag{4.30}$$

β is a normalization factor and l_p the length of the traveled path, i.e. until completion or failure. As the start conditions for each trial of a map are always the same, the values of the indicators t_r^j, $status^j$ and l_p^j are only dependent on the choice of way^j. Accordingly, t_r^j, $status^j$ and l_p^j are stochastically independent between consecutive trials once the robot decides on a specific choice of way^j. The current choice on way^i in turn depends on the knowledge about $P(ratio)$ which is with respect to (4.30) clearly dependent on the prior trials. In order to retrieve the actual estimate of the time $\hat{t}_r^i$ for trial i, two cost functions

are used. In the first case the expected value

$$\hat{c}_{\mu}(t_r^i) = \frac{\hat{t}_r^i}{s} = \frac{1}{s} E[t_r^i]$$

and in the second the ς-quantile

$$\hat{c}_{\varsigma}(t_r^i) = \frac{\hat{t}_r^i}{s} = \frac{1}{s} Q_{t_r^i}(\varsigma)$$

is calculated. In both cases the values are divided by seconds s to obtain a dimensionless scalar quantity. The expected value is in the continuous case derived by

$$\mathrm{E}_{\mathfrak{c}}[t_r] = \int_0^{\infty} t \ \mathsf{f}_{t_r}(t) \ dt$$

and for a discrete variable by

$$\mathrm{E}_{\mathfrak{d}}[t_r] = \sum_{k=1}^{k_{max}} t_{r,k} P(t_{r,k}), \tag{4.31}$$

where k_{max} is the number of discretization intervals and $t_{r,k}$ is the mean of the k-th interval. Accordingly the ς-quantile is in the continuous case derived by

$$\begin{aligned} Q_{t_r,\mathfrak{c}}(\varsigma) = \underset{t_{r,max}}{\arg\min} \int_0^{t_{r,max}} t \ \mathsf{f}_{t_r}(t) \ dt, \\ s.t. \int_0^{t_{r,max}} \mathsf{f}_{t_r}(t) \ dt \overset{!}{\geq} \varsigma \end{aligned}$$

and for a discrete variable holds

$$\begin{aligned} Q_{t_r,\mathfrak{d}}(\varsigma) = \min_{k_{max}} \sum_{k}^{k_{max}} t_{r,k} P(t_{r,k}), \\ s.t. \sum_{k=1}^{k_{max}} P(t_{r,k}) \overset{!}{\geq} \varsigma. \end{aligned} \tag{4.32}$$

Hereby needs to be noted that Proposition 4.3 is only valid for continuous pdfs. However, even though the current implementation learns a discrete pdf for t_r this represents a quantized version of the continuous f_{t_r}. From this follows for the discrete quantile function $Q_{t_r,\mathfrak{d}}(\varsigma)$ that it is a sampled version of its continuous counterpart $Q_{t_r,\mathfrak{c}}(\varsigma)$, where holds

$$Q_{t_r,\mathfrak{d}}(\varsigma) \geq Q_{t_r,\mathfrak{c}}(\varsigma)$$

with equality at the bounds of the discretization intervals. Accordingly the discrete pdf yields an even more conservative estimation that satisfies

$$\widehat{rel}_{\mathfrak{d}} \geq \widehat{rel}_{\mathfrak{c}}, \tag{4.33}$$

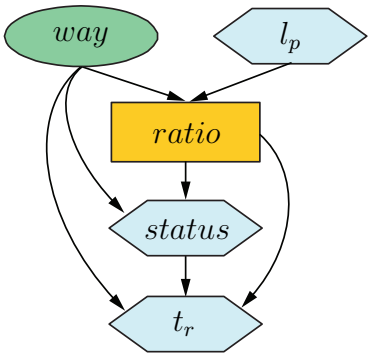


Fig. 4.13.: Bayesian graph structure used for the inference policies. The structure was manually suited to the inference requests. Following Fig. 4.8 the indicators are grouped into perceptual (rectangles), planning (ellipses) and execution (hexagons) indicators.

where $\widehat{rel}_{\mathfrak{d}}$ and $\widehat{rel}_{\mathfrak{c}}$ are the reliabilities of estimations based on the discrete and the continuous pdf respectively. As in the following only discrete distributions are considered (4.31) and (4.32) provide the essential cost functions for the reward estimation. Which of these two alternatives is chosen depends on the used policy.

Selection policies: For the selection of the *way* the following policies were used:

- short policy ϖ_{short}: the robot takes always the shorter path. This policy was only used for training.
- long policy ϖ_{long}: the robot takes always the longer path. This policy was only used for training as well.
- random policy ϖ_{rand}: the path was randomly selected with equal probabilities of both choices.
- defensive policy ϖ_{def}: the robot takes always the path with less dynamic objects in case these are known. If $ratio = 1/1$ the shorter path is used. In the `limited` condition ϖ_{rand} is used.
- inference policy ϖ_{inf}: the robot decides based on an environment model and its current knowledge which *way* to choose. For this policy both described cost models were used, where $\varpi_{inf,\mu}$ relates to a mean-based estimation according to (4.31) and $\varpi_{inf,\varsigma}$ to a quantile-based estimation according to (4.32).

Learning the environment model: For the inference policy ϖ_{inf} the method described in Section 4.4 is used to learn a model of the environment. However, in Section 4.4 the model structure as well as the parameters are learned from scratch, wherefore the Bayesian Information Criterion (BIC) is used to score candidate structures. BIC gives those structures a higher score that are less complex and better explain the indicator interdependencies.

However, for a model to be suitable for the online application further aspects are of importance that are not taken into account by the BIC measure. In this respect the structure needs to ensure conditional dependence among the variables where needed. For example in case two variables A and B are only connected with each other over a third variable C, where A is the parent and B the child of C, A and B are conditionally independent in case C is given.

So, in order to ensure that all inference requests are satisfied, it is beneficial to determine the graph structure with respect to the reward estimation. In consideration of (4.27) - (4.30) three types of inference requests – (4.28) and (4.29) are of same type – are needed in order to retrieve an estimation of $\hat{\varrho}(p_{\mathcal{A}}(\tau_{goto}), \boldsymbol{s})$. In this respect the graph shown in Fig. 4.13 has been manually derived, which satisfies direct conditional dependence among all variables as needed by (4.27) - (4.30).

In analogy to the graph shown in Fig. 4.8 the indicator variables also are grouped into the categories perception, planning and execution. However, in contrast to Section 4.4.3 this graph is not the one that explains the data best but the one that fits to the inference requests most. In this respect any unnecessary interdependencies have been neglected such as an intuitive dependence between *way* and l_p for example. Moreover, the path length l_p is now classified as execution indicator, since for the current setup no explicit path planner was used.

For the parameter learning, the structure was trained with data gathered by two complete runs with ϖ_{short} and two with ϖ_{long} respectively. Thus data for all potential constellations of *way* and the environment parameters is obtained. While *ratio*, *status* and *way* are already of discrete type, l_p and t_r had to be discretized. Therefore, for l_p three uniform intervals and for t_r four uniform intervals were chosen after manual observation of the data. For the parameter training the Bayes Net Toolbox for Matlab [83] was used and as prior a Bayesian Dirichlet distribution with a sample size of one. The resulting model provides the basis for a situation-aware action selection for which results are given next.

4.5.3. Experimental Results

For the evaluation of the described approach three types of the inference policy were applied. Besides the mean-based version $\varpi_{inf,\mu}$ two quantile-based versions $\varpi_{inf,\varsigma}$ with $\varsigma \in \{0.5, 0.75\}$ were used. For comparison, additional results with the random policy ϖ_{rand} and the defensive policy ϖ_{def} were retrieved. In order to test the approach also under different levels of environment dynamic three different object speeds

$$\left\langle v_{o,\frac{1}{4}}, v_{o,\frac{1}{2}}, v_{o,1} \right\rangle = \left\langle 0.15\frac{m}{s}, 0.3\frac{m}{s}, 0.6\frac{m}{s} \right\rangle$$

were used. The robot speed was controlled by the *vfh* algorithm but has been limited to $v_r \in \left[0\frac{m}{s}, 1\frac{m}{s}\right]$. For each v_o a separate environment model was learned as described above. Thereafter a set of 15 runs was conducted for each combination of the five policies with the object speeds. As mentioned before, throughout one entire run the robot is faced with all maps in the `limited` condition and thereafter by all maps in the `overview` condition. The maximum number of trials per map was set to $\alpha = 3$. The parameters of the experimental runs are also summarized in Table 4.2.

Policies:	$\{\varpi_{rand}, \varpi_{def}, \varpi_{inf,\mu}, \varpi_{inf,0.5}, \varpi_{inf,0.75}\}$
Robot speed:	$v_r \in \left[0\frac{m}{s}, 1\frac{m}{s}\right]$
Object speeds:	$\left\langle v_{o,\frac{1}{4}}, v_{o,\frac{1}{2}}, v_{o,1} \right\rangle = \left\langle 0.15\frac{m}{s}, 0.3\frac{m}{s}, 0.6\frac{m}{s} \right\rangle$
Runs conducted for each $v_{o,*}$*:*	15
Trials per map:	$\alpha = 3$
Used object ratios:	$ratio \in \{1/1,\ 1/3,\ 3/1,\ 1/5,\ 5/1\}$
Occurences of ratio per condition:	$n_{maps}(ratio, condition) = 60$

Tab. 4.2.: Parameters used for the experimental runs.

In the following, results conducted with the different policies are presented. In Section 4.5.3.1 the performances achieved by the policies are compared. Section 4.5.3.2 discusses the reliabilities achieved by the inference-based policies and Section 4.5.3.3 shows briefly how the performance could be further increased by failure forecasting.

4.5.3.1. Situation-Aware Action Selection

In the following is shown how the situation-awareness of the inference-based policies yields a performance improvement. First the error rates of the policies are discussed followed by an examination of those performances.

The relative error rate $n_{err}/n_{maps} \in [0, \alpha]$ is given by the number of errors, i.e. collisions, n_{err} in relation to the number of maps n_{maps}. It is shown in Fig. 4.14 where it is partitioned according to condition and *ratio*. As each *ratio* appears exactly four times per condition, $n_{maps}(ratio, condition) = 60$ in each set of 15 runs.

The blind policy refers to the choice the robot makes in case no knowledge about the objects is available at all. In the `limited` condition (Fig. 4.14(left)) the learned blind policy of all ϖ_{inf} is for all cases of v_o to take the *short way*. As a result the three ϖ_{inf} yield especially for a *ratio* of 1/3 and 1/5 a lower error rate compared to ϖ_{rand} and ϖ_{def}, which both choose randomly in this situation. For a *ratio* of 3/1 and 5/1 the ϖ_{inf} outperform ϖ_{rand} and ϖ_{def} only for $v_{o,1}$ (Fig. 4.14(a)). In the less dynamic environments all policies yield quite similar error rates given these two ratios. The reason is that for $v_{o,1}$ the collision risk is much higher such that the robot has only a very small chance to successfully complete the *way* with three or more objects. This leads to a stronger importance of the actual *way* choice on the performance in this setup. Even though the initial choice of the ϖ_{inf} in the `limited` condition and for $ratio \in \{3/1, 5/1\}$ is the more crowded *way*, in most situations the robot switched its decision in a second trial. More specifically, it switched in 59% for $v_{o,\frac{1}{4}}$, in 88% for $v_{o,\frac{1}{2}}$, and in 95% for $v_{o,1}$. So obviously higher dynamic in the environment lead to faster decision changes. In general, in around 70% of the decision changes the robot with an ϖ_{inf} took the *way* with less objects and in average 80% of the switches in the `limited` condition occurred in maps with a *ratio* of 3/1 or 5/1.

In the environments with slower objects the robot has even on a more crowded *way*

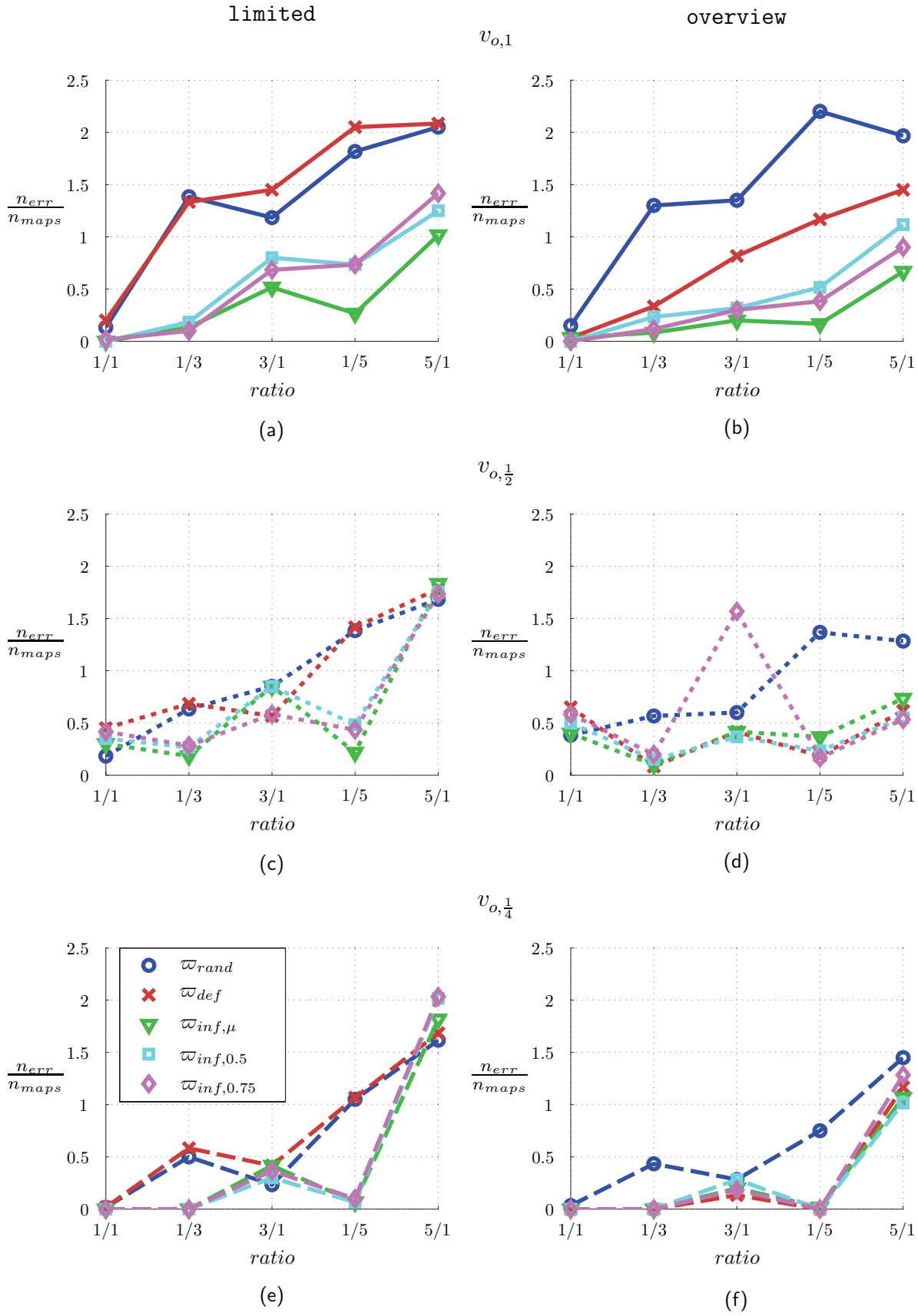


Fig. 4.14.: Number of errors n_{err} in relation to the number of maps n_{maps} for all ratios: shown for the `limited` (left) and `overview` condition (right) and all object speeds v_o (top to bottom).

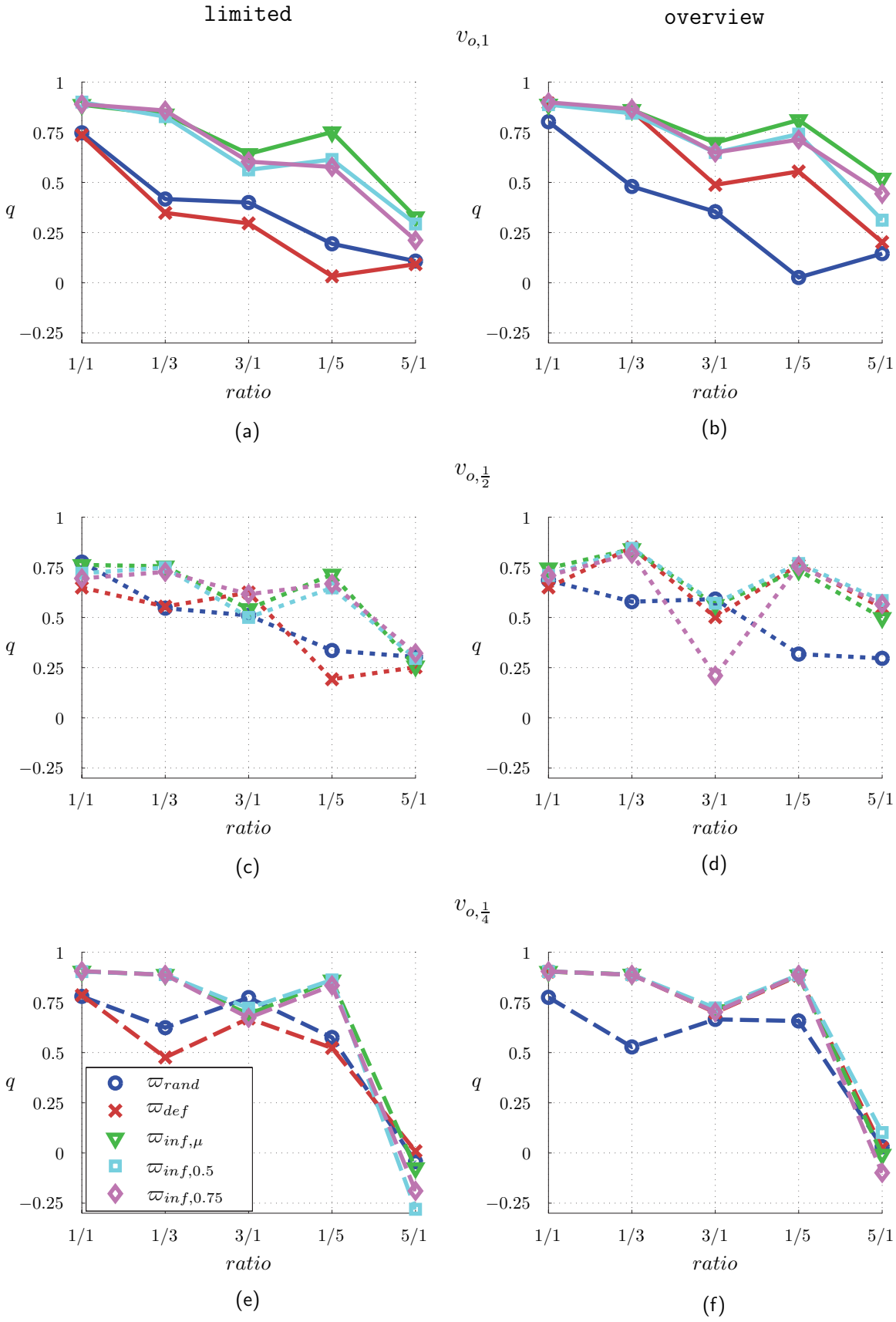


Fig. 4.15.: Performance $q \in]-\infty, 1]$ achieved per ratio: shown for the `limited` (left) and `overview` condition (right) and all object speeds v_o (top to bottom).

a realistic chance for a successful run, which is observable in the on average lower error rates (Fig. 4.14(c) and (e)) compared to the $v_{o,1}$ situation (Fig. 4.14(a)). Nevertheless, the success chances on the *way* with only one object are still higher and since all ϖ_{inf} take for $ratio \in \{3/1, 5/1\}$ initially the more crowded choice they are not able to surpass the random selection strategy in these situations.

For the `overview` condition, the learned initial policy is mostly similar to the defensive policy. Only $\varpi_{inf,0.5}$ deviates in the $v_{o,\frac{1}{4}}$ and $\varpi_{inf,0.75}$ in the $v_{o,\frac{1}{2}}$ situation by initially selecting the *short way* in a *ratio* of 3/1. The choice of $\varpi_{inf,0.75}$ is also observable by the comparably large error rate in Fig. 4.14(d). This will be further discussed later on. Except for this outlier, for the `overview` condition the error rates of the three ϖ_{inf} and ϖ_{def} are quite similar what is also intuitive as their choice in the first trial is mostly the same. Only ϖ_{rand} has more failed trials as it is the only policy that does not take the given information into account at all.

However, in the current application the robot is not supposed to primarily minimize the error rate but rather to maximize the performance q given by the average achieved reward shown in Fig. 4.15. It is observable that the q-curves strongly reflect the course of the respective error rates, which indicates the large influence of the error rate on the resulting performance in this specific application. This leads to the conclusion that in the current scenario the heuristically chosen defensive policy ϖ_{def} is already a quite good choice. This is also identified by the proposed approach as for all ϖ_{inf}, except for the two cases mentioned above, for the `overview` condition exactly ϖ_{def} was learned as initial choice. However, the error rate is not of primary interest.

The performance is the measure to be optimized. Even though the error rate has a strong influence, it does not fully determine the performance as for example observable in the `limited` condition for $ratio = 5/1$. While the respective n_{err}/n_{maps} of the $v_{o,\frac{1}{4}}$ compared to the $v_{o,\frac{1}{2}}$ situation are fairly similar (Fig. 4.14(c) and (e)) the respective performances (Fig. 4.15(c) and (e)) differ quite a lot. So while all policies yield a $q \approx 25$ in case of $v_{o,\frac{1}{2}}$ (Fig. 4.15(c)), the performance gets even negative in the less dynamic case (Fig. 4.15(e)). This is ascribable to the circumstance that the robot failed frequently for $ratio = 5/1$, even in the environment with $v_{o,\frac{1}{4}}$. However, in the latter it collided more often later, e.g. with the fourth or fifth object, in contrast to the environments with faster objects where it mostly collided already with the first or second one. As a consequence, the failure costs $c_f(t_r)$ are higher in the less dynamic case due to more lost time. This in turn results in a lower performance even though the error rates are similar to the ones of the $v_{o,\frac{1}{2}}$ environment.

Considering the average performance shown in Fig. 4.16 it is observable that the inference-based policies yield in average a better performance q than ϖ_{def}. This results mainly from the fact that, according to (4.30), both ϖ_{inf} also make use of the experience from prior failed trials. Especially in the `limited` condition this yields a significantly better q compared to the random choices.

In summary, it is shown that the inference-based policies were able to outperform the heuristic ones, even though in the current case ϖ_{def} was already a very good heuristic choice. This verifies the strong benefit of a situation-aware action selection that takes

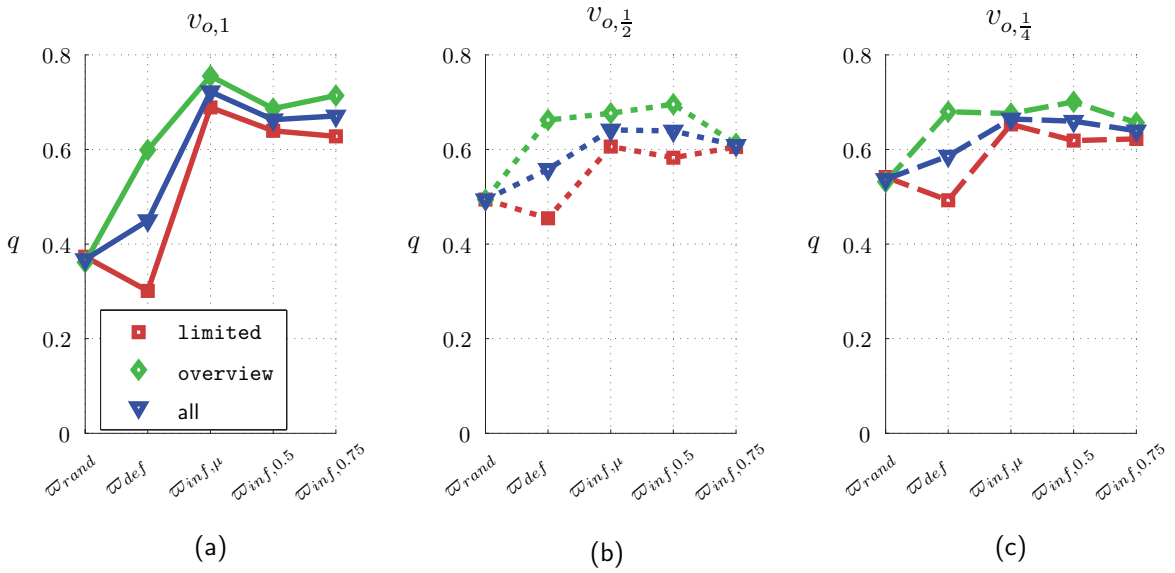


Fig. 4.16.: Performance $q \in]-\infty, 1]$ achieved in average by the used policies after 15 runs, shown for each condition separately as well as the combination of both.

the available knowledge about environmental factors and those mutual interdependencies already during the cost and reward estimation into account. However, as discussed in Section 4.5.1 besides the actual reward maximization also the related reliability of its estimation is of importance for an efficient cooperative action selection.

4.5.3.2. Reliable Reward Estimation

The motivation to use a quantile-based estimation originated from the demand for a reliable and risk-aware estimation, which is according to Definition 4.2 determined by the ratio of reward underestimations. The achieved reliabilities of the three inference policies are shown in Fig. 4.17 for all used v_o.

The quantile-based policies $\varpi_{inf,0.5}$ and $\varpi_{inf,0.75}$ reach their target reliability of 0.5 and 0.75, except for $rel(\varpi_{inf,0.75}) = 0.73$ in the $v_{o,\frac{1}{4}}$ environment, which is slightly below (Fig. 4.17(c)). While such small deviations are a result of the underlying stochasticity, the larger overstepping in some cases is ascribable to the discretization of t_r and l_p. However, the discretization may only lead to a more risk-averse behavior as shown by (4.33).

From Fig. 4.17 is further observable that also for $\varpi_{inf,\mu}$ all values are larger than 0.5, even though $\varpi_{inf,\mu}$ does not take its $rel(\varpi_{inf,\mu})$ into account at all. This indicates that the outcome distribution of f_{t_r} is asymmetric as for a symmetric distribution a $rel(\varpi_{inf,\mu}) = 0.5$ is to be expected. While in the current case the asymmetry leads also for $\varpi_{inf,\mu}$ to a risk-averse estimation, in other setups exactly the opposite might be the case such that rel gets very small. That the quantile-based policies $\varpi_{inf,\varsigma}$ yield also in such situations their dedicated target reliability is for example observable in Fig. 4.17(c) for the `overview` condition, where $rel(\varpi_{inf,0.75}) > 0.75$ while $rel(\varpi_{inf,\mu})$ is around 0.5.

While these results verify that the quantile-based policies are able to yield the desired risk-averse reward estimation, the $\varpi_{inf,\varsigma}$ also bare the risk of being less optimal than

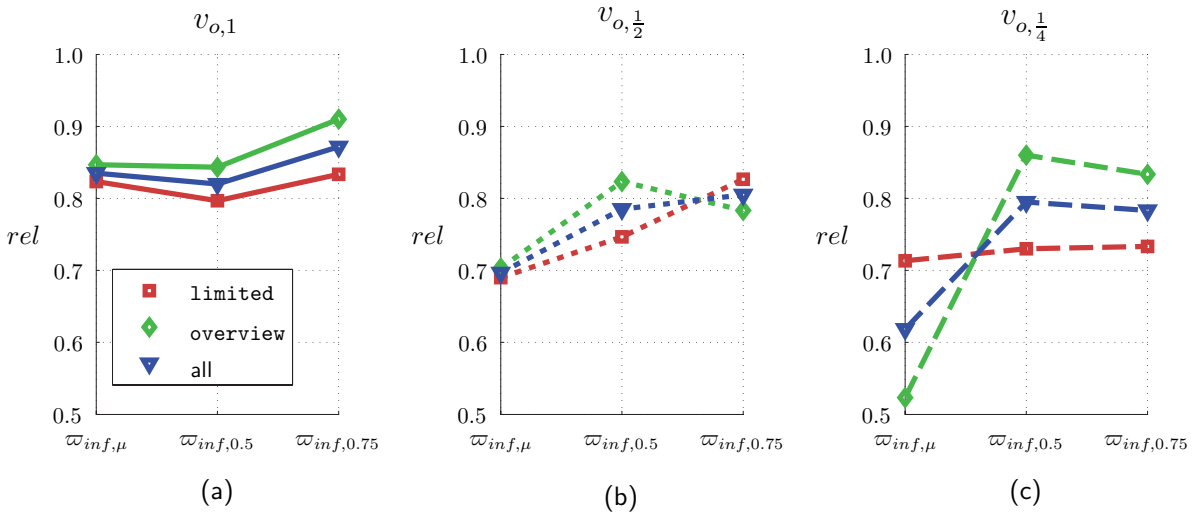


Fig. 4.17.: Reward reliability achieved by the inference policies: shown in the `limited`, the `overview` and both conditions for each object speed v_o (left to right).

$\varpi_{inf,\mu}$, since in contrast to the latter $\varpi_{inf,\varsigma}$ performs a multi-objective optimization where the reward receives only second priority. Considering Fig. 4.16, in all environments the overall performance of the $\varpi_{inf,\varsigma}$ is indeed below $\varpi_{inf,\mu}$, where the maximum deviation is 8%, 5%, and 4% for $v_{o,1}$, $v_{o,\frac{1}{2}}$, and $v_{o,\frac{1}{4}}$ respectively. This indicates the tendency that higher dynamic also leads to a larger deviation from the best achievable performance $q(\varpi_{inf,\mu})$.

To further clarify the cause of such deviations it is in the following illustrated by an exemplary case. As previously mentioned, in two specific situations the inference-based policies deviate from the mean-based one as they initially select the *short way* in a *ratio* of 3/1. The case striking out most is the one with $\varpi_{inf,0.75}$ in the $v_{o,\frac{1}{2}}$ situation as it led to a three times higher error rate (Fig. 4.14(d)) and a performance drop of more than 50% compared to $\varpi_{inf,\mu}$ (Fig. 4.15(d)) for this specific situation. The reason why the robot chose the apparently worse *way* gets clear by considering the conditional distribution $\mathsf{f}_{t_r}(t_r \mid status, 3/1, way)$, which is required in (4.29) and (4.28) – to compute $\hat{c}\left(\hat{t}_r \mid status, ratio, way\right)$ – and shown in Fig. 4.18. For the case of failure on the *short way* (Fig. 4.18(c)) is observable that already for the first interval $\Delta t_{r,1} = [0s, 10s]$ holds $P(t_r \in \Delta t_{r,1}) > \varsigma = 0.75$. So in the cases where the robot failed, the collision occurred mostly in the beginning. Accordingly is the constraint of (4.32) already satisfied by $\Delta t_{r,1}$ such that $Q_{t_r,\mathfrak{d},short}(0.75) = 10s$ is set to the larger bound of the interval. Instead, for the *long way* (Fig. 4.18(d)) holds $P(t_r \in \Delta t_{r,4}) > (1 - 0.75)$ for the interval $t_{r,4} =]30s, \infty s[$, from that follows that $Q_{t_r,\mathfrak{d},long}(0.75)$ lies within the fourth interval. In this case a maximal upper bound of $40s$ has been selected in the course of the prior discretization.

Even though the probability of failure $P(failed \mid 3/1, short) = 0.73$ is higher for the *short* than for the *long way* – $P(failed \mid 3/1, long) = 0.53$ – the large difference in the failure cost dominates the reward estimation in this case what led the robot choose the *short* path. In order to handle such occasions a larger number of discretization intervals and/or more training data may be used. However, it needs to be considered that more intervals lead to a higher model complexity and moreover also imply the second, more

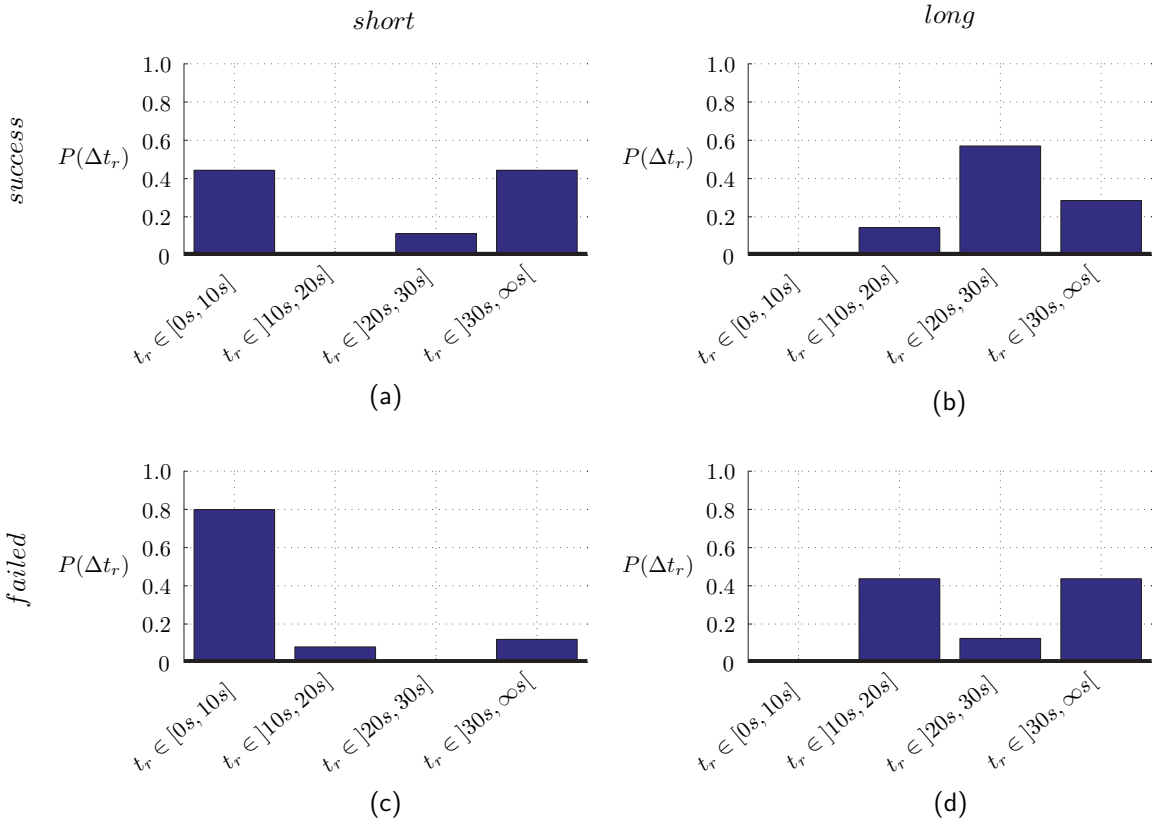


Fig. 4.18.: The conditional distribution $\mathsf{f}_{t_r}(t_r \mid status, ratio = 3/1, way)$ of the cost metric t_r in the $v_o = 0.3\frac{m}{s}$ environment: partitioned into $status \in \{success, failed\}$ (top to bottom) and $way \in \{short, long\}$ (left to right).

training data, which is especially for real-world systems often not easily retrievable. In this respect, even though in the current example more training data and a smoother discretization could have been easily used, this was refrained on purpose in order to verify the suitability of the described approach also for systems where only little data is available. This specific example exemplified the action selection and the related reward estimation on a single case.

Next the performance on a whole is examined in relation to the uncertainty of the environment. In order to obtain a general measure of the estimation quality relative to the given uncertainty the ratio of the mean square error $\mathrm{MSE}(\hat{\varrho})$ of the expected reward compared to the variance $\mathrm{Var}(\varrho)$ of the achieved reward may be used. It is shown in Fig. 4.19 for both conditions. While the ratios for $\varpi_{inf,\mu}$ and $\varpi_{inf,0.5}$ are quite similar $\varpi_{inf,0.75}$ results in a larger relative MSE. This is explainable as follows. In contrast to $\varpi_{inf,\mu}$ aims the quantile-based estimation not at minimizing the MSE, such that a higher ratio is to be expected with larger ς. Even though the variance in the estimation was except for one case still lower than the variance in the actual outcome. Furthermore, the above discussed trend that the approach yields better results with lower environment dynamic is also observable in Fig. 4.19.

In summary the quantile-based policies yielded the desired risk-averse estimation for the sacrifice of a minor local performance decrease. However, this related only to the

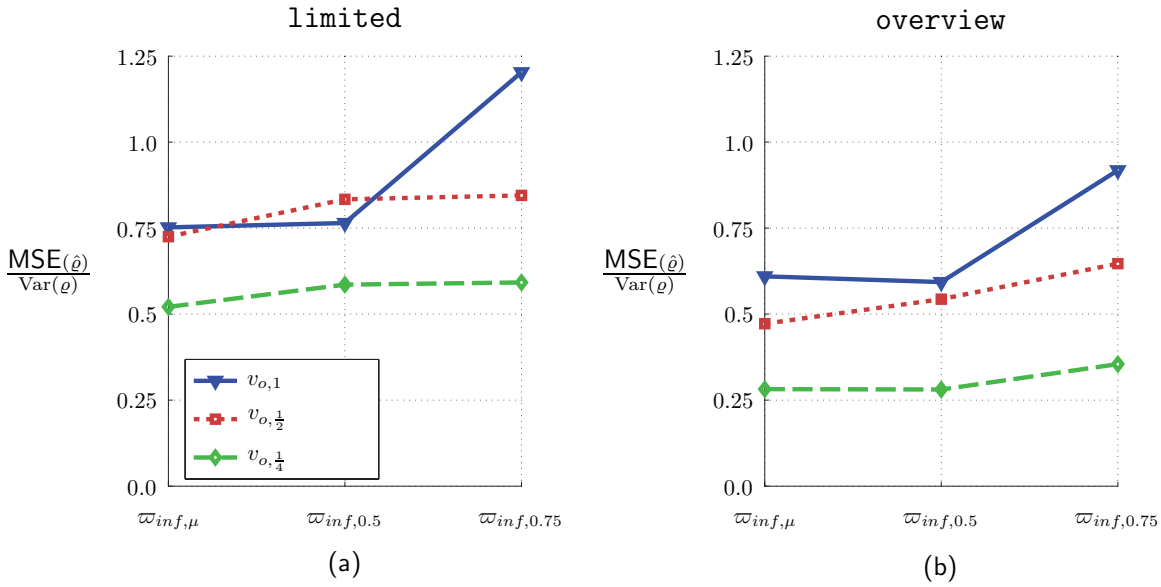


Fig. 4.19.: Mean squared error (MSE) of the reward estimation $\hat{\varrho}$ in relation to the variance of the achieved reward ϱ.

performance of a single robot. As shown by the exemplary cost function in (4.23) this may still lead to a better global performance of the entire MRS. A further improvement is achievable by an early enough forecasting of an upcoming performance deterioration.

4.5.3.3. Forecasting of Poor Performance

In Section 4.3, four aspects were formulated that are essential for a persistent action selection. Three of these aspects have been discussed so far. The fourth relates to a failure-forecasting during the execution by observation of the current cost deviation $\Delta c(a \mid \boldsymbol{s})$ and the ability of adequate counteractions. Even though the actual execution of actions is not in focus of this work a brief example is given how this may be achieved.

In general, the robot aims to maximize the reward achieved per map, which results according to (2.6) (p. 22) from the difference between obtained utility and arising costs. Assuming the robot has now as additional action the option to decide before every new trial, whether it wants to continue with the current map or to proceed with the next one. In case it decides to stop with the current map it will neither receive the utility nor cause any further costs resulting in $\varrho = 0$ for the next trial. So in case the robot expects for the latter a $\hat{\varrho} < 0$ stopping is most probably the better choice.

This scenario was examined by respectively post-evaluating the results from the previous part. However, in the current application such an additional action had almost no impact since $\hat{\varrho}$ was in most cases positive. Only in the environment with high dynamic ($v_{o,1}$) the robot with $q_{\varpi_{inf,0.75}}$ would have achieved a performance gain of 16.5%. Nevertheless, in general is to expect that in environments with even higher dynamic or by selecting a lower utility value the potential gain and thus the importance of such a strategy become considerably higher.

4.5.4. Discussion

A situation-aware and risk-concerted reward estimation by usage of a quantile-based cost computation has been described. Benefits of the presented method are that nonlinear effects of the environment on the cost metric are taken into account. Furthermore, a single model of the environment suffices to serve multiple types of inference requests, keeping the demand for training data comparably low.

The experimental results showed that the proposed inference-based policies were able to outperform the heuristic alternatives, even though a very good heuristic choice was already made. The quantile-based policies reached the desired risk-aversion levels for the sacrifice of some local performance decrease. Finally an outlook on how to avoid an upcoming performance deterioration by an adequate failure-forecasting was given.

In summary the strong benefit of a situation-aware action selection that takes the available knowledge about environmental factors and those mutual interdependencies already during the cost and reward estimation into account has been verified. Furthermore has been shown that the risk of not reaching the estimated rewards is reducible to a pre-desired level.

The possibilities to incorporate Proposition 4.3 into a multi-robot optimization are expected to be manifold and also depend on the specific application and the respective cost function. A strong benefit is for example expectable in scenarios in which frequently multiple global optima w.r.t. the criteria of first priority are found, or in applications that are faced with the tradeoff between efficiency and risk. Such typical cost functions may be similar to (4.23) and relate for example to scenarios where the robots are supposed to act time-optimal. In these cases MRSs are often faced with situations where the robots have to wait for each other such as in cooperative surveillance or joint manipulation tasks for example.

Consequential next steps in this respect would be to combine the proposed cost estimation with the cooperative planning frameworks by focusing on a suitable adaptation of the risk-aversion level in order to optimize the overall MRS performance. In order to give an exemplary outlook how a usage of Proposition 4.3 in a MRS may look like, one potential incorporation into the framework MuRoCo – described in Chapter 3 – is demonstrated. It is based on the previous exemplary cost function (4.23) that is formulated more generally by the cost

$$c_m(p_{\mathcal{T}}(\tau)) = \max_{\forall \tau_j \in p_{\mathcal{T}}(\tau)} c(\tau_j)$$

for action plan $p_{\mathcal{T}}(\tau)$. In other words $c_m(p_{\mathcal{T}}(\tau))$ is given by the largest cost of its subtasks τ_j. Accordingly follows for the set $\mathcal{Z}_p$ of pareto-optimal coalitions

$$\mathcal{Z}_p = \left\{ z \in \mathcal{Z}_{cap} \,\middle|\, c_m(z, p_{\mathcal{T}}(\tau)) = \min_{z \in \mathcal{Z}_{cap}} \left(\max_{r_i \in z} c(r_i, \tau_j) \right) \right\},$$

where $\mathcal{Z}_{cap}$ is the set of all capable coalitions according to Definition 3.12 (p. 40). $\tau_j = \psi_{z,p_{\mathcal{T}}(\tau)}(r_i)$ is the subtask $\tau_j \in p_{\mathcal{T}}(\tau)$ assigned to r_i according to the subtask assignment $\psi_{z,p_{\mathcal{T}}(\tau)}$ of coalition z, as described in Section 3.4.4 (p. 42). A specific coalition

among the pareto-optimal solutions $\mathcal{Z}_p$ may be then derived by

$$z^* = \arg\max_{z \in \mathcal{Z}_p} \left(\min_{r_i \in z} rel\left(c(r_i, t_j)\right) \right),$$

i.e. the coalition with the least low reliability. In this respect the coalition is chosen that mostly guarantees to achieve its promised reward. Note that in case the robot with least low reliability is member of multiple coalitions this returns a pareto-optimal set as well. These cases may be handled by also optimizing w.r.t. the robot with second lowest reliability and so on. This example demonstrates the benefit derivable from the proposed approach. A further investigation of respective opportunities goes beyond the scope of this thesis but presents promising perspectives for future research.

4.6. Summary

A group of robots can improve their joint performance by a cooperative selection of actions. This demands a joint planning in order to find a consensus on the actions to be executed by each single robot. In this respect, a major problem of robotics is, that the performance estimation during the planning phase commonly deviates from the actual performance, which is achieved during the subsequent execution. This chapter focused on this, often disregarded, yet fundamental problem and presents a novel bipartite approach for a situation- and uncertainty-aware performance estimation. The questions in focus are to identify the crucial factors that influence the performance most, to learn how the performance is influenced and finally to use this learned information for a better decision making in order to improve the system performance. The entire approach was validated in extensive experiments.

More specifically, the first major contribution is a method for a system interdependence analysis, which quantitatively evaluates the interdependencies among indicators of the internal robotic system, as well as, the environment. The analysis enables not only the determination of the limitations of an autonomous robotic system, but also insight on what influences its performance most. It further examines whether the current situation has an effect on the accuracy of the cost estimation and if this effect can be taken into account beforehand. The system interdependence analysis was exemplary applied to the autonomous mobile robot *ACE*, based on experimental data from an extended field experiment that is described in Appendix B. For the *ACE* robot was shown, which of the proposed indicators were suitable to represent the system performance and which were not. Furthermore, the influence of the environment on these suitable performance indicators, for example the map uncertainty, could be identified.

Such acquired knowledge enables the consideration of the influences occurring from the environment and also from the tasks executed by other robots in parallel. This may help the designer to guide future research and/or, as shown in the second part of this chapter, can be utilized by the system itself for an improved cost estimation. In this context, the second major contribution is a quantile-based cost estimation, which enables the consideration of large variances and even of asymmetries in the respective indicator

distributions. Accordingly, the quantile-based policies take the characteristics of the cost deviations into account and thereby yield a more reliable action selection, where reliability is defined as the probability of reaching the estimated rewards. The method was tested in a highly dynamic environment. The experimental results showed that the method is able to outperform the heuristic alternatives, even though here a very good heuristic choice was already made. Also, the desired risk-aversion levels were reached, for the expected sacrifice of a minor local performance decrease.

The benefits of the proposed approach are that it requires only few training data and allows for a rather coarse parameter discretization, which keeps the model complexity low. Moreover, the approach allows for the inclusion of any given prior knowledge about the robot or the environment. Such information can be incorporated within the indicator prioritisation, for the model structure and/or the parameters, or during the online operation through indicator evidence. Nevertheless, the methods are still applicable to learn models also from scratch, i.e. without any prior knowledge. A further advantage, compared to regression models, for example, is that only a single model needs to be learned to serve various types of inference requests. A limitation of the approach is the environment-dependence of the models that always demands an on-site model learning in order to take location-specific factors into account. This makes it not suitable for most exploration scenarios for example. Further is assumed that input and output are observable. Additionally, the presented methods are not capable to handle situations where the output responds erratically to the input.

Future work may concentrate on extending the model parameters for continuous distributions. This would remedy quantization errors but probably demand a greater amount of training data. A further beneficial extension is an online model update. This would extend the applicability to unexplored environments or areas that are subject to long-term changes. Respective literature may be found in the field of incremental learning. A large potential is to be expected by an integration of the quantile-based estimation into procedures for multi-robot decision making, such as the task allocation framework presented in the preceding chapter. Particularly in cooperative MRSs where robot-robot interaction takes place, the performance, as well as the success probability, of a single robot is influenced by the respective measures of the other robots. In this respect for example, an adequate consideration of the cost reliabilities may be used to determine better coalition matches, as discussed in the previous section. Generally, the presented approach is suitable for many robotic systems that are faced with the discrepancy between performance maximization and risk minimization. In case the functional relationship between performance and risk is not known, which is to be expected for many real-world systems, the introduced approach enables the approximation of this relationship and utilization of this knowledge for a risk-aware decision making. In this respect, an adaptive selection of the desired reliability level may be investigated, in order to achieve a desired tradeoff between risk and efficiency.

5. Conclusion and Outlook

This thesis investigated the problem of a cooperative action selection in multi-robot systems (MRSs). This problem arises in scenarios where multiple robots, which act simultaneously in the same environment, can benefit from cooperation. As discussed in Chapter 1, along with the progress in robotics, such scenarios gain more and more in importance, especially in the area of service robotics. In order to bring such robotic systems from laboratory settings closer to their intended operational areas, this thesis introduced solutions to overcome the challenges of an efficient and robust action selection in cooperative real-world MRSs. In this respect, the presented solutions contribute primarily to a more efficient and reliable handling of these challenges, namely the rapidly growing problem complexity and the existing information uncertainty.

More specifically, the MuRoCo framework for a multi-robot task allocation was introduced, which yields a robust action selection and a better scaling of the problem complexity. To take the information uncertainty into account, a bipartite approach was proposed, which first learns a probabilistic cost model that is then used during system operation by a more reliable and situation-aware performance estimation. In the following, these contributions are discussed in more detail and an outlook on promising directions for future research is given.

5.1. Conclusion

In Chapter 2, this thesis provided a generic formulation of the action-selection problem and gave a structured illustration of how respective solvers can be integrated within an overall operating system for MRSs. In this respect, cooperative action selection refers to the choice of actions by multiple robots in order to maximize the joint performance of the group. By selecting appropriate sequences of actions, the robots complete dedicated tasks and obtain respective rewards. In general, the feasible sequences of actions that lead to the completion of specific tasks can be described in plans. While the plans themselves are assumed to be situation-independent, the quality of those execution is not. As a consequence, all possible plans can be pre-computed offline and stored in the knowledge base of the robots. Accordingly, the actual problem during the robotic operation – and thus the one under focus in this thesis – is to choose the plan which is best in a specific situation. Concerning this problem, the major challenges arise from the related complexity and the given information uncertainty.

The arising complexity relates mainly to the multi-robot task allocation. In the field of multi-robot task allocation the primary challenge is not to determine approaches that are able to solve the problem, but approaches that scale well with the number of robots and tasks. The market-based task allocation framework MuRoCo, introduced in Chapter 3,

contributes in this respect, by yielding a lower bound of the worst-case complexity for the formation of robot teams, so-called "coalitions". By splitting the subtask and the task assignment, the required computational effort is reduced while still guaranteeing the solution quality of an exhaustive search. In other words, MuRoCo yields optimal solutions for the sequential assignment of tasks that require a tight cooperation of multiple robots (ST-MR-IA), commonly known as "coalition formation". However, employed MRSs are usually seldom confronted with worst-case situations. Instead, they are more frequently faced with situations where part of all possible constellations is not only suboptimal, but even unfeasible. This is exploited in MuRoCo by several pruning strategies, which identify infeasible candidates in an early phase and thereby reduce the number of potential solutions to be evaluated. The resulting computational savings have been examined in a benchmark evaluation, which is – to the author's best knowledge – the first quantitative evaluation of such a framework. Moreover, of further importance for an efficient operation of MRSs in practice, are adequate means to handle the environmental uncertainty and execution failures. Depending on the dynamic characteristic of the environment, such means may turn out to be most crucial with respect to the system performance. MuRoCo incorporates respective means for failure forecasting and handling, which aim to prevent errors and, in case of occurrence, allow for a fast replanning. This enables a robust operation of complex multi-robot systems, which also has been verified in a service scenario with a MRS of four heterogeneous robots. Overall, MuRoCo presents an exhaustive framework that is based on a generic problem formulation and has been developed for the employment in real-world systems. This makes it suitable to a broad range of applications and domains. The given results have shown, that the benefits of MuRoCo are, especially in systems with high heterogeneity and/or ones operating in highly dynamic environments, of importance. Even though, industrial or military applications provide these characteristics to a greater or lesser extent as well, the primary scope of MuRoCo is to be expected in the field of service robotics. In these settings, the processes are commonly less predefined but rather determined by the current needs of humans. This matches the strengths of MuRoCo, namely a situation-aware, efficient, and robust, action selection in real-world settings.

As previously mentioned, MuRoCo incorporates means to avoid and to handle failures. Of significant importance is the ability to handle such hard errors but also of soft deviations, between the expected and the achieved performance. Even though this problem is of high importance for an efficient action selection, as the accuracy of the cost/reward estimation strongly determines the validity of the planned solutions, it has been often neglected in literature so far. This thesis highlighted the importance of this problem, provided a structured problem statement, and introduced a novel approach to handle the problem in highly dynamic environments. In this respect, the risk- and situation-aware performance estimation, described in Chapter 4, provides a generic solution to learn the environmental influence on the robotic system and utilize the gained knowledge for an improved action selection. An approach is introduced, to determine the influence of environmental factors on the system performance and thereafter utilize this knowledge to yield a more accurate cost and reward estimation. The influence is modeled in form of a Bayesian network, which is learned offline based on previously gathered data. One major advantage is that nonlinear effects of the environment on the cost metric are taken into account.

The gained knowledge is then used during online system operation to yield a situation-aware and risk-concerted estimation of the future performance. Besides a mean-based estimation, a quantile-based cost and reward estimation is proposed that enables the consideration of large variances and even of asymmetries in the respective indicator distributions. This allows for a risk-adjusted decision making, which means a regulation of the probability to over- or underestimate the achieved costs. In the experimental evaluation, the proposed inference-based policies achieved better results than the heuristic alternatives, even though the latter performed quite well. The quantile-based policies reached the desired risk-aversion levels for the expected sacrifice of some local performance decrease. A further major benefit of the approach is that a single model of the environment suffices to serve multiple types of inference requests, thus keeping the demand for training data comparably low. The approach is highly generic as it enables the incorporation of any prior knowledge that is given about the robot or the environment. Respective information can be included in the form of the indicator prioritisation, the model structure and/or the parameters, or during the online operation in form of indicator evidence. Still, in case no knowledge is given, the methods are perfectly applicable to learn models from scratch. Furthermore, a single model is sufficient to serve various types of inference requests. This presents an essential advantage compared to regression models, for example. A restriction of the approach is that the models are only valid in the respective environment. Accordingly, in order to take location-specific factors into account, an on-site model learning is required, which makes the approach not suitable to exploration scenarios, for example. Further assumptions are that input and output are observable and that the output does not respond erratically to the input. The method provides an alternative in comparison of deriving the deterministic system model, what may be quite hard for complex systems, or it also can be used to verify the latter. Besides the direct incorporation of the gained knowledge during system operation, it also may help the designer to guide future research. In this respect, the method is also generic enough, to be applied to many robotic systems that are faced with the discrepancy between performance maximization and risk minimization. In case the functional relationship between performance and risk is not known, which is to be expected for many real-world systems, the introduced approach enables the approximation of this relationship and the utilization of this knowledge for a risk-aware decision making.

5.2. Outlook

Finally, possible directions of future work are proposed, to point out opportunities on how to act on the presented approaches in order to extend these by new functionalities or apply them to new fields of application. In the domain of multi-robot task allocation, still part of future research remain adequate approaches to tackle the very hard problem of the simultaneous assignment of a set of multi-robot tasks. Similarly, so far no approach to efficiently handle the assignment to multi-task robots (MT-MR-IA) is known. However, the relevance of multi-tasking in real applications may be limited due to physical constraints and thus should be explicitly considered in advance. Even though the worst-case complexity for the coalition formation could be lowered by MuRoCo, and feasible system sizes will increase

with the rising power of computers, it is still insufficient for large-scale systems. As discussed in Chapter 3, in such settings, heuristics are required to focus the search on the most relevant candidates. The usage of robot clusters is promising for practical systems, as for example the negotiation limited amongst robots in the vicinity. Another idea is to prioritize past solutions as they might be good candidates for future assignments as well. However, one needs to bear in mind that such heuristics always entail the sacrifice of optimality. How close the found solutions are to the optimum is strongly determined by the characteristics of the specific practical setting. As a consequence, a scenario-independent valuation of the suitability of specific heuristics can not be given.

Concerning the system interdependence analysis, during its evaluation the performance indicators had to be chosen application-dependent in order to demonstrate the proposed approach. In this respect, further steps should concentrate on the generality of indicators, in the sense of their suitability for representing the performance of different purpose systems. This would allow the specification of application-independent benchmark tests with respect to system robustness, in order to facilitate system comparability. Concerning the method itself, different algorithms for the BN structure search may be evaluated, for example whether they provide a better tradeoff between complexity and solution quality. This would improve the scalability of the approach. Additionally, instead of a static network, dynamic Bayesian networks may be learned, which allow the examination of temporal interdependencies of dynamic systems. Also, extending the model parameters for continuous distributions, as well as an autonomous determination of the discretization levels, and an online model update, would provide further flexibility to the approach.

The introduced risk-aware cost estimation creates manifold opportunities for future research. It provides a very generic approach that is not limited to multi-robot applications but applicable to single-robot scenarios, which are likewise faced with the tradeoff between efficiency and risk. A promising direction for future research is a tighter interconnection of the presented inference-based cost estimation with action selection and/or planning methods. In MRSs, large potential is to be expected by an integration of the quantile-based estimation into the presented action planning frameworks. Taking the cost reliabilities during the action selection into account, for example by adaptively setting the reliability level, may help to determine better coalition matches as discussed in the previous chapter. For robotic systems in general, a better understanding of the current environmental situation allows for a more focused search in the space of possible solutions and an increased validity of the estimated quality. In this respect, the proposed approach enables opportunities to combine a better understanding of the situation with the robotic reasoning capabilities in order to achieve a more cognitive system behavior.

A. Distribution of the Coalition Responsibilities in MuRoCo

In Section 3.4.4 a method for the assignment of multi-robot tasks is described. During its initialization phase Algorithm A.1 is used to distribute the coalition responsibilities equally among the robots. As each robot $r \in \mathcal{R}$ is part of $2^{\mathcal{R}-1}$ coalitions the total number of coalition members, i.e. the sum of all coalition sizes $|z|$, results in $|\mathcal{R}|2^{\mathcal{R}-1}$. Distributing this again equally among the robots leads to $2^{\mathcal{R}-1}$. Since Algorithm A.1 ensures that each robot obtains exactly those $2^{\mathcal{R}-1}$ coalition members to evaluate, the computational effort is theoretically the same for all robots assuming that all coalitions need to be evaluated. The time complexity of Algorithm A.1 scales exponentially with the number of robots $|\mathcal{R}|$, but it is only required at the system start or when $\mathcal{R}$ changes.

Table A.1 shows all possible coalitions for an exemplary multi-robot system composed of four robots. Table A.2 shows the respective sets of coalition responsibilities, which are obtained with Algorithm A.1 after four iterations.

Robots	Coalitions														
	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
r_1	✓	✓	✓	✓		✓	✓	✓				✓			
r_2	✓	✓	✓		✓	✓			✓	✓			✓		
r_3	✓	✓		✓	✓		✓		✓		✓			✓	
r_4	✓		✓	✓	✓			✓		✓	✓				✓
$\lvert z\rvert$	4	3	3	3	3	2	2	2	2	2	2	1	1	1	1

Tab. A.1.: List $l_{\mathcal{Z}}$ of coalitions for a multi-robot system of four robots.

Iteration	Coalition Responsibilities			
	$\mathcal{Z}_{\text{resp},r_1}$	$\mathcal{Z}_{\text{resp},r_2}$	$\mathcal{Z}_{\text{resp},r_3}$	$\mathcal{Z}_{\text{resp},r_4}$
1	$\{z_1\}$	$\{z_2\}$	$\{z_3\}$	$\{z_4\}$
2	$\{z_1, z_5\}$	$\{z_2, z_6\}$	$\{z_3, z_7\}$	$\{z_4, z_8\}$
3	$\{z_1, z_5\}$	$\{z_2, z_6, z_9\}$	$\{z_3, z_7, z_{10}\}$	$\{z_4, z_8, z_{11}\}$
4	$\{z_1, z_5, z_{12}\}$	$\{z_2, z_6, z_9, z_{13}\}$	$\{z_3, z_7, z_{10}, z_{14}\}$	$\{z_4, z_8, z_{11}, z_{15}\}$
$\sum_{z \in \mathcal{Z}_{\text{resp},r}} \lvert z\rvert$	8	8	8	8

Tab. A.2.: Sets $\mathcal{Z}_{\text{resp},r}$ of coalition responsibilities for the robots $\{r_1, r_2, r_3, r_4\}$.

Algorithm A.1: Distributes the coalition responsibilities among the robots such that each robot is responsible for $2^{\mathcal{R}-1}$ coalition members.

`distributeCoalitionResponsibilities`($\mathcal{R}$):

```
1  generate list l_Z of all z ∈ Z sorted by descending size |z|;
2  generate list l_R of all r ∈ R;
3  set maxcount = 2^(R-1);
4  set count(r) = 0 for all r ∈ R;
5  set Z_resp,r = ∅ for all r ∈ R;
6  set i = 1;
7  while l_Z ≠ empty do
8      take first z from l_Z;
9      while z unassigned do
10         set r_i to i-th r from l_R;
11         if (count(r_i) + |z|) ≤ maxcount then
12             Z_resp,r ∪← z;                          // assign z to r_i
13             count(r_i) = count(r_i) + |z|;
14         end
15         if i < R then
16             i = i + 1;
17         else
18             i = 1;                                  // next iteration
19         end
20     end
21 end
```

B. The Autonomous City Explorer (ACE) Project: Mobile Robot Navigation in Highly Populated Urban Environments

One of the greatest challenges nowadays in robotics is the advancement of robots from industrial tools to companions and helpers of humans, operating in natural, populated environments. In this respect, the Autonomous City Explorer (*ACE*) project aims to combine the research fields of autonomous mobile robot navigation and human robot interaction. A robot has been created that is capable of navigating in an unknown, highly populated, urban environment, based only on information extracted through interaction with passers-by and its local perception capabilities.

This chapter describes the algorithms and architecture that make up the navigation subsystem of *ACE*. Besides the algorithms used for Simultaneous Localization and Mapping (SLAM) and path planning in uneven dynamic environments, the navigation behaviors of the robot are described. More specifically, the selection and control of these behaviors, as well as the system architecture that integrates all modules into a complete working system are presented. Results from an extended field experiment, where the robot navigated autonomously through the downtown city area of Munich, are analyzed and show that the robot is capable of long-term, safe navigation in real-world settings.

B.1. Motives for the Project

Robots are generally understood as tools, performing well-defined tasks in structured industrial or laboratory settings. One of the biggest challenges of robotics is to evolve them to companions and helpers capable of operating efficiently and safely in natural, populated environments. However, such systems require complex cognitive capabilities in order to achieve higher levels of cooperation and interaction with humans and cope with rapidly changing objectives.

Several systems have been developed in this direction that are capable of autonomously navigating in unstructured terrains, e.g. [116], and more recently in urban environments, e.g. [119]. In all of these approaches, global waypoints, in the form of GPS coordinates, topological information about the route as well as specific mission information are provided in advance. Therefore the robots are not required to interact or cooperate with humans to reach their goals. Human robot interaction on the other side has been mainly investigated in structured indoor environments, leading to the creation of guide or mall robots [41,

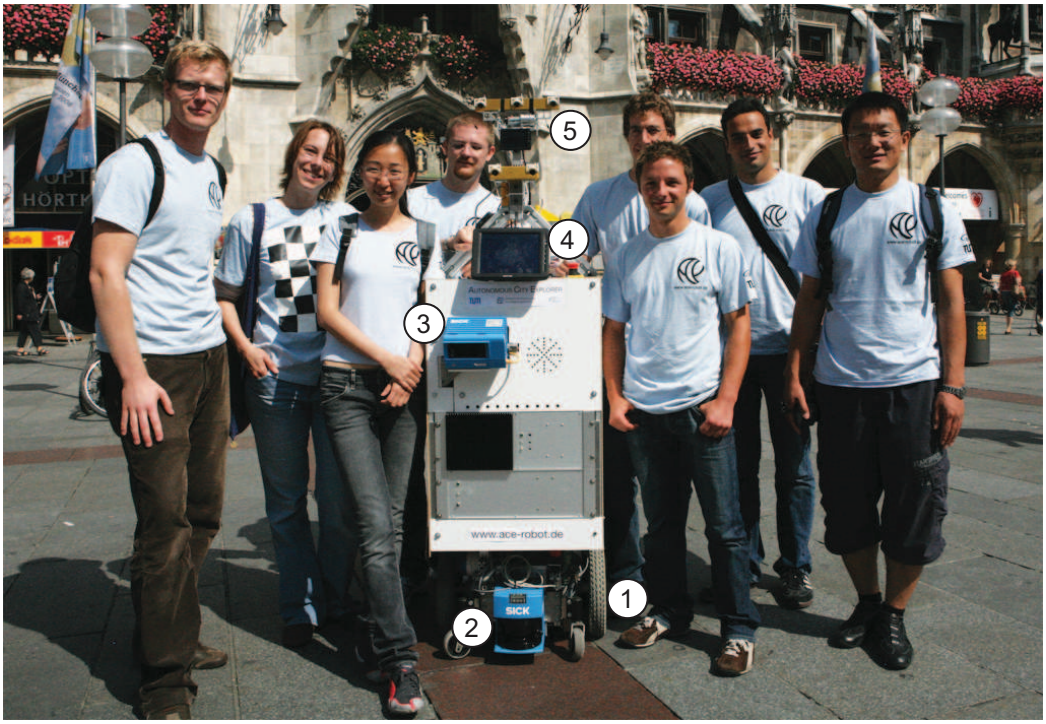


Fig. B.1.: The *ACE* robot and its developer team. Components of the robot: (1) Differential drive mobile platform, (2) LMS200 laser range finder, (3) LMS400 laser range finder, (4) Touch screen, (5) Robotic head with five cameras.

115], which can autonomously interact with people and provide them with information. Complete environment knowledge is again required, simplifying navigation.

The Autonomous City Explorer *(ACE)* project [130] aims to combine these two lines of research. A robot has been created that is capable of navigating in an unknown urban environment, based only on information extracted through interaction with passers-by and its local perception capabilities. More specifically, the project objective was to develop a robot that manages to find its way from the Technical University of Munich to the central square of Munich, without any prior map knowledge or GPS information. Directions are solely obtained by asking pedestrians for the way. The objective was successfully accomplished on 31 August 2008.

In the following the algorithms and subsystems used by *ACE* for reliable navigation and behavior selection, as well as results from the conducted field experiments are presented. Sec. B.2 briefly reviews the hardware and software components of the system. Sec. B.3 highlights how the system can model its environment based on noisy sensor data and use this self-acquired model to navigate safely. Sec. B.4 describes the behaviors that make up the system and how they are selected, coordinated and translated to navigation commands. The latter are executed as described in Sec. B.5. Finally, experimental results are presented in Sec. B.6.

B.2. System Description

The *ACE* robot is based on a differential drive platform capable of carrying a payload of 150kg. Fig. B.1 shows the robot and its components. A SICK LMS200 laser range finder is used for navigation and is mounted with a horizontal scan plane. A SICK LMS400 laser range finder is mounted at a 45° angle from the plane. This is deployed for terrain traversability assessment and curb detection. For interaction with humans a touchscreen and speakers are used. A robotic head which comprises of five cameras and an animated mouth is mounted on the robot, for gesture recognition and people tracking [78]. More details on the hardware components can be found in [130].

The software architecture of the navigation and behavior selection subsystem of the *ACE* robot, is illustrated in Fig. B.2. It is divided into three layers: (i) *Processing Layer*, (ii) *Control Layer* and (iii) *Execution Layer*.

Raw sensor data are given as input to the *Processing Layer* of the architecture, which is described in Sec. B.3. The SLAM module processes the data in order to create a representation of the environment. This representation is fused with a grid containing terrain traversability information, produced by the *Traversability Assessment* subsystem, to obtain a 2.5D map used for planning the path of the robot to a goal point. The latter is selected by the *Control Layer*.

Commands, processed sensor data, and trigger signals requesting behavior switches, are retrieved from the rest of the *ACE* subsystems, such as the *Vision*, the *Human-Robot Interaction* (*HRI*) or the *Traversability Assessment* subsystem. The *Control Layer* is responsible for selecting, controlling and monitoring the current navigation behavior according to these inputs and the perceived world state. This way, a global path is generated which is send to the *Execution Layer*. There the global waypoints are transformed to appropriate motor commands. An obstacle avoidance module and the platform control ensure the safe motion of the robot.

B.3. Processing Layer

In order to navigate safely to a certain goal, the *ACE* robot must be capable to localize itself, generate a representation of the environment and to find a drivable path through it. This section describes the approaches used for Simultaneous Localization and Mapping (SLAM), grid fusion to obtain a 2.5D map and path planning.

B.3.1. Simultaneous Localization and Mapping

The problem of SLAM has been studied extensively over the last years. Within the *ACE* project a grid-based approach has been chosen that makes use of particle filters. Particle filters allow the approximation of arbitrary probability distributions, making them more robust to unpredicted events such as small collisions which often occur in challenging environments and cannot be modeled. Furthermore, grid-based SLAM does not rely on predefined feature extractors, which are dependent on the assumption that the environment

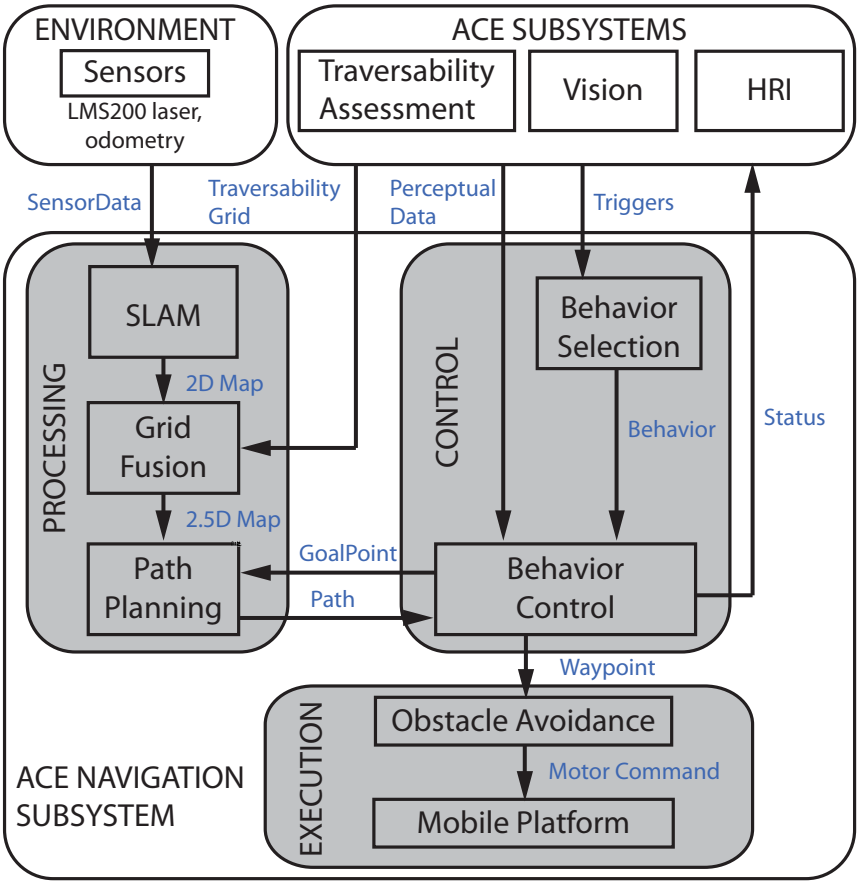


Fig. B.2.: The software architecture of the navigation subsystem of *ACE*.

exhibits a known structure. In cluttered outdoor environments the grid-based approach provides a more robust and accurate mapping.

The idea of Rao-Blackwellization is to evaluate some of the filtering equations analytically and some others by Monte Carlo sampling. This leads to estimators with smaller variance than those obtained by pure Monte Carlo sampling. In the context of SLAM the posterior distribution $P(\mathcal{X}_t, m^b|\mathcal{O}_t, \mathcal{U}_t)$ needs to be estimated. Specifically, the map m^b and the trajectory $\mathcal{X}_t$ of the robot need to be calculated based on the observations $\mathcal{O}_t$ and the odometry measurements $\mathcal{U}_t$, which are obtained by the robot and its sensors. The Rao-Blackwellization technique allows the factorization of the posterior $P(\mathcal{X}_t, m^b|\mathcal{O}_t, \mathcal{U}_t) = P(\mathcal{X}_t|\mathcal{O}_t, \mathcal{U}_t)P(m^b|\mathcal{X}_t, \mathcal{O}_t)$.

The posterior distribution $P(\mathcal{X}_t|\mathcal{O}_t, \mathcal{U}_t)$ can be estimated by sampling, where each sampled particle represents a potential trajectory. This is the localization step. Next, the posterior $P(m^b|\mathcal{X}_t, \mathcal{O}_t)$ over the map can be computed analytically, as described in [77], since the history of poses $\mathcal{X}_t$ is known.

An algorithm similar to [40] is used to estimate the SLAM posterior. Due to space limitations only the main differences are highlighted. Each particle i is weighted according to the recursive formula

$$w_t^i = \frac{P(o_t|m_{t-1}^i, x_t^i)P(x_t^i|x_{t-1}^i, u_{t-1})}{P_0(\mathcal{X}_t^i|\mathcal{X}_{t-1}^i, \mathcal{O}_t, \mathcal{U}_{t-1})} w_{t-1}^i. \tag{B.1}$$

The term $P(x_t^i|x_{t-1}^i, u_{t-1})$ is an odometry-based motion model. The motion of the robot in the interval $(t-1, t]$ is approximated by a rotation Δ_{rot1}, a translation Δ_{trans} and a second rotation Δ_{rot2}. All turns and translations are corrupted by noise. In the specific implementation presented here, noise is assumed zero mean normally distributed.

The likelihood of an observation given a map and a position estimate is denoted as $P(o_t|m_{t-1}^i, x_t^i)$. It can be evaluated for each particle by using the particle map constructed so far as well as the map correlation. More specifically, a local map $m_{local}^i(x_t^i, o_t)$ is created for each particle i and its correlation with the most recent particle map m_{t-1}^i,

$$corr = \frac{\sum_{x,y}(m_{x,y}^i - \bar{m}^i) \cdot (m_{x,y,local}^i - \bar{m}^i)}{\sqrt{\sum_{x,y}(m_{x,y}^i - \bar{m}^i)^2 \sum_{x,y}(m_{x,y,local}^i - \bar{m}^i)^2}} \tag{B.2}$$

is evaluated. The term $\bar{m}^i$ symbolizes the average map value in the overlap between the two maps. Finally, the observation likelihood is considered proportional to the correlation value $P(o_t|m_{t-1}^i, x_t^i) \propto corr$.

An important issue for the performance and the effectiveness of the algorithm is the choice of the proposal distribution $P_0(\mathcal{X}_t^i|\mathcal{X}_{t-1}^i, \mathcal{O}_t, \mathcal{U}_{t-1})$, from which particles are drawn from. In order to acquire it, new odometry measurements are corrected based on the current laser data and the map of the particle with the highest likelihood. A scan matching technique, similar to the one described in [44], is used. The corrected odometry measurements are applied to the motion model.

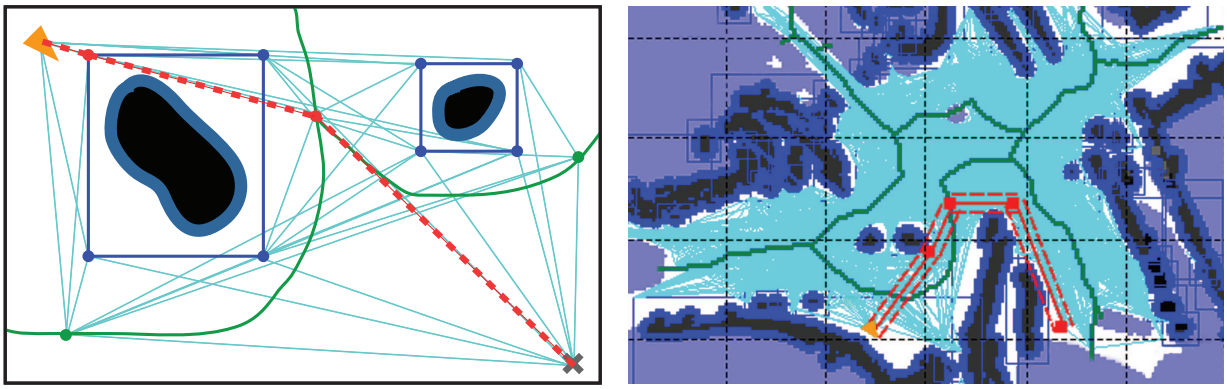
B.3.2. Grid Fusion

In order to integrate traversability information, such as detected curbs, the grid m^b from the SLAM module is fused with the grid m^n retrieved from the *Traversability Assessment* subsystem, to obtain the combined 2.5D grid m^f.

The fused grid m^f is initialized with the parameters of m^b such as width, height or resolution. The probabilities are set to

$$P^f(x,y) = \begin{cases} P_{max}^n(x,y) & \text{if } P^n(x,y) \geq max(P_{occ}, P^b(x,y)), \\ P^b(x,y) & \text{otherwise,} \end{cases} \tag{B.3}$$

where $P(x,y)$ is the occupancy probability of the cell containing the global point (x,y). For path planning this grid is thresholded by the value P_{occ}, i.e. all $P(x,y) \geq P_{occ}$ are interpreted as occupied. To consider the case when the resolution of the traversability information is finer than the resolution of the occupancy grid m_b, the probability $P_{max}^n(x,y)$ is the highest occupancy estimate from all cells in m^n which overlap with the respective cell in m^b. Equation (B.3) ensures that the traversability information is only fused, if it indicates that the respective cell is classified as occupied and the respective probability is higher than the estimate of the SLAM module. The resulting 2.5D grid m^f is send to the *Path Planning* module. This way the non-traversable regions not detected from the horizontally mounted laser range finder are also taken into account during path planning.



Schematic illustration

Path planned by the *ACE* robot

Fig. B.3.: The path planning approach: Shown are obstacles (black), which are blown up to C-Space (blue), the respective bounding boxes (blue rectangles) and the Voronoi graph (green) with corresponding nodes (dots), the visibility graph (solid cyan lines) and the resulting A* path (dashed line) from the robot (triangle) to the goal point (cross).

B.3.3. Path Planning

The *Path Planning* module generates safe paths to a goal point chosen by the *Behavior Control* module. To reach the next goal point, the robot has to navigate outdoors in unknown, dynamic, and unstructured environment. Therefore a hybrid visibility graph is generated on which A*-search is performed.

First, all frontiers and obstacles within the occupancy grid are determined. Frontiers are defined as connected unexplored cells having each at least one free neighboring cell, while obstacles are defined as connected occupied cells. For the creation of the visibility graph a dual approach is used, which is illustrated in Fig. B.3. A C-Space transformation is performed on the global map which is created incrementally by the SLAM module. The C-Space (*configuration space*) is commonly used for motion planning and refers to the set of possible positions the robot can reach. Bounding boxes are derived which enclose each obstacle in the map with a certain offset, shown as blue rectangles in Fig. B.3. All of the box vertices that lie in free space are inserted as nodes into the visibility graph. The robot position and the goal point are also inserted in the graph. If only these nodes were used to compute a path, this would bypass the obstacles as close as possible. Such a path is advantageous in open places, where enough space is available and the most efficient path is desired. However, in situations where the robot has to go along a narrow sidewalk for example, a maximum clearance path is preferable. Therefore the above is combined with an extension of the Voronoi method.

Voronoi graphs belong to the class of distance transformation algorithms and have been used in the planning literature for defining collision free paths in bounded environments, such as corridors or streets. They have also been used in order to discretize the continuous environment into a finite set of places for topological map-building. Recently, an extension has been proposed [6], which effectively computes a connected graph in incrementally updated occupancy grids. This method yields paths aligned with obstacles in the known

space and ending at the frontiers of unknown space. The set of points of a Generalized Voronoi Graph is extended by the points that are closer than a distance threshold from any obstacle. The resulting graph which is shown in Fig. B.3 with green, is then pruned in order to eliminate spurious junctions and branches.

The presented path planning approach makes use of the alignment attribute of the resulting Voronoi graph as follows. Junctions, red dot in the middle of Fig. B.3, of the generated Voronoi graph are detected and inserted into the visibility graph. Nodes between junctions are then sampled on the Voronoi graph according to a predefined distance and also added to the visibility graph.

Finally, edges are inserted between all nodes of the visibility graph that have direct visibility and are assigned edge costs proportional to their Euclidean distances. By applying A*, a heuristic based best-first search is performed on the extended visibility graph, which results in a fast determination of the shortest distance paths.

The presented *Processing Layer* creates a representation of the environment and searches for a drivable path in it. The required control input is coming from the *Control Layer* which is described in the next section.

B.4. Control Layer

When navigating through urban environments, a robot is faced with different kinds of situations that require different kinds of navigation behaviors. It must be able to explore the environment, follow a sidewalk, safely cross a street, approach a person or follow a certain direction. As can be seen in Fig. B.2 the *Control* is split into the *Behavior Selection* and the *Behavior Control* modules, which are described later in this section. Next the behaviors of the *ACE* robot are introduced.

B.4.1. Robot Behavior Description

The behaviors available to the robot are (1) *Explore*, (2) *FollowPerson*, (3) *GoInDirection*, (4) *ReachGoal* and (5) *Idle*. In the rest of this section each of them is going to be presented.

Explore: The ability to explore its environment in order to find people to interact with and increase its map knowledge, is fundamental for the robot. Optimization is performed to choose its next goal, in order to achieve a tradeoff between maximizing its information gain and minimizing traveling costs. Given an occupancy grid map, frontier regions between known and unknown areas are identified, as described in [127]. The cells of the grid m that belong to a frontier region $\mathfrak{f}$, are denoted by $m_{\mathfrak{f}}$. The expected information gain $I(m_{\mathfrak{f}}, x_t)$ acquired by reaching a frontier region from the current robot position x_t, is calculated as in [112]. The traveling costs associated with reaching a frontier region, $c(m_{\mathfrak{f}}, x_t)$, are proportional to the path length to it. In order to achieve the aforementioned tradeoff, the autonomous explorer chooses its next goal, on the frontier region that maximizes the following objective function

$$m_{\mathfrak{f}}^{*} = \arg\max_{m_{\mathfrak{f}}} \{ I(m_{\mathfrak{f}}, x_t) - \gamma_{\mathfrak{f}}\, c(m_{\mathfrak{f}}, x_t) \}. \tag{B.4}$$

The parameter $\gamma_{\mathfrak{f}}$ is used to define how much the path cost should influence the exploration process.

FollowPerson: The robot must be able to follow a person in order to cross a street safely, or in case a human wants to lead the robot to an interesting location. The vision system of the robot is used to track the person [78]. The tracker runs at 10 Hz and its outputs are send to the *Behavior Control*. There they are filtered in order to get rid of ambiguous measurements and a goal w_{gp} for the robot to reach is generated at 1 Hz.

GoInDirection: If the robot needs information, it stops and looks for passers-by with its active camera head. On detection, a human is asked to touch the screen and point in the direction of its designated goal location [78]. The coherence of the information received during the interactions is checked within the *HRI* subsystem of the *ACE* robot [130]. The *Behavior Control* module receives the verified direction and sets the necessary constraints to the *Path Planning* module, so that the robot travels on the sidewalk toward the specified direction until a crossroad is detected by the vision system. Safe navigation on the sidewalk is achieved by incorporating the curbs into the map as described in Sec. B.3.2. Whenever a crossroad is detected the robot stops and asks for help [130].

ReachGoal: In this behavior the robot tries to reach a given goal position w_{gp}. A minimum distance path is searched. If no direct path to w_{gp} can be found, a goal-directed exploration takes place, by choosing the frontier closest to w_{gp}:

$$m_{\mathfrak{f}}^{*} = \arg\min_{m_{\mathfrak{f}}} \{c(m_{\mathfrak{f}}, w_{gp})\}. \tag{B.5}$$

To avoid fluctuation of the robot motion, the *Behavior Control* module constrains the variance of the path as described in the next section.

Idle: In this behavior the robot stops immediately and no motion commands are send to the *Mobile Platform*. It is chosen either in emergency situations or when interaction with humans is taking place. It is also the default behavior in case the previous behavior has been terminated and no new trigger requests are available.

The correct selection and control of these behaviors is conducted by the homonymous modules which are presented in the rest of this section. By coordinating the described behaviors appropriately, they enable the robot to fulfill its overall objective even in unstructured environments.

B.4.2. Behavior Selection

The *Behavior Selection* module is responsible for choosing the appropriate navigation behavior depending on the current situation. Several *ACE* subsystems c, such as the Human-Robot-Interaction (*HRI*), the vision system or the internal navigation system modules, react on environmental events and emit trigger signals $\text{trigger}(c, b)$ to request a desired

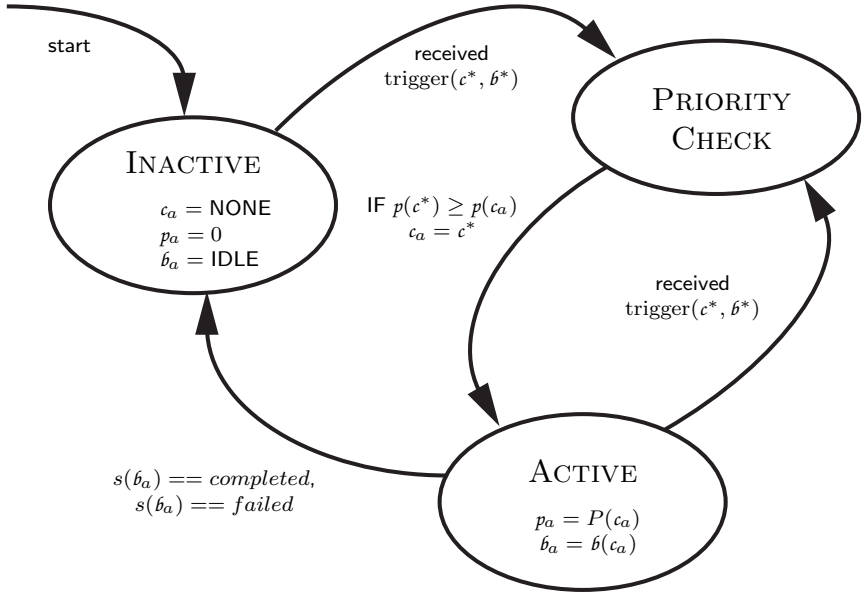


Fig. B.4.: Finite State Machine of the *Behavior Selection* module.

navigation behavior b. Such events are e.g. information input from humans, the detection of an intersection or the completion of the previous behavior.

These requests are handled by the *Behavior Selection* through the Finite State Machine (FSM) illustrated in Fig. B.4. The FSM is composed of three control states INACTIVE, ACTIVE and PRIORITYCHECK. A priority $p(c)$ is assigned to each subsystem c, which can emit command signals, according to its relevance for safety and goal completion. In the presented implementation the priorities were assigned empirically. At the beginning, the *state* of the *Control Layer* is set to INACTIVE and the active behavior b_a = Idle. Whenever a new trigger(c^*, b^*) is received, the *state* switches to PRIORITYCHECK, which is performed between c^* and the active command module c_a. If $p(c^*) \geq p(c_a)$, c^* becomes c_a and the new behavior $b_a = b^*$ is forwarded to the *Behavior Control* where it is executed. The *Behavior Selection* remains in ACTIVE state until either a new trigger signal is received, or the *Behavior Control* reports a change of the behavior state $s(b_a)$ to *completed* or *failed*.

B.4.3. Behavior Control

The *Behavior Control* module is responsible for the proper execution, monitoring and failure handling of the active behavior b_a. The pseudo-code shown in Algorithm B.1 highlights the control procedure. As long as the *state* of the *Control Layer* does not become INACTIVE, the *Behavior Control* remains in an endless loop (ll.1-7). During each cycle it first requests b_a from the *Behavior Selection* and any additionally required data directly from external modules. Depending on b_a, this can be the robot pose x_t, a goal direction φ_{gp} or a specific goal point w_{gp}.

In the next step, the behavior state $s(b_a)$ and any constraints need to be updated as shown in Fig. B.5. In case the *failedtrials* (i.e. the number of consecutive prior runs where b_a has not been successfully executed) exceeds the upper bound MAXTRIALS, the claimed *goalaccuracy* is decreased. This enables the *ACE* robot to react in situations where a goal

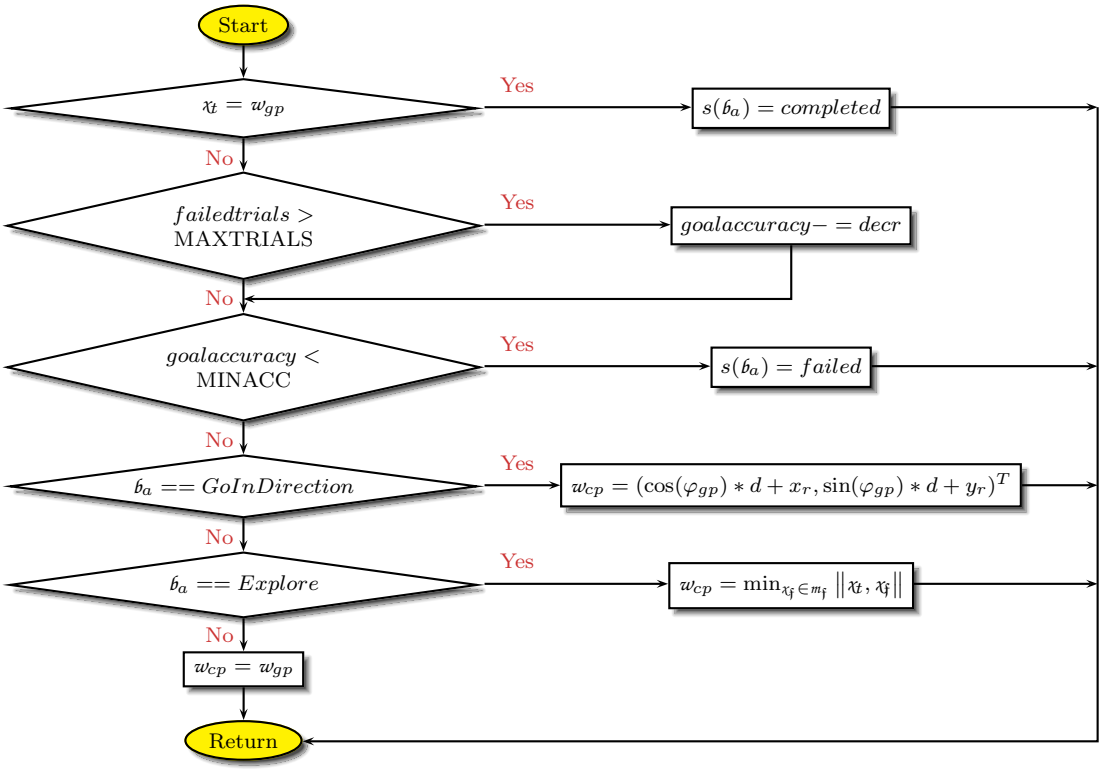


Fig. B.5.: Flow chart of the `checkConstraints()` procedure.

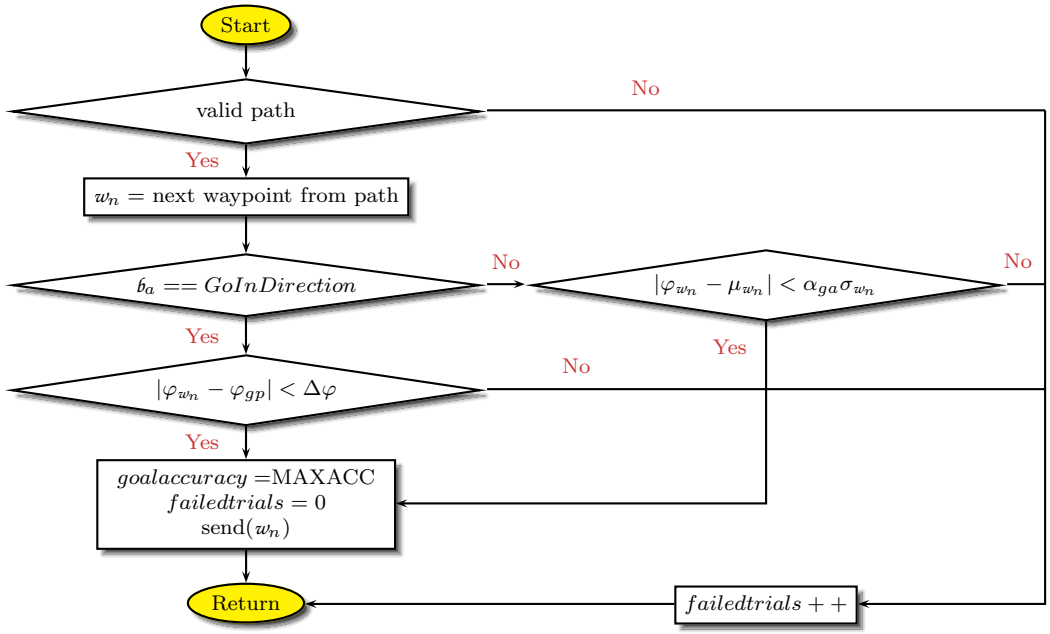


Fig. B.6.: Flow chart of the `checkConsistency()` procedure.

Algorithm B.1: The *Behavior Control* of the *ACE* robot.

```
BehaviorControl():

1 while state ≠ INACTIVE do
2 |   behavior b_a = requestBehavior();
3 |   ⟨robot pose x_t, goal direction φ_gp, goal point w_gp⟩ = updateInfo();
4 |   w_cp = checkConstraints(b_a, x_t, φ_gp, w_gp);                    // see
  |   Fig. refalg:endconstraints
5 |   w_n =computePath(w_cp) ;                          // Path Planning module
6 |   checkConsistency(b_a, w_n, φ_gp);              // see Fig. refalg:endconsis
7 end
```

point lies within non-traversable space, e.g. stairs, or the goaldirection is blocked, e.g. by a car. Decreasing the *goalaccuracy* leads to a less strict consistency check (described below) and allows short-term deviations from the goal. In situations where *ACE* encounters larger barriers, e.g. when the received w_{gp} lies within a building or φ_{gp} points into a wall, the *goalaccuracy* would permanently drop leading to a growing deviation from the actual goal. Therefore, whenever the *goalaccuracy* drops below a threshold MINACC, the *Behavior Control* will report $s(b_a) = failed$, enabling the *Behavior Selection* to stop the current behavior and the HRI module e.g. to ask for new directions.

In case the `checkConstraints()` is passed, the next checkpoint w_{cp} for the planning module has to be derived. In the *GoInDirection* behavior, w_{cp} is set into the direction φ_{gp} at a distance d from the robot. In the *Explore* behavior, w_{cp} is set to the closest point from the frontier f derived by (B.4). For the remaining behaviors w_{cp} is set to w_{gp}. Then w_{cp} is send to the path planning module described in Sec. B.3.3, which returns a path in form of a list of waypoints. An exception is made in the *Follow* behavior, where a direct line of sight to w_{cp} is assumed, since it corresponds to the position of the person tracked by the vision system. In that case the planning module is set inactive and the waypoint w_n is directly set to w_{cp}.

To verify that the found path satisfies the demanded *goalaccuracy*, the next waypoint w_n is applied to the `checkConsistency()` shown in Fig. B.6. In the *GoInDirection* behavior, i.e. that the deviation of the direction of the waypoint φ_{w_n} from φ_{gp} must be within a bound $\Delta\varphi$. For the other behaviors the deviation of φ_{w_n} from the mean μ_{w_n} of the n prior w_n must be smaller than the corresponding standard deviation σ_{w_n} weighted with α_{ga}. The parameters n, α_{ga} and $\Delta\varphi$ are set with respect to the current *goalaccuracy*. The parameters and thresholds presented in this section have been derived experimentally. In case the consistency check is passed successfully, w_n is forwarded to the *Execution Layer*.

B.5. Execution Layer

The *Execution Layer* obtains global waypoints from the *Behavior Control* module which has been described in the previous section. These are given as input to the *Obstacle Avoidance* module, which generates motor commands for the mobile platform. This module takes into account dynamic obstacles in the vicinity of the robot and ensures safe local

navigation. A method similar to [94] is used to generate smooth and safe robot trajectories.

Scenes and corresponding output of different behaviors from the conducted field experiment with the *ACE* robot are shown in the next section.

B.6. Experimental Results

The navigation subsystem of the *ACE* robot has been tested in several indoor and outdoor scenarios, over several months. In this section results and experiences from the major field experiment are presented.

The objective of the experiment was for the *ACE* robot to reach Marienplatz, the central square of Munich, starting from the Technical University of Munich. This is a distance of approximately 1.5 km in the most active part of downtown Munich. The robot did not have prior map knowledge or GPS and relied only on interactions with passers-by to get directions and on its on-board sensors, described in Sec. B.2, in order to navigate safely. The experiment was conducted successfully on 31 August 2008. The robot interacted with 38 persons before reaching its final goal after approximately 5 hours. The large number of interactions explains also the relatively long completion time. Many people were curious and wanted to try out the robot, therefore frequently stopping it in order to interact with it. In some not so populated parts of the route, the robot was able to safely navigate on the sidewalk in a given direction, for up to 250 m before arriving at a crossroad.

For illustrative purposes, an aerial photo of the area where the robot navigated can be found in Fig. B.7. The approximate trajectory of the robot is indicated by a yellow line. During the experiment, a large occupancy grid map was build online by the *SLAM* module. The algorithm, which was described in Sec. B.3.1, ran at 2 Hz, used 200 particles and retrieved data from the LMS200 laser range finder. Parts of the acquired map are illustrated in Fig. B.7(a)-(b). Both occupancy grids have been overlapped with satellite images taken from Google Earth to illustrate the accuracy of the maps, which have a resolution of 15 cm. It can be seen that the described algorithm delivers accurate metric maps from intersection to intersection and across larger areas.

Fig.B.8 shows three exemplary scenes encountered during the experiment (left side) and the corresponding outputs of the *Control Layer* and *Processing Layer* (right side). Scene (a) shows a situation in *Follow* behavior, where the robot is guided by the vision system. In scene (b), the robot is located in a narrow sidewalk passage, where it retrieves a direction from a passer-by and plans a drivable path by making use of the dual planning approach. Scene (c) is in a highly populated street, where the straight direction of travel is blocked by people. *ACE* circumvented the people by dynamically adjusting the behavior constraints.

The right side of Fig. B.8 shows the part of the occupancy grid, which is transformed to C-Space and used for path planning. The transformation is indicated by the dark blue regions around the obstacles (black). The black coordinate raster has a cell size of 5×5 m and is only used for illustration purposes. Replanning was performed at 2 Hz. In scene (a), the *ACE* robot (green triangle) crossed a street in *Follow* behavior. It can be observed from the corresponding output (a2), that the planning module is inactive. The robot is directly send to the position of the person (red dot), which is received from the vision system [78].

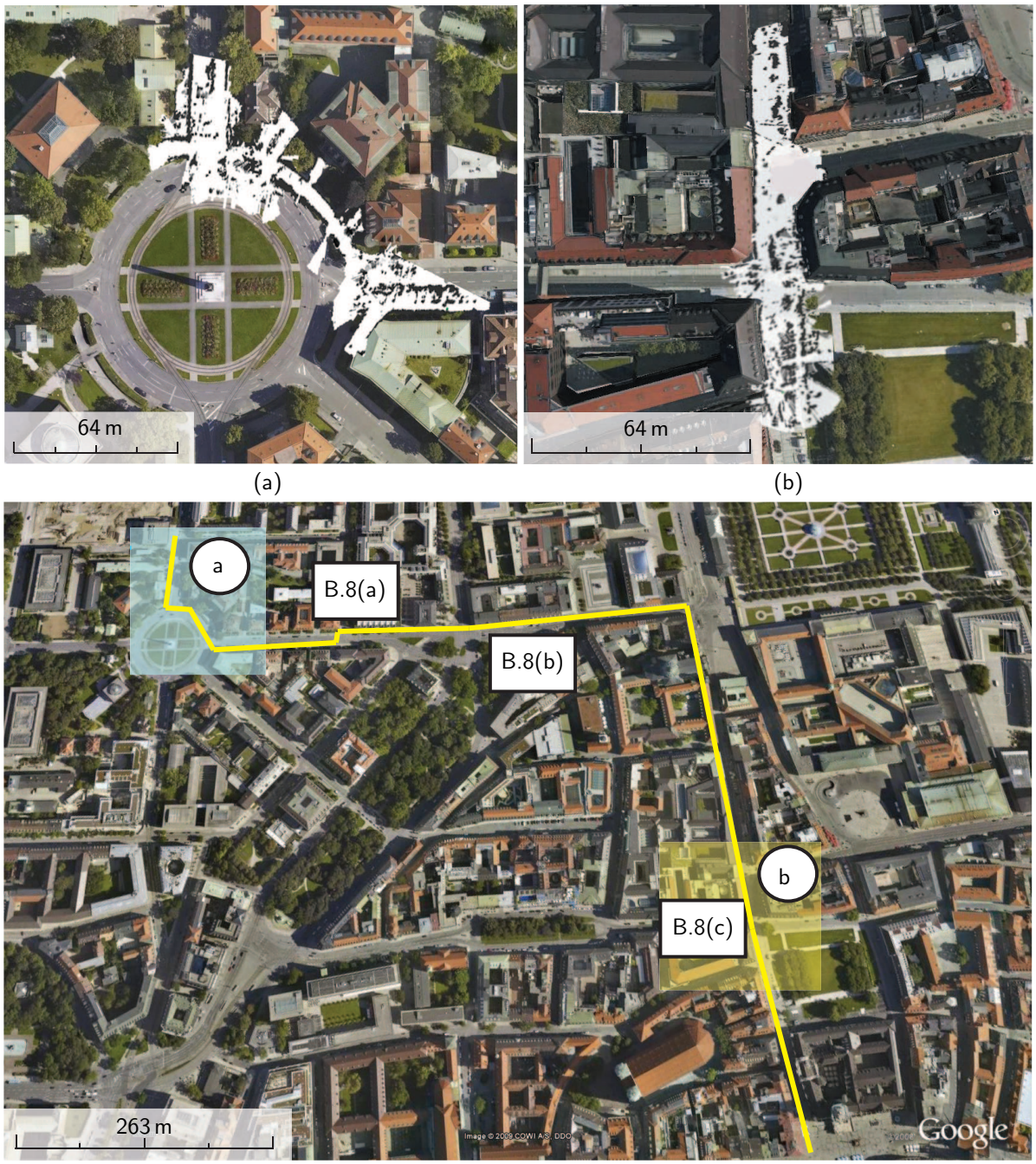


Fig. B.7.: Downtown area of Munich. The route ($\sim 1.5\,\mathrm{km}$) of the robot from the Technical University of Munich to Marienplatz is indicated by the yellow line. (a)-(b): Parts of the map generated by the *SLAM* module during navigation that correspond to the two representative situations which were chosen for the interdependence analysis in Section 4.4.3. The occupancy grids have been overlapped with satellite images taken from Google Earth to illustrate the accuracy of the maps. The rectangles mark the positions of the scenes described in Fig. B.8.

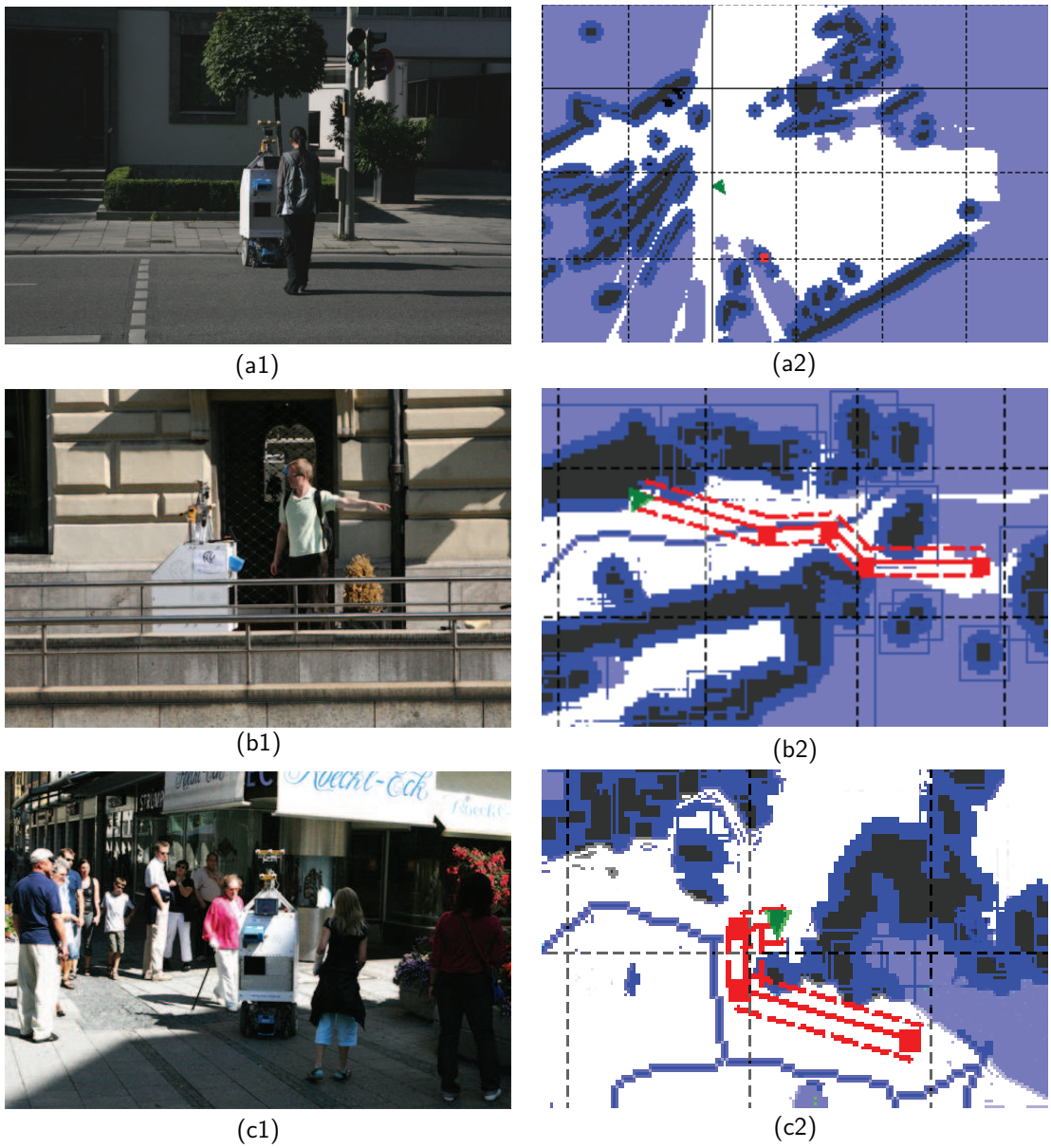


Fig. B.8.: Three different scenes encountered during the experiment (left side) and the corresponding outputs of the *Control* and *Processing Layer* (right side). (a) shows the *ACE* robot in *Follow* behavior, (b) shows the *GoInDirection* behavior on a narrow sidewalk and (c) shows the *GoInDirection* behavior in a crowded place. The coordinate raster (black lines) has a cell size of 5 m×5 m and is only used for illustrative purposes.

(a) (b)

Fig. B.9.: Two scenes from the field experiment in the *Deutsches Museum*.

In scene (b), a person was pointing into the direction the *ACE* robot should follow in order to reach Marienplatz. The direction is perceived by the vision system with a detection rate of 80.2% and an accuracy of 6.8°, and send to the *Behavior Control* module. There it is transformed and forwarded to the *Path Planning* module, which is active in this situation. In Fig. B.8(b2) the Voronoi Graph (blue lines), which traverses the free space (white), and resulting path (three parallel red lines) can be seen. The shown path illustrates the advantage of the dual path planning approach. The first two waypoints (red dots) correspond to nodes retrieved from the Voronoi Graph. While the path would proceed straight to the corner of the obstacle in the bottom (below the second waypoint) if only the bounding boxes were used, the extension with the Voronoi Graph keeps the robot in the center of the narrow sidewalk leading to a maximal clearance from obstacles. However, the passage in the area of the third and fourth waypoint is closer than the preferred obstacle distance used for the Voronoi Graph. To drive through this area, the *ACE* robot needs to get as close to the obstacles as possible. This is enabled by using the nodes from the bounding boxes. The solid red center line indicates the actual path through free C-Space. The dashed red lines on the left and right indicate the path through free space (i.e. without C-Space transformation) broadened by the robot width.

Fig. B.8(c) shows the *GoInDirection* behavior in a crowded street where the direct way is blocked. To proceed into the goal direction φ_{gp}, indicated by the line between the 2nd and 3rd waypoint, the *ACE* robot is forced to bypass the persons by driving almost into the opposite direction (1st waypoint). This leads to an inconsistency with the desired goal accuracy, what is mastered by temporarily relaxing the goal accuracy and consequently allowing the *ACE* robot to temporarily deviate from φ_{gp}. The maximum allowed deviation has been experimentally set to $\Delta\varphi = 90°$.

In 2008 also a second field experiment was conducted where the robot was part of a museum exhibition from which scenes are shown in Fig. B.9. The demonstration was successfully performed without any required algorithmic modifications to the system, what indicates the suitability of the navigation subsystem for both indoor and outdoor settings.

B.7. Conclusion

This chapter described the navigation subsystem of the *ACE* robot, which has been designed to navigate autonomously in highly populated urban environments, based only on information extracted by interactions with humans and its on-board sensors. *ACE* integrates innovative algorithms for dealing with the problems of SLAM, path planning in unstructured environments and the selection, monitoring and control of different navigation behaviors. Results from an extensive field experiment, conducted in the downtown city area of Munich (`www.ace-robot.de`), show that the robot is capable of navigating safely in highly populated environments over long time periods, while dealing with unexpected situations which occur frequently in real-world settings.

C. Map Overview

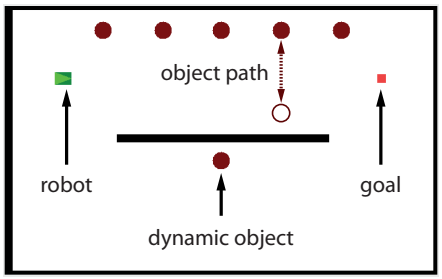
During the experimental evaluation described in Section 4.5.3 the maps described in the following were used. The maps differ with respect to the four parameters shown in Table C.1. The $ratio = x/y$ specifies that x dynamic objects are located on the short and y dynamic objects on the long way between robot and goal. *Swapped* indicates whether the start and goal positions are mirrored, e.g. map 13 has the mirrored positions of map 1. *Shifted* relates to a clockwise shift of start and goal position by the respective percentage of the short path length. The latter always remains the same. The parameter *Rotation* indicates a clockwise rotation of the entire map by the specified value.

The maps 1 to 20 were obtained by generating all possible combinations of *Swapped*, *Shifted* and *Rotation* and then randomizing those sequence. Next each map was assigned a specific *ratio*, such that each *ratio* occurs exactly four times.

Parameter	Range of values
Ratio	1/1, 1/3, 3/1, 1/5, 5/1
Swapped	yes, no
Shifted	0%, 20%, 40%
Rotation	0°, 90°, 180°, 270°

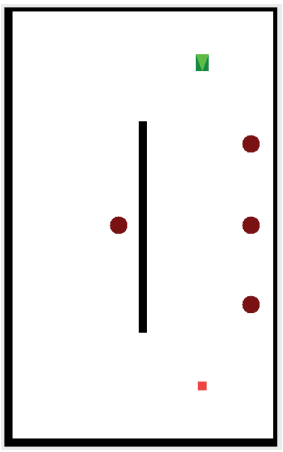
Tab. C.1.: Map parameters and respective ranges of values.

Map 1

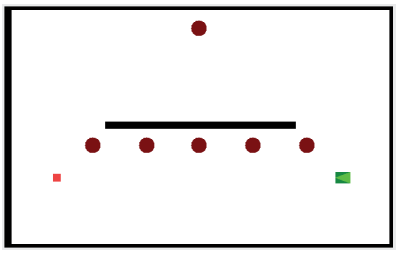


	Ratio	Swapped	Shifted	Rotation
Map 1	5/1	No	0%	0°
Map 2	3/1	No	0%	90°

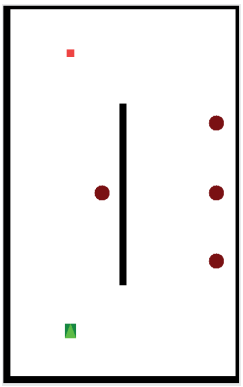
Map 2



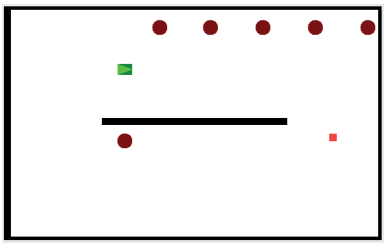
Map 3



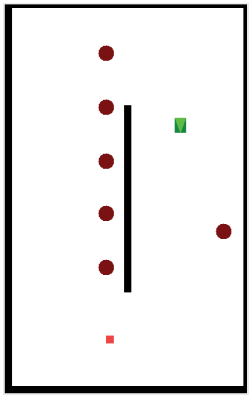
Map 4



Map 5

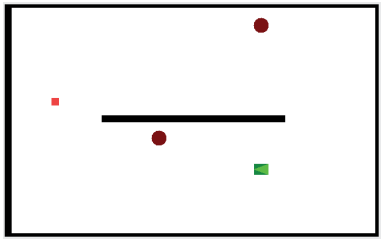


Map 6

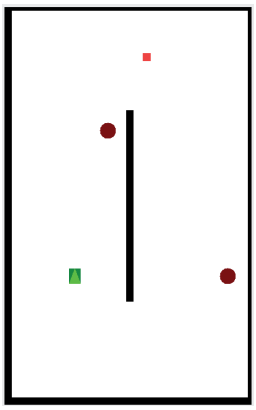


	Ratio	Swapped	Shifted	Rotation
Map 3	5/1	No	0%	180°
Map 4	1/3	No	0%	270°
Map 5	5/1	No	20%	0°
Map 6	1/5	No	20%	90°

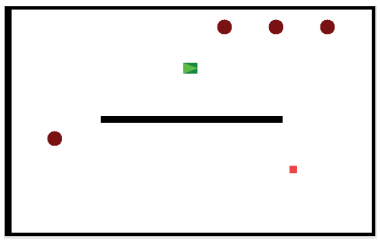
Map 7



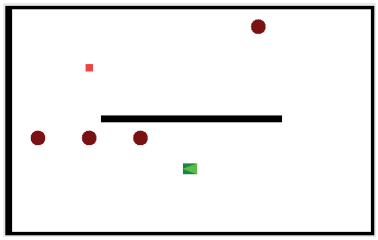
Map 8



Map 9

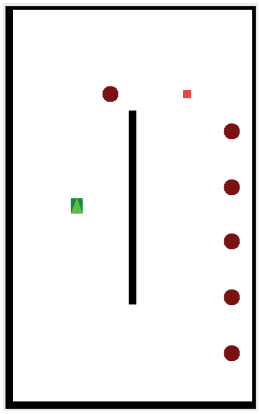


Map 10

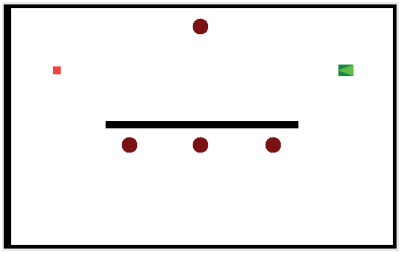


	Ratio	Swapped	Shifted	Rotation
Map 7	1/1	No	20%	180°
Map 8	1/1	No	20%	270°
Map 9	3/1	No	40%	0°
Map 10	3/1	No	40%	180°

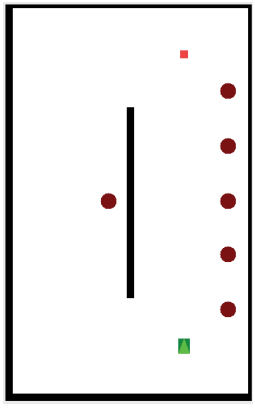
Map 11



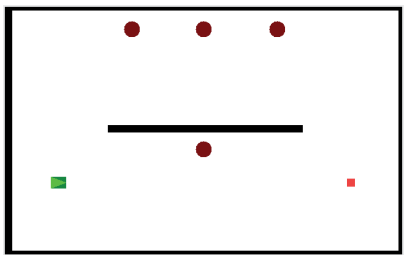
Map 12



Map 13

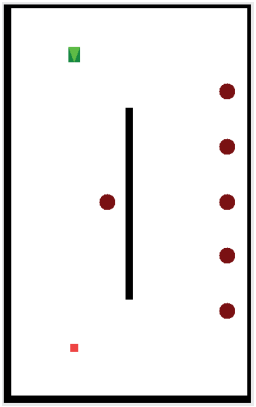


Map 14

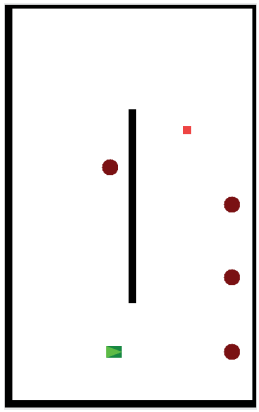


	Ratio	Swapped	Shifted	Rotation
Map 11	1/5	No	40%	270°
Map 12	1/3	Yes	0%	0°
Map 13	5/1	Yes	0%	90°
Map 14	1/3	Yes	0%	180°

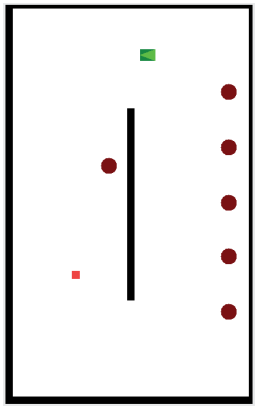
Map 15



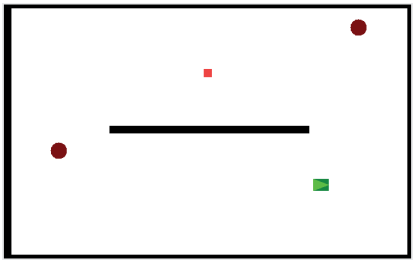
Map 16



Map 17

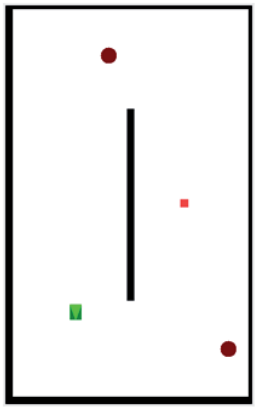


Map 18

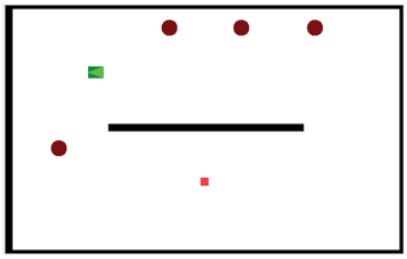


	Ratio	Swapped	Shifted	Rotation
Map 15	1/5	Yes	0%	270°
Map 16	3/1	Yes	20%	90°
Map 17	1/5	Yes	20%	270°
Map 18	1/1	Yes	40%	0°

Map 19



Map 20



	Ratio	Swapped	Shifted	Rotation
Map 19	1/1	Yes	40%	90°
Map 20	1/3	Yes	40%	180°

Bibliography

[1] G.T. Anderson and Y. Gang. A proposed measure of environmental complexity for robotic applications. In *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, 2007.

[2] T. Arai, E. Pagello, and L.E. Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18:655–661, 2002.

[3] K. J. Arrow. *Essays in the theory of risk-bearing.* Markham Publishing Company, 1971.

[4] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An integrated humanoid platform for sensory-motor control. In *Proc. of the Int. Conf. on Humanoid Robots (Humanoids)*, 2006.

[5] A. Avizienis, J.C. Laprie, and B. Randell. Fundamental concepts of computer system dependability. In *Proc. of the Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments*, 2001.

[6] P. Beeson, N. K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[7] D.P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20:133–149, 1990.

[8] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:237–256, 1995.

[9] S. Bothelho and R. Alami. Robots that cooperatively enhance their plans. In *Proc. of the Int. Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2000.

[10] J. Bredin, R. T. Maheswaran, C. Imer, T. Basar, D. Kotz, and D. Rus. A game-theoretic formulation of multi-agent resource allocation. In *Proc. of the Int. Conf. on Autonomous Agents*, 2000.

[11] P. Brucker. *Scheduling Algorithms.* Springer Verlag, 2007.

[12] A. Campbell, A. S. Wu, and R. Shumaker. Multi-agent task allocation: Learning when to say no. In *Proc. of the Conf. on Genetic and Evolutionary Computation*, 2008.

[13] H. Chan. *Sensitivity Analysis of Probabilistic Graphical Models*. PhD thesis, University of California, Los Angeles, 2005.

[14] J. Chen, X. Yan, H. Chen, and D. Sun. Resource constrained multirobot task allocation with a leader-follower coalition method. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[15] F. Chenavier and J.L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1992.

[16] H.I. Christensen and W. Förstner. Performance characteristics of vision algorithms. *Machine Vision and Applications*, 9:215–218, 1997.

[17] E.J. Clarke and B.A. Barton. Entropy and MDL discretization of continuous variables for Bayesian belief networks. *Int. Journal of Intelligent Systems*, 15:61–92, 2000.

[18] T.H.J. Collett, B.A. MacDonald, and B.P. Gerkey. Player 2.0: Toward a practical robot programming framework. In *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, 2005.

[19] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11:138–148, 1999.

[20] G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[21] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[22] J.W. Crandall and M.A. Goodrich. Measuring the intelligence of a robot and its interface. In *Proc. of the Workshop on Performance Metrics for Intelligence Systems (PerMIS)*, 2003.

[23] O. Cuisenaire. *Distance Transformations: Fast Algorithms and Applications to Medical Image Processing*. PhD thesis, Universite Catholique de Louvain, 1999.

[24] DARPA. Urban challenge - rules, 2007.

[25] M.B. Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.

[26] S. Dubowsky, F. Genot, S. Godding, H. Kozono, A. Skwersky, H. Yu, and L.S. Yu. PAMM - a robotic aid to the elderly for mobility assistance and monitoring: A "helping-hand" for the elderly. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[27] J.M. Evans. HelpMate: An autonomous mobile robot courier for hospitals. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 1994.

[28] A. Farinelli, L. Iocchi, and D. Nardi. Multi robot systems: A classification focused on coordination. *IEEE Transactions on System Man and Cybernetics*, part B:2015–2028, 2004.

[29] A. Farinelli, L. Iocchi, D. Nardi, and V.A. Ziparo. Assignment of dynamically perceived tasks by token passing in multi-robot systems. In *Proc. of the IEEE Special issue on Multi-Robot Systems*, 2006.

[30] R.B. Fierro, A.K. Das, J.R. Spletzer, J.M. Esposito, V. Kumar, J.P. Ostrowski, G.J. Pappas, C.J. Taylor, Y. Hur, R. Alur, I. Lee, G.Z. Grudic, and B. Southall. A framework and architecture for multi-robot coordination. *Int. Journal of Robotics Research*, 21:977–998, 2002.

[31] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2:189–208, 1971.

[32] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34:596–615, 1987.

[33] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.

[34] G. Garibott, S. Masciangelo, M. Ilic, and P. Bassino. Robolift: A vision guided autonomous fork-lift for pallet handling. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 1996.

[35] E. Gat. *Artificial intelligence and mobile robots*, chapter Three-layer architectures, pages 195–210. MIT Press, Cambridge, MA, USA, 1998.

[36] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multirobot coordination. *IEEE Transactions On Robotics And Automation*, 18:758–768, 2002.

[37] B. P. Gerkey and M. J. Matarić. A formal analysis and taxomony of task allocation in multi-robot systems. *Int. Journal of Robotics Research (IJRR)*, 23:939–954, 2004.

[38] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: The planning domain definition language, version 1.2. Technical report, Yale Center for Computational Vision and Control, 1998.

[39] P. J. Gmytrasiewicz and E. H. Durfee. Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems*, 4:233–272, 2001.

[40] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[41] H.-M. Gross, H.-J. Böhme, Ch. Schröter, S. Müller, A. König, C. Martin, M. Merten, and A. Bley. ShopBot: Progress in developing an interactive mobile shopping assistant for everyday use. In *Proc. of the Int. Conf. on Systems, Man and Cybernetics*, 2008.

[42] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45:26–34, 2008.

[43] G.S. Guthart and J.K. Salisbury. The intuitive^{TM} telesurgery system: Overview and application. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[44] D. Haehnel, W. Burgard, D. Fox, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003.

[45] U. D. Hanebeck, N. Saldic, and G. Schmidt. A modular wheel system for mobile robot applications. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 1999.

[46] J. Held, A. Lampe, and R. Chatila. Linking mobile robot performances with the environment using system maps. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[47] J.M. Hollerbach et al. A roadmap for US robotics: From internet to robotics. Technical report, Computing Community Consortium, 2009.

[48] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19:281–316, 2005.

[49] M. Hosseini. *Statistical Models for Agroclimate Risk Analysis*. PhD thesis, University of British Columbia, 2009.

[50] H.M. Huang, E. Messina, R. Wade, R. English, B. Novak, and J. Albus. Autonomy measures for robots. In *Proc. of the Int. Mechanical Engineering Congress & Exposition*, 2004.

[51] M.N. Huhns and M.P. Singh. *Readings in Agents*. Morgan Kaufmann Publishers, 1998.

[52] T.L. Huntsberger, A. Trebi-Ollennu, H. Aghazarian, P.S. Schenker, P. Pirjanian, and H.D. Nayar. Distributed control of multi-robot systems engaged in tightly coupled tasks. *Autonomous Robots*, 17:79–92, 2004.

[53] A. Jacoff, E. Messina, and J. Evans. Performance evaluation of autonomous mobile robots. *Industrial Robot: An Int. Journal*, 29:259–267, 2002.

[54] E. Jones, B. Browning, M.B. Dias, B. Argall, M. Veloso, and M. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[55] N. Kalra, M.B. Dias, R.M. Zlot, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proc. of the IEEE*, 94:1257–1270, 2006.

[56] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[57] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi. Humanoid robot HRP-3. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[58] J. Karvanen. Estimation of quantile mixtures via L-moments and trimmed L-moments. *Computational Statistics & Data Analysis*, 51:947–959, 2006.

[59] J. Kiener and O. von Stryk. Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[60] R. Koenker. *Quantile Regression*. Cambridge Univ Press, 2005.

[61] R. Korn and E. Korn. *Option Pricing and Portfolio Optimization*, volume 31. American Mathematical Society, 2001.

[62] B.H. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag, 2008.

[63] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, pages 83–97, 1955.

[64] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *Pervasive Computing, IEEE*, 3:24–33, 2005.

[65] A. Lampe and R. Chatila. Performance measure for the evaluation of mobile robot autonomy. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[66] P. Langley, J.E. Laird, and S. Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10:141–160, 2009.

[67] R.N. Lass, E.A. Sultanik, and W.C. Regli. *Metrics for Multiagent Systems*, chapter 1, pages 1–19. Springer, 2009.

[68] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[69] D. Lecking, O. Wulf, V. Viereck, J. Tödter, and B. Wagner. The RTS-STILL robotic fork-lift. EURON Technology Transfer Award, 2005.

[70] V.J. Leon, D. Kortenkamp, and D. Schreckenghost. A planning, scheduling and control architecture for advanced life support systems. In *NASA Workshop on Planning and Scheduling for Space*, 1997.

[71] K. Lerman, C. Jones, A. Galstyan, and M.J. Matarić. Analysis of dynamic task allocation in multi-robot systems. *Int. Journal of Robotics Research*, 25:225–241, 2006.

[72] B. Liepert et al. Strategic Research Agenda for robotics in Europe. Technical report, The European Robotics Technology Platform (EUROP), 2009.

[73] H. Lim, Y. Kang, J. Lee, J. Kim, and B.J. You. Multiple humanoid cooperative control system for heterogeneous humanoid team. In *Proc. of the IEEE Int. Symposium on Robot and Human Interactive Communication*, 2008.

[74] M. Matarić, G. Sukhatme, and E. Stergaard. Multirobot task allocation in uncertain environments. *Autonomous Robots*, 14:255–263, 2003.

[75] G. A. Mills-Tettey, A. Stentz, and M. B. Dias. The dynamic Hungarian algorithm for the assignment problem with changing costs. Technical Report CMU-RI-TR-07-27, Robotics Institute, Carnegie Mellon University, 2007.

[76] G. Montemayor and J. Wen. Decentralized collaborative load transport by multiple robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[77] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *Sensor Devices and Systems for Robotics*, pages 243–276, 1989.

[78] Q. Mühlbauer, S. Sosnowski, T. Xu, T. Zhang, K. Kühnlenz, and M. Buss. Navigation through urban environments by visual perception and interaction. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[79] Armin Müller, Alexandra Kirsch, and Michael Beetz. Transformational planning for everyday activity. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2007.

[80] N. Munoz, J. Valencia, and N. Londono. Evaluation of navigation of an autonomous mobile robot. In *Proc. of the Workshop on Performance Metrics for Intelligence Systems*, 2007.

[81] R. Murai, T. Sakai, H. Uematsu, H. Nakajima, K. Mitani, and H. Kitano. Conveyance system using autonomous mobile robots. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2009.

[82] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*, 1999.

[83] K. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33:1024–1034, 2001.

[84] M. Nagarajan and G. Sosic. Game-theoretic analysis of cooperation among supply chain agents: Review and extensions. *European Journal of Operational Research*, 187:719–745, 2008.

[85] J.E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: A Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66:751–769, 2004.

[86] T. Odashima, M. Onishi, K. Tahara, K. Takagi, F. Asano, Y. Kato, and H. Nakashima. A soft human-interactive robot RI-MAN. Video Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS), 2006.

[87] RIKEN: The Institute of Physical and Ltd. Chemical Research, Tokai Rubber Industries. Realization of transfer operations by nursing-care assistant robot RIBA. Company News, 2009.

[88] H.K. Park, H.S. Hong, H.J. Kwon, and M.J. Chung. A nursing robot system for the elderly and the disabled. *Int. Journal of Human-friendly Welfare Robotic Systems (HWRS)*, 2:11–16, 2001.

[89] L.E. Parker. ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14:220–240, 1998.

[90] L.E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2:5–14, 2008.

[91] L.E. Parker and F. Tang. Building multi-robot coalitions through automated task solution synthesis. *Proc. of the IEEE*, 94:1289–1305, 2006.

[92] H.V.D. Parunak. Applications of distributed artificial intelligence in industry. *Foundations of distributed artificial intelligence*, 4:139–164, 1996.

[93] E.P.D. Pednault. ADL and the state-transition model of action. *Journal of Logic and Computation*, 4:467–512, 1994.

[94] R. Philippsen and R. Siegwart. Smooth and efficient obstacle avoidance for a tour guide robot. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003.

[95] D.V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002.

[96] R. Robinson. Counting unlabeled acyclic digraphs. *Combinatorial mathematics V*, 622:28–43, 1977.

[97] RoboCup@Home. www.ai.rug.nl/robocupathome.

[98] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[99] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation: Mobile robot navigation with uncertainty in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1999.

[100] S. Russell and P. Norvig. *Artifical Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[101] A. Saltelli, K. Chan, and E.M. Scott, editors. *Sensitivity Analysis*. John Wiley & Sons Inc, 2004.

[102] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238, 1999.

[103] S. Sariel, T. Balch, and N. Erdogan. Incremental multi-robot task selection for resource constrained and interrelated tasks. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[104] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[105] S. Seuken and S. Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17:190–250, 2008.

[106] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[107] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, 2009.

[108] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[109] RoboCup Soccer. http://www.robocup.org/robocup-soccer/.

[110] S. Sosnowski, A. Bittermann, K. Kühnlenz, and M. Buss. Design and evaluation of emotion-display EDDIE. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[111] S. Srinivasa, D. Ferguson, M.V. Weghe, R. Diankov, D. Berenson, C. Helfrich, and H. Strasdat. The robotic busboy: Steps towards developing a mobile robotic home assistant. In *Proc. of the Int. Conf. on Intelligent Autonomous Systems*, 2008.

[112] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. of Robotics Science and Systems (RSS)*, 2005.

[113] B. Stanczyk. *Developement and Control of an Anthropomorphic Telerobotic System*. PhD thesis, Technische Universität München, Institute for Automatic Control Engineering, 2006.

[114] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8:345–383, 2000.

[115] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Int. Journal of Robotics Research (IJRR)*, 19:972–999, 2000.

[116] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley - the robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23:661–692, 2006.

[117] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. In *Proc. of the Int. Multi-Robot Systems Workshop*, 2005.

[118] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.

[119] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M.N. Clark, J. Dolan, D. Duggins, C. Galatali, T. andGeyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25:425–466, 2008.

[120] J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[121] L. Vig and J. A. Adams. A framework for multi-robot coalition formation. In *Proc. of the 2nd Indian Int. Conf. on Artificial Intelligence*, 2005.

[122] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22:637–649, 2006.

[123] N. Vlassis. A concise introduction to multiagent systems and distributed AI. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 1:1–71, 2007.

[124] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT press, 2000.

[125] B.B. Werger and M. J. Matarić. Broadcast of local eligibility: Behavior-based control for strongly cooperative robot teams. In *Proc. of the Fourth Int. Conf. on Autonomous Agents*, 2000.

[126] WillowGarage. Pr2: www.willowgarage.com/pages/pr2/overview.

[127] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Int. Conf. on Autonomous Agents*, 1998.

[128] M.M. Zavlanos, L. Spesivtsev, and G.J. Pappas. A distributed auction algorithm for the assignment problem. In *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2008.

Own Publications

[129] D. Althoff, O. Kourakos, M. Lawitzky, A. Mörtl, M. Rambow, F. Rohrmüller, D. Brščić, S. Hirche, and M. Buss. An architecture for real-time control in multi-robot systems. In *3rd Int. Workshop on Human-Centered Robotic Systems*, 2009.

[130] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss. The Autonomous City Explorer: Towards natural human-robot interaction in urban environments. *Int. Journal of Social Robotics*, 1:127–140, 2009.

[131] K. Klasing, G. Lidoris, A. Bauer, Florian Rohrmüller, D. Wollherr, and M. Buss. The Autonomous City Explorer: Towards semantic navigation in urban environments. In *1st Int. Workshop on Cognition for Technical Systems*, 2008.

[132] G. Lidoris, F. Rohrmüller, D. Wollherr, and M. Buss. System interdependence analysis for autonomous robots. *Int. Journal of Robotics Research*, 30:601–614, 2011.

[133] G. Lidoris, F. Rohrmüller, D. Wollherr, and M. Buss. The Autonomous City Explorer (ACE) Project - mobile robot navigation in highly populated urban environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[134] M. Rambow, F. Rohrmüller, O. Kourakos, D. Brščić, D. Wollherr, S. Hirche, and M. Buss. A Framework for Information Distribution, Task Execution and Decision Making in Multi-Robot Systems. *IEICE TRANSACTIONS on Information and Systems*, E93-D:1352–1360, 2010.

[135] F. Rohrmüller, M. Althoff, D. Wollherr, and M. Buss. Probabilistic mapping of dynamic obstacles using Markov chains for replanning in dynamic environments. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, September 2008.

[136] F. Rohrmüller, O. Kourakos, M. Rambow, D. Brščić, D. Wollherr, S. Hirche, and M. Buss. Interconnected performance optimization in complex robotic systems. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[137] F. Rohrmüller, G. Lidoris, D. Wollherr, and M. Buss. System interdependence analysis for autonomous mobile robots. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.