İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

AN INTEGRATED PLANNING, SCHEDULING AND EXECUTION FRAMEWORK FOR MULTI-ROBOT COOPERATION AND COORDINATION

Ph.D. Thesis by

Sanem SARIEL, M.Sc.

Department : Computer Engineering

Programme: Computer Engineering

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

AN INTEGRATED PLANNING, SCHEDULING AND EXECUTION FRAMEWORK FOR MULTI-ROBOT COOPERATION AND COORDINATION

Ph.D. Thesis by

Sanem SARIEL, M.Sc.

(504022202)

Date of submission : 4 May 2007

Date of defence examination: 22 June 2007

Supervisors (Chairman):	**Prof. Dr. Nadia ERDOĞAN**
	Assoc. Prof. Dr. Tucker BALCH
	(Georgia Institute of Technology)
Members of the Examining Committee	**Prof.Dr. Emre HARMANCI**
	Prof.Dr. Levent AKIN (BÜ.)
	Prof.Dr. Coşkun SÖNMEZ (YTÜ.)
	Assoc. Prof. Dr. Sabih ATADAN
	Assist. Prof. Dr. Feza BUZLUCA

JUNE 2007

Dedicated to the memory of my father,
Pilot, Mehmet Sarıel...

PREFACE

I first came to focus on distributed intelligence and multi-entity cooperation in 2002, and I am always impressed by the way a group or community constitute a great power morally or physically and get around difficulties by acting together, as we observe in nature and in our daily affairs. Working on robots and investigating of intelligence, particularly distributed intelligence, have been my research objectives since my early years at Istanbul Technical University.

Prof. Tucker Balch helped make many of my research dreams come true. He has my deepest appreciation for believing in my research, continually supporting me, and advising me in the way I could improve myself. It has been a pleasure to work with him in the BORG Laboratory at Georgia Institute of Technology.

In conducting this PhD research and writing this thesis book, I have been very fortunate to have Prof. Nadia Erdoğan as my co-advisor. I'm deeply indebted to her for helping me advance my research. In many cases her affectionate encouragement helped me keep my motivation up.

I feel lucky to have met many great scientists, especially those who greatly support and encourage young researchers in their research. I am grateful to Prof. Martin Savelsbergh, Prof. Pınar Keskinocak, Prof. Sven Koenig, Prof. Michail Lagoudakis and Prof. Ashok Goel for their valuable comments on my research.

I would also thank Prof. Emre Harmancı, Prof. Levent Akın, Prof. Tevfik Akgün, Prof. Bilge Günsel and Prof. Turgay Altılar for the continuous moral support they provided during my research studies at Georgia Tech.

Many thanks to young geniuses: Ram Ravichandran, Keith O'hara, Arya Irani, Victor Bigio, Ananth Ranganathan, Adam Feldman, Michael Kaess, Matt Powers, Sang Min Oh and all the members of the robotic-discussion-group for the valuable discussions on robot research and for their friendship.

Pınar Taşkıran, Sibel Yaman, Faik Başkaya, Can Envarlı and Ping Wang have always been very supportive especially when I felt lonely during this long journey. My special thanks go to them.

I specifically thank Jacquelyn Berry and Zuhal Yılmazer for their precious efforts in official tasks and for their valuable friendship.

I am deeply grateful to A. Çağatay Talay for his friendship, continuous support and encouragement throughout this research and his willingness to take over all my administrative burdens while I was in Atlanta.

I would finally thank my mother, Necla Sarıel, and my sister, Selen Sarıel. This PhD thesis research could not have been conducted and completed without their continuous support, love and patience.

Finally, this PhD book has come to a conclusion. However, I would like to emphasize what my advisors always mention:
"This is not the end of my research. It's just the beginning..."

June 2007 Sanem SARIEL

CONTENTS

ABBREVIATIONS ix
LIST OF TABLES xi
LIST OF FIGURES xi
LIST OF SYMBOLS xiv
SUMMARY xvi
ÖZET xix
1. INTRODUCTION 1
1.1. Open Issues in Multi-Robot Coordination Research 2
1.2. A Brief Overview of the Proposed Approach 2
1.3. Contributions of the Thesis 3
1.3.1. DEMiR-CF 3
1.3.2. Formulation of the Cooperative Mission Achievement and Coordinated Task Selection Problems 5
1.3.3. Integration of Task Allocation, Execution and Contingency Handling into a Single Framework 5
1.3.4. Real-world Suitability with Limited Assumptions 6
1.3.5. Investigation and Generation of Novel Solutions for Different Application Problems 6
1.3.6. Combination of The Methods from Different Disciplines 6
2. PROBLEM STATEMENT AND MOTIVATION 7
2.1. Multi-Robot Mission Achievement 7
2.2. Real-Time Issues and Requirements for Multi-Robot Task Achievement 9
2.3. Multi-Robot Cooperation vs. Coordination 10
2.4. Formulation of the Cooperative Mission Achievement Problem 10
3. BACKGROUND AND RELATED WORK 13
3.1. A Brief Review on the Classification of Earlier Multi-Robot Systems 13
3.2. Organization and Control Hierarchy 14
3.2.1. Centralized Approaches 14
3.2.2. Decentralized Approaches 14
3.3. Task Representation for Coordination 15
3.4. Task Allocation and Coordination Type Based on the Mission Structure 16
3.4.1. Tight/Loose Coordination 17

3.4.2. Allocation of Tasks with Dependencies 17
3.4.3. Combinatorial Effects on Task Assignment 18
3.5. Instantaneous Task Assignment/Scheduling 18
3.6. Reallocation and Dynamic Task Switching 19
3.7. Bounds on the Solution Quality 20
3.8. Organizational Requirements on Multi-Robot Task Execution and Coordination 20
3.8.1. Coalition Formation 21
3.9. Communication and Coordination Tools 22
3.9.1. Negotiations 22
3.9.2. Auctions 23
3.9.3. Contract-Net-Protocol 23
3.10. Failure Detection and Recovery 25
3.11. Interleaving Planning and Execution 26
3.12. Application Domains 27
3.13. Multi-Robots vs. Multi-Agents 28
3.14. Summary and Discussion 29

4. DEMiR-CF: DISTRIBUTED AND EFFICIENT MULTI-ROBOT - COOPERATION FRAMEWORK 30
4.1. Integrated Modules of DEMiR-CF 30
4.2. Mission Representation 33
4.3. Inputs and Outputs of DEMiR-CF 35
4.4. Dynamic Priority-based Task Selection Scheme 36
4.4.1. Rough Schedule Generation Scheme 38
4.4.2. DPTSS Algorithm 39
4.5. Cost Evaluation for Rough Schedule Generation and Dynamic Task Selection 40
4.5.1. Cost Function Design Criteria 41
4.5.1.1. Independent Tasks with Combinatorial Structures 41
4.5.1.2. Interrelated Tasks with Multi-Robot Requirements 42
4.6. Task Allocation 43
4.6.1. Distributed Task Allocation Scheme 43
4.6.2. Communication in DEMiR-CF 44
4.6.3. Roles 45
4.7. Coalition Maintenance/Dynamic Task Switching Scheme 47
4.8. Plan B Precautions: A System-wide Contingency Handling Mechanism 49
4.8.1. Representation of The System Model In Each Robot's World Knowledge 49
4.8.2. Plan B Precaution Routines 52

5. EMPIRICAL EVALUATION OF DEMiR-CF ON THE MULTIPLE TRAVELING ROBOT PROBLEM 54
5.1. MTRP Problem Statement 54
5.1.1. Remarks on the MTRP Characteristics 55
5.1.2. Operations Research Methods for the MTRP 57
5.1.2.1. Integer Programming Formulation 57
5.1.2.2. Branch and Bound Approach 58

5.1.2.3. Heuristics 58
5.1.3. Robotic Research Methods for the MTRP 58
5.1.3.1. Prim Allocation Method 59
5.1.3.2. Other solutions for the MTRP 60
5.1.4. Cooperation Objectives 60
5.1.5. Application Domains for the MTRP 61
5.1.6. Communication Requirements 63
5.1.7. Formation of the Mission Structure for the MTRP 63
5.2. Applying DEMiR-CF to the MTRP 64
5.2.1. The Dynamic Priority Based Task Selection Scheme 64
5.2.1.1. Selecting is Eliminating the Other: Incremental Allocation through Unconditional Commitments 64
5.2.1.2. Cost Estimation and Evaluation 65
5.2.1.3. Dynamic Task Selection and Distributed Target Allocation 66
5.2.2. Failure Detection and Recovery 68
5.3. Bounds on the Solution Quality 69
5.3.1. Analysis of the Approach 69
5.4. Implementation Details 70
5.5. MTRP Experiments 71
5.5.1. Experiment 1: Evaluation of the FAC Heuristic Cost Function 73
5.5.2. Experiment 2: Performance Comparison of DEMiR-CF and the Prim Allocation Approach with the Optimum 73
5.5.3. Experiment 3: Real-Time Dynamic Simulation Experiments 76
5.5.4. Experiment 4: Real-Time, Real-World and Dynamic Environment Experiments 80
5.6. Summary and Discussion 84

6. EMPIRICAL EVALUATION OF DEMiR-CF ON NAVY MISSIONS 86
6.1. Naval Mine Countermeasure Missions 86
6.2. Applying DEMiR-CF to the Naval MCM 87
6.2.1. Task Representation for The MCM 87
6.2.2. Exploration for Detection and Classification of MLO Locations 88
6.2.3. Identification of Mine-like Objects 89
6.3. Experimental Results on the Naval MCM 90
6.4. A NAVY Homeland Security Application 94
6.5. Summary and Discussion 98

7. EMPIRICAL EVALUATION OF DEMiR-CF ON RESOURCE CONSTRAINED AND INTERRELATED TASKS 99
7.1. Complex Multi-Robot Mission Problem Statement 99
7.2. Applying DEMiR-CF to Complex Missions 100
7.2.1. Dynamic Priority-based Task Selection Scheme and Online Scheduling 100
7.2.1.1. DPTSS Algorithm 103
7.2.2. Distributed Task Allocation Scheme 104
7.2.3. Cost/Bid Evaluation and Tie Breaking Rules 104
7.2.4. Analysis of the Approach 105
7.3. Complex Mission Experiments 105

7.4. Summary and Discussion 110

8. DISCUSSION AND CONCLUSION 111
8.1. Future Work 115

BIBLIOGRAPHY 116

APPENDIX 123

A.TSP HEURISTICS 124

B.TSPLIB INSTANCES 126

BIOGRAPHY 131

ABBREVIATIONS

ALWSE-MC	:	Autonomous Littoral Warfare Systems Evaluator-Monte Carlo
AUV	:	Autonomous Underwater Vehicle
CA	:	Christofides Algorithm
CC	:	Closest Cost
CNP	:	Contract-Net-Protocol
CDP	:	Cooperative Distributed Planning
CIH	:	Cheapest Insertion Heuristic
CMAP	:	Cooperative Mission Achievement Problem
CTSP	:	Coordinated Task Selection Problem
DAG	:	Directed Acyclic Graph
DEMiR-CF	:	Distributed and Efficient Multi-Robot - Cooperation Framework
DPTSS	:	Dynamic Priority-Based Task Selection Scheme
ECT	:	(Smallest) Earliest Completion Time
EDD	:	Earliest Due Date
EST	:	(Smallest) Earliest Starting Time
FAC	:	Farthest Addition Cost
FIH	:	Farthest Insertion Heuristic
FIPA	:	Foundations of Intelligent Physical Agents
FSM	:	Finite State Machine
GRR	:	Greatest Resource Requirements
GRPW	:	Greatest Rank Positional Weight
IP	:	Integer Programming
KQML	:	Knowledge Query Manipulation Language
LCT	:	(Smallest) Latest Completion Time
LIS	:	Least Immediate Successors
LPT	:	Longest Processing Time
LST	:	(Smallest) Latest Starting Time
LTS	:	Least Total Successors
MCM	:	Mine CounterMeasure
MILP	:	Mixed Integer Linear Program
MIS	:	Most Immediate Successors
MLO	:	Mine-like Object
MPCP	:	Multiagent Plan Coordination Problem
MRTA	:	Multi-Robot Task Allocation
MSLK	:	Minimum Slack
MTS	:	Most Total Successors
MTSP	:	Multiple Traveling Salesman Problem
MTRP	:	Multiple Traveling Robot Problem
NDP	:	Negotiated Distributed Planning
NIH	:	Nearest Insertion Heuristic
NMH	:	Nearest Merger Heuristic
NNH	:	Nearest Neighborhood Heuristic

OPT	:	Optimal
OR	:	Operations Research
RCPSP	:	Resource Constrained Project Scheduling Problem
SA	:	Search Area
ScP	:	Scheduling Problem
SLAM	:	Simultaneous Localization and Mapping
SPT	:	Smallest Processing Time
SR	:	Search and Rescue
TBD	:	To be determined
TDL	:	Task Description Language
TSP	:	Traveling Salesman Problem
UAV	:	Unmanned Aerial Vehicle
UGV	:	Unmanned Ground Vehicle
UUV	:	Unmanned Underwater Vehicle

LIST OF TABLES

		Page No
Table 4.1	Robot Team and Capabilities in the Box Mailing Mission	34
Table 4.2	Representation of the tasks of the Box Mailing Mission	35
Table 4.3	Message types in DEMiR-CF	46
Table 4.4	Precautions for contingencies and conflicts	52
Table 4.5	Model checking for tasks and system robots	53
Table 4.6	Model updates related to the messages	53
Table 5.1	FAC heuristic function performance results for the known TSP instances	72
Table 6.1	The performance results for different message loss rates	94
Table 6.2	The cost evaluations for the application domain	97
Table 7.1	Cost evaluations for the tasks	105

LIST OF FIGURES

		Page No
Figure 3.1	A sample mission representation as an AND/OR task tree	16
Figure 3.2	A sample multi-agent plan representation as task graphs	17
Figure 3.3	Contract Net Protocol	24
Figure 4.1	DEMiR-CF Modules	32
Figure 4.2	Directed acyclic mission graph for the Box Mailing Mission	33
Figure 4.3	The Box Mailing Mission initial state	33
Figure 4.4	Basic steps of an auction negotiation process	44
Figure 4.5	An invalid auction cancellation	45
Figure 4.6	Coalition maintenance	48
Figure 4.7	States of the FSMs for single-robot task models	50
Figure 4.8	States of the FSMs for multi-robot task models	51
Figure 5.1	The optimal solution can be obtained by clustering the targets	55
Figure 5.2	Clustering the targets does not necessarily result in the optimal solution	56
Figure 5.3	MSF allocations	57
Figure 5.4	Effects of the MST traversal strategy on the total cost for the open traversal version of the TSP	59
Figure 5.5	Two different optimization objectives for the MTRP	61
Figure 5.6	Target selection strategy by the FAC Heuristic function	66
Figure 5.7	The Archimedean spiral and two simple implementations	70
Figure 5.8	The robot with correct localization finds the target	70
Figure 5.9	An implementation of the Archimedean spiral with the corresponding robot behavior	71
Figure 5.10	Results for the ATT48 TSP instance	72
Figure 5.11	Runtime comparison of the approaches for single robot route generation	73
Figure 5.12	Performance results of the heuristic approaches	74
Figure 5.13	Allocations generated by each approach for an instance of 15 robots and 50 targets	75
Figure 5.14	Scalability analysis for different number of robots	76
Figure 5.15	Performance results for the Contingency Handling Mechanism	77
Figure 5.16	The initial configuration of Scenario 1	78
Figure 5.17	The final configuration of Scenario 1	78
Figure 5.18	Maps and paths for Scenario 1	79
Figure 5.19	The final configuration of Scenario 2	79
Figure 5.20	Maps and paths for Scenario 2	79
Figure 5.21	The Khepera II robot visits six targets in the environment	80

Figure 5.22 Three Khepera II robots divide the labor of visiting the six targets 80
Figure 5.23 The team handles the failure of the third robot in real-time 81
Figure 5.24 Single robot cases for different initial deployment locations of the robots 81
Figure 5.25 Scenario 1: the multi-robot case without failure 82
Figure 5.26 Scenario 2: the multi-robot case in which the third robot fails after completing its task 83
Figure 5.27 Scenario 3: the multi-robot case in which the third robot fails before completing its task 83
Figure 5.28 Scenario 4: the multi-robot case in which the first robot fails before after completing its task 84
Figure 5.29 Scenario 5: the multi-robot case in which the second robot fails before completing its task 84

Figure 6.1 Conceptual flowchart related to the AUV operations 90
Figure 6.2 NAVY MCM Mission Scenario 1 91
Figure 6.3 NAVY MCM Mission Scenario 2 92
Figure 6.4 NAVY MCM Mission Scenario 3 93
Figure 6.5 Initial mission graph consists of only the search task 95
Figure 6.6 Robots patrol the area in the corresponding regions 95
Figure 6.7 A sample execution trace under highly dynamic situations 96
Figure 6.8 Mission graph and allocations evolving through time accordingly 97
Figure 6.9 Execution conflicts under limited communication are resolved 98

Figure 7.1 Mission completion time results for the pick-up/delivery mission 106
Figure 7.2 Total path length results for the pick-up/delivery mission 107
Figure 7.3 Scenario 1 and 2 108
Figure 7.4 Khepera II robots achieve the overall complex mission 109
Figure 7.5 Real scenario mission graph with interrelated tasks 109

Figure B.1 ATT48 TSPLIB instance with 48 nodes 127
Figure B.2 Optimal open-loop route for the ATT48 TSPLIB instance 127
Figure B.3 EIL51 TSPLIB instance with 51 nodes 128
Figure B.4 Optimal open-loop route for the EIL51 TSPLIB instance 128
Figure B.5 BERLIN52 TSPLIB instance with 52 nodes 129
Figure B.6 Optimal open-loop route for the BERLIN52 TSPLIB instance 129
Figure B.7 EIL101 TSPLIB instance with 101 nodes 130
Figure B.8 Optimal open-loop route for the EIL101 TSPLIB instance 130

LIST OF SYMBOLS

$\boldsymbol{\alpha}$	:	FAC heuristic parameter
$\boldsymbol{\mu}$	:	Average
$\boldsymbol{\sigma}$	:	Standard Deviation
$\boldsymbol{B_{ij}}$	:	Bidder robot r_j for task t_i
$\boldsymbol{c_{ij}}$	:	The cost value to traverse between i and j
$\boldsymbol{C_i}$	:	Completion time of task t_i
$\boldsymbol{CL_i}$	:	Coalition leader of $Coal_i$
$\boldsymbol{CM_i}$	:	Coalition member of $Coal_i$
$\boldsymbol{c\{i,j\}}$	:	Euclidean distance between i and j
$\boldsymbol{cap_j}$	:	Set of capabilities of robot r_j
$\boldsymbol{Coal_i}$	:	Coalition formed to execute task t_i
$\boldsymbol{curcs_j}$	:	Remaining capacity of robot r_j
$\boldsymbol{deplist_i}$	:	Dependency list of task t_i
$\boldsymbol{M}$	:	Mission to be achieved
$\boldsymbol{M^*}$	:	Minimum matching
$\boldsymbol{M_C}$	:	Coverage mission
$\boldsymbol{M_I}$	:	Identification mission
$\boldsymbol{M_{MTRP}}$	:	MTRP mission
$\boldsymbol{O}$	:	Objective function
$\boldsymbol{p_i}$	:	Actual processing time of task t_i
$\boldsymbol{P(t_i)}$	:	Set of all predecessor tasks of task t_i
$\boldsymbol{r_j}$	:	A single robot of the multi-robot team indexed by j
$\boldsymbol{relinfo_i}$	:	Task related information set for task t_i
$\boldsymbol{reldist}$	:	Relative distance
$\boldsymbol{precinfo_i}$	:	Precaution information set for task t_i
$\boldsymbol{reqcap_i}$	:	Set of capabilities of to execute task t_i
$\boldsymbol{reqcs_i}$	:	required capacity to execute task t_i
$\boldsymbol{reqno_i}$	:	Minimum required number of robots to execute task t_i
$\boldsymbol{R}$	:	Set of the robots in the multi-robot team
$\boldsymbol{R_{csj}}$	:	Roughly estimation on the remaining capacity for robot r_j
$\boldsymbol{R_{UUV}}$	:	Set of UUV robots
$\boldsymbol{S_i}$	:	Actual starting time of task t_i
$\boldsymbol{S_{Rj}}$	:	Rough schedule of robot r_j
$\boldsymbol{t_\phi}$	:	Ineligible task
$\boldsymbol{t_{Aj}}$	:	Active task
$\boldsymbol{t_{Cj}}$	:	Critical task for robot r_j
$\boldsymbol{t_{Ej}}$	:	Eligible task
$\boldsymbol{t_{Ij}}$	:	Inactive task
$\boldsymbol{t_i}$	:	A single task of the overall mission indexed by i
$\boldsymbol{t_{ie}}$	:	The task in execution
$\boldsymbol{t_{iej}}$	:	The task in execution by robot r_j
$\boldsymbol{t_b}$	:	Boundary target

$\boldsymbol{t_{sj}}$	:	The most suitable task to execute for robot r_j
$\boldsymbol{T}$	:	Set of the tasks of the overall mission
$\boldsymbol{T_{\phi}}$	:	Set of all ineligible tasks
$\boldsymbol{T_{Aj}}$	:	Set of all active tasks
$\boldsymbol{T_{Cj}}$	:	Set of all critical tasks for robot r_j
$\boldsymbol{T_{Ej}}$	:	Set of all eligible tasks
$\boldsymbol{T_{Ij}}$	:	Set of all inactive tasks
$\boldsymbol{T_{Rj}}$	:	Rough target set for r_j
$\boldsymbol{T_{TSP}}$	:	Partial tour for the TSP
$\boldsymbol{type}$	:	Type of task t_i
$\boldsymbol{w_i}$	:	Waypoint indexed by i
$\boldsymbol{w_b}$	:	Boundary waypoint
$\boldsymbol{W_{jk}}$	:	Rough Region Schedule for robot r_j and region k
$\boldsymbol{x_{ij}}$	:	Indicator (0/1) variable

AN INTEGRATED PLANNING, SCHEDULING and EXECUTION FRAMEWORK FOR MULTI-ROBOT COOPERATION and COORDINATION

SUMMARY

Although several architectures have been proposed for multi-robot coordination previously, the field is still in its early stages and robot teams are still test subjects in research laboratories. Fortunately, a growing community has been researching techniques for multi-robot systems. The basis has been formed for multi-robot systems to serve humanistic needs in the new future in many domains, such as search and rescue operations and space explorations in the real world. In fact, even though some of the open problems have been solved, some open issues still remain even for single bodies of robots.

The real dynamics of physical task performance force unplanned actions to be taken. Since the world is beyond the control of robots and changes continuously in real-world applications, the difficulty of the multi-robot task execution problem goes beyond the task allocation problem. In particular, multi-robot systems deal with difficulties arising from noisy sensor information, unexpected outcomes of actions, environmental limitations (especially in communication) and the presence of failures of hardware. Furthermore, the problem instance may be evolving through real time which is another important research challenge. All these factors may affect the overall solution. Against this background, research in this thesis addresses issues of real-time execution when managing an overall team by a central authority is not possible due to limitations of the real-world environments. Therefore, each individual robot should find a way to solve the global problem from a local perspective in a decentralized way.

The main contribution of this PhD research is the design of a general framework, **D**istributed and **E**fficient **M**ult**i**-**R**obot-**C**ooperation **F**ramework (DEMiR-CF), that can be used to solve problems on a wide variety of application domains. DEMiR-CF is suitable for multi-robot teams cooperating to achieve a global mission. Team members cooperate to fulfill a mission by dividing the labor of task execution through individual decisions that coordinate their actions, contributing to the achievement of the goal in a distributed manner.

The cooperative mission execution problem is formulated as the Cooperative Mission Achievement Problem ($CMAP$) and each individual robot contributes to solve the $CMAP$ by solving the formulated Coordinated Task Selection Problem ($CTSP$) for itself. These two problem formulations are introduced to represent the generalized problem and stated formally for the first time in this PhD thesis. DEMiR-CF is capable of resolving the $CMAP$ and $CTSP$ for robots.

In the design of DEMiR-CF, the following issues were particularly investigated as the design criteria: efficient and realistic representation of missions, efficient allocation of tasks to cooperatively achieve the global goal, maintenance of the system coherence and consistency by the team members, detection of the contingencies and recovery from various failures that may arise during runtime, efficient reallocation of tasks (if necessary) and reorganization of team members (if necessary).

DEMiR-CF is designed to address different types of missions from the simplest to more complex ones, including missions with interrelated tasks and multi-resource (robot) requirements. In the proposed mission representation, components of individual tasks are also allowed to be updated by robots depending on real-world requirements.

The framework ensures an efficient way of integrating task planning, allocation and execution for multi-entity (agent/robot) teams independent of the underlying low-level behavior architecture, yet without ignoring it. The integrated components of the framework ensure solving the $CMAP$ in a robust and efficient way. As a result, global planning, scheduling and execution is carried on by the cooperative work of robots performing under DEMiR-CF. Even though an incremental task selection approach is adopted, a global plan consideration is also preserved to make the system both sound and complete.

The performance evaluations of the framework were implemented both on simulations and on real robots for different application domains. Each application domain is a separate problem domain which requires in depth research.

The Multiple Traveling Robot Problem (MTRP) to explore several targets is the first application domain on which tests were performed. This domain forms a basis for different application domains. Although tasks of this domain are independent of each other, there is a combinatorial structure of the problem when efficiency of the solution is concerned. Therefore, optimization of the generated solutions is investigated. Some heuristic cost functions to solve the $CTSP$ is proposed to be used with DEMiR-CF in the thesis. The performance of the proposed heuristics and the framework is compared with that of one of the well-known allocation approaches.

The evaluations on NAVY domains were performed where cooperation of underwater vehicles is achieved for homeland security missions, such as mine countermeasure missions. In this domain, the robot team is modeled as a heterogeneous team and the mission is constructed from different types of tasks, where each one needs to be performed by a different vehicle. The domain is modeled as containing both the coverage problem and the MTRP in itself. Robustness of the framework against both communication and robot failures and efficiency of its response to the dynamically varying conditions are tested in simulations.

In general, DEMiR-CF targets complex missions involving tasks with resource constraints and interrelations. Therefore, these types of missions are perfect candidate domains to apply the full functionality of the framework. In the last experimental setup, specifically, pick-up/delivery and object construction domains are investigated as complex domains. Different from previous experiments, the robots are involved in more complex tasks where they interact with the objects in the environment. The objective is not only optimizing cost functions but also obeying rules and resolving constraints on task execution during runtime. The base mechanisms of DEMiR-CF are used to design the solution. The efficiency of the proposed solution for complex domains is evaluated through experiments.

ÇOKLU-ROBOT SİSTEMLERİNİN ORTAK ÇALIŞMASI ve KOORDİNASYONU İÇİN TÜMLEŞİK BİR PLANLAMA, GÖREV ATAMA ve YÜRÜTME MİMARİSİ

ÖZET

Literatürde çoklu-robot koordinasyonu için birçok mimari önerilmiş olmasına rağmen, bu konudaki araştırma alanı henüz gelişmekte olup üzerinde çalışılması gereken birçok açık nokta bulunmaktadır. Çoklu-robot sistemleri henüz laboratuvarlarda araştırma denekleri olarak kullanılmakta olsalar da yakın gelecekte birçok uygulama alanı için insanlık yararına hizmet etmek üzere ilk adımlar atılmıştır. Bu uygulama alanlarının başında arama-kurtarma operasyonları ve uzay araştırmaları gelmektedir. Gittikçe artan bir ilgiyle genişleyen çoklu-robot sistemleri araştırma grupları, çoklu-robot sistemleri için çeşitli metodlar üzerinde araştırma yapmaktadır. Ancak, tek robotlu sistemler için bile henüz çözülmemiş birçok problem bulunmaktadır.

Gerçek dünyada ortaklaşa otonom olarak çalışan donanımların görev başarımı daha önceden planlanmamış davranışlar ve olayları göz önüne almayı gerektirir. Robotlar kendilerinin kontrol altında tutamadıkları ve sürekli dinamik olan ortamlar ve uygulama alanlarında çalıştıklarından, çoklu-robot görev yürütme problemi görev paylaşımı probleminin de ötesinde çok daha zor boyutlara taşınır. Özel olarak, çoklu-robot sistemleri gürültülü sensör verileri, beklenmeyen davranışlar ve sonuçları, ortamsal kısıtlamalar (özellikle iletişimde) ve sık olabilecek donanım bozulma ve hataları gibi çeşitli durumlardan kaynaklanan zorluklar ile yüzleşmek zorundadır. Bunlara ek olarak problemin kendisi de ortam durumu veya başka bir sebeple değişim gösteriyor olabilir ki bu da problemi daha zor kılmaktadır. Tüm bu faktörler sistemin genel başarımını etkiler. Bu doktora tezi araştırması, tüm bu durumlara karşı gerçek zamanlı görev yürütme durumlarını da göz önüne alarak ortamdan kaynaklanan kısıtlamalar sonucu robot sistemini tek bir merkezden yönetmenin mümkün olmadığı durumlar için geçerli yöntemler üzerine odaklanmıştır. Dolayısıyla her bir robot tüm problemi yerel olarak dağıtılmış bir şekilde çözmek için bir yol bulmalıdır.

Bu tezin en önemli katkısı, birçok uygulama alanında kullanılabilecek olan genel bir mimarinin -DEMiR-CF (**D**istributed and **E**fficient **M**ult**i**-**R**obot-**C**ooperation **F**ramework)- tasarımı ve ele alınan uygulama alanları için bu mimari kullanılarak çözümler önerilmesidir. DEMiR-CF karmaşık bir ana işin başarılması için robotların ortaklaşa çalışmasını öngören bir mimaridir. Ortak çalışma, robotların dağıtılmış olarak karar alması ve görev paylaşımı yaparak bütün işin başarımına katkıda bulunması ile sağlanır.

Ortak iş yürütme problemi -$CMAP$ (Cooperative Mission Achievement Problem)- her bir robotun bu bütün problemi çözmek üzere kendisi için koordineli görev seçim problemini -$CTSP$ (Coordinated Task Selection Problem)- çözmesi ile gerçeklenir. Bu problem tanımları ve formülasyonları genelleştirilmiş ortak görev yürütme problemini temsil etmek üzere ilk kez bu tezde ortaya konmuştur. Önerilen mimari de bu problemleri çözmek üzere tasarlanmıştır.

DEMiR-CF tasarımında özellikle şu problemler ve çözümleri üzerine çalışılmıştır: görevlerin etkin ve gerçekçi şekilde temsili, görevlerin genel amacı gereçeklemek üzere etkin şekilde atanması, sistem bütünlüğünün ve tutarlılığının robotlar tarafından korunması, robotların yürütme zamanı oluşabilecek olağan dışı durumlar, hatalar ve bozulmaları tespit edip uygun hata kotarma yöntemlerini gerçeklemesi, gerekiyorsa görevlerin etkin şekilde yeniden atanması ve gerekli durumlarda takım üyelerinin yeniden organize olmaları.

DEMiR-CF en basit işlerden en karmaşık işlere kadar farklı tipte görevler üzerinde çalıştırılabilecek şekilde tasarlanmıştır. Karmaşık işler birbirlerine bağımlı ve çoklu kaynak (robot) gereksinimi olan görevler içerebilir. Önerilen mimaride, görevler, yürütme sırasında gerçek dünya gereksinimlerini karşılayacak şekilde güncellenebilmesine açık şekilde temsil edilmektedir.

Mimari, çoklu-robot (etmen) sisteminin görev planlaması, ataması ve yürütmesini alt katman davranış modelinden bağımsız olarak fakat onu gözardı etmeyecek şekilde gerçekleme imkanı sunar. Mimarinin tümleşik bileşenleri $CMAP$ problemini hataya karşı dayanıklı ve etkin şekilde çözmek üzere tasarlanmıştır. Dolayısıyla genel planlama, görev ataması ve yürütmenin eş zamanlı şekilde gerçeklenmesi, DEMiR-CF mimarisinde tasarlanmış robotlar tarafından mümkün olmaktadır. Robotlar dinamik ve artımlı olarak görev seçimini yürütürler, ancak tüm işi etkin olarak tamamlayacak şekilde genel plan değerlendirilmesi yapılır.

Mimarinin başarımı hem benzetim ortamları, hem de gerçek robotlar üzerinde, farklı uygulama alanlarında sınanmıştır. Aslında, her bir uygulama alanı derinlemesine çalışma gerektiren ayrı bir problemdir.

Çoklu-robot çoklu-hedef ziyaret etme problemi, başarım analizinin yapıldığı ilk uygulama alanıdır. Bu problem birçok uygulama alanı için bir temel oluşturmaktadır. Problem görevleri birbirlerinden bağımsız olsa da, çözüm kalitesi göz önüne alındığında, problemin kombinasyonal bir yapısı vardır. Dolayısıyla üretilen çözümlerin eniyilemesi üzerinde durulmaktadır. Bu tezde bu problemi çözmek üzere DEMiR-CF mimarisinde kullanılabilecek maliyet hesap fonksiyonları önerilmektedir. Önerilen bu hesap fonksiyonları mimari ile birlikte gerçeklenmiş ve literatürde iyi bilinen bir görev atama yöntemi ile karşılaştırılarak başarım analizi yapılmıştır.

Bir başka uygulama alanı olarak mimari, kıta sahanlığını korumaya yönelik askeri Deniz Kuvvetleri problemleri üzerinde gerçeklenmiştir. Buradaki robotlar sualtı araçları olup, örnek bir uygulama problemi olarak sualtı mayın tarama ve imha problemi ele alınmıştır. Robotlar farklı yeteneklere sahip olup heterojen

birimler olarak modellenmiştir. Ana iş, farklı yetenekler gerektiren ve bu nedenle farklı robotların yürütmesi gereken alt görevlerden oluşmaktadır. Bu problem içinde, hem ortam tarama, hem de "çoklu-robot çoklu-hedef ziyaret etme" problemi irdelenmiştir. Ayrıca, mimarinin iletişim problemleri ve robot bozulmalarına karşı hataya dayanıklılığı ve ortam dinamizmine karşı başarım iyileştirme yetenekleri irdelenmiştir.

DEMiR-CF mimarisi, en genel anlamda çoklu kaynak gereksinimleri olan ve birbirleri ile bağlantılı görevlerden oluşan karmaşık işleri çözmeyi hedef alır. Dolayısıyla bu tür karmaşık görevler, mimarinin tüm fonksiyonelliğini ölçmek üzere en uygun aday uygulama alanlarını oluşturur. Son deneysel platformda, karmaşık görevler olarak nesne alma/taşıma ve nesneler ile inşa uygulama alanları üzerinde çalışılmıştır. Önceki deneylerden farklı olarak, robotlar ortamdaki nesnelerle de etkileşim kurdukları karmaşık görevler üzerinde çalışmışlardır. Amaç sadece eniyileme değil, aynı zamanda kısıtlama ve kurallara uygun şekilde görev yürütmektir. Bu uygulama alanlarında, DEMiR-CF mimarisinin temel bileşenleri kullanılmıştır. Gerçeklenen deneylerde yöntemin etkinliği irdelenmiştir.

1. INTRODUCTION

Research on multi-robot coordination has come into prominence due to the recent demand for Unmanned Ground Vehicle (UGV), Unmanned Aerial Vehicle (UAV), Unmanned Underwater Vehicle (UUV) or space rover teams for both humanitarian and military applications.

Multi-robot systems are suitable for domains in which the completion of the mission is not possible with a single robot or situations in which coordination of more than one robot contributes to the efficiency of the overall system. Sample situations observed in nature also reveal how utility is gained when entities behave in a cooperative manner. Cooperation among animals and/or humans is the most apparent evidence of this theory.

Another reason for using multi-robot systems is the innate requirements for distributing the system and the solution. In fact, the structure of the problem domain may inherently possess distributed parts in space, time or functionality. Such sample domains are listed below:

- Cooperative object transportation and manipulation, manufacturing, site preparation, underwater construction, or pick-up/delivery
- Urban search and rescue
- Urban reconnaissance/surveillance
- Coverage for map building, searching, snow removal, lawn mowing, car body painting, harvesting, etc.
- Formation generation and reconfiguration for special purposes
- Robot soccer
- Cooperative multi-target search and tracking
- Hazardous cleanup, waste cleaning, mine cleaning (ground/underwater)
- Planetary exploration

Sometimes robot application domains contain assemblage of these missions. Some of these domains are adversarial, that is, robots compete against each other, such as in multi-robot soccer teams. However, in this research, we focus on cooperative multi-robot systems. Our proposed approach may also fit in adversarial domains as well, interpreting the competition as facing difficulties of the environment in a cooperative domain.

1.1. Open Issues in Multi-Robot Coordination Research

Although several architectures have been proposed for multi-robot coordination, the field is still in its early stages with a great deal of open questions in many dimensions. In fact, even though some of the open problems have been solved in single robot research, some open issues still remain even for single bodies of robots.
Open issues of multi-robot coordination that we would like to address are:

- How can tasks be represented for effective cooperation?
- Which robot should perform which task in order to cooperatively achieve the global goal? Who should take which role in the team?
- How is group coherence and consistency maintained?
- Which information is shared when there are environmental, communication and/or process power limitations?
- How are contingencies detected?
- How are allocations re-organized when contingencies are detected?

In addition to these open questions, Parker (2004) brings forward the open issue of "whether a general architecture can be implemented to cover a wide area of applications" in her multi-robot research review.

1.2. A Brief Overview of the Proposed Approach

This PhD research aims to focus on the investigation of the open issues in multi-robot task allocation and execution problem in the context of autonomous achievement of a mission and to propose novel algorithms to address open issues, explicitly for task allocation and execution. We do not touch on research on low-level essential robotic elements such as localization, mapping, etc. Our

main objective is to present a new effective strategy, integrating task planning, allocation and execution for multi-entity (agent/robot) teams, independent of the underlying low-level behavior architecture, yet without ignoring it. Our research is closely related to Operations Research and Scheduling Theory, Distributed AI/Multi-Agent Systems, Planning, and undoubtedly Robot Research.

Several real-time and real-world issues and limitations are analyzed for different application domains in this PhD research. Then, the problem is formulated as Coordinated Task Selection Problem for each robot in a decentralized setting to solve the reformulated the Cooperative Mission Achievement Problem. To solve the formulated problem, a novel decentralized multi-robot cooperation framework is proposed. The new approach integrates incremental task selection, distributed task allocation and contingency handling mechanisms to be performed during runtime in a single framework for efficient achievement of complex missions involving both resource-constrained and interrelated tasks. Experiments to analyze the performance of the new approach are conducted on different application domains in both simulations and in real environments with real robots. In addition to the proposal of this new framework, this study also provides real-world applicable solutions using the new framework for separate application domain problems.

1.3. Contributions of the Thesis

The main contributions of this PhD research are threefold. The first one is the formulation of the Cooperative Mission Achievement Problem ($CMAP$) and the Coordinated Task Selection Problem ($CTSP$) based on the Operations Research, Robot Research and Distributed Artificial Intelligence Research problems. The second and most important contribution is the proposal of an integrated cooperation and coordination framework, **D**istributed and **E**fficient **M**ulti-**R**obot-**C**ooperation **F**ramework (DEMiR-CF), for a multi-robot team to solve the $CMAP$ and $CTSP$ efficiently. The third contribution is the set of near optimal solutions generated for several application domains with DEMiR-CF. We will briefly explain these contributions in the following subsections.

1.3.1. DEMiR-CF

DEMiR-CF is an incremental and dynamic task allocation framework suitable for a multi-robot team to achieve a complex mission cooperatively in a distributed manner, simultaneously handling real-time contingencies.

In the design of DEMiR-CF, its suitability and applicability to many types of robot platforms is targeted and computationally tractable procedures are proposed in the framework to be performed on robots even with limited computational capabilities. The following properties in DEMiR-CF are endeavored to be achieved:

- **Integrity:** DEMiR-CF integrates task planning, scheduling and execution into a single framework for real-time task achievement. From our point of view, in order to obtain globally (near-)optimal solutions, task allocation, execution and contingency handling should be integrated into the cooperation framework without assuming they are achieved separately. This is the main rationale behind our framework.

- **Efficiency:** The main strength of DEMiR-CF is its incremental and dynamic task selection and allocation strategy through forming rough schedules which is proved to be preferable when compared to the complete allocation strategies for multi-robot systems. This approach can generate highly acceptable solutions for real-time task achievement with a significantly shorter response time. The efficiency of DEMiR-CF is validated through evaluations performed on different simulators and in real experimental environments with robots, as well for different NP-Hard problems.

- **Flexibility:** DEMiR-CF is a highly flexible framework that responds immediately to changes at runtime and adapts its behavior accordingly. Its flexible task representation design allows for dynamic changes in the decomposition of tasks during runtime. Online tasks that are generated during runtime by either operators or active robots can be easily integrated into the problem instance and the required actions are performed immediately by means of the incremental task selection approach. Resource constraints on task execution can also be dynamically updated based on the physical requirements determined by robots during runtime task execution.

- **Robustness:** *Plan B Precautions*, an extensive design of the precaution routines and the solution quality maintenance schemes are integrated into DEMiR-CF for robots to efficiently recover from contingencies. Therefore, even when (1) the communication is not reliable, (2) there is the potential risk of failures, and (3) the environment is dynamic or unstructured, the framework can ensure robustness by reconfiguring the allocations in an efficient and completely distributed manner.

- **Consistency:** DEMiR-CF ensures a globally consistent system and updated robot models regardless of the availability of resources in a completely distributed manner. System-wide consistency is achieved through the designed precaution routines, which results in the elimination of redundant efforts.

- **Generality:** The intended application domains of DEMiR-CF cover a broad range. This is validated through different implementations of the framework in different domains. It is expected that the evaluations of DEMiR-CF on different domains also contribute to the research on the individual problems and the application domains.

- **Applicability:** DEMiR-CF can easily be used even on very small robots with limited computational capacities and capabilities without dependencies on specific software and/or hardware.

1.3.2. Formulation of the Cooperative Mission Achievement and Coordinated Task Selection Problems

We formulate the multi-robot mission achievement problem stating the requirements on the effective task allocation and reallocations against the real dynamics. The Coordinated Task Selection Problem is a part of the Coordinated Mission Achievement Problem for robots to select the most suitable task by a time-extended view of the problem.

1.3.3. Integration of Task Allocation, Execution and Contingency Handling into a Single Framework

The main strength of DEMiR-CF is that it integrates continual task allocation and execution capabilities along with the contingency handling mechanisms. In every phase of the design, the main consideration has been to combine different aspects of the framework so that they form a coherent whole. The implementation of DEMiR-CF, a high level framework, takes place on top of the main robot architecture. The dynamic task selection, distributed task allocation and contingency handling mechanisms of DEMiR-CF are smoothly integrated into each other to achieve the real-world complex missions by a multi-robot team. DEMiR-CF can easily be implemented on different robot platforms along with low-level procedures for motor and sensor interface, localization and mapping.

Target allocation and route construction are integrated into each other by an incremental assignment approach for the multi-target exploration domain.

Multi-robot planning is integrated into task allocation for complex missions of interrelated and multi-resource dependent tasks.

1.3.4. Real-world Suitability with Limited Assumptions

In our research, we endeavor to limit assumptions and make the system mirror reality to the greatest extent. Therefore, we have focused on real-time situations and designed the framework as capable of detecting and recovering from various real-world failures efficiently. This goal is achieved through an extensive design of precaution routines, which ensure the suitability and robustness of the framework in real-world domains. Several design criteria are investigated for the potential situations that may occur in real-time and suitable recovery actions are activated.

1.3.5. Investigation and Generation of Novel Solutions for Different Application Problems

While designing DEMiR-CF, evaluations on different domains were performed on both simulated and real robots. Each separate domain is a stand alone problem domain for multi-robot systems. By applying the framework to separate domains, both the generality of DEMiR-CF is evaluated and these individual problems are investigated in detail. Different formulations of the problems and sometimes integration of the domains appeared as new testbeds. Since different domains require different cost function designs, the treatment of these individual tasks is also investigated.

1.3.6. Combination of The Methods from Different Disciplines

We are inspired by the methods in Robot Research, Distributed Artificial Intelligence Research and Operations Research in the design of DEMiR-CF. This interdisciplinary research highlights both the synergy between the problems of these different domains and the ability to form new routes on different views of the problems.

Finally, in doing this research, we aimed at developing ideas, approaches and algorithms for robots to serve our humanistic and protective needs in a better way, to challenge disasters beyond humans and/or to make use of technology to better understand nature (e.g., in space exploration experiments). Although these ideas may be used for protecting ourselves from each other, we are totally against the use of these ideas for any destructive and inhuman objectives.

2. PROBLEM STATEMENT AND MOTIVATION

In this chapter, we present the problem investigated in this Ph.D. research and its formulation based on the problem formulations in Operations Research (OR).

We are inspired by the Resource Constrained Project Scheduling Problem (RCPSP) (Brucker, 2002) treated in OR to formulate the general multi-robot multi-task allocation problem. Beyond the base problem, unpredictability of the exact processing times of tasks, unstable cost values during runtime and inconsistencies due to uncertain information form the main difficulties of the task allocation problem for robot systems. Particularly, robots also deal with real-world missions that may change their forms by introducing new online tasks during execution, making the problem more challenging besides the real-world dynamism.

2.1. Multi-Robot Mission Achievement

In this research, we investigate a framework for a multi-robot team to efficiently allocate tasks among themselves and achieve the overall mission. This problem simply deals with "who executes which task and when?" Therefore, the problem can be divided into several sub-problems: First, the overall mission should be decomposed into tasks to be executed by different robots. After this, the tasks should be allocated to the robots with an efficient representation. Based on the allocations, the robots begin performing the corresponding tasks. During task execution, group coherence and consistency should be maintained in several different situations that may appear in the real world. The robots need to exchange information and adapt to the changing situations. Meanwhile robustness should be ensured by detecting the anomalies, and the system should recover from them.
There are three main allocation schemes to find schedules for robots:

1. Applying a centralized approach which executes on the operational leader robot(s),

2. Applying a centralized approach which executes on each robot, or

3. Applying a distributed approach which executes on each robot

The overall schedule may be carried out by using operations research methods (the first allocation method). However, shortcomings of the central authorities and scheduling all tasks from scratch, especially in executing real-world tasks are numerous, as listed below:

- For the decision to be made by a centralized (semi-centralized) algorithm, the related information should be collected from all robots (even current fuel levels). In dynamic environments, updated information related to the resources (robots) should be continually recollected.
- Results of the scheduling process should be continually delivered to the individual robots.
- Additional complications for determining the decision-making authority exist, if there is more than one decision-making authority.
- These former issues should be handled in dynamic real-task environments which are often unpredictable and noisy.
- A huge amount of redundant scheduling cost arises, if globally (near-)optimum solutions are desired.
- The system becomes sensitive to single/multiple point(s) of failures.

Furthermore, besides the real-time execution burdens, the robots have to deal with others' plans in the first allocation scheme.

The second allocation method does not suggest coordination among robots. The robots may plan themselves without knowledge about the intentions of the other robots, and consequently, further inconsistencies may arise.

The third allocation scheme, on the other hand, can be applied by an effective, explicit task allocation approach and can provide a scalable and efficient way of distributing tasks. In this approach, robots announce their intentions regarding a task selected to execute, may reason about the tasks that others have selected and can make future plans based on this information. The decision-making authority is therefore distributed among the robots, thus arriving at a scheduling solution in a distributed manner. However, it may not guarantee finding the optimal schedules. By introducing efficient heuristic cost functions, the solution quality may be improved.

Our research addresses issues of real-time execution when the managing of the overall team by a central authority is not possible due to limitations of the real-world environments. Therefore, each individual robot should find a way to solve the global problem from a local perspective while thinking as globally as possible as in the third allocation scheme.

2.2. Real-Time Issues and Requirements for Multi-Robot Task Achievement

Even in the case of carefully written orchestra scores or playbooks, the real dynamics of physical task performance force some unplanned actions to be taken. Since the world is beyond the control of the robots and changes continuously in real-world applications, the difficulty of the multi-robot task execution problem goes beyond the task allocation problem. In particular, multi-robot systems deal with difficulties arising from noisy sensor information, unexpected outcomes of actions, environmental limitations (especially in communication) and the presence of failures of hardware. All these factors may affect the overall solution. We list evolving circumstances that may change the solution as:

- Self failure detection: Robots detecting their own failure.
- Robots detecting the failure of another robot.
- Change in the estimated task execution cost/time: Environmental dynamics, uncertain knowledge, or hardware problems may cause delays in task execution or early achievements of tasks. Uncertain sensor and/or localization information may also result in incorrect estimations.
- Change in the task definitions: Task dependencies, priorities, or the overall objective (goal) may change. Some tasks may become invalid during runtime.
- New online tasks introduced by human operators or discovered by the robots themselves.
- New robots being released, or some failed robots being repaired or recovering from trap-like threats.
- Intervention and manual changes on assignments by external agents.

Some of these situations may arise after either internal or external events. Given these contingencies, even the result of an approach capable of finding the optimal solutions may become suboptimal under the uncertainties of the real-world

applications. Verification of the solution optimality is also a difficult issue for real-world applications.

2.3. Multi-Robot Cooperation vs. Coordination

Malone and Crowston (1994) define coordination as managing dependencies among activities. Durfee (2001) defines coordination as an agent's fundamental capability to decide on its own actions in the context of the activities of other agents around it. According to Jennings' definition (Jennings, 1996), coordination is the process by which an agent reasons about its local actions and the (anticipated) actions of others to try and ensure that the community acts in a coherent manner.

Cooperation on the other hand refers to the practice of people or greater entities working in common with commonly agreed-upon goals and possibly methods instead of working separately in competition (Wikipedia-Cooperation).

Therefore, there is an important distinction between cooperation and coordination. This distinction is made in desJardins et al. (1999), where multi-agent interaction is classified under two planning perspectives: Cooperative Distributed Planning (CDP) and Negotiated Distributed Planning (NDP). Although there is no clearly defined boundary between these models, CDP agents typically exchange information about their plans, which they iteratively refine and revise until they fit together, whereas NDP agents negotiate over planned activities to ensure that their local objectives are met by their plan, when viewed in a global context. Coordination is defined as "incremental merging of individual agent sub-plans (multi-agent filtering)".

2.4. Formulation of the Cooperative Mission Achievement Problem

General multi-robot task allocation problem may be formulated based on the well known Resource Constrained Project Scheduling Problem (RCPSP) in OR (Brucker, 2002). RCPSP is known to be an NP-Hard problem (Weglarz, 1999). The adapted version of the formulation for our multi-robot task allocation problem on project tasks is given as follows. A complex mission consists of a set of tasks $T = \{t_1, ..., t_n\}$ which have to be performed by a team of robots $R = \{r_1, ..., r_m\}$. The tasks are interrelated by two types of constraints. First, precedence constraints are defined between activities. These are given by relations

$t_i \prec t_j$, where $t_i \prec t_j$ means that task t_j cannot start before task t_i is completed. Second, a task t_i requires a certain set of capabilities $reqcap_i$ and certain number of robots (resources) $reqno_i$ to be performed.
Using the given notation, Scheduling Problem (ScP) is defined as determining starting times of all tasks in such a way that:

- at each execution time, the total $reqno_i$ for a task t_i is less than or equal to the number of available robots ($R_{Sj} = \cup r_j$) with $reqcap_i \subseteq cap_j$ (Condition-1).
- the given precedence conditions (Condition-2) are fulfilled, and
- the makespan $C_{max} = max(C_i),\ 1 \leq i \leq n$ (Objective, O) is minimized, where $C_i = S_i + p_i$ is assumed to be the completion of task t_i, where S_i is the actual starting time and p_i is the actual processing time respectively.

This problem can be also be stated as a multiprocessor task scheduling problem and it is proved to be an NP-Hard problem by Brucker (2001).

Beyond this base problem, the real-time issues presented earlier add further dimension into this problem. It's not always possible to estimate the exact processing times (p) of tasks in real-world missions, especially those in which robots are involved. However, to form a complete schedule, it is necessary to make an approximation in terms of the best knowledge available.

The cost values are unstable during runtime and usually inaccurately estimated. Since the sensor values of the robots are noisy and world knowledge is uncertain, inconsistencies are unavoidable. Particularly, the robots also deal with real-world missions that may change their forms by introducing new online tasks during execution, making the problem more challenging besides the real-world dynamism.
Difficulty of the task allocation/reallocation problem arises when communication is limited and robots should autonomously perform task allocation at the same time as task execution. Simultaneous execution requirements make the problem more challenging because each robot should be in its most suitable execution in a future formation and estimate it correctly before making a decision with the minimum communication possible.

The Coordinated Task Selection Problem ($CTSP$) we propose is a part of the Scheduling Problem (ScP) and is stated for each robot. When each $CTSP$ is solved in a time-extended manner, an overall schedule of the tasks and their

executors can be found. The new formulated task selection problem that robots try to solve is given as follows: $CTSP$ for each robot (after either being idle or completing a task) is determining the next task t_i to be selected in such a way that:

- task t_i is not achieved yet,
- total $reqno_i$ for task t_i is less than or equal to the number of available robots ($R_{Sj} = \cup r_j$) with $reqcap_i \subseteq cap_j$ (Condition-1),
- the given precedence conditions (Condition-2) are fulfilled,
- and the selected objective (O) is minimized.

Including the $CTSP$, the Cooperative Mission Achievement problem ($CMAP$) for each robot is formulated as follows:

1. Select the task in such a way that the $CTSP$ is satisfied,
2. Determine the most appropriate robot (coalition) according to communication or beliefs to execute the task; resolve conflicts, if any,
3. Execute the selected task efficiently, if it is appropriate to execute, and
4. Simultaneously respond to contingencies and return to Step 1, when necessary, until the mission is achieved.

The second step in the formulation of $CMAP$ is required due to the uncertainties in the knowledge of robots. If robots had complete knowledge over the world state, then this step would become redundant. Real-world limitations make the fourth step an inevitable part of this problem.

3. BACKGROUND AND RELATED WORK

The goal of this section is to introduce the main concepts and basic vocabulary needed to comprehend the proposed approach. The material has been classified under different subtitles including corresponding related work to give better insight into the principles used in the research. Each corresponding section first introduces coordination mechanisms, then reviews the earlier implementations that utilize them.

3.1. A Brief Review on the Classification of Earlier Multi-Robot Systems

There are two different types of approaches for multi-robot coordination: implicit/emergent coordination and explicit/intentional coordination. Implicit coordination methods take their grounds from biological inspirations. These systems are suitable for large teams to effectively achieve an overall mission with the local view of each individual team member. This approach produces highly effective solutions when combined with behavior based architectures for domains such as foraging (Balch and Arkin, 1998). However, these systems are usually domain-dependent. Explicit coordination schemes, on the other hand, embody more complex representations and algorithms compared to the implicit case, so their generalization is much easier. They may be extended to address a wide variety of applications if implemented effectively. In this section, we review the existing work in explicit multi-robot coordination for team tasks in detail, leaving implicit approaches out of our scope.

Gerkey and Mataric (2004) present a taxonomy for the Multi-Robot Task Allocation (MRTA) problem. In their analysis of this problem, they state that utility is the core subject of optimization of overall solution quality. Since utility is a measure of both the robot's state and that of the environment, its inexactness due to sensor noise, uncertainties, and changes in the environment makes the multi-robot coordination problem difficult. Additionally, it is emphasized that it is often difficult to measure the main objective being optimized during execution. In their taxonomy, the problems can be classified within three different dimensions:

- Single-task robots (ST) vs. Multi-task robots (MT): Robots may execute only single tasks at a time or may be in execution of more than one task.
- Single-robot tasks (SR) vs. Multi-robot tasks (MR): A single task execution may require one or more robots to execute.
- Instantaneous assignment (IA) vs. Time-extended assignment (TA): Arrival of new tasks may be instantaneous, that is, there may be no knowledge of the release time of tasks, or task information may be given to the system initially.

According to this classification, current multi-robot systems are considered as an instantiation of each dimension and a combination of them.

3.2. Organization and Control Hierarchy

Intentional multi-robot coordination can be achieved by either centralized control or decentralized control. In the following subsections, we review these approaches along with recent research.

As presented in the previous chapter, the centralized approach is not usually successful, especially when communication is limited between the operator and the robots or with the presence of a high possibility of single point of failure. Therefore, this work focuses on distributed coordination frameworks, and we broadly review the literature on this subject.

3.2.1. Centralized Approaches

The initial focus in multi-robot research was on centralized approaches. One of the earlier works proposed by Tews (2001) is a centralized coordination framework, emphasizing the importance of sharing the same timing and model of environmental parameters concerned with the activity for coordinating entities. Recently, Koes et al. (2005) have proposed a centralized architecture with a Mixed Integer Linear Program (MILP) approach for multi-robot coordination in the search and rescue domain.

3.2.2. Decentralized Approaches

Parker (1998) has presented one of the earlier works for instantaneous multi-robot task assignment with a behavior based framework, ALLIANCE, and further extended it by integrating learning into the system in L-ALLIANCE.

Recent studies have revealed that the distributed approach, especially when complemented with the auction-based methods, has shown great promise for multi-robot task allocation during the last decade because of its scalability. M+ (Botelho and Alami, 1999) is one of the most successful architectures for distributed task allocation and achievement, addressing many real-time issues, including plan merging paradigms. MURDOCH (Gerkey and Mataric, 2002) is a framework that achieves publisher/subscriber type allocation for instantaneous assignment. Dias (2004) proposes a combinatorial auction-based task allocation scheme: TraderBots. Zlot and Stentz's work (Zlot and Stentz, 2006) on task tree auctions is presented as an extension to the Traderbots approach to address more complex tasks which can be decomposed into task trees. Lemaire et al. (2004) propose a task allocation scheme for multi-UAV (Unmanned Aerial Vehicles) cooperation with balanced workloads of robots. Recently Gancet et al. (2005) have proposed a framework for multi-UAV coordination. The application problem requires consideration of temporal constraints, uncertainties of task execution, reactivity to contingencies and changing priorities during runtime by operators. Their approach supports both centralized and distributed coordination and uses synchronization signals to coordinate synchronization among UAVs. We review these architectures in the following corresponding subsections.

3.3. Task Representation for Coordination

Tasks need to be efficiently represented before they are allocated among the robots. This may require that the overall mission be decomposed into several tasks.

In earlier systems, usually ad-hoc representations are used to represent the mission structure. Some models for task representations for robots such as Task Description Language (TDL) (Simmons and Apfelbaum, 1998) exist in the literature; however, to our knowledge, a TDL representation supporting multi-robot coordination has not been released or published yet. Goldberg et al. (2002) use TDL in the executive layer of their layered market-based coordination architecture. TAEMS framework ensures ways to define interrelations among tasks in a detailed expression syntax (Decker, 1996) for multi-agent coordination. Zlot and Stentz (2006) use AND/OR trees (Nilsson, 1986) to represent alternative and entailed solutions for task execution. In Figure 3.1 a sample task tree representation can be seen for the reconnaissance mission. In this mission description, the area should be covered by dividing the region into different parts (AND connections) in different ways (OR connections).

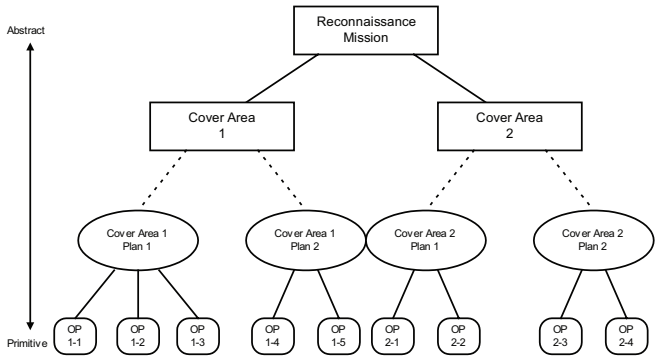


Figure 3.1: A sample mission representation as an AND/OR task tree (Zlot and Stentz, 2006)

The task dependency issue, on the other hand, presents orderings among tasks and has not been given much attention in earlier multi-robot systems. The interdependencies make the NP-Hard multi-robot coordination problem even harder. The interrelations among tasks require effective task representations. As Malone and Crowston (1994) state; if there is no interdependence, there is nothing to coordinate. Lemaire et al. (2004) represent temporal interdependencies as task trees and the task execution is synchronized among robots by temporary master-slave relationships.

Graph representations may be more suitable for representing tasks with precedence relations as in activity on node graphs. This representation enables the usage of graph algorithms on these graphs. In multi-agent research, each agent's abstract plan is given as a separate graph and interrelations are defined on this graph. A sample multi-agent plan (Cox et al., 2005) for the Multiagent Plan Coordination Problem (MPCP) is given with interrelations between tasks (represented as boxes), preconditions and post-conditions in Figure 3.2. The preconditions are labeled on the arcs, whereas the postconditions are stated near the task boxes.

3.4. Task Allocation and Coordination Type Based on the Mission Structure

We classify task coordination for multi-robot systems into four subclasses, namely, tightly coupled tasks, loosely coupled tasks, interrelated tasks and tasks as parts of a combinatorial structured mission.

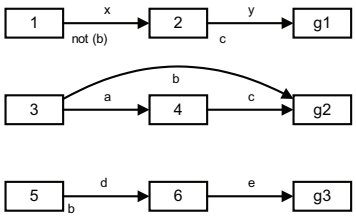


Figure 3.2: A sample multi-agent plan representation as task graphs (Cox et al., 2005)

3.4.1. Tight/Loose Coordination

In tightly coupled tasks, the actions implemented by each robot are highly dependent on the actions of others. Therefore, tightly coupled tasks are represented as non-decomposable atomic units. These tasks require cooperative work of all the participants, resolving constraints among each other. Interactions may have unexpected outcomes affecting each other. A sample application of tightly coupled task execution is presented in Kalra et al. (2005), in which robots try to sweep a perimeter while coordinating their movements. Task dependencies are considered in keeping a formation while simultaneously obeying some rules.

In loosely coupled task execution, the actions performed for individual tasks do not have effects on other tasks. Separate individual tasks can be assigned to different robots in the team. MURDOCH framework (Gerkey and Mataric, 2002) is evaluated on both a simple task domain with loosely coupled tasks and a box pushing domain as a tightly coupled task domain. In the latter domain, the monitoring for task execution is handled and the directives are given by a watcher robot.

3.4.2. Allocation of Tasks with Dependencies

Tasks with precedence/resource constraints are represented in task graphs in which nodes represent the individual tasks, whereas the arcs represent the interrelations. These interrelations may correspond to shared resources, producer/consumer, simultaneity and task-subtask dependencies (Ossowski, 1999). The Pick-Up/Delivery domain tasks can be classified in this class because of the producer/consumer type of dependency relation for the pick-up tasks and the delivery tasks. More complicated interrelations may be involved in the mission representations.

Task dependency has been analyzed in some earlier multi-robot cooperation schemes. The work by Alami et al. (1998) on multi-robot coordination presents a generic scheme based on a distributed plan-merging process. Although optimality is not guaranteed, Plan Merging Operation (PMO) provides a coordinated plan in their approach. A deadlock resolution is implemented in a distributed manner. The M+ scheme (Botelho and Alami, 1999) combines local planning and negotiation for task allocation, and cooperative reaction for contingencies. In their framework, a mission is a set of partially ordered tasks. Each robot has its own local world knowledge. Tasks are allocated through negotiation processes. Alami and Botelho (2001) introduce the mechanism concept in the framework M+CTA as an improvement to the M+ scheme. Mechanism is used for the resources in multi-robot cooperation. Each robot has an individual plan and tasks are initially decomposed and then allocated. After this planning step, robots negotiate with each other in order to incrementally adapt their plans in a multi-robot context.

3.4.3. Combinatorial Effects on Task Assignment

Tasks with combinatorial structures take part in the task classes with soft interrelations (i.e., not as hard as in tightly coupled tasks) in our task classification. In this class, although there is no interrelation among tasks in the mission definition, because of the combinatorial structure of the problem, the solution quality highly depends on which task is performed first and by which robot. Multi-robot exploration as a Multiple-TSP problem, the commonly studied application domain, belongs to this class of tasks. Separately Lagoudakis et al. (2004), Dias and Stentz (2002) and Lemaire et al. (2004) have studied this problem.

3.5. Instantaneous Task Assignment/Scheduling

Multi-robot task allocation is better viewed as a scheduling problem when there are interrelations and dependencies among tasks. When the problem solving time is limited, Branch and Bound or MILP methods may not be convenient. In this case, heuristic methods are preferred to find a good solution in reasonable time. Furthermore, finding a solution with these approaches in a decentralized setting may need considerably greater efforts in both computation and communication.

Given these limitations, instantaneous task assignment becomes profitable as it provides a dynamic solution allocating tasks to robots whenever resources are available (Parker, 1998; Gerkey and Mataric, 2002). However, in this case,

the global solution quality may be degraded if the decisions are made by just using the up-to-date knowledge available, ignoring the global solution quality. Paquet (2006) models the multi-agent task assignment problem as a scheduling problem for the RoboCupRescue simulation domain. The main objective is the maximization of the number of rescued civilians in a simulated disaster environment. They compare both centralized and distributed scheduling in their work. They conclude that the distributed scheduling performance is as good as that of the centralized scheduling and the decentralized scheduling approach is more robust. In the distributed approach, each agent locally chooses its best task to accomplish using a scheduling algorithm. Then, the best local task information is exchanged among agents to find the global best task to perform. The Earliest Due Date (EDD) algorithm is used to schedule the tasks both locally and globally. If there is no overload, this algorithm is optimal for the given objective of maximization of the number of rescued civilians. The EDD algorithm is a useful tool to schedule tasks that have no interdependencies and to make instantaneous assignments. In the EDD algorithm, future considerations and interrelations are not taken into consideration.

3.6. Reallocation and Dynamic Task Switching

Depending on the application domain and the frequency of the change in the world correspondingly, task reallocation is needed to adapt to changing situations. In dynamic games, such as RoboCup soccer domains, this frequency is high. However, this frequency may be much lower when robots run in fully structured environments with known maps and few unexpected events (e.g., robots working in a fully structured and isolated factory environment.)

Lagoudakis et al. (2004) propose a task reallocation method to be applied whenever the world knowledge of the robots changes in the multi-target exploration domain. On the other hand, dynamic scheduling and task selection (Paquet, 2006) prevent rescheduling all the tasks that have been previously scheduled. As stated in Paquet (2006), rescheduling each time that the world knowledge changes could make agents switch between tasks frequently. In their work, task preemption is not allowed to circumvent this situation.

In ALLIANCE (Parker, 1998), responding to unexpected events and dynamic task reallocation are provided through the use of motivations: robot impatience and robot acquiescence. The impatience motivation enables a robot to handle

situations when other robots fail in performing the given tasks. The acquiescence motivation enables a robot to handle situations in which it fails to properly perform its task. In Traderbots (Dias et al., 2004), task reallocation is achieved through continuous auctioning by one of the robots. In MURDOCH (Gerkey and Mataric, 2002), task reallocation is provided through the observations and directives of the leader robot for tight coordination, and a publisher/subscriber type of instantaneous task allocation approach for loosely coupled tasks. In M+ (Botelho and Alami, 1999), contingencies in task execution are handled by re-planning for execution of the goals at hand by each robot. Chaimowicz et al. (2002) address the task dependence and role exchange issues in their work. Utility calculation is implemented to perform role exchanging. Task announcement is used to call for additional help. In the work by Lemaire et al. (2004), allocations are implemented whenever the world knowledge of robots changes (there may be new online tasks or robots may fail to achieve their plans).

3.7. Bounds on the Solution Quality

The exact bounds of the solution quality is hard to evaluate for robot systems due to uncertainties. Therefore, in the last decade researchers have proposed effective approaches and opportunistic methods without giving boundaries on the overall solution quality. The one exception is Lagoudakis et al. (2004); however, their work assumes perfect communication and contingencies are not considered in these boundaries.

3.8. Organizational Requirements on Multi-Robot Task Execution and Coordination

Tasks can be classified according to single robot/multiple robot requirements for task execution, as in the taxonomy given in Gerkey and Mataric (2004). Furthermore, robots in multi-robot task execution may be part of either a homogeneous or a heterogeneous group. The heterogeneity may be in the possessing capabilities or in the task execution performance. For example, the robots may have the same equipment capable of achieving all the tasks of the mission but may differ in abilities such as speed. Dahl et al. (2004) address this issue in their task allocation method through vacancy chains. This method ensures differentiation between robots based on their individual performances not related to their physical and sensor capabilities. High-value tasks are assigned to the high-performance robots.

Horling and Lesser (2005) classify the multi-agent organizations depending on the objective. According to their classification, coalition as an organizational paradigm is used in systems where the agents form coalitions (agent groups) to perform a task in cooperation, and the coalition dissolves when the corresponding task no longer needs to be executed by them. From our perspective, coalitions are suitable to achieve the tasks which require a subteam to be formed during a time period.

3.8.1. Coalition Formation

A coalition is an alliance between entities, during which they cooperate in joint action, each in their own self-interest. This alliance may be temporary or a matter of convenience. A coalition thus differs from a more formal covenant (Wikipedia). The selection of the members of a coalition at one step has a great effect on future coalition formation.

Shehory and Kraus (1998) present one of the earlier algorithms for coalition formation for cooperative multi-agent systems. During coalition value calculations, agents' capabilities are taken into consideration. In multi-robot systems, the cost values are a function of not only capabilities but also the physical conditions which change during execution, such as robot's location, object/subject's location, etc. When the robots decide to perform a task, both subjects in the environment and robots' physical entities (e.g., fuel) change.

Vig and Adams (2005) state the differences of multi-robot and multi-agent coalition formation issues from a sensor possessive point of view. Locational sensor capabilities are considered in their work. They propose an approach based on the coalition evaluation step in Shehory and Kraus' algorithm (Shehory and Kraus, 1998). Vig and Adams' approach assumes that capabilities are known apriori before coalitions are formed. This approach may be applicable in the beginning of mission execution. However, another important factor in multi-robot systems for evaluating coalitions is the changing cost values during runtime. They assume robot capabilities do not change (which also is not a realistic assumption); however, this is not the case for the costs. They analyze the trade-off between distributing the coalition value evaluation among robots and implementing coalition value evaluation for each robot. This approach may result in a robust system where all robots are informed about the robot failures. Coalition imbalance issue is addressed to make the coalitions more robust by distributing the required resources. This issue is debatable based on the objective function which is selected.

ASyMTRe (Parker and Tang, 2006), uses reconfigurable schema abstraction for collaborative task execution by providing sensor sharing among robots. The fundamental building blocks of ASyMTRe are collections of environmental sensors (ES), perceptual schemas (PS), motor schemas (MS), and also communication schemas (CS). In ASyMTRe, connections among the schemas are dynamically formed at runtime. The information labels provide a method for automating the interconnections of schemas, enabling robots to share sensory and perceptual information as needed. In their approach, initially the schema set is reduced to contain only the individual separate schemas. Then potential solutions are found. Finally the corresponding schemas are instantiated on robots. This approach is used to form low-level coalitions to share robot capabilities.

3.9. Communication and Coordination Tools

Being a part of the real environment, robots trying to achieve common goals need to interact with others to perform the tasks. Interaction may take place in two forms:

- by a communication language (direct interaction)
- by observing others (indirect interaction)

Communication is one of the most important coordination tools for robots when they have joint goals or interacting actions. Some tasks may be achieved without communication (Balch and Arkin, 1994) but mostly intentional cooperation/coordination requires some level of communication. Research on observing other robots is in its early stages with current technology.
Communication protocols are specified at several levels:

- the bottom layer specifies the method of interconnection,
- the middle layer specifies the format, or syntax of the information being transferred,
- the top layer specifies the meaning, or semantics of the information.

There are two main types of interaction to allocate tasks and reaching a consensus on task execution: Negotiations and Auctions.

3.9.1. Negotiations

Negotiation is a process by which a joint decision is reached by two or more agents, each trying to reach an individual goal or objective (Huhns and Stephens, 1999,

Muller, 1996). At this point, it differs from cooperative multi-robot objectives. However, even when the robots work cooperatively, this mechanism can be used to reach an argument for the global goal from local perspectives.

3.9.2. Auctions

Auctions are inevitable parts of the Market-Based approach which is based on economy theory. An auction consists of an auctioneer and potential bidders. Auctioneers want to sell items and get the highest possible price, whereas bidders want to acquire the item with the lowest possible price (Sandholm, 1999).

The objects of the auctions can either be single items or a bundle of items. Both single item (Lagoudakis et al., 2004) and multiple item allocation for multi-robot task allocation (Berhault et al., 2003; Dias, 2004) are studied in the literature. As a special case of the combinatorial auction approach, the task tree auctions presented in Zlot and Stentz (2006) propose a way to offer task allocation from any point in the task tree with the payoff of the expensive bid clearing algorithms. In the combinatorial auction methods, even when effective methods are used to define bundles of items, communication requirements and efforts for negotiations and bidding grow exponentially with the task size.

3.9.3. Contract-Net-Protocol

Even though the issue of task allocation is analyzed in a wide variety of works, there still are not formalisms to decide which one to implement in a particular domain. Both single item and combinatorial auctions are proposed for multi-robot systems. However, there is not a perfect answer for the suitability of these methods for different domains. Since in many domains the environment is highly dynamic, the solution to task allocation problem in the initial world state may be obsolete or suboptimal in later steps. This is due to the environmental changes after actions of the robots, changes in the robots' capabilities, changes in the team mission, or changes in the team capabilities or composition. Auction-based methods and Contract Net Protocol (CNP) (Randall and Smith, 1983; Smith, 1980) seem to be an efficient way to allocate tasks in a distributed manner and these methods are applied to both software agents and real robots.

CNP is modeled on the contracting mechanism used by businesses to govern the exchange of goods and services. The contract net provides a solution for finding the most appropriate agent to work on a given task. In the contract net protocol,

an agent that has a task to be solved is called the manager. Agents that might be able to solve the task are called potential contractors. The manager decomposes its larger problem into a set of subproblems and announces each subproblem to the network, along with the specifications about which other agents are eligible to accept the subproblem, when the deadline is reached, and how they should specify a bid for the problem. A recipient of the announcement decides whether it is eligible, and if so it formulates a bid. The manager collects bids, and awards the subproblem to the contractors with the best bid. A contractor receives the subproblem details, solves the subproblem, and returns the solution to the manager. The contract net protocol procedural steps can be seen in Figure 3.3. There is a mutual selection between the manager and the contractors. Any agent

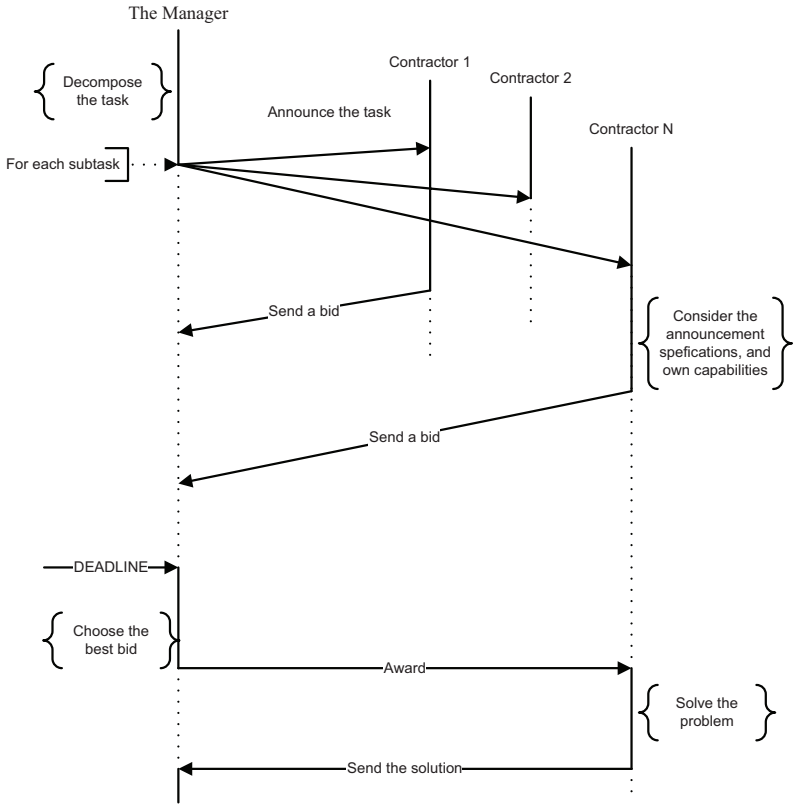


Figure 3.3: Contract Net Protocol

can act as a manager by making task announcements, and any agent can act as a contractor by responding to the task announcements. The task announcement can be made by broadcasting to the network or by directed contracting by prior experience. One drawback of the contract net protocol is that a task might be awarded to a contractor with limited capability if a better qualified contractor is busy at the award time. There are also some situations when the manager does not get any bid from contractors. The contractors might be busy with other

tasks. They might consider other announcements and prefer them by ranking, or they might not be capable of working on the task in consideration. If no contractors are found, the manager may request a response from contractors indicating their situation such as: eligible but busy, ineligible, or uninterested. The manager can retry the announcement periodically. The manager can also relax the eligibility requirements, but there may be no contractors having capability for the announced subproblems. In this case, the manager tries to decompose the problem differently. Another solution is that the contractors can announce availability. It is also possible to alternate between task and availability announcements.

Although Contract Net Protocol presents the formalism on the relationships between managers and contractors for task allocation, it does not present details for the following questions:

- What is the communicated information in auctions for multi-robot systems?
- When should task announcement be made? Who should announce the task?
- How should bid values be defined to get globally good solutions for different domains?
- Which subset (or all) of the already allocated tasks should be re-allocated?
- When should reallocations be announced?

These questions are usually left open in most multi-robot systems.

3.10. Failure Detection and Recovery

Robustness is an important issue for multi-entity systems (Kaminka and Tambe, 2000). The dynamic task allocation problem has been investigated in the face of robot failures (including partial/complete failures) or environmental changes. In most cases which include failure detection, the test domain missions include independent subtasks that can be executed by a single robot.

ALLIANCE (Parker, 1998), provide a mechanism to handle robot failures through using the motivational behaviors. Dias et al. (2004) investigate the performance of their framework, Traderbots, against different kinds of failures such as communication failures, partial malfunction or death. In their work, execution conflicts between robots are resolved by continuous auctioning by one of the robots. Since the robots sell different portions of their tasks during execution,

robot failures are handled by complicated queries on earlier communications made by the failed robot and communicating with other subcontractors for the tasks of the failed robot. Gerkey and Mataric (2002) evaluated the MURDOCH against the robot failures for tightly coordinated task execution in which there is a leader giving appropriate directives for the changing situation of the system after failures. In M+ (Botelho and Alami, 1999), contingencies in task execution (task failures) are handled by re-planning for execution of the goals at hand by a robot. The watch-out task introduced in the work by Lemaire et al. (2004) has an interesting property providing cooperative work against communication failures.

3.11. Interleaving Planning and Execution

According to desJardins et al. (1999), there are three ways to accommodate planning and execution into one framework:

1. Conditional planning: For each contingency, an alternative course of actions is provided.

2. Plan monitoring and repair: The plan-and-execute cycle is repeated sequentially whenever the execution does not match the estimated model.

3. Interleaving planning and execution together: This method corresponds to continual planning.

As desJardins et al. (1999) state, an agent should plan continually:

- when aspects of the world can change beyond the control of the agent,
- when aspects of the world are revealed incrementally,
- when time pressures require execution to begin before a complete plan can be generated, and
- when goal objectives change.

desJardins et al. (1999) come up with an important corollary that states it is better to delay plan refinement as long as possible, so that detailed decisions are made with as much information as possible. They argue that simply combining distributed and continual planning methods independently may not be sufficient and more intelligent integration approaches are needed.

3.12. Application Domains

RoboCup domain (Kitano, 2000) is a testbed to develop and improve robot architectures and algorithms by motivating through competitions held annually. Vail and Veloso (2003) present a dynamic assignment based coordination approach for RoboCup soccer robots. Their method is based on shared potential functions related to the positions of the relevant obstacles. The bidding mechanism is implemented for distributed coordination. Kose et al. (2005) present a market-driven coordination approach for RoboCup domain. They present different bid functions to assign roles and further extend the system by integrating reinforcement-learning to learn the role assignment process. Paquet (2006) models the multi-agent task assignment problem as a scheduling problem for the RoboCupRescue simulation domain. The main objective is the maximization of the number of saved civilians in a simulated disaster environment.

As a humanitarian domain the real Search and Rescue (SR) domain is one of the attractive application domains for deploying multiple robots. Jennings et al. (1997) propose an algorithm for a distributed team of autonomous mobile search and rescue robots. In their experimental setup with two robots, the tasks are explicitly defined and communication is assumed to be perfect. The experimental robots are homogeneous and they both aim to manipulate an object with no constraints on the task requirements. Recently, Koes et al. (2005) propose a centralized architecture with a Mixed Integer Linear Program (MILP) approach to multi-robot coordination for the search and rescue domain. The evaluations are performed in simulations.

In the cooperative transportation domain, a group of robots locate and cooperatively transport several objects scattered in the environment. Chaimowicz et al. (2002) address the task dependence and role exchange issues in this domain. Dahl et al. (2004) present the results of their approach based on vacancy chains in this domain. The scheme by Alami et al. (1998) is tested on the multiple transportation of containers in different environments.

Another attractive domain for researchers is surveillance, monitoring and reconnaissance domains. These domains could be modeled by either the multi-robot coverage (Hazon and Kaminka, 2005, Rekleitis et al., 2004) or the multi-target exploration problem formulation. In multi-target exploration, the robots visit targets (special observation points or object locations). Goldberg

et al. (2003) present evaluations of their market-based architecture for the MARS exploration application domain. Dias and Stentz (2002) evaluate the performance of their framework, Traderbots on the multi-robot exploration problem in which robots are homogeneous and tasks have the same type of capability requirements. Zlot and Stentz (2006) evaluate the market-based complex task allocation approach work on the area reconnaissance problem. The system proposed by Lemaire et al. (2004) aims to be used in surveillance and monitoring, specifically forest fire monitoring applications. There are other military applications, such as perimeter sweeping (Kalra et al., 2005) in which robots try to sweep an area while coordinating their movements to form security barriers.

The Multiple-TSP problem, the commonly studied application domain, usually is ipart of one of the domains presented above. This domain deserves investigation due to its easy applicability into different applications such as SR and space exploration domains. Lagoudakis et al. (2004), Dias and Stentz (2002) and Lemaire et al. (2004) have studied this problem.

3.13. Multi-Robots vs. Multi-Agents

How multi-robot research differs from multi-agent research is questioned by both robot and software agent researchers. The main difference is in the environment and the body of the entities; the robots live in the real physical world, whereas the software agents live in electronic environments. Therefore, the assumptions made in multi-agent systems make the real robot systems far from reality. While the agents can travel easily over communication channels, the robots need to avoid real obstacles while simultaneously localizing themselves and mapping their environment.

Usually the agents in the multi-agent systems are modeled as self-interested. The agents have their own abstract plans and should coordinate their activities/actions in a compatible and mutually supporting manner. The suitability of self-interest in multi-robot systems is a part of ongoing debates.

The main difficulty with robot systems is the requirement for simultaneous computation and physical actions. The capabilities of the agents are usually assumed to be shared by other agents. However, considerable effort is needed to achieve this in robot systems. Simultaneity and synchronization between robots is much harder with limited sensors and communication capabilities.

3.14. Summary and Discussion

Different frameworks proposed in the literature touch separate aspects of the research questions, presented in different subsections of this chapter. Current research in multi-robot coordination addresses the issue of dynamic task allocation in the presence of robot failures and environmental changes. Earlier experiments are usually on either simulated or real robot teams implementing simple/independent tasks. Complex task domains requiring heterogeneous robot teams working on interrelated tasks facing with time constraints, sensing uncertainties, and nondeterministic actions have not yet been fully investigated with a complete framework.

Although it is questioned if a general architecture can be found spanning all types of domains for multi-robot systems, in this research our objective is to investigate and find an acceptable answer to this question by offering a distributed multi-robot coordination scheme at least spanning different multi-robot mission domains. This framework should serve to achieve a global goal with its integrated scheduling and execution capabilities while handling contingencies and using resources effectively.

We have seen that market-based approach ensures a scalable way of solving the multi-robot task allocation problem. However, following the remarks made by Dias et al. (2005), existing market mechanisms are not fully capable of re-planning task distributions, changing decomposition of tasks, rescheduling commitments or re-planning coordination during execution. From the dynamic events dimension, there is not a formalized study of response speed for any multi-robot coordination approach. Scalability in the market-based approaches may be limited by the computation and communication needs that arise from increasing auction frequency, bid complexity and planning demands.

On the other hand, Operation Research methods may not be directly applicable to the multi-robot coordination for complex tasks.

Therefore, we need to use the approaches from different disciplines to construct our multi-robot coordination framework as a consistent and efficient architecture formed from the marriage of Operations Research, Planning, Distributed Problem Solving and Autonomous Robots Research fields.

4. DEMiR-CF: DISTRIBUTED AND EFFICIENT MULTI-ROBOT - COOPERATION FRAMEWORK

This chapter presents our solution to the Cooperative Mission Achievement problem ($CMAP$) as a generalized framework and the integrated components of the framework. Since our objective is to span different types of domains, we present the details of the components in our framework by analyzing them for missions involving both independent and interrelated tasks (Sariel and Balch, 2005b, Sariel et al., 2006a, Sariel and Balch, 2006a).

4.1. Integrated Modules of DEMiR-CF

DEMiR-CF, **D**istributed and **E**fficient **M**ulti**-R**obot - **C**ooperation **F**ramework, is designed for complex missions including interrelated tasks that require diverse (heterogeneous) capabilities and simultaneous execution. The framework combines *The Dynamic Priority Based Task Selection Scheme*, *Distributed Task Allocation* and *Coalition Formation Schemes* as cooperation components and *Plan B Precaution Routines*, some of which are implemented by *The Coalition Maintenance/Dynamic Task Switching Scheme*. These components are integrated into a single framework to provide an overall system that finds near-optimal solutions for real-time task execution.

DEMiR-CF is classified as SIZE-LIM, COM-NEAR/COM-INF, TOP-ADD, BAND-MOTION, ARR-COMM/ARR-DYN, PROC-TME, CMP-HET according to the taxonomy of multi-robot systems given in Dudek et al. (1996). To clarify this notation, the abbreviations are explained as follows. The number of robots in the system is classified as SIZE-LIM, since the system is not designed as an emergent swarm robot collective. Communication range can be limited (COM-NEAR) or robots may have unlimited communication range in the operation environment (COM-INF). Communication topology is classified as TOP-ADD since both broadcast and peer-to-peer communication is possible. Communication bandwidth is not assumed to be costless; therefore, due to this limitation, DEMiR-CF is classified as BAND-MOTION. Reconfigurability of DEMiR-CF is classified as both ARR-DYN, that is, the relationship of members of the system can change arbitrarily, and ARR-COM, where coordinated rearrangement is needed for multi-robot requirements for single task execution.

The processing ability of each robot is classified as PROC-TME, Turing machine equivalent. Collective composition is heterogeneous (CMP-HET) where both a heterogeneous and homogeneous team is addressed.

The overall objective of the robot team ($r_j \in R$, $0 < j \leq ||R||$) in our framework is to achieve a mission (M) consisting of independent or interrelated tasks T_i ($0 < i \leq ||M||$), by incremental assignment of all $T_i \in M$ to $r_j \in R$ while optimizing the specified objective function. Tasks are preemptive: The activity of task execution can be split during runtime if another advantageous situation arises or environmental conditions impel.

Coalitions ($Coal_i$) (Horling and Lesser, 2005) are formed to meet simultaneous execution requirements of tasks (T_i) synchronously by a group of robots. An example of such a task that needs to be executed by a coalition of robots is pushing a heavy object requiring more than one robot. Sizes of coalitions vary according to the minimum number of robots required ($reqno_i$) to execute the tasks. A coalition [1] may involve only one robot for a task that can be executed by a single robot.

The robots can detect and recover from different types of contingencies by keeping the models of the system tasks and other robots as corresponding Finite State Machines (FSM) in their world knowledge. Details of these FSMs are presented in Section 4.8. *The Model Update Module* is responsible for checking and updating a robot's own models. The modules that embody the framework and the information which flows among them are given in Figure 4.1. Each robot keeps a model of the other robots and the mission tasks. *The Model Update*, *The (System) Consistency Checking Module*, and *The Dynamic Task Selector Module* perform *Plan B Precaution Routines* by either updating the model maintained by the robot or activating the warning mechanisms. Model updates are initiated by either incoming information from the other robots or information perceived by the robot itself. If a system inconsistency exists, *The Consistency Checking Module* is responsible to initiate warning mechanisms and inform the corresponding robots. *The Dynamic Task Selector Module* selects the most suitable task by considering the model of the robot. *The Allocation Scheme* ensures the distributed task allocation by executing the required negotiation procedures for the selected task. *The Execution/Coalition Scheme* implements synchronized task execution and *coalition maintenance* procedures. According to the selected task and the task

[1] The term coalition is also used for a single robot executing a task for the sake of generality.

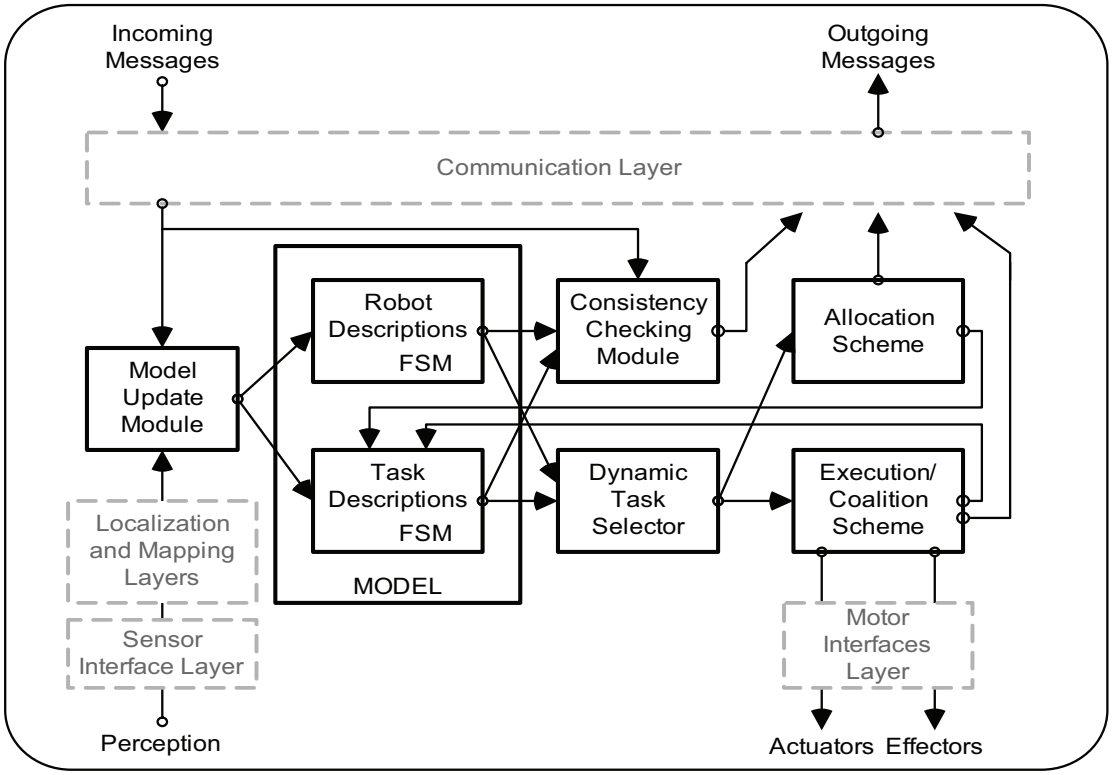


Figure 4.1: DEMiR-CF Modules

currently in execution, the task models are updated accordingly. A sample flow of the operations in the framework is summarized below:

1. Initially the robots are delivered the mission task definitions (time-extended representation of tasks with precedence constraints to achieve the overall mission).

2. Each robot selects the most suitable candidate task to execute through global cost consideration (dynamic task selection/switching).

3. Robots offer auctions for the tasks they have selected. During auction steps, inconsistencies are cleared and conflicts are resolved.

4. Coalitions are formed for the announced tasks, making sure that each robot takes part in the most suitable coalition when the global solution quality is considered.

5. Dynamic task selection/switching proceeds simultaneously with task execution. This allows the robot to switch between tasks when executing the candidate task becomes more profitable than continuing with the current task, handling real-time contingencies at the same time. Thus,

corresponding auction and coalition formation procedures (2-4) are applied continually.

Real-time situations in which task switching becomes necessary are given in Section 2.2.

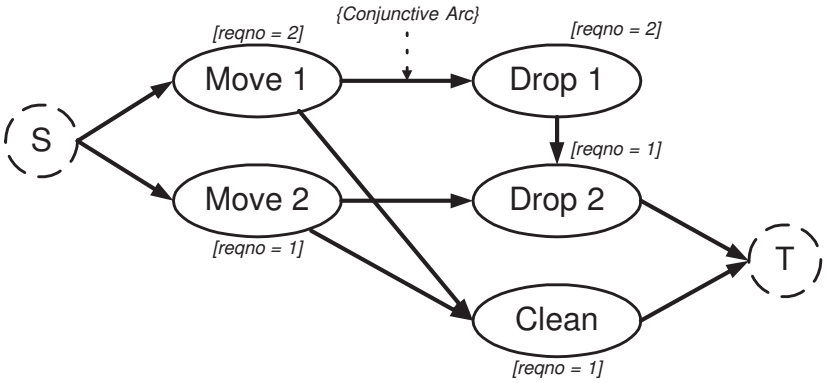


Figure 4.2: Directed acyclic mission graph for the Box Mailing Mission

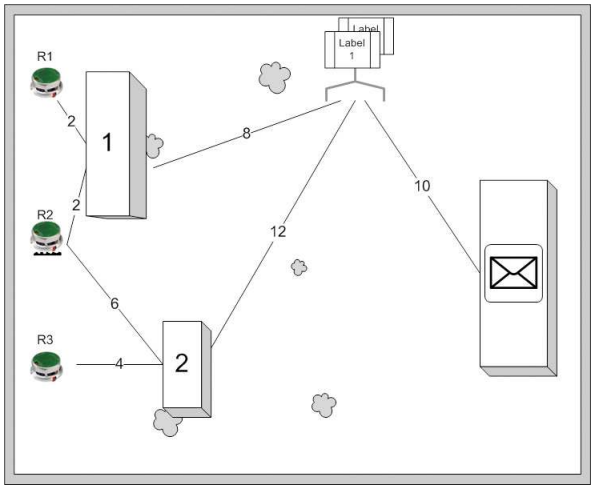


Figure 4.3: Box Mailing Mission initial state is illustrated. The robots are located on the left. The two boxes, the stamping machine and the mailbox are located in different places in the environment.

4.2. Mission Representation

A mission in DEMiR-CF is represented by a directed acyclic graph (DAG) where each node represents a task and the directed arcs (conjunctive arcs) represent the precedence constraints among tasks.

Example 1 *To clarify, Figure 4.2 depicts the graph representation of a small sized mission illustrated in Figure 4.3. The mission involves moving boxes to a stamping machine, dropping them in a given order, and then cleaning the room. The room can only be cleaned after both boxes are moved. Since box 1 is heavy, two robots are needed to move and drop the box (hence, $reqno = 2$). Unique dummy nodes are added to the graph to represent initial (S) and termination tasks (T). Therefore, even when the task graph is not connected, after adding these task nodes, it becomes connected. Although this graph shows the relationships between dependencies among tasks, it does not show which robot performs which task and in what sequence. The decision of which robot will be involved in the task execution of a particular task has an important effect on the performance. There may be alternative solutions to this problem according to the number of available robots, their capabilities and task requirements. A sample set of robot capabilities is given in Table 4.1.*

Table 4.1: Robot Team and Capabilities in the Box Mailing Mission Domain

Robot ID	Capability
R1	BUMPER, GRIPPER
R2	BUMPER, GRIPPER, BRUSH
R3	BUMPER, GRIPPER

In our representation, interrelations among tasks can be represented either by adjacency-lists or adjacency matrices where each node represents a task. Tasks are represented as septuples containing information regarding task execution requirements and task status: $<$ *id*, *type*, *reqcap*, *deplist*, *reqno*, $relinfo$, $precinfo>$.

1. *id*: A system-generated unique task identification number common to all robots before mission execution.

2. *type*: A description of task type and corresponding action definitions.

3. *reqcap*: Requirements defining special sensors and capabilities required to execute the task.

4. *deplist*: The two types of dependencies representing precedence relations. Hard dependency implies sequential execution of the related tasks while soft dependency allows parallel execution. Dependencies are represented by two letters H and S followed by the dependent task id.

Table 4.2: Representation of the tasks of the Box Mailing Mission

id	*type*	*reqcap*	*deplist*	*repno*	*relinfo*	*precinfo*
0	$MOVE$	$1, 2$	$-$	2	$<$ *locations of object* 1 *and the final destination* $>$	*available*
1	$MOVE$	$1, 2$	$S0$	1	$<$ *location of object* 2 *and the final destination* $>$	*available*
2	$DROP$	$1, 2$	$H0$	2	$<$ *location of object* 1 $>$	*available*
3	$DROP$	$1, 2$	$H1, H2$	1	$<$ *location of object* 2 $>$	*available*
4	$CLEAN$	3	$H0, H1$	1	$<$ *cleaned portion of the environment* $>$	*available*

5. *reqno*: The minimum number of robots required to execute the task, determined either before mission execution or during runtime.

6. $relinfo$: Descriptive information regarding task type, such as the latest location, the target location, etc.

7. $precinfo$: Precaution information used for contingency handling: the task state, the estimated task achievement time and the current execution cost.

Information in a task representation can dynamically be modified during execution. In particular, $relinfo$, $precinfo$ and *reqno* are subject to change during execution.

Task representation for the tasks of the sample mission described is given in Table 4.2. Capabilities required for task execution, such as possessing a bumper (to push an object), possessing a gripper (to hold and drop a box) and possessing a brush (to clean the environment) are encoded as 1, 2 and 3, respectively. Soft (S) and Hard (H) dependencies are identified with the corresponding task ids. $precinfo$ values of the tasks are initialized to state *available* before mission execution. These values are updated during runtime. These issues are explained in Section 4.8.

4.3. Inputs and Outputs of DEMiR-CF

DEMiR-CF uses the planning graph of a mission for interrelated tasks represented as explained. Although planning is assumed to be performed outside the framework, it is achieved by DEMiR-CF for independent tasks. For example, the multi-target exploration mission has independent tasks, but the selection of the targets to be visited by robots is a form of planning which is done by DEMiR-CF.

Another input to the system is the robot capabilities. The system may involve a heterogeneous team of robots both possessing different equipment (e.g., sensors,

different hardware tools, etc.) and/or having different capabilities with the same equipment (e.g., speed). The robots are informed about the other robots in the system initially. But this does not preclude that they discover other robots working on the same mission during runtime in peer-to-peer communications. The initial information is better to be fed into the robots' world knowledge, although DEMiR-CF can also handle distributed information update in runtime.

At the end of the mission execution, all achievable tasks of the mission are performed by robots working in a cooperative manner. In the meantime, DEMiR-CF also ensures efficiency with its integrated task allocation and contingency handling capabilities.

4.4. Dynamic Priority-based Task Selection Scheme

In DEMiR-CF, the robots make instantaneous decisions (from their local perspectives) which are both precedence and resource feasible in the context of the global time extended view of the problem. While the completion of the mission is the highest priority objective, performance related objectives can additionally be targeted. Each robot initially forms a rough schedule of its activities for an overall time extended resolution of the mission. Since these schedules are highly probable to change in dynamic environments and robots also have the real-time burdens of path planning, mapping etc., the formed rough schedules are tentative and constructed by computationally cheap methods (explained in Section 4.4.1.). Therefore, the robots in our framework come up with their rough schedules and refine their plans during actual fast execution when information available in the current context enables them to make specific, detailed decisions.

Task selection and allocation is performed by evaluating each task according to the selected cost function. Depending on the objective, different cost functions can be defined, from the simplest functions to more complex and composite evaluations. Cost evaluation is one of the key issues to make the framework suitable for different domains in an efficient and effective manner. Cost functions are analyzed in Section 4.5.

Since schedules are subject to change, we propose an approach in which tasks are not scheduled initially but instead allocated to robots incrementally, without ignoring the overall global solution quality. Therefore, the main objective

becomes determining a particular task to be assigned whenever it is convenient in a precedence and resource feasible manner, instead of scheduling all the tasks from scratch. Although not a concern during assignments, preemption (i.e., yielding) is possible to maintain the solution quality and to handle failures during execution. Therefore, the allocation problem turns into a selection problem and is stated as the $CTSP$, which is introduced in Section 2.

Note that the $CTSP$ presented earlier is an optimization problem as in ScP, and it is desirable to find a solution by considering the problem from the global perspective. Therefore, the instantaneous task selection scheme needs to be strengthened by considering the problem as a whole, with the designed cost evaluation functions. Depending on the objective function, either priorities or penalties can be applied to find a near-optimal solution ensuring a time-extended view of the problem. This issue is analyzed in detail in Section 4.5.

The following definitions are needed to present our formulation to solve the $CMAP$.

Definition (*suitable task and suitable robot*) t_i is a suitable task for robot r_j, if $reqcap_i \subseteq cap_j$ and r_j is a suitable robot for t_i.

Definition (*executable task*) t_i is an executable task, if at least $reqno_i$ number of robots can be assigned for its execution.

Definition (*task in execution*) t_{iej} is a task in execution by robot r_j or coalition C_j. T_{ie} is a union of tasks in execution.

Definition (*eligible task*) t_{Ej} is an eligible task, if it is an *executable task* and is neither in execution (t_{ie}) nor achieved. T_{Ej} is a union of *eligible tasks* for robot r_j.

Definition (*ineligible task*) t_ϕ is an ineligible task if it is not an *executable task*, if it is already achieved or if it is not a *suitable task*. T_ϕ is a union of *ineligible tasks*.

Definition (*predecessor task set*) $P(t_i)$ is defined as the set of all predecessor tasks of task t_i.

Definition (*active task*) t_{Aj} is an active task if it is *suitable*, *executable* and tasks in $P(t_{Aj})$ are completed. $T_{Aj}(\subseteq T_{Ej})$ is a union of the *active* tasks for robot r_j.

Definition (*inactive task set*) is the set of all all inactive tasks (t_{Ij}), $T_{Ij} = T_{Ej} \setminus T_{Aj}$ contains the tasks that are *suitable* but not *executable* yet for robot r_j.

Definition (*critical task*) A critical task t_{Cj} is a task that has inflexibility from the point of view of resources and robot r_j is suitable for that task. T_{Cj} is a union of *critical tasks* for robot r_j.

Definition (*rough schedule*) A rough schedule S_{Rj} for robot r_j is a priority queue of mission tasks that r_j assumes it will execute.

4.4.1. Rough Schedule Generation Scheme

Each robot r_j generates its rough schedule as a dynamic priority queue similar to runqueues, by considering its critical task set (T_{Cj}), the eligible tasks (T_{Ej}), the conjunctive arcs (if any) and the requirements. If there are no new online tasks or invalidations, the order of the tasks which are connected by the conjunctive arcs remains the same in the priority queue, even though there may be additional intermediate entries into the queue at runtime.

Since each robot r_j has different capabilities, the eligible task sets (T_{Ej}) and the priority queue entries will be different. Sometimes uncertain information (e.g., related to a local online task) or unexpected (internal or external) events (e.g., detection of a fuel leakage) may result in this difference, even when robots possess the same type of capabilities. The critical tasks may be determined either by negotiations or by beliefs. Critical task information is used for determining the task requirements such as power, fuel etc.

Intuitively, robots do not deal with the ineligible tasks (T_ϕ), the union of tasks that are already achieved or those that are not eligible from the capabilities perspective while forming the rough schedules. The eligible tasks ($T_{Ej} = T \setminus T_\phi$) for robot r_j consists of active and inactive tasks.

The rough schedule of a robot is generated by execution of Algorithm 1. $curcs_j$ represents the remaining capacity of robot r_j and $reqcs(t_i)$ represents the required capacity for task t_i in terms of the consumable resources (e.g., fuel). In the rough schedule generation algorithm, while the rough schedule is being formed, the remaining capacity of the robot is also monitored. If the capacity of the robot is

Algorithm 1 GenerateRoughSchedule for robot r_j

input: Eligible task set (T_{Ej}), active task set (T_{Aj}), critical task list (L_{Cj}), remaining capacity ($curcs_j$) of robot r_j
output: Rough schedule (S_{Rj}) of tasks, the top most suitable active task t_s

$t_s = \phi$; $R = curcs_j$; $achievable = true$;
S_{Rj} = GeneratePriorityList(T_{Ej}, T_{Aj})
/*Determines if the mission is achievable*/
for each $t_i \in L_{Cj}$ **do**
  $R = R - reqcs(t_i)$
  if $R < 0$ **then**
    $achievable = false$
    $R = curcs_j$
    break
  end if
end for
if $S_{Rj} \neq \phi$ and $(top(S_{Rj}) \in L_{Cj} \parallel R - reqcs(top(S_{Rj})) \geq 0)$ **then**
  $t_s = top(S_{Rj})$
end if

not sufficient for executing all of its critical tasks and the mission is believed to be unachievable accordingly, then the robot may select an active task to execute even if it is not a critical task for the robot in case new robots can be deployed. However, if the mission is believed to be achievable, the robot may select to stay idle until its critical tasks become active. This selection is done after forming the rough schedule. The active task on top of the rough schedule that can be executable is the most suitable task to be executed for the robot. Sometimes the rough schedule of the robot may be empty. In this case, the robot selects to stay idle as determined in the DPTSS algorithm. The priority values to form rough schedules are determined based on the mission and the objective function which will be explained in Section 4.5.

4.4.2. DPTSS Algorithm

In our incremental allocation approach, the fundamental decision that each robot must make is the selection of the most suitable task from the active task set (T_A) by considering eligible task set (T_E). Algorithm 2 presents the DPTSS in which a rough schedule is generated before making a decision. The four different decisions made by robots after performing DPTSS are:

- continue to execute the current task (if any),
- join a coalition,
- form a new coalition to perform an available task, or
- stay idle.

The dynamic task switching scheme is used by robots to dynamically switch between tasks if updates in the world knowledge compel. Therefore, issues related to both online scheduling and scheduling under uncertainty are addressed.

Algorithm 2 DPTSS Algorithm for robot r_j

input: Mission (M) task descriptions
output: Action to be performed depending on the selected task

Determine the T_{Ej}, $T_{Aj} \subseteq T_{Ej}$
L_{Cj} = GenerateListOfCriticalTasks(T_{Ej})
[S_{Rj}, t_s] = GenerateRoughSchedule (T_{Ej}, T_{Aj}, L_{Cj}, $curcs_j$)
if $t_s \neq \phi$ **then**
  if t_s is the current task **then**
    Continue with the current execution
  else
    Offer an auction to form a new coalition or directly begin execution
  end if
else
  if $t_{iek} \in T_{ie}$ and $R + reqcs(t_{iek}) \leq curcs_j$ and it is profitable to join a coalition **then**
    Join a coalition
  else
    Stay idle
  end if
end if

The DPTSS process is repeated whenever a robot completes its current task execution or detects a change in its world knowledge. Instead of regenerating the rough schedule at each call of the DPTSS, the rough schedule may be repaired whenever it is desirable.

4.5. Cost Evaluation for Rough Schedule Generation and Dynamic Task Selection

The impact of cost evaluation on solution quality is inevitable for systems that need some forms of optimization procedures, and research in this area requires more investigation. Unless efficient cost evaluation strategies are designed, it is not possible to observe globally near-optimal solutions for NP-Hard problems, and additional adjustments are required to change allocations with an additional cost of communication as in combinatorial auctions.

According to the taxonomy given in (Gerkey and Mataric, 2004), multi-robot task allocation problems are divided into two classes based on the mission description: instantaneous vs. time-extended. Most multi-robot architectures

offer solutions for instantaneous assignments. DEMiR-CF can address both types of classes by implementing incremental allocation of tasks with efficient bidding strategies. Therefore, global solution quality is achieved from a time-extended view of the problem by means of bid considerations. However, the approach is also capable of offering solutions for instantaneous changes on task descriptions. Therefore, our framework is classified as capable of addressing both classes.

Incremental assignments eliminate redundant considerations for environments in which a current best solution is highly probable to change, and efficient and intelligent bidding strategies ensure solutions to be close to optimal with a time-extended view of the problem in a computationally tractable way.

It is shown that by an efficient bid evaluation approach, globally near optimal solutions can be observed in an auction-based approach. This is validated in the following chapters by evaluating the performance of the framework in different domains.

4.5.1. Cost Function Design Criteria

Depending on the selected application domain and a regular objective function, cost functions should be designed appropriately. Different types of rough schedule generation schemes can be performed by using the designed cost functions. At this point, it is necessary to distinguish the rough schedule generation schemes for independent tasks and interrelated tasks with workforce constraints.

4.5.1.1. Independent Tasks with Combinatorial Structures

There are several Operation Research methods to allocate independent tasks to robots.

Integer Programming Formulation. Optimal results can be obtained by an efficient Integer Programming (IP) formulation in IP solvers (e.g., the commercial IP solver CPLEX (ILOG-CPLEX-9.0-UserMan)).

Branch and Bound Algorithms. In these algorithms, a search tree is enumerated (branching) by constructing the smaller subtrees (subproblems) within a feasible search space and searched through investigating lower and upper bounds of each subtree (bounding). The procedure continues until all nodes are either pruned or solved. The performance of the approach is dependent on the branching and bounding algorithms (Toth and Vigo, 2001).

Heuristics. There are several heuristic approaches to find approximate solutions to the optimal solution. The methods are classified into two subclasses: Classical Heuristics and Meta-heuristics. In the classical heuristic approach, standard construction and improvement methods are applied to the solution. These approaches perform limited exploration of the search space and typically produce good results within modest computing times. In the meta-heuristics approach, the algorithm searches a large solution space. Evolutionary algorithms, Tabu Search and Simulated Annealing methods are the common methods belonging to this class. Although the quality of the solution is much higher than that of the classical approaches, time complexity increases dramatically in these approaches. The applied procedures are usually context dependent and require finely tuned parameters (Toth and Vigo, 2001).

All these cost evaluation procedures can be used in DEMiR-CF. Each robot may simply perform the explained operations to generate the rough schedules and select a task to perform. However, as experiments in the following chapters illustrate, classical heuristic cost evaluations, which are more applicable to robots, produce highly acceptable and efficient results.

4.5.1.2. Interrelated Tasks with Multi-Robot Requirements

The interrelations among tasks and resource requirements are represented as directed acyclic graphs in each robot's world knowledge. The generated rough schedules respect the precedence and resource constraints. For the makespan objective in general, branch and bound methods can be applied. However, since there are interrelations and resource dependencies, reaching a consensus by communication may sometimes be intractable.

There are two efficient heuristic methods to generate feasible schedules: Forward-Backward Schedule Generation Schemes (Pinedo, 2005) for the RCPSP and another method that we propose which generates a topological sort of the directed acyclic graph fed by different priority rules. There are various priority rules that can be applied (Brucker and Knust, 2006) for evaluating cost values to select tasks. These are classified into four types: activity-based rules, network-based rules, critical path-based rules and resource-based rules. Different priority rules are listed as follows: activity-based rules by selection of tasks with the smallest processing time (SPT) or the longest processing time (LPT); network-based rules by selection of tasks with the most immediate successors (MIS), the least immediate successors (LIS), the most total successors (MTS),

the least total successors (LTS) or the greatest rank positional weight (GRPW) (the largest processing time of all successors); critical path-based rules with the smallest earliest starting time (EST), the smallest earliest completion time (ECT), the smallest latest starting time (LST), the smallest latest completion time (LCT) or minimum slack (MSLK); resource-based rules with the greatest resource requirements (GRR).

4.6. Task Allocation

DEMiR-CF uses the standard auction procedures of CNP (Smith, 1980) to announce the *intentions* of robots on task execution and to select the *reqno* number of robots for a coalition in a cost-profitable, scalable and tractable way. Additionally, *Plan B Precaution Routines* are added to check validity, consistency and coherence in these negotiation steps. Each robot intending to execute a task announces an auction after determining its rough schedule and performing the DPTSS.

4.6.1. Distributed Task Allocation Scheme

Basically, auction announcements are ways to illustrate intentions to execute tasks for which $reqno = 1$ or to select members of coalitions to execute tasks for which $reqno > 1$. Therefore, if more than one robot declares intentions to execute the same task, the more suitable one(s) is selected in the auction by considering the cost values. Auction negotiations and selection of the suitable robots are performed in a completely distributed manner by the auctioneers. Single task items are auctioned and allocated in auctions. Auction negotiation implemented in the framework consists of standard steps to clear an auction. Robots can get the necessary task details from the auction offers, and then check the validity of the auction. If the auction is invalid, related precaution routines (explained in Section 4.8.) are activated. Otherwise, the candidate robots send their cost values as bids. (The other candidate robots may behave as auctioneers as well. If the auctioneer does not receive the required number ($reqno$) of bids (also counting in its own bid) from the other robots by the predefined deadline, it cancels the auction. Otherwise, it ranks all bids and assigns the best suitable robot with the lowest cost value to the executable task (if $reqno = 1$), or suitable coalition members (if $reqno > 1$). The framework allows multiple auctions to be carried out simultaneously. The basic steps of an auction negotiation process are illustrated in Figure 4.4. Validity controls are performed to ensure system consistency as a part of the *Plan B*

Precautions on top of the standard CNP protocol procedures. If the auction is for an already achieved task, it becomes invalid and if this situation is detected by any of the candidates, a warning message is sent to the auctioneer (Figure 4.5).

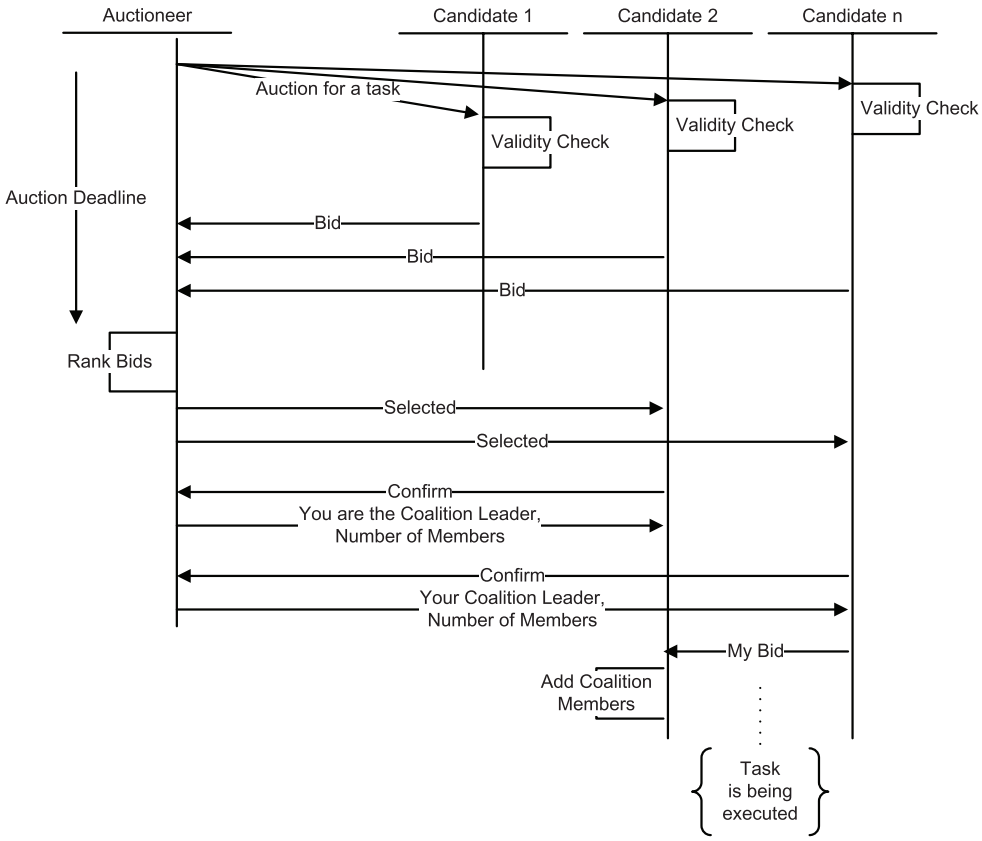


Figure 4.4: Basic steps of an auction negotiation process

4.6.2. Communication in DEMiR-CF

Interactions among robots is assumed to be implemented by explicit communication. However, if robots are capable of observing each other, this utility can also be used by DEMiR-CF to improve the performance and the solution quality. Different types of communication are available for robot systems. While ground robots use wireless communication, underwater robots may use acoustic modems.

An extensive set of communication primitives and different message types are designed for robots to have a common ontology. These messages can be implemented in KQML (Finin et al., 1997) or FIPA (FIPA ACL) compliant

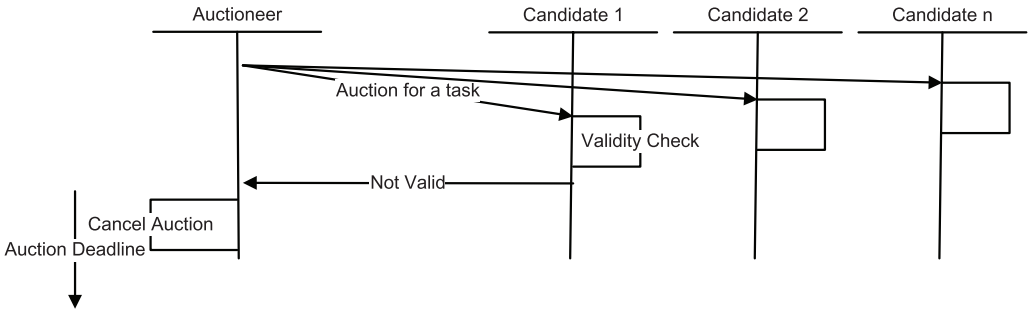


Figure 4.5: An invalid auction for an achieved task is canceled by the auctioneer after being warned by one of the candidates

formats.

Most of the time, the broadcast type of message propagation is used in DEMiR-CF. For some special negotiations peer-to-peer messaging is also used. The message types designed for DEMiR-CF are given in Table 4.3.

4.6.3. Roles

Members of coalitions are selected by auctions. The auctioneers are also active robots in the system. A robot (r_j) may take different roles for task t_i, such as auctioneer, bidder (B_{ij}), coalition leader (CL_i) or coalition member (CM_i) during runtime.

- An Auctioneer is responsible for managing auction negotiation steps and selecting $reqno_i$ of suitable members of a coalition.
- A Bidder is a candidate to become a member of a coalition to execute a task.
- A Coalition Leader is the robot responsible for maintaining the coalition and providing synchronization while performing the coalition task.
- A Coalition Member is one of the members of the coalition, and it executes a portion of the task.

A robot r_j may be in more than one B_{ij} roles for different tasks, but may not be in multiple roles as an auctioneer, a CM_i or a CL_i at the same time. The auctioneer is responsible to select the required number of robots (the coalition leader and members) for task execution. The auctioneer may or may not take place in the coalition for which it offers an auction. The coalition leader, selected by the auctioneer as the one with the minimum cost for executing the task,

Table 4.3: Message types in DEMiR-CF

Message Type	*Description*
EXECUTING	Robots broadcast messages regarding the tasks they execute along with additional information on task execution.
ACHIEVED	Robots broadcast this type of message when they believe a task is achieved.
CANCEL_EXEC	Robots declare cancellation of task execution with this type of message.
QUERY_LEADER	Sometimes it is desirable to send a heart-beat signal to check if the coalition leader is alive.
SYNCHRONIZATION	Robots send synchronization messages to coordinate tightly coupled task execution (e.g., while pushing a box simultaneously).
LEADER_INFO	Coalition leader robots send their information to the coalition members.
AUCTION	Auction messages are broadcast to all robots to declare the intention of task execution.
BID	Bid messages are sent to an auctioneer robot to declare the cost of executing the corresponding task.
AWARD_COAL_LEADER	The best suitable leader robot is awarded with this type of message.
AWARD_COAL_MEMBER	The best suitable member robots are awarded with this type of message.
WARNING	Warning messages are sent whenever there are inconsistencies detected in another robot's knowledge.
CONFIRM	Awarded robots send a confirmation message to the auctioneer.
CANCEL_AUCTION	Auction cancellation is declared to the others.
JOIN/SWITCH_REQUEST	Robots may declare their requests to join a coalition or switch a task to execute.
JOIN_ACCEPT	Robots willing to participate in task execution are accepted.
TERMINATE_COAL	Coalitions may be terminated by the coalition leader.
RELEASE	Robots may be released from the coalitions.
OBS_LOCATE	Newly detected obstacle locations may be broadcast.
OBS_CLEAR	Previously occupied but cleared for the time being location information can also be sent.
MY_LOCATION	Robots may broadcast their location information.
UNACHIEVABLE	Robots inform others about the tasks that are believed to be unachievable (e.g., tasks for untraversable targets).

maintains the coalition and keeps track of the members' conditions and their updated information. After the execution of the task is completed, the coalition ends. Each robot is allowed to take part in only one coalition until it leaves the coalition. Coordination between coalition members is implemented through synchronization messages. We assume that robots are not able to infer the state of others by observation, although such capabilities would only provide more reliability (e.g., Balch and Arkin (1994)).

4.7. Coalition Maintenance/Dynamic Task Switching Scheme

In the framework, instead of using complicated re-allocation procedures, we propose incremental selection and task switching schemes for behaving myopically while thinking globally using bid evaluation heuristics. Provided with an efficient bid evaluation heuristic, the dynamic task selection scheme ensures task switching whenever it is profitable. Each robot, independent of its current status from executing a task or not, can offer a new auction or select to execute a task already being executed by another robot with a worse cost value than it will cost for itself. If task switching occurs with a coalition member, the corresponding coalition member is released from the coalition and becomes a suitable robot for other tasks.

To ensure maintaining the solution quality against environmental changes, we propose a dynamic reconfiguration approach. Coalition members can leave a coalition when there is a sufficient number of members to execute the task. The coalition leader is responsible for broadcasting the maximum cost of execution for the task by one of the coalition members in each execution step. In the decision stage, each robot receiving these messages evaluates the maximum cost value of each coalition. If a robot detects that its cost is lower than the maximum cost of the coalition and it is released from its current coalition, it sends a join request message to the coalition leader. The leader, getting a join request message, directly adds the robot to the coalition. If the coalition leader detects that the size of the coalition is larger than required, it can release coalition members with the maximum cost value for the current task. Getting a released message, a robot can proceed to select another suitable task. When the coalition leader considers the size of the current coalition, it also checks the failures. Since each robot in the coalition broadcasts an *executing* message along with the updated cost values, their failure can be detected by the coalition leader. The failed robots are also released from the coalition. If the number of members to execute the task is below the required number for a period of time, the coalition leader cancels the

coalition. An illustrative example of coalition reconfiguration is given in Figure 4.6. Such a situation may occur if a robot is not reachable when the auction announcement is made (a). When the situation changes (b), the robot may take over the role of the member with the maximum cost value in the coalition (c). The released member can select another task to execute.

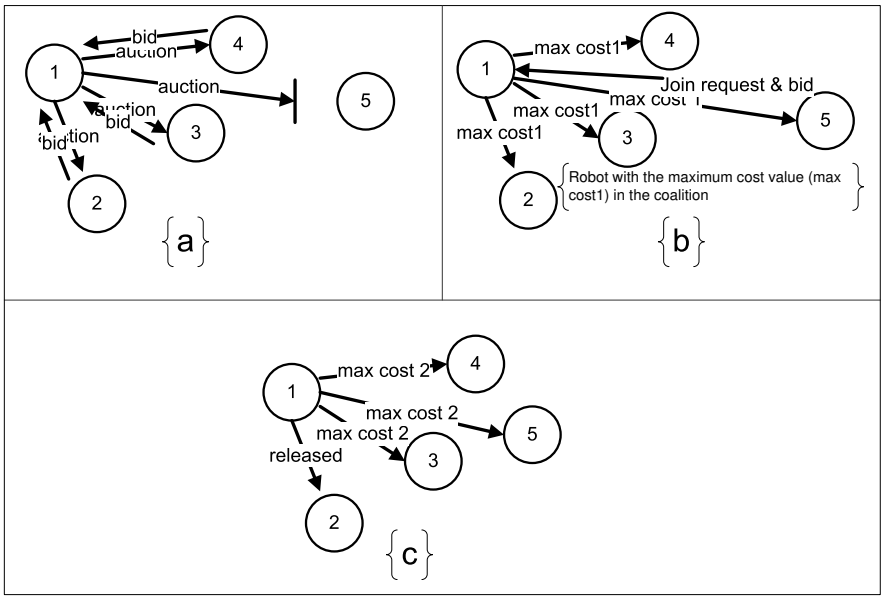


Figure 4.6: An illustrative reconfiguration scenario for maintaining the coalitional solution quality

The decision regarding which robot/agent is a member of which coalition (task execution) is an important issue. Since coalitions are disjoint in our case, assigning a robot/agent to a coalition may prevent another advantageous situation in which one of the already assigned robots may take a role in a near future formation. Further negotiations, other than auctions are needed to reach consistent agreements.

One important issue that should be addressed in robot systems is ensuring ways to plan for the global problem from local views. From our point of view, this can be achieved through extensive bid evaluation designs and additional routines to improve solution quality. We generalize statements given in desJardins et al. (1999) for continual planning. Added to the integration of planning and execution, task allocation/scheduling should also be integrated into a continual planning process. Currently most multi-agent/robot coordination architectures

implement decomposition, allocation and execution steps sequentially, and respond to the contingencies (mostly embedded in the model) in the execution phase. However, this integration is provided in DEMiR-CF by means of the *Plan B Precaution Routines* which are explained in the following section.

4.8. Plan B Precautions: A System-wide Contingency Handling Mechanism

Plan B Precautions are taken in DEMiR-CF by *The Model Update Module* which updates the system model of the robot and *The System Consistency Checking Module*. *The Model Update Module* uses incoming information and its own perception data to update the models of the robot. *The System Consistency Checking Module* provides warning messages to keep the system consistent.

In DEMiR-CF, information is not assumed to be complete and robots allocate and execute tasks in a distributed manner. This framework can take advantage of communication when it is reliable. The consistency of the current task states is ensured by *Plan B Precautions* which are carried out by the robots in a completely distributed manner. Consequently, the system immediately responds to various failure modes and can recover from them. Current implementation uses explicit communication to detect conflicts and contingencies. However, failures in communication can also be handled by precaution routines. (If robots can observe each other implicitly, model updates can be implemented in a similar manner.) Appropriate precaution routines according to the contingent situations are activated to either correct the models or to initiate recovery actions.

4.8.1. Representation of The System Model In Each Robot's World Knowledge

Each robot keeps the models of the system tasks and other robots in its knowledge base. Models of different robots may become inconsistent because of the uncertainties, incomplete knowledge, assumptions etc. It is not always possible to share a common world knowledge in decentralized systems as in the case of ours.

Task models are stored as Finite State Machines (FSM) where each task can be in a different state. Task FSMs are illustrated in Figure 4.7 and Figure 4.8 for single-robot and multi-robot task models, respectively. The difference in these two types of representations is mainly the synchronization issue that robots need to achieve when participating in a coalition. The state transitions are activated

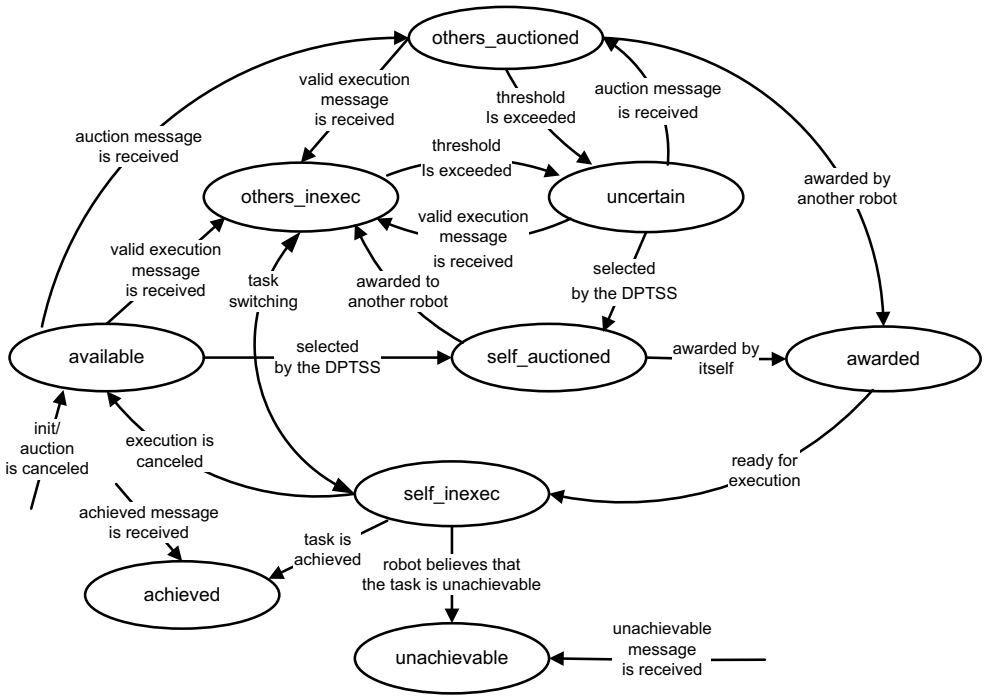


Figure 4.7: States of the FSMs for single-robot task models

by *The Model Update Module* for both types of FSMs. Different task states that tasks can be in are as follows:

- *available*: This is the initial state of the tasks that are neither in execution nor in auction.
- *uncertain* (interpreted as state *available*): This state is activated whenever there is incomplete information regarding a task which was previously being *auctioned* or *executed*.
- *self_auctioned*: A task in this state is under the auction negotiation process managed by the corresponding robot as an auctioneer.
- *others_auctioned*: A task in this state is under the auction negotiation process managed by another robot as an auctioneer.
- *awarded*: A task is in this state if the robot either selects itself or is awarded the task at the end of an auction negotiation process.
- *self_inexec_wait_sync*: A task in this state is being executed by the corresponding robot in a coalition with more than one robot but not synchronized yet.

- *self_inexec*: A task in this state is being executed by the corresponding robot in a coalition and synchronized or the coalition involves only the robot itself.
- *others_inexec*: A task in this state is being executed by other robots.
- *achieved*: This is the final state of the tasks that are achieved.
- *unachievable*: This state is used for the tasks that are not traversable or achievable.

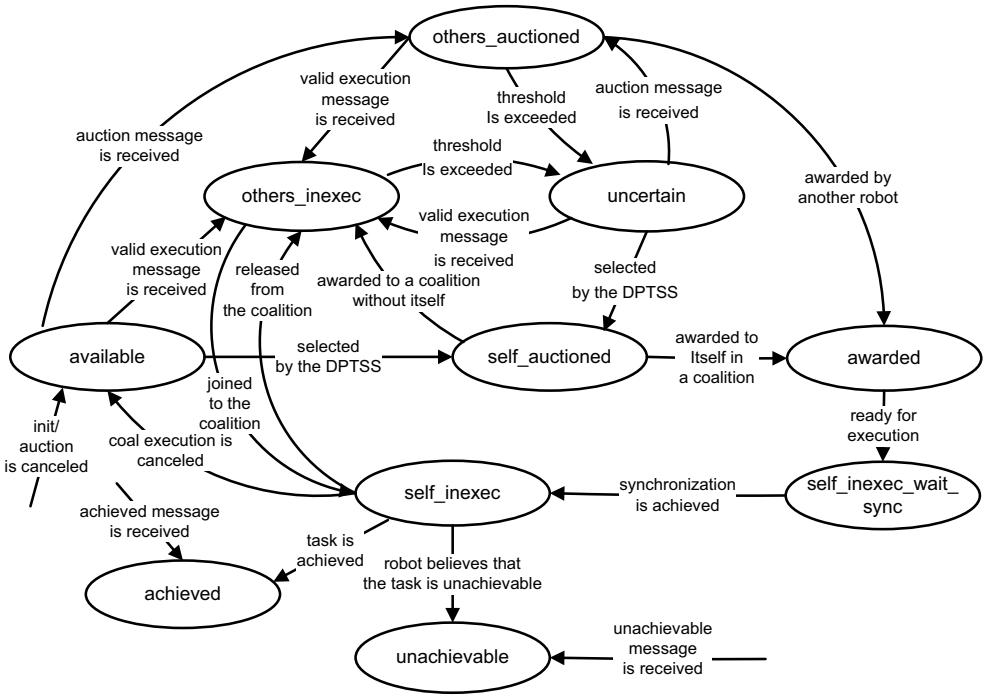


Figure 4.8: States of the FSMs for multi-robot task models

Robot models are also stored in FSMs, where each robot model has a state which is assumed by the corresponding robot. A robot may be in one of the following states:

- *idle*: A robot is assumed to be in this state when there is no evidence that it is executing any task.
- *executing*: A robot is assumed to be running properly and executing a task whenever there is supporting evidence.
- *failed*: A robot is assumed to have failed when there is no evidence that it has been running properly for a long time.

- *auctioneer*: A robot is assumed to have auctioned for a task when there is recent evidence.

Table 4.4: Precautions for contingencies and conflicts

Contingency or Conflict by inconsistencies	**Precaution**
Any message from an unrecognized system robot is received.	Robot model is created with the corresponding state derived from the message.
Any message related to an unrecognized task is received.	The task is added to the task list with the corresponding state.
An already achieved task is announced as a new task/being executed/canceled/auctioned.	Warning message is sent to the sender.
A task being executed/auctioned is announced as being executed/auctioned.	Only the robot with the minimum cost continues the operation.
Cancellation message is received for a task already being executed by the receiver.	Robot state is set as *idle*.
A cancellation message is received for a task being executed by the sender robot.	The task and robot states are set as *available* and *idle*, respectively.

4.8.2. Plan B Precaution Routines

Recovery operations may include warning other robots about the problem or changing the model accordingly. Inconsistencies usually arise when robots are not informed about tasks that are achieved, under execution, or under auction in real-world operations.

To keep system consistency, robots use explicit communication and broadcast the following information:

- known achieved tasks in predefined time periods to prevent redundant executions (This feature provides a bucket-brigade type of information sharing which enables information transition from one robot to another where point-to-point access is not possible, and consequently communication range limitations are resolved.)
- newly discovered online tasks which are unachieved yet
- task execution messages in predefined time periods (These messages contain the updated cost value and estimated task achievement deadline information. Therefore, they serve as clues, meaning that the executer robot is still alive and the task is under execution.)
- task achievement message when the task is achieved
- cancellation message if task execution is canceled

- task invalidation message when an invalidity is detected

Designed precautions are given in Table 4.4. Most of the contingencies are detected by checking models, and corresponding model updates are implemented (Table 4.5-Table 4.6).

Table 4.5: Model checking for tasks and system robots

Status	**Action**
The time duration from the latest communication with a robot is longer than the threshold.	Robot state is set as *failed*. Related task state is set as *uncertain*.
Task in execution is not achieved although the estimated deadline is reached.	Task state is set as *uncertain*.
Task state is *others_auctioned* for longer than predefined time period.	The task state is set as *uncertain*.

One standard way of detection of robot failures is sending heart-beat signals. However, in our framework, incoming messages from other robots are taken as clues for being marked as running properly. More complicated prediction models may be used for more accurate failure prediction. Some misleading beliefs such as setting the state of a robot as *failed* although it is running properly may cause parallel executions. This is a desired feature from the mission completion point of view. Designed precautions resolve these kinds of inconsistencies if communication resources permit in later steps. In designing precautions, it is assumed that robots are trusted and benevolent.

Table 4.6: Model updates related to the messages

Message Type	**Action**
Any type	Current time is registered as the latest comm. time both with the robot and for the task.
"achieved" - valid	The robot and task states are set as *idle* and *achieved*, respectively. If the task is in consideration (in schedule or in execution), it is canceled.
"execution" - valid	If there are other tasks with state *self_inexec*, these states are changed to *uncertain*.

5. EMPIRICAL EVALUATION OF DEMiR-CF ON THE MULTIPLE TRAVELING ROBOT PROBLEM

Search and rescue operations, space exploration, and reconnaissance/surveillance applications require effective multi-robot exploration. Although the coordination problem seems to be similar, the overall objective for cost optimization may be different in these domains. Search and rescue operations may require time minimization, while space operations require minimization of total path length traversed by all robots being proportional to the total energy consumed by robots.

In this chapter, we investigate the performance of several heuristic functions integrated into the framework for multi-robot exploration tasks as a case study (Sariel and Balch, 2005a, Sariel and Balch, 2006c). Because this problem area is well studied in operations research, optimal solutions are available, so we can analyze the deviations of results generated by the framework from optima. Note, however, that optimal solutions sometimes require time consuming computations, while our solutions can be found quickly. Although the problem domain consists of the same types of tasks that can be executed by a homogeneous team of robots, still it is NP-Hard due to the combinatorial structure of the problem.

5.1. MTRP Problem Statement

The single robot exploration problem, a variation of the well known NP-Hard Traveling Salesman Problem (TSP), is to find the minimum cost traversal of a given number of targets (T) without considering the return cost from the last target to the initial location for a single robot. The problem can be stated as finding the minimal Hamiltonian path on a given fully connected graph with all nodes to be visited. The travel costs are assumed to be symmetric. Although the TSP is NP-hard, there are many efficient k-OPT heuristic methods in the literature (Lawler et al., 1985).

The Multi Traveling Robot Problem (MTRP) or the multi-robot multi-target exploration problem is a generalization of the TSP in which there is a team of robots (R) to visit targets ($t_i \in M_{MTRP}$) at least once (ideally at most once). This problem may be stated for different objectives such as minimizing

the overall path length traversed by robots or minimizing the total time of traversing all targets (similar to the makespan minimization objective). In the MTRP, besides the quality of the solution constructed by the paths of robots, allocation of the targets is quite affective on the overall solution quality.

Optimal results can be obtained using Integer Programming (IP) formulations. However, these approaches may become impractical even when the size of the mission is moderate or the cost values change frequently due to uncertain knowledge, changes in the environment (including failures) or the changing structure of the mission (e.g., online tasks). Furthermore, robots have continuous path planning burdens for target sets in dynamic environments. Expensive computational efforts for initial allocations may become redundant.

The Vehicle Routing Problem (VRP) from transportation and logistics research (Toth and Vigo, 2001, Ghiani et al., 2003) is similar to the MTRP. Especially the dynamic and stochastic multi-depot VRP problem has a similar structure to the problem presented here. In the multi-depot VRP, vehicles may be in different depots, similar to our case where robots may be in different locations initially or during run-time.

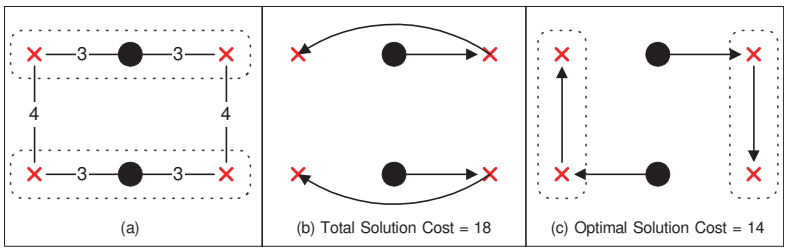


Figure 5.1: The optimal solution can be obtained by clustering the targets

5.1.1. Remarks on the MTRP Characteristics

While allocating targets, separation by considering distances between robot-target pairs and assignment of the corresponding targets to robots ignore the additional cost of returns. A sample situation is given in Figure 5.1. In (a), the locations of robots and the targets are marked with circles as crosses, respectively, and the distance values are given. Allocations and final paths generation by separating targets based on the distances to robots result in the total cost value 18 (Figure 5.1 (b)). However, the optimal allocation cost value is indeed 14 (Figure 5.1 (c)).

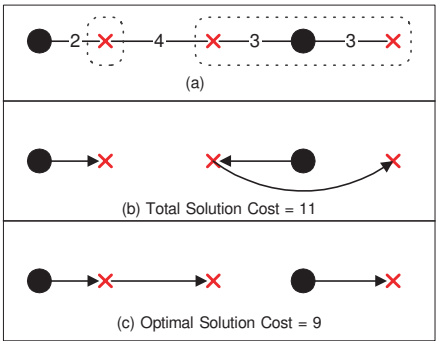


Figure 5.2: Clustering the targets does not necessarily result in the optimal solution

The optimal allocation can be obtained by clustering the targets. Clustering methods can be used to form target clusters. However, these techniques use distance information. Therefore, in some situations, clustering methods also do not produce optimal results because of ignoring additional costs as in Figure 5.2.

Lagoudakis et al. (2005) present an extensive analysis of the multi-robot exploration problem from the point of view of solution guarantees. In their work, they analyze allocation approaches for both sequential tree construction and route generation, and direct allocation while constructing paths. From our point of view, as it is suggested in their work, generating a Minimum Spanning Forest (MSF) and constructing routes on separate MSTs may not always be an effective method. One reason is that there may be different MSF solutions for the MTRP case in which some of the distances are the same. It may result in different allocation alternatives, and if there is not a reasonable allocation strategy other than selecting the minimum distance, the solution quality may degrade accordingly. The other reason is that while allocating targets, considering only distance is not an effective approach because additional costs added in the route construction phase are ignored. A sample situation is given in Figure 5.3.

Even though constructions of the tree-like structures are computationally efficient, tree construction and making allocations from scratch may result in suboptimal allocations. An IP approach may be used for finding optimal allocations. However, for even moderate sized instances, there is no guarantee of the solution time. Changes in the distance values between target and robot pairs may be frequent in dynamic environments. In this case, the solution should be reexamined. Even very small changes may completely change the overall solution.

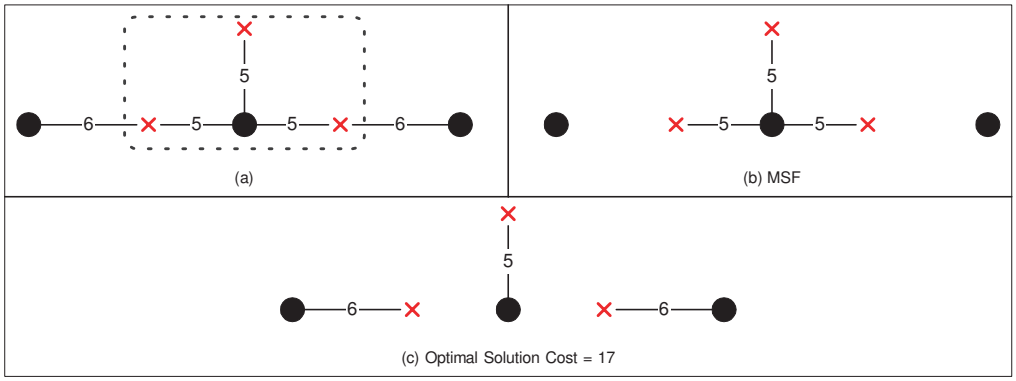


Figure 5.3: The optimal allocations may be completely different than the allocations performed after computing the MSF

5.1.2. Operations Research Methods for the MTRP

The MTRP problem is investigated as Multi-Depot VRP problem or Multiple TSP problem in Operations Research (OR). There are several methods to obtain solutions either with exact optimal value or with bounded optimality.

5.1.2.1. Integer Programming Formulation

Optimal results can be obtained by an efficient Integer Programming (IP) formulation. The optimal results in this work are generated by the commercial IP solver CPLEX using the following IP formulation given in Lagoudakis et al. (2005).

minimize:

$$\sum_{i \in R \cup T, j \in t_i} c_{ij} x_{ij}$$

subject to

$$\sum_{i \in R \cup T} x_{ij} = 1, \forall j \in T$$

$$\sum_{j \in T} x_{ij} \leq 1, \forall i \in R \cup T$$

$$\sum_{i,j \in U} x_{ij} \leq |U| - 1, \forall U \subseteq T : |U| \geq 2$$

In this formulation, R denotes the set of robot vertices and T the set of target vertices. x_{ij} is an indicator (0/1) variable for $i \in T \cup R$ and $j \in T$. If $x_{ij} = 1$, then location j must be visited directly after location i. c_{ij} is the cost value to traverse between $i \in R \cup T$ and $j \in R \cup T$.

The first set of constraints ensures that target locations are visited exactly once, while the second set ensures that robot and target locations are left at most once and, finally, the third set guarantees that there are no cycles among the target locations (subtour elimination constraints) (Lagoudakis et al., 2005).

5.1.2.2. Branch and Bound Approach

The MTRP problem can be solved by Branch and Bound algorithms. In these algorithms, a search tree is enumerated (branching) by constructing the smaller subtrees (subproblems) within a feasible search space and searched through investigating lower and upper bounds of the current subtree (bounding). The procedure continues until all nodes are either pruned or solved. The performance of the approach is dependent on the branching and bounding algorithms (Toth and Vigo, 2001).

5.1.2.3. Heuristics

There are several heuristic approaches to find approximate solutions close to the optimal solution. The methods are classified into two subclasses: Classical Heuristics and Meta-heuristics. In the classical heuristic approach, standard construction and improvement methods are applied to the solution. These approaches perform limited exploration of the search space and typically produce good results within modest computing times. In the meta-heuristics approach, the algorithm searches a large solution space. Evolutionary algorithms, Tabu Search and Simulated Annealing methods are the common methods belonging to this class. Although the quality of the solution is much higher than that of the classical approaches, time complexity worsens dramatically in these approaches. The applied procedures are usually context dependent and require finely tuned parameters (Toth and Vigo, 2001).

5.1.3. Robotic Research Methods for the MTRP

Operations Research methods are applied and integrated into robot systems in earlier works.

5.1.3.1. Prim Allocation Method

The Prim Allocation method (Lagoudakis et al., 2004), based on Prim's Algorithm (Jarnik, 1930; Prim, 1957), generates an MSF of targets and robots. An MSF consists of separate robot trees. These trees are constructed by adding each unallocated target to the closest robot path containing the node with the minimum distance to the target, until all targets are allocated. In other words, a new target is added by considering the distances between the target and the nodes of the robot tree instead of considering the last position of the robot. Each robot offers an auction for a target and one of the targets is allocated in each round. Before robots run and visit the targets, all targets are allocated. Whenever the world knowledge is changed, the remaining unvisited targets are reallocated using the same algorithm. Like Prim's Algorithm, the Prim Allocation method is bounded by 2*OPT for the MTRP. Since this method offers a single item allocation approach, it is the best candidate in the literature to compare the task allocation approach of DEMiR-CF. Furthermore, the Prim Allocation approach is discussed with its performance bounds and the details necessary to implement it, so we have been able to implement and compare the performance of DEMiR-CF to that of the Prim Allocation method.

The depth-first traversal solution of an MST is bounded by 2*OPT, and the traversal and subtree selection does not affect the solution quality in solving the TSP. However, for the open loop version of the TSP, as in the MTRP, selection of the subtree that is traversed affects the solution quality to a great extent. To improve the solutions, traversal may begin with the shortest depth subtree and continue with traversal on the subtree with the next shortest depth. A sample situation is given in Figure 5.4. We added this improvement to the PRIM Allocation method while making the comparisons.

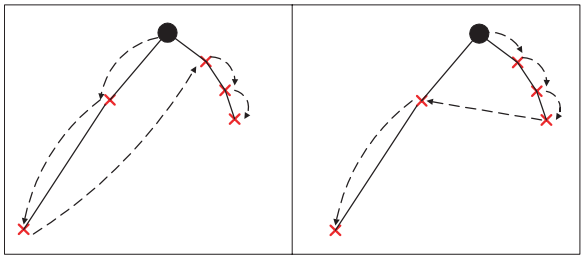


Figure 5.4: Effects of the MST traversal strategy on the total cost for the open traversal version of the TSP

5.1.3.2. Other solutions for the MTRP

Both combinatorial (Dias and Stentz, 2002, Berhault et al., 2003) and single auction methods are studied for the MTRP in the literature. In GRAMMPS (Brumitt and Stentz, 1998), one of the earliest works to solve the MTRP, a mission planner works centrally either on one of the robots or as an operator. The mission planner selects a robot for each target. The system can regenerate plans when the environment changes. Authors claim that for the initial state of the system, the allocations may be suboptimal. However, in later steps, when the number of open missions decreases, the system can find close to optimal solutions. By using the simulated annealing algorithm, a randomized search over all possible allocations is made. In their latest work, Dias and Stentz (2002) propose a market-based scheme introducing a leader approach for combinatorial task exchanges. These leaders are responsible for multi-party multi-task optimizations for obtaining optimal results. In combinatorial auctions, different combinations of tasks are offered and bidders bid by considering all tasks in these combinations. Thus, this method may become intractable for large instances or for dynamic situations in which calculations should be made frequently. Especially when the environment is dynamic, allocations may become suboptimal. Then, a combinatorial exchange mechanism is necessary to maintain optimality. Computational requirements for combinatorial auctions increase dramatically for dynamic environments.

5.1.4. Cooperation Objectives

Different types of objectives complementary to the main goal may be selected to optimize the performance of the system, as in scheduling problems. Examples of such objectives are total energy minimization (total path length minimization), time minimization, average energy minimization, makespan minimization (Lemaire et al., 2004), etc. Based on the selected objective function, cost evaluation may need to be designed differently. In Figure 5.5, the paths that robots traverse in order to optimize target allocations for two different objectives are shown. The robots are located at the bottom of the figure in the initial configuration facing the three targets to be visited. Achievement of the total path length minimization objective is illustrated in the first row of the figures (a-e), whereas in the second row of the figures, robots minimize the time needed to achieve the mission (f-j). When the total path length is minimized, one of the robots visits all targets. For the time minimization, all robots are involved in the target visiting process. Related videos of these runs are available at the website-ref: SarielKh2Mov.

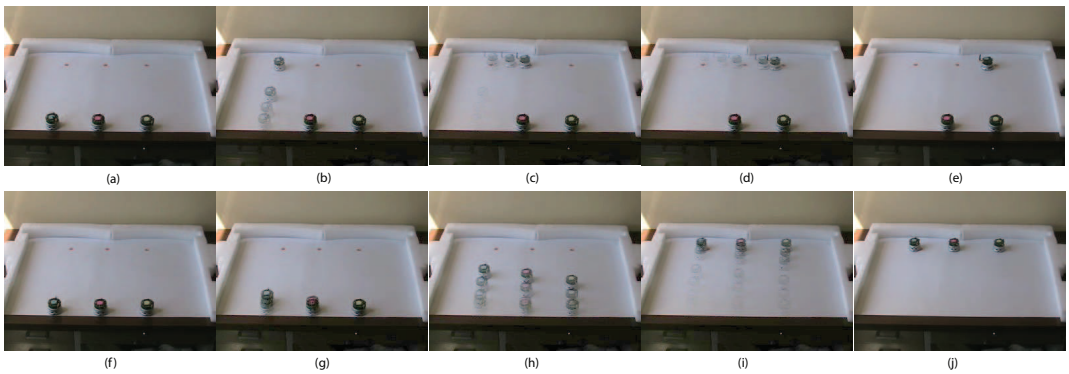


Figure 5.5: Two different optimization objectives for the MTRP: The first row of the figures illustrates achievement of the total path length minimization objective. In the second row, robots minimize the time needed to achieve the mission.

In a more general setting, incorporating composite objectives (sometimes for pareto optimal solutions) into the system may be more useful for the success of the mission (e.g., both the path length and the time need to be optimized). Besides these objectives, there are other real-time execution issues that should also be analyzed. Some of them are listed below:

- reducing collision risk,
- target priorities/strategic target selection,
- time windows, and
- multi-robot requirements to visit a target, etc.

5.1.5. Application Domains for the MTRP

The MTRP domain forms an important basis for many other domains such as Search and Rescue (SR), Space Explorations, Object Construction, Pick-up/Delivery, etc. "Goto_Target" task is one of the main tasks for these domains. Many high level tasks or assemblages of behaviors require this task to be performed. Even "pushing an object" task can be performed by simply going to the next point in the desired pushing path; that way the object is assumed to be pushed when the target location is reached.

In SR operations (Davids, 2002), the mission is challenging due to short time constraints and success is crucial. Victim locations in a disaster area are not known in advance. Incoming sensor data can be used to determine probabilistic information regarding the possible victim locations (Sariel and Akin, 2005).

In space explorations (Boddy et al., 2004), the mission is exploring unknown outer space to collect scientific data. In this domain, robots communicate through satellite links that are highly prone to communication failures and latency. Instead of optimizing time, the battery/fuel life of the robots may be optimized in this domain.

In the object construction or pick-up/delivery domain, the mission is locating the objects so that they can be carried or pushed to the desired destinations. Depending on the evidence strength of the estimated locations of the objects, the estimated task achievement time may vary. There are two situations for the problem structure: There may be (strong/weak) evidence about the estimated locations of objects, whereas in another setting, the object locations cannot be estimated in advance.

All domains listed above have the MTRP ingredients in their problem representations although the implementations and the complementary objectives may be different.

In summary, target visiting can be performed:

- to be in a specific location for a purpose,
- to participate in a formation,
- to locate an object to either affect it or collect information from it,
- to pick up an object,
- to drop off an object at/deliver an object to the destination.

The frontier cell-based exploration to cover unknown environments and create maps (Burgard et al., 2005) is one of application domains related to the MTRP. In this case, the coverage problem is efficiently reduced to the assignment of frontier cells to the robots. The frontier cells can be modeled as the targets of the MTRP.

In another application domain, the coverage problem is represented as visiting waypoints, which can also be formulated as the MTRP problem (Sariel et al., 2006b). Therefore, although the coverage problem and the MTRP seem to be different, they share common structures when appropriately represented.

5.1.6. Communication Requirements

One of the main issues in multi-robot systems is the communication among robots. The following questions from Balch and Arkin (1994) determine the design criteria for multi-robot systems: How much communication do robots need? Given the communication limitations, what is the best strategy? From our point of view, communication is the top most priority issue in task allocation for multi-robot systems. Regardless of the task selection mechanism, the outside connection has a superior effect on the task achievement time.

5.1.7. Formation of the Mission Structure for the MTRP

Although not investigated in detail previously, the generation of the targets for the MTRP is an important issue. Even if the target locations are known in advance, if robots use different global reference frames, high uncertainty and many inconsistencies are unavoidable.

There are two ways of online target assignment:

- the operator agent(s) assign the targets to the robots, or
- robots discover new targets to be visited by themselves or by others. (e.g., in the search and rescue case, a robot may discover more than one potential victim location.)

The important question which arises is reaching a consensus on the determination of the target: Which robot is talking about which target? Our solution to this problem is assigning system-wide known id numbers (reserved) to the system generated tasks. Alternatively, the robots can communicate the believed locations of the targets. Given a region threshold, relatively close targets are treated as the same (Sariel et al., 2006b), although this brings uncertainty into the system.

The problem with online target generation by robots themselves is that each robot executes its own localization and mapping procedures and may come up with different localization errors. These errors may accumulate resulting in greatly overestimated target locations.

5.2. Applying DEMiR-CF to the MTRP

In practical applications, computing the true optimal solutions is not always required due to several reasons (Reinelt, 1994). Those reasons may be the incorrect modeling of the underlying problem (targets) or the lack of sufficient time to find the optimal solution. These are common cases in robot applications along with the real-time issues presented in Chapter 2. To meet all these limitations, we propose a dynamic and distributed task allocation scheme, DEMiR-CF, to coordinate robots that cooperate to fulfill different parts of a mission. Dynamism is achieved through the incremental selection and allocation of the targets. By means of the distributed characteristic of DEMiR-CF, each robot is allowed to select a candidate task for itself and, next, the robots proceed to cooperate in the process of selecting the most suitable robots for the tasks.

DEMiR-CF is designed with the capability to deal with the situations presented in Chapter 2. The framework can efficiently respond to these events and the solution quality is maintained simultaneously with the real-time task execution. We propose a general mechanism for multi-robot cooperation for the MTRP but not necessarily specific heuristic functions to solve the problem, although we validate their successes.

5.2.1. The Dynamic Priority Based Task Selection Scheme

The dynamic priority based task selection and allocation scheme ensures two types of selection. Each robot selects a target to visit. The system provides a CNP-based selection method to select the appropriate robot to allocate a task. This mechanism ensures distributed, robust and scalable allocations. The incremental assignment approach eliminates the complexity of the decision of allocating all targets to all robots at a time.

5.2.1.1. Selecting is Eliminating the Other: Incremental Allocation through Unconditional Commitments

Each robot selects candidate targets that are suitable for itself and forms a target set (rough schedule). This set consists of the targets suitable to be visited by the robot. The robot then selects the most suitable candidate task to perform among these targets in its corresponding set. After selecting the most suitable target for itself, the robot announces its intention. A CNP-based selection is implemented and the most appropriate robot among the team of robots to perform the task (to visit the target) is selected. When robots receive task execution intention messages, they either send their cost values as bids for the announced target or

send warning messages to the sender robot. These warning messages are sent if the auction is for an invalid target, is one which has already been achieved or is being executed by another robot. The design of the bid generation rules and the reply messages is very affective on the quality of allocations. We analyze two different bid generation rules for different optimization criteria.

Since there is a tight connection between route generation and allocations for the MTRP, robots initially generate rough routes (rough schedules) in our heuristic approach. Next, each robot (r_j) selects its most suitable target among the targets in the rough schedule (T_{Rj}). T_{Rj} is constructed by selecting the targets close to robot r_j, among unvisited targets (T_U) according to Equation 5.1, where *dist* function returns the Euclidean distance between two points. Targets in T_{Rj} are considered as the candidate targets for robot r_j. Therefore, before selecting the most suitable target, each robot constructs these rough route sets. This heuristic does not compel an actual commitment, and the targets in the rough routes are not necessarily assigned to the corresponding robots in the future auctions. Instead, it provides a global view to the problem from a local perspective.

$$\begin{aligned} reldist(r_j, t_i) &= dist(r_j, t_i) - min(dist(r_k, t_i)) \\ &\quad \{\forall k \neq j, r_k \text{ is active}\} \\ T_{Rj} &= \cup t_i, \quad reldist(r_j, t_i) < 0, \ \forall t_i \in T_U \end{aligned} \tag{5.1}$$

5.2.1.2. Cost Estimation and Evaluation

Although the TSP problem is NP-hard, there are many efficient heuristic methods in the literature generating k-OPT solutions (Lawler et al., 1985). Some of the heuristic methods use the triangle inequality principle given in Equation 5.2 The triangle inequality principle for cities i, j and k assumes that the shortest distance (c) between two cities is a straight line (direct route).

$$c_{ik} \leq c_{ij} + c_{jk} \tag{5.2}$$

Since these heuristic cost functions form the basis of our inspiration to design our cost functions, we review them in Appendix A. We are inspired by some of these route generation heuristics in the design of our multi-robot multi-target route construction heuristics. We extensively focus on two heuristic cost functions combined within our framework in this work. These cost values are evaluated for the targets in T_R for each robot. The CC (Closest Cost) heuristic cost value for robot r_j and target t_i is evaluated by Equation 5.3. This heuristic cost function

only considers the distance between targets in T_{Rj} and the robot r_j.

$$c_{ji} = dist(r_j, t_i)\ t_i \in T_{Rj} \tag{5.3}$$

The FAC (Farthest Addition Cost) heuristic function considers costs as if there is a route for T_{Rj} as in Equation 5.4 and applies a penalty for not visiting the boundary targets. Boundary targets, t_{b1} and t_{b2}, are the targets in T_{Rj} with the maximum distance value. The FAC heuristic forwards robots to these targets in T_{Rj} to some degree. The main idea behind this approach is that the open loop traversal should contain both t_{b1} and t_{b2}s. If the robot heads towards one of these targets, if profitable (α), the longest path can be traversed by traversing other targets on the path. A sample illustration of this cost function is given in Figure 5.6. In this figure, although t_2 is closer to r_1 than t_1, with the FAC heuristic applied, t_1's cost value is smaller than that of t_2 ($3 < 3.6$), hence resulting in a better route shown by the dashed arrows. We have performed an empirical analysis of the parameter α; the best results are observed for a value of 0.6.

$$\begin{aligned} c_{ji} \quad = \quad & \alpha * dist(r_j, t_i) + (1-\alpha) * [dist(t_{b1}, t_{b2}) \\ & -max(dist(t_i, t_{b1}), dist(t_i, t_{b2})] \\ & \{dist(t_{b1}, t_{b2}) = max(dist(t_k, t_l))\ t_{i,k,l} \in T_{Rj}\} \end{aligned} \tag{5.4}$$

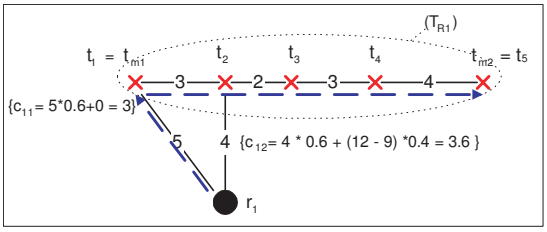


Figure 5.6: Target selection strategy by the FAC Heuristic function

5.2.1.3. Dynamic Task Selection and Distributed Target Allocation

Each robot executes Algorithm 3 to generate its rough schedule. The robot then selects the most suitable candidate task (t_s, the most suitable target among the rough schedule targets) to perform.

Algorithm 4 forms the main loop for incremental task allocation and it is called in the beginning of the mission execution and whenever the world knowledge

Algorithm 3 MTRP-FormRoughSchedule for r_j

input: T_U
output: T_{Rj} and t_s

$T_{Rj} = \phi$ (a heap with task cost as the key)
$t_s = \phi$

while T_U is not empty **do**
  if t_i is in the rough schedule region according to Equation 5.1 **then**
    c_{ji} = evaluateCost($t_i \in T_U$) (*)
    insert t_i into T_{Rj}
  end if
end while

if $\|T_{Rj}\| > 0$ **then**
  $t_s = top(T_{Rj})$
end if
$*$ c_{ji} is evaluated by Equation 5.3

of the robot changes. Each robot executes the same algorithm concurrently until the end of the mission, when all traversable targets are visited. The given algorithm may be used to allocate all targets from scratch. However, an incremental assignment approach eliminates the redundant allocations for dynamic environments. The cost function design can be determined based on the capabilities of the robot. The cost function that we use can be successfully implemented for very small robots, as is shown in our experiments.

Algorithm 4 MTRP-DPTSS algorithm for robot r_j

input: T_U
output: An action to be performed

$[T_{Rj}, t_s]$ = **MTRP-FormRoughSchedule**(T_U)
if $t_s \neq \phi$ **then**
  if t_s is the current task **then**
    continue with current execution
  else
    if t_s is an *available* task **then**
      offer an auction to announce intention on execution
    else
      directly execute the task (task switching or being awarded)
    end if
  end if
else
  stay idle
end if

After selecting the most suitable target for itself, each robot announces its intentions by a single-item auction. Selection of the best robot for a task is performed by using the Contract Net Protocol (CNP) in our approach. Although the CNP presents the formalism on the relationships between managers and contractors, some simple decisions are left to the designer. Most auction based task allocation schemes offer solutions for allocating one/a subset of task(s) of the overall mission. However, there is usually little information about when task announcements and reassignments are made.

The approach we propose allows for multiple auctioneers and winners for different tasks, depending on the optimization objective. In the case of the total path length minimization objective, ending one auction at a time results in better performance when there are relations between targets. This is the reason why some auctions are canceled when there are multiple auctions going on at the same time. On the other hand, auctions/executions are canceled only if there are relations between the targets in consideration for the time minimization objective. These two different responses are generated by Algorithm 5 and Algorithm 6, respectively.

Algorithm 5 Response for Path Optimization - r_j

if auction/execution is in progress & ($c_{ji} > c_{lk}$) **then**
  cancel auction/execution for t_i
end if
if $c_{jk} < c_{lk}$ **then**
  send bid value for t_k
end if

Algorithm 6 Response for Time Optimization - r_j

if auction/execution is in progress & ($c_{ji} > c_{jk}$) || (($c_{ji} > c_{lk}$) & (t_k or r_l is close to the T_{Rj})) **then**
  cancel auction/execution for t_i
end if
if $c_{jk} < c_{lk}$ **then**
  send bid value for t_k
end if

5.2.2. Failure Detection and Recovery

Plan B Precaution Routines integrated into the dynamic task selection mechanism is applied for the MTRP as explained in Chapter 4. The single-robot task models

are used to represent the states of the FSMs for individual tasks. Robots continually perform model updates and consistency checking operations using information received from the incoming messages and the data perceived from the environment during runtime. When robots receive task execution intention messages as auctions, they either send their cost values as bids for the announced targets or send warning messages to the sender robots. Failures can be detected immediately and recovery routines are activated for the failures or contingencies.

5.3. Bounds on the Solution Quality

The performance of the Prim Allocation algorithm is proved to be bounded by 2*OPT (Lagoudakis et al., 2004). The difference between the Prim allocation and the CC heuristic approach is in the robot location considered. Our framework combined with the CC heuristic considers the latest location of robots, while the Prim Allocation algorithm considers their initial locations. Assuming there are two subtrees of the MST, the CC heuristic approach forwards robots into either of the subtrees of the MST, leaving the other subtree to be traversed by another robot or by itself. If in the end, the first robot traverses the subtree, the solution cost is the same as the Prim Allocation solution cost. However, if another robot traverses the other subtree, the generated solution is better than depth-first traversal since the other robot has been favored because of its cheaper cost value. The CC heuristic can be classified as one of the BidSumPath heuristics (Lagoudakis et al., 2005) and it is shown that the solution is bounded by 2*OPT. The FAC heuristic forwards robots to one of the border subtrees. In the worst case scenario, the next selection phase forwards a robot to the next subtree in the MST before the completion of the traversal of a subtree (usually this results in the elimination of long connections among subtrees and gives better results). However, by making use of triangle inequality, going back to the previous subtree cannot be greater than two times the traversal of the corresponding MST edges in this worst case.

5.3.1. Analysis of the Approach

The approach we have proposed for the MTRP offers a polynomial time solution. Sorting the distance values to find the boundary targets takes $O(nlog(n))$ for all n number of tasks. Both cost and queue initialization are implemented in $O(n)$. Top element selection and deletion is performed in $O(logn)$. Therefore, the total complexity is bounded by $O(nlog(n))$. In the worst case, the environment is dynamic and cost values change frequently in the order of $O(l)$. Then the total complexity becomes $O(lnlog(n))$ for each robot.

5.4. Implementation Details

"Morse decomposition" and spinning behavior is stated to be efficient for searching an area (Acar et al., 2002). This is used in our system for locating objects whose estimated locations are not exactly known or when there is uncertainty about the location of either an object or a robot. This idea is useful for locating specific objects that are very likely to be located in the search area. The framework implements the Archimedean spiral in robot behavior, an is inspired from the geometric implementations given in Figure 5.7.

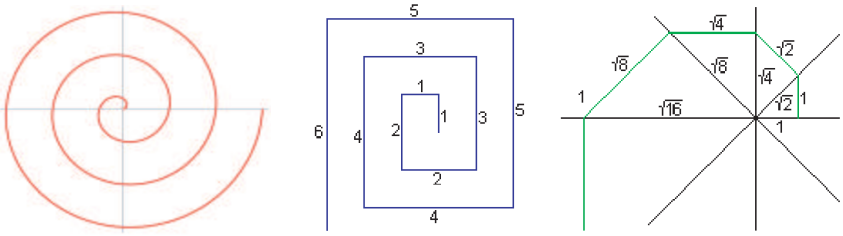


Figure 5.7: The Archimedean spiral and two simple implementations

A search and rescue domain in which victim locations are not exactly known but may be estimated, is a candidate domain for applying this idea. However, if the robot is only visiting waypoints, spinning behavior may be time consuming in the application.

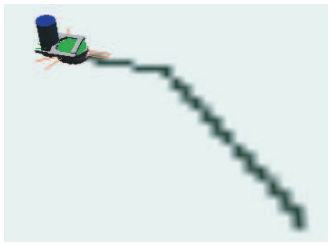


Figure 5.8: The robot can directly find the target with a reasonably accurate estimation

We define a deadline threshold for the spinning behavior in our implementation. The robot searches the area to find the object until the deadline is reached. If the object is not found within the deadline, the robot gives up its search for that target and continues to look for other targets, if any. In Figure 5.8, the robot

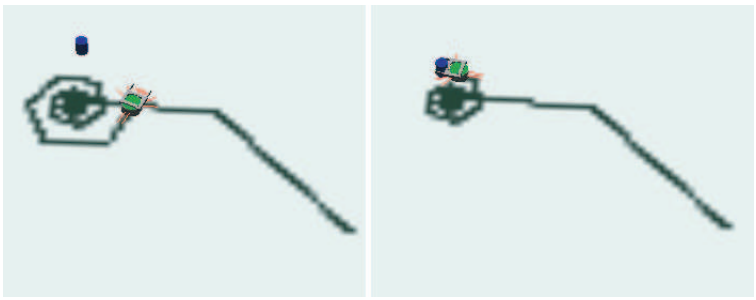


Figure 5.9: An implementation of the Archimedean spiral with the corresponding robot behavior

directly finds the object (The scenario illustration is taken from a WEBOTS simulation with a Khepera robot) since the uncertainty is in an acceptable degree. Figure 5.9 presents two situations; one in which the robot can find the object before the deadline is reached, and one in which it cannot find the object in due time. Due to a localization error or inexact information regarding the location of the object, the robot should search the area for the object. On the left, the robot continues to search until a deadline is reached. On the right, it finds the object while spinning before the deadline. In some of our experiments illustrating different cases, robots simply visit targets without looking for objects and without using the approach presented above.

5.5. MTRP Experiments

MTRP experiments are designed in four sets. In the first set, the performance of the proposed heuristic cost function for the MTRP is analyzed. In the second set of experiments, the heuristic approaches are evaluated to measure the solution quality. All methods are implemented by coding all methods in different programs with the same interface to give the test instances on CPLEX by using the Integer Programming formulation given in Lagoudakis et al. (2004). Basically, in this set of experiments, the algorithms are run on distance matrices. In the optimal result generation for large instances of the problem, the constraints (3rd) cannot be fed into CPLEX. As explained in Lagoudakis et al. (2004) we used a cutting plane method to solve the integer program. In our experiments, we observed this limitation for instances with 18 targets or more. The results are taken by the cutting plane method after iterative calling of CPLEX for a previously found solution, as explained in the original paper (Lagoudakis et al., 2004). The environment is represented as a grid with 100*100

nodes; the number of robots change in the range [1-50] and the number of targets in [10-50]. Time comparison results are taken for the centralized implementation of the methods. The distance calculations among targets and robots are excluded from the run time period while the IP model generation is included in the time period because it is part of the solution. Both approaches are assumed to be running on the given distance matrices. The results are presented as averages of 100 independent runs for the randomly generated test instances. The running time results are averaged over 30 runs.

In the third set of experiments, the real-time performance of DEMiR-CF on the MTRP is evaluated as dynamic simulation experiments on the professional mobile robot simulation software, WEBOTS (Michel, 1998). WEBOTS contains a rapid prototyping tool allowing the user to create 3D virtual worlds with physics properties, such as mass repartition, joints, friction coefficients, etc. The fourth set of experiments is performed on real robots, namely Khepera II.

Table 5.1: FAC heuristic function performance results for the known TSP instances

TSP Instance	FAC	Optimum	% Deviation from the Optimum
ATT48 - 48 nodes	33537.83	31470.4	6.5
EIL51 - 51 nodes	444.01	413.51	7.37
BERLIN52 - 52 nodes	8104.99	7305.38	10.94
EIL101 - 101 nodes	725.31	629.38	15.24

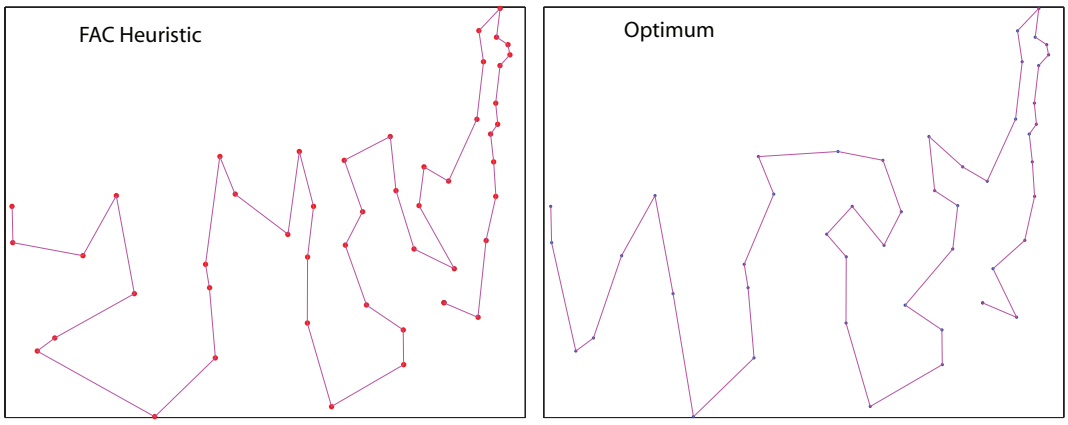


Figure 5.10: Open-loop routes of the FAC Heuristic function and the Optimum for the ATT48 TSP instance

5.5.1. Experiment 1: Evaluation of the FAC Heuristic Cost Function

In the first set of experiments, an analysis of the performance of the FAC heuristic for the TSP is performed for the known TSP instances (Reinelt, 1991). Problem instances are presented in Appendix B. The results are given in Table 5.1 as the cost of traveling through visiting all nodes for a single robot. These results reveal the near-optimal performance of the FAC heuristic function with at most 15.24% deviation from the optimum (for a large TSP instance). Note that these results are for the open loop TSP and present construction solutions without any additional improvements applied. In Figure 5.10, open loop routes of the FAC heuristic function and the Optimum are given for a sample TSP instance, ATT48, where targets represent geographical locations of capitals of US cities. The geographical representations and the optimal routes for other instances are given in Appendix B.

5.5.2. Experiment 2: Performance Comparison of DEMiR-CF and the Prim Allocation Approach with the Optimum

In the second set of experiments, heuristic functions integrated into DEMiR-CF and the Prim Allocation method are evaluated on randomly generated test sets for different numbers of robots and targets.

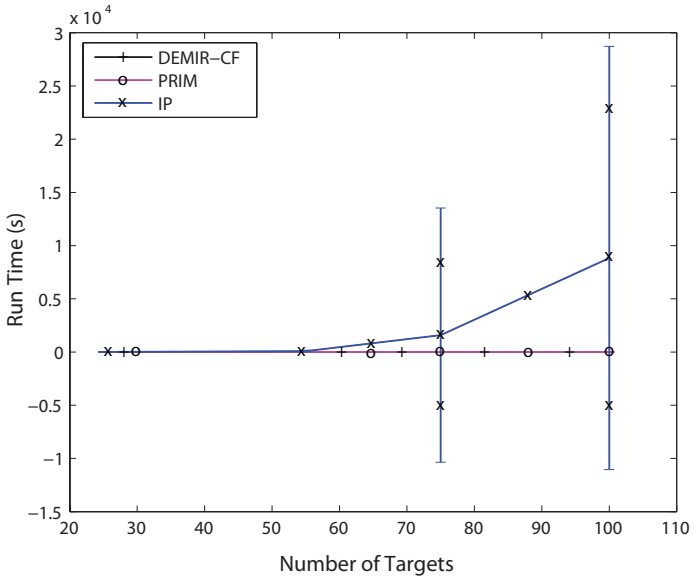


Figure 5.11: Runtime comparison of the approaches for single robot route generation

Comparison of running time requirements of DEMiR-CF, the Prim Allocation and the IP approach for the MTRP solution approaches are given in Figure 5.11.

Large standard deviation values for the IP approach present the dependency of the solution time on the problem instance structure. The performance of the IP approach is close to the worst case performance for some instances, not given in these statistics. Depending on the application and the instance size or the frequency of the changes in the distance values, the IP approach may be impractical without guarantees on the solution time. Allocations by using a heuristic approach (either DEMiR-CF or the Prim Allocation) can be implemented in a very short time as expected and are given in Figure 5.11. This graph presents running time results of the approaches for the single robot case. With a decreasing amount of target densities, the IP approach solution time decreases accordingly.

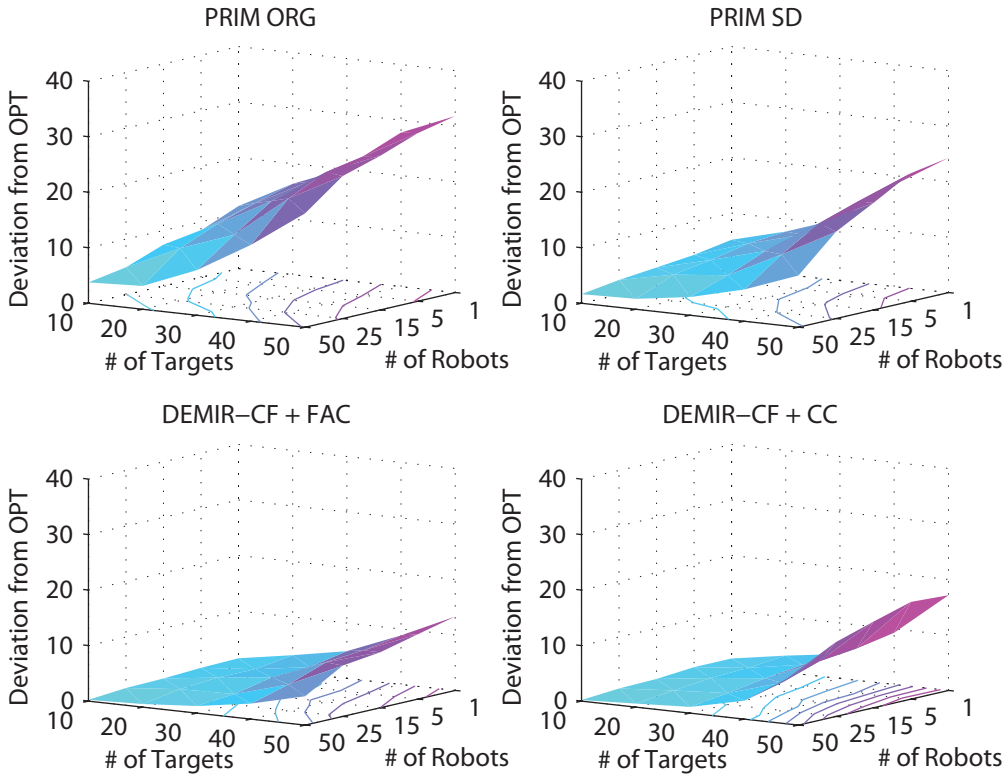


Figure 5.12: Performance results of the heuristic approaches

The overall performance results are given in Figure 5.12 as deviations from the optima with standard deviation, averaged over 100 runs. PRIM-ORG values represent the results of the Prim Allocation method without considering subtree sizes on the traversal, while PRIM-SD values represent the results with shortest subtree selection improvement. Results of the FAC heuristic approach are

promising even for single robot instances. With increasing number of robots, the solution quality is also affected by the target allocation. Therefore, the CC and the FAC heuristic results become closer with only a very small value of deviation from the optima. The decrease in target/robot proportion results in a decrease in the deviations of the results from the optima. However, results obtained for the Prim Allocation method still deviate from the optima because of the allocation method. This is prevented in DEMiR-CF by the dynamic selection of T_Rs. Allocation samples generated by the algorithms and the optimal allocations for a 15 robots 50 targets instance is given in Figure 5.13. Note that our results can be further improved by using 2-OR exchange (Toth and Vigo, 2001) type of improvements, if the communication is reliable.

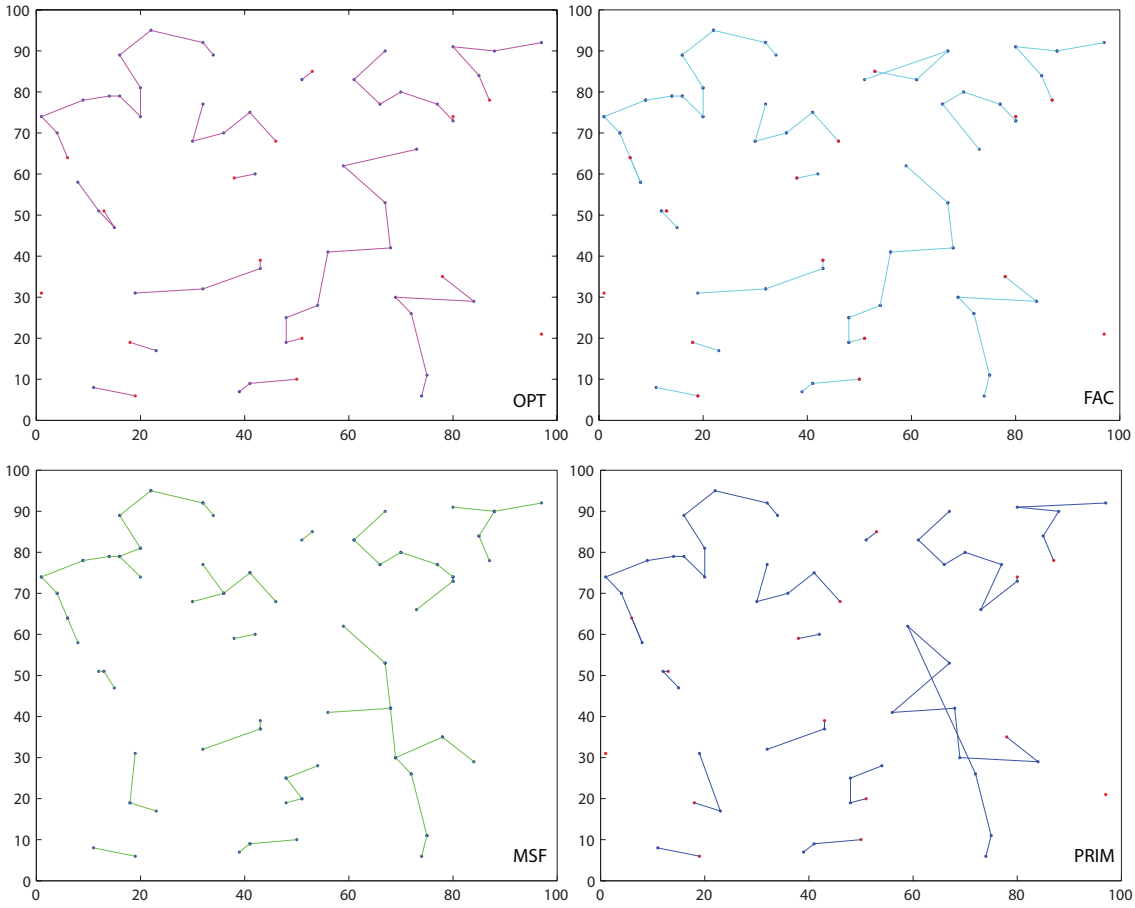


Figure 5.13: Allocations generated by each approach for an instance of 15 robots and 50 targets

5.5.3. Experiment 3: Real-Time Dynamic Simulation Experiments

In our simulation experiments, each environment is represented as a 5m by 5m 3D virtual world where 70mm-sized simulated Khepera II robots and objects are located. The environments are randomly generated VRML files containing the robots, the objects and the MTRP targets.

Robot kinematic calculations are done on the low-level design of the robots by using the odometry information coming from the encoders for both simulation and real-world experiments. The robots perform mapping of the environment by using an occupancy grid approach (Moravec and Elfes, 1985) simultaneously with online localization. In the simulator, slipping and encoder errors are simulated whereas the real world has its own uncertainties. Due to the slipping errors, as expected from differential wheel robots, usually odometry errors are encountered.

The scalability of the system for different numbers of robots is evaluated through 10 run-time experiments for each set. The robots are randomly located in the environments with 10 targets at fixed locations. Since the approach proposed is for a multi-robot team, we expect to see a linear decrease in the total path length traversed by the robots as the number of robots increases as in Figure 5.14. These results validate our expectations.

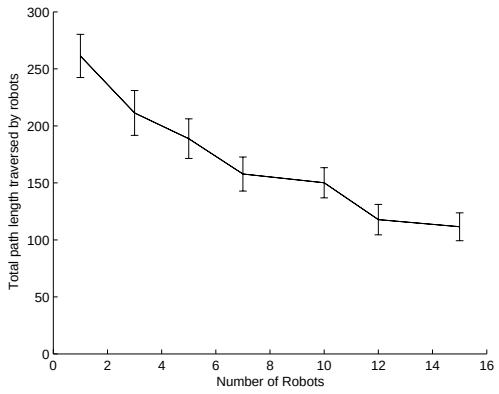


Figure 5.14: Scalability analysis for different number of robots

In another experiment using the same setting presented above, the performance of the contingency handling mechanism is evaluated and the results are depicted in Figure 5.15. NO_PREC indicates that capabilities of the contingency handling mechanism are not used, while USE_PREC indicates that they are

used. The figure on the left is the total number of messages sent by the robots. The surprising result here is the increase in the number of messages for the NO_PREC case. Although the contingency handling mechanism seems to inject additional messages into the system at first glance (for ensuring the system consistency), these results reveal that it also eliminates the redundancy in the number of messages for multiple bids and auctions. The figure on the right shows the total path length traversed by the robots. The path length is presented by discretized values (each unit is 70mm). The experiments reveal that using the contingency handling mechanisms reduce both the number of messages sent and the total path length traversed by the robots. This result validates the efficiency of integrating the contingency handling mechanism into the system. Dynamic simulation experiments where the robots perform both

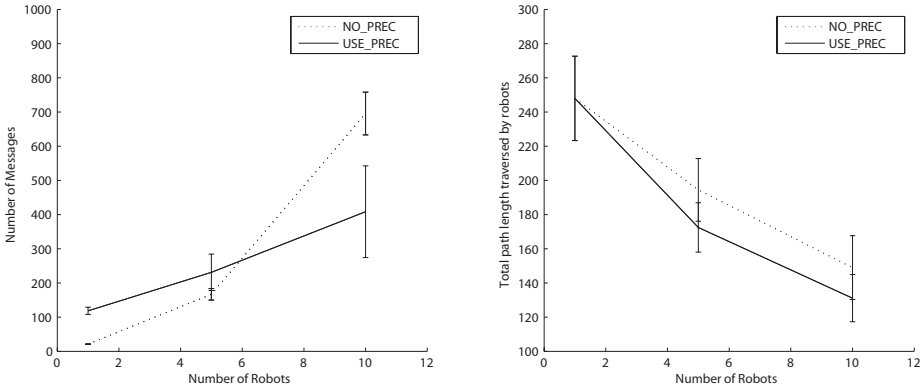


Figure 5.15: Performance results for the Contingency Handling Mechanism

task allocation and real-time sensing, localization and mapping can demonstrate how the knowledge of map information changes the problem instance and the requirements on reallocations when new information is encountered by robots. Two sample scenarios with 3 robots and 10 targets present this hypothesis. The initial configuration of the environment is illustrated in Figure 5.16. The targets are presented by the blue landmarks. Two robots are located at the bottom of the environment and the third robot is located in the center of the environment.

In scenario 1, lacking the map information, the robots initially assume that the environment is empty, even though there are obstacles. Thus, visiting the predetermined target locations takes much more time than expected. The final state of the environment at the end of the mission is presented in Figure 5.17. Robot paths are drawn by the simulator in this figure. The robots carry out localization and mapping operations simultaneously to visit targets efficiently.

The actual paths and the estimated paths traversed by the robots using odometry information and kinematics calculations are illustrated on the left side of Figure 5.18. The estimated paths are represented by dotted lines. As can be seen from the figures, due to the slipping and encoder errors, the estimated paths do not completely overlap with the actual paths traversed. In such cases, target locations may sometimes be incorrectly interpreted. The errors on the traversed paths may be reduced by using landmarks in the environment and adjusting the localization error accordingly. The occupancy mapping approach (Moravec and Elfes, 1985) is used in our experiments. The maps generated by the robots are illustrated on the right side of Figure 5.18. Rough schedules are generated and robots visit targets based on the up-to-date selections. During mission execution, one of the robots detects that a target is not traversable after discovering the obstacles surrounding the target. In scenario 2, the map of the environment is given to the robots (Figure 5.19). As expected, robots do not intend to visit untraversable targets. Paths and maps of this scenario are illustrated in Figure 5.20.

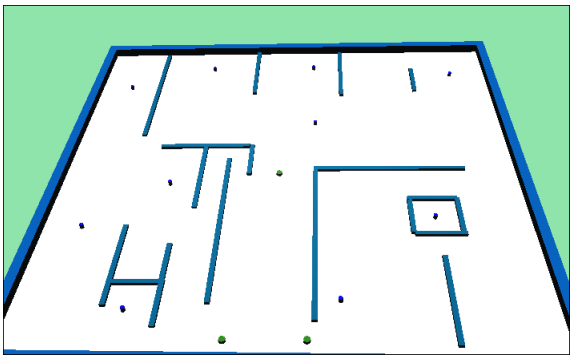


Figure 5.16: The initial configuration of Scenario 1

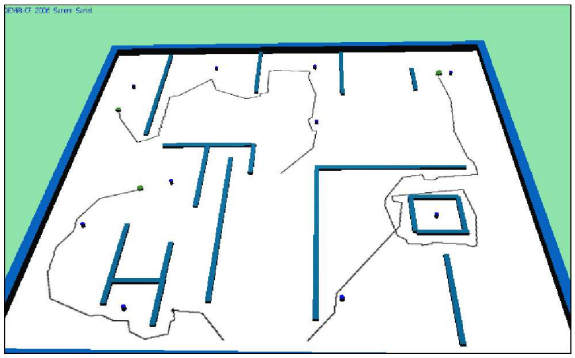


Figure 5.17: The final configuration of Scenario 1

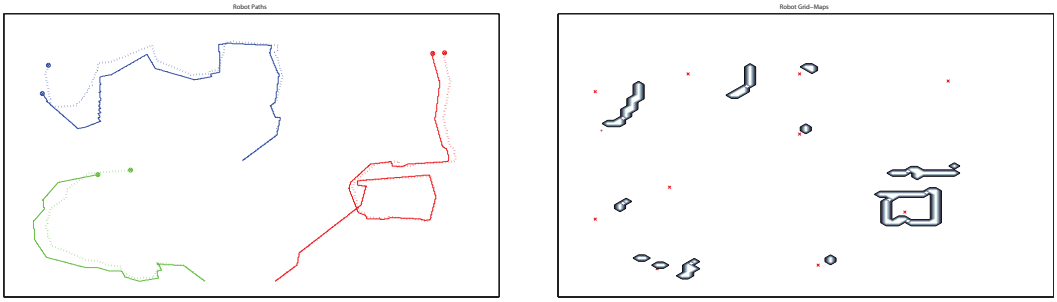


Figure 5.18: (left) Ground truth and estimated paths of the robots. (right) Runtime maps generated by the robots in Scenario 1.

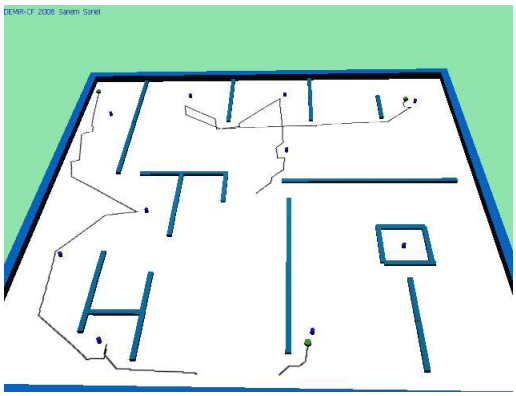


Figure 5.19: The final configuration of Scenario 2

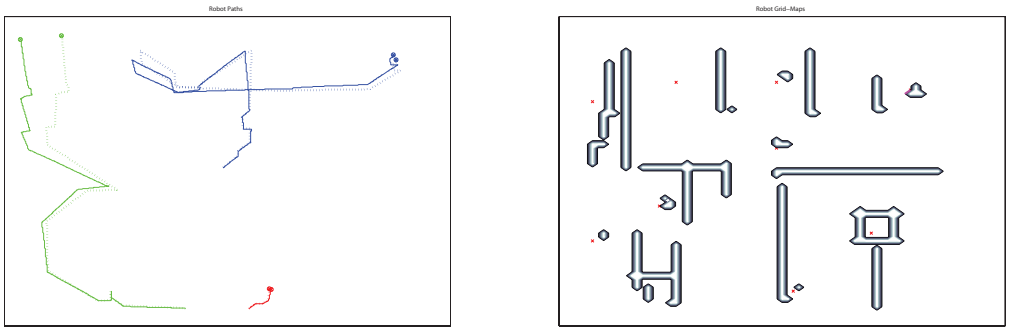


Figure 5.20: (left) Ground truth and estimated paths of the robots. (right) Runtime maps generated by the robots in Scenario 2.

5.5.4. Experiment 4: Real-Time, Real-World and Dynamic Environment Experiments

Each Khepera II robot is equipped with a 25MHz MC68331 micro-controller, 512K Flash and 512K RAM memories and 8 infra-red sensors with a limited obstacle detection range as simulated in WEBOTS. Communication is achieved through wireless links. Real Kheperas have standard radio turrets mounted on them to communicate through the selected radio frequency. Emitters in the simulator are configured to have a baud rate of 9600 and a buffer size of 1024B as in the receiver modules.

Figure 5.21: The Khepera II robot visits six targets in the environment

Figure 5.22: Three Khepera II robots divide the labor of visiting the six targets

Each robot executes a multi-threaded controller software to achieve the functionality of DEMiR-CF. Different modules on the task allocation layer are integrated with the low-level Sensory Interface, Motor Interface, Motion Model and Mapping modules in the multi-threading structure.

The overlaid video frames from real robot experiments that we have conducted are illustrated in Figure 5.21 - Figure 5.23. Figure 5.21 illustrates a path

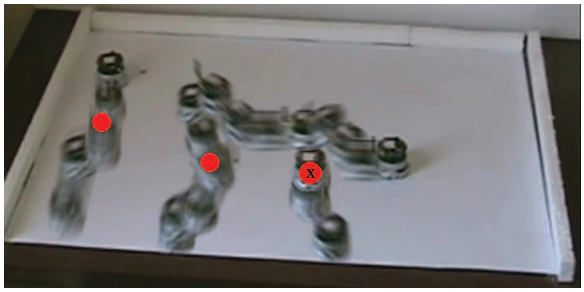


Figure 5.23: The team handles the failure of the third robot in real-time

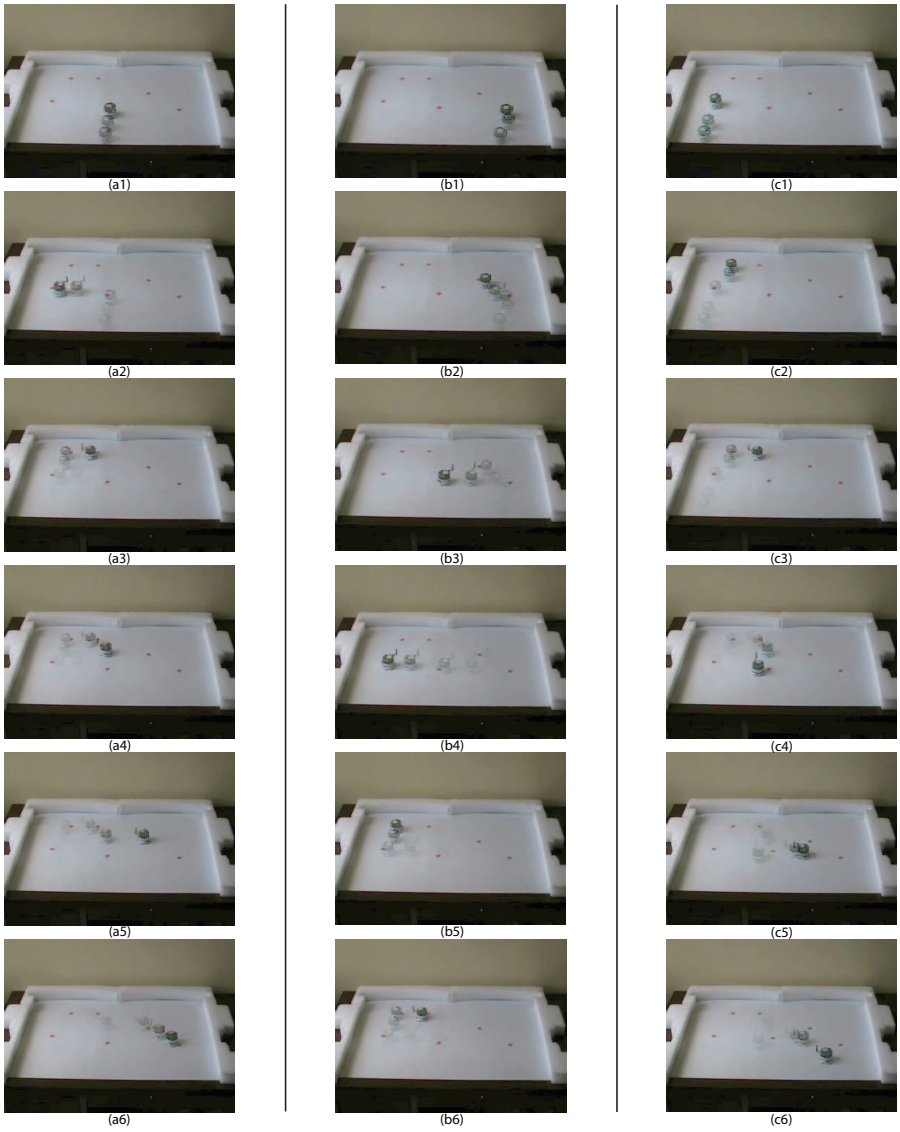


Figure 5.24: Single robot cases for different initial deployment locations of the robots

traversed by one Khepera II robot visiting six targets in the environment. The target locations are shown with blue flags and fixed in all real robot experiments. When we have a multi-robot team, the mission completion time is reduced by the robots' division of labor. The robot paths are illustrated in Figure 5.22. The multi-robot team can successfully handle robot failures as illustrated in Figure 5.23. All robot positions at the time of the failure are marked with red filled circles.

Depending on the initial location of the robot, the path constructed to traverse the targets differentiate for the single robot case. This is illustrated in Figure 5.24. In this figure, each column (a-c) represents the illustration of an independent run with a different initial deployment location for the robot. The continuous video frames are divided into episodes in which the images of the videos are overlapped and the final overlapped image is illustrated (e.g., a1,a2..a6) in the figure.

Figure 5.25 illustrates the scenario (Scenario 1) with three robots for the same setting of the target set. Time is minimized by using a three-robot team. Robots use DEMiR-CF to select and allocate tasks in a distributed manner.

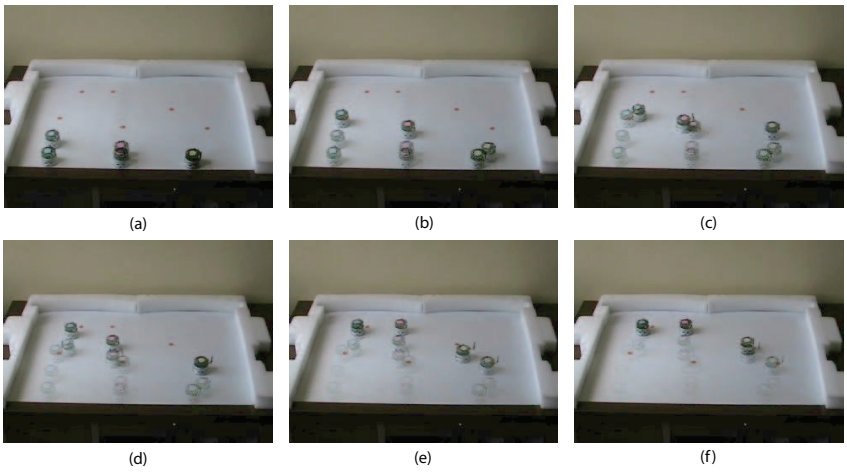


Figure 5.25: Scenario 1: the multi-robot case without failure

The failure of the third robot and the rest of the run are illustrated in Figure 5.26 (Scenario 2). In this scenario, the third fails after visiting its assigned task. The failure of the robot is forced by the human agent isolating the related robot. At the time of the failure, the other robots are busy with their own target visiting tasks. The failure of the third robot does not block the execution. Since the

allocations are performed incrementally in DEMiR-CF, the target that is assigned to the third robot in Scenario 1 is not allocated to the third robot in this situation because it fails immediately after achieving its first task. After the failure, the second robot selects this target as an available target among the other targets. There is no recovery in this scenario, but redundant allocation procedures are eliminated by DEMiR-CF.

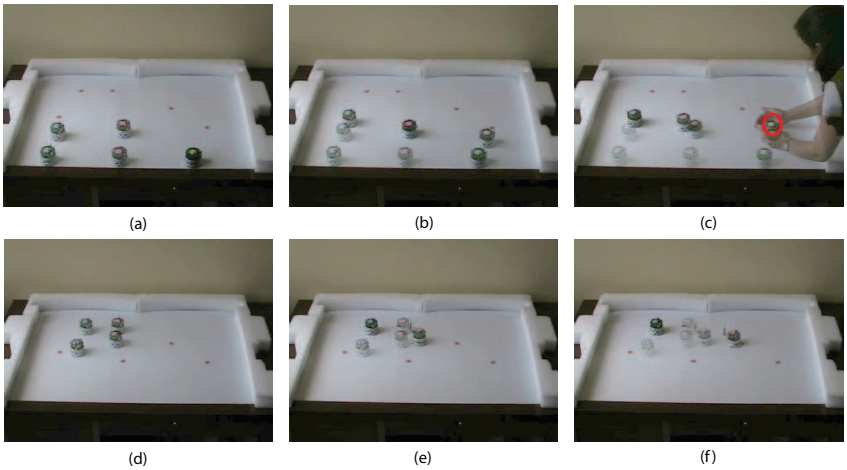


Figure 5.26: Scenario 2: the multi-robot case in which the third robot fails after completing its task

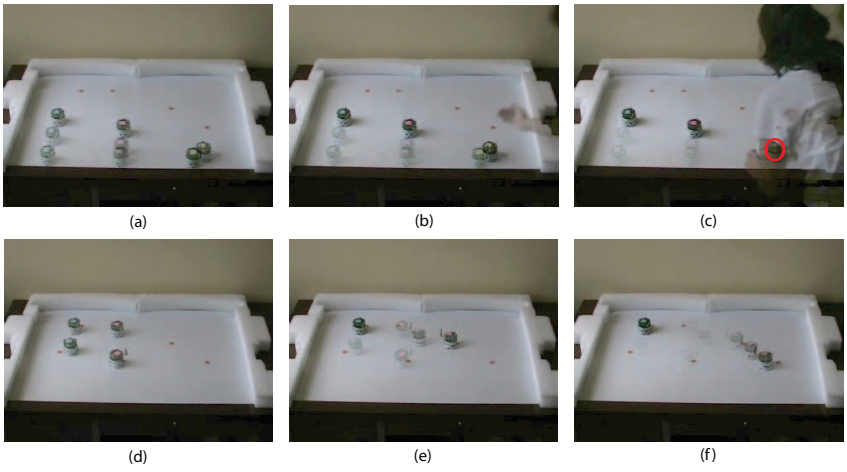


Figure 5.27: Scenario 3: the multi-robot case in which the third robot fails before completing its task

If the failure of the third occurs before completing its assigned task, then a recovery is needed (Figure 5.27). Using *Plan B Precaution Routines*, DEMiR-CF

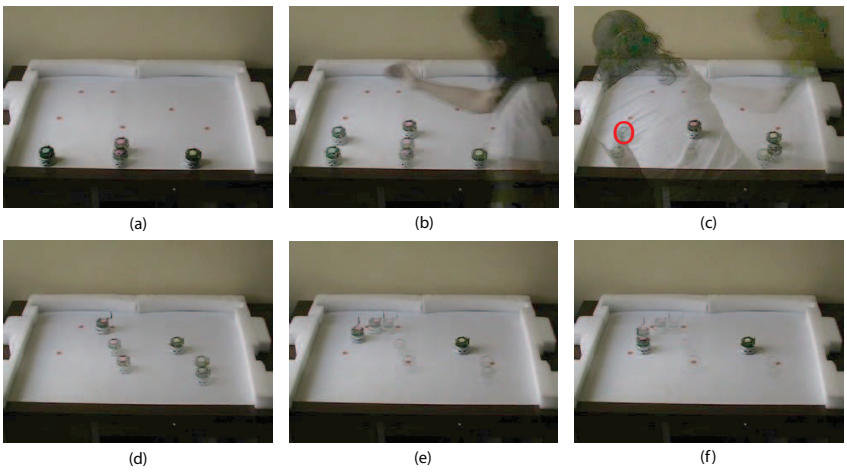


Figure 5.28: Scenario 4: the multi-robot case in which the first robot fails before after completing its task

can handle these types of contingency cases and the mission is successfully and efficiently accomplished. The failure cases for the first and the second robot are also illustrated in Figure 5.28 and Figure 5.29, respectively. All videos of these scenarios are available at the website-ref: SarielKh2Mov.

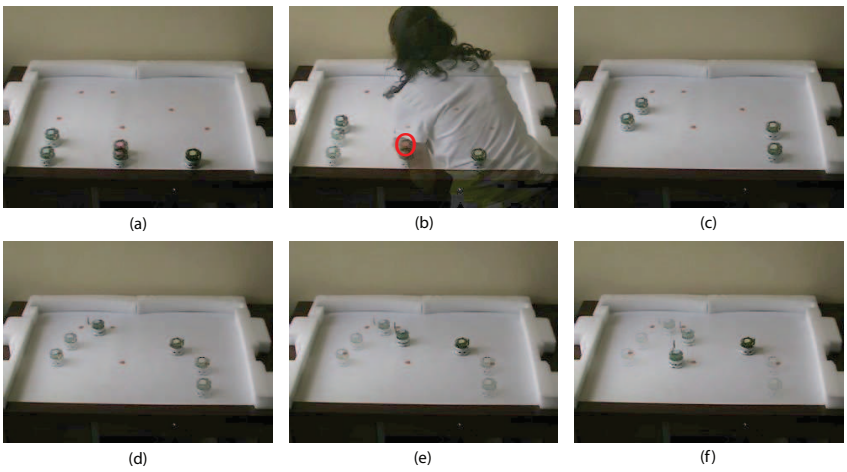


Figure 5.29: Scenario 5: the multi-robot case in which the second robot fails before completing its task

5.6. Summary and Discussion

In this chapter, we introduce the MTRP problem, the solution methods and our solution by using DEMiR-CF. In conclusion, the IP formulation generates optimum results for a given configuration of the robot and target locations

in small instances. However, these approaches may become impractical when the number of targets increases or the distances change frequently because of uncertain knowledge, the dynamism of the environment or the changing structure of the mission. Therefore, the IP approach may be too expensive when added to the path planning calculations for large target sets. As the results illustrate, allocating all targets and generating routes of robots from scratch may result in highly suboptimal solutions.

DEMiR-CF eliminates the redundant efforts by means of the incremental assignments based on the up-to-date situations of the environment for this domain. It can also handle the contingencies by *Plan B Precaution Routines*. Communication failures may sometimes prevent target allocations from being optimal. The framework can also detect these situations and maintains high solution quality by the dynamic task selection and task exchange scheme. As a final remark, as our experiments and given sample situations reveal, we argue that target allocation and route construction should be integrated for better results in this domain. This integration and incremental allocation is useful for eliminating redundant calculations in highly dynamic or unknown environments.

6. EMPIRICAL EVALUATION OF DEMiR-CF ON NAVY MISSIONS

In this chapter, we evaluate the performance of DEMiR-CF in Naval Mine CounterMeasure (MCM) missions and Naval Homeland Security Missions. Marine applications using AUVs (Autonomous Underwater Vehicle) involve challenges in addition to noisy communication, position uncertainty and the likelihood of failures. In particular, in undersea operations communication windows are restricted and bandwidth is limited. Consequently, coordination among agents is correspondingly more difficult. It is highly likely that the initial task assignments are subject to change during run time in these kinds of environments. DEMiR-CF on NAVY missions can ensure robust execution and efficient completion of missions against several different types of failures. The experiments are performed on the US Navy's Autonomous Littoral Warfare Systems Evaluator-Monte Carlo (ALWSE-MC) simulator against different contingencies that may arise at run time.

Empirical evaluations are performed on a cooperative NAVY mine clearance mission (Sariel et al., 2006b) and a cooperative NAVY homeland security mission (Sariel and Balch, 2006b).

6.1. Naval Mine Countermeasure Missions

Naval Mine CounterMeasures are actions taken to counter the effectiveness of underwater mines. MCM operations include finding and seizing mine stockpiles before they are deployed, sweeping desired operational areas, identifying mined areas to be avoided, and locating and neutralizing individual mines (Stack and Manning, 2004).

Our research is focused on the subset of MCMs that involve locating and mapping all individual mines in an operational area. In general, recognizing proud mines on the seafloor is not overly difficult; the difficulty arises with the abundance of non-mine objects on the sea floor that possess mine-like characteristics (e.g., geologic outcroppings, coral, man-made debris, etc.). This ample supply of false alarms has necessitated the following strategy typically employed by the Navy: detect and classify the mine-like objects (MLOs) with high-coverage rate

sensors (e.g., sidelooking sonar), employ advanced signal processing techniques for maximal false alarm reduction, then revisit the remaining MLOs with identification-quality assets (e.g., electro-optic sensors) to confirm them as mines or dismiss them as false alarms.

The reference mission in this research is to detect, classify, and identify underwater mines in a given operational area simulated in a PC-based software, ALWSE-MC (ALWSE), analysis package designed to simulate multiple autonomous vehicles performing missions in the littoral regions including mine reconnaissance, mapping, surveillance, and clearance. This mission employs two types of vehicles: unmanned underwater vehicles (UUVs) which are free swimming AUVs and possess large-footprint sensors (e.g., side-scan sonar) for detection and classification (D/C) of mines and sea-floor crawlers equipped with short-range, identification-quality sensors (e.g., camera). The crawlers have the ability to stop at an object and take a picture with a camera.

6.2. Applying DEMiR-CF to the Naval MCM

To apply DEMiR-CF framework to the NAVY MCM, the task types and the vehicle operations are determined and the general representation is adopted to be used in this domain. Different operations are needed for different underwater vehicles depending on their capabilities and the task types that they can execute.

6.2.1. Task Representation for The MCM

Our general task representation is capable of describing complex tasks with interdependencies. However, in this particular case study, tasks do not have interdependencies. Two types of tasks are defined for vehicles: "visit waypoint" (w) and "identify MLO" (t). The task representation includes the capabilities required for each type of task: $reqcap_w$ contains side-scan sonar and $reqcap_t$ contains cameras besides the standard capabilities of AUVs common in both types of vehicles. The coverage mission (M_C) contains a predefined number of known waypoints ($w_i \in M_C$, $0 < i \leq ||M_C||$) to be visited by all UUVs ($R_{UUV} \subset R$). One way of task representation is to directly assign a task for each waypoint. However, this representation has a drawback of high communication requirements for the efficient completion of the mission. Instead, we represent tasks as interest points of regions/search areas ($W_k = \cup w_i$, $\forall w_i$ is unvisited, and $W_k \subseteq M_C$). Therefore, both the allocation of the waypoints to the robots and the paths constructed to traverse these waypoints are determined online by negotiations. Negotiating the interest points (regions) instead of the individual waypoints

reduces the communication overhead. Regions determined by different UUVs may vary during runtime and may sometimes overlap. However, the uncertainty related to the region determination is within an acceptable range, especially when the cost is compared to the requirements of complete knowledge sharing by representing each waypoint as a task. Before defining the regions, the relative distance values, $reldist(r_j, w_i)$, are determined for each unvisited waypoint w_i using Equation 6.1, where function *dist* returns the Euclidean distance between points. r_k locations are the latest updated locations of the robots. If there is no known active robot assumed to be running properly, $reldist(r_j, w_i)$ is the value of the distance between the robot and the waypoint.

$$reldist(r_j, w_i) = dist(r_j, w_i) - min_{\forall k \neq j}(dist(r_k, w_i)),\ r_k \text{ is active} \tag{6.1}$$

Each robot defines its rough regions ($W_{jk},\ 1 \leq k \leq ||R_{UUV}||$). The number of regions equals the number of UUVs believed to be running properly. After sorting the $reldist(r_j, w_i)$ values of the unvisited waypoints in descending order as an array, the array is cut into subarrays which represent the regions. Each region contains approximately an equal number of waypoints. Each robot specifies the region of the highest interest as its "first" region. If the robots are closely located, the regions of highest interest may overlap. In this case, negotiations are needed to resolve conflicts and to assign only one robot for each region.

The identification mission (M_I) contains an unknown number of tasks for the MLO locations ($t_i \in M_I,\ 0 < i \leq ||M_I||$) to be visited by the crawlers. Therefore, the tasks in M_I are generated during runtime.

6.2.2. Exploration for Detection and Classification of MLO Locations

To begin the mission, the UUVs survey the operational area following waypoints determined *a priori*; however, corresponding regions containing waypoints may be reassigned by the negotiations among UUVs autonomously. After determining the regions, each UUV proposes an auction for the region of highest interest (interest point). After negotiations on several auctions, each UUV is assigned to the closest region (interest point). If more than one robot is almost the same distance from the interest point, the one with the smaller id number is assigned to the region. The other UUVs continue to offer auctions for the remaining regions. Allocations of the regions may also change during run time to maintain the solution quality. Whenever UUVs detect UUV failures or recoveries from failures, they change their region definitions accordingly and offer new auctions. After the region

assignments are completed, each robot visits waypoints in its region (W_j) in a sequence identified by an ordering from the smallest to the largest of their cost values, which are computed using the heuristic function given in Equation 6.2.

$$\begin{aligned} c(r_j, w_i) &= \alpha * dist(r_j, w_i) + (1-\alpha) * [dist(w_{b1}, w_{b2}) \\ &\quad -max(dist(w_i, w_{b1}), dist(w_i, w_{b2}))] \\ \{dist(w_{b1}, w_{b2}) &= max(dist(w_k, w_l)),\ w_{i,k,l,f1,f2} \in W_j\} \end{aligned} \tag{6.2}$$

This heuristic function considers boundary targets, w_{b1} and w_{b2} in W_j, the targets with the maximum distance value. The basic idea of this function is to forward the robot to one of these boundary targets since these targets determine the diameter of the region (W_j) and both of them should be visited. If the robot initially heads towards one of the boundary targets, the diameter (the longest path) can be traveled by visiting other targets along the path. The cost penalty applied to forward the robot to the boundary targets is limited to a small degree. By introducing a constant (α), this degree of direction can be adjusted. When α is assigned a value of 2/3, this heuristic function produces close to optimal results for multi-robot multi-target domains as explained in the previous chapter. If more than one pair of boundary targets exist, the pair which has a member the smallest distance from the UUV is selected.

As UUVs detect the MLOs on their way, they broadcast these estimated target positions to all AUVs (i.e., tasks for crawlers are generated online during execution). Then MLO information can propagate in bucket-brigade style to all other AUVs in the group that can possibly be reached. Periodic broadcasting of important information coming from either owned sensors or external agents is a way to handle communication range limitations.

6.2.3. Identification of Mine-like Objects

When the crawlers are informed about the MLO locations, they update their world knowledge and dynamically select the best MLO targets to visit and propose auctions. Therefore, they can switch among tasks when new tasks appear, if it is more profitable. It is also possible that a crawler may inadvertently discover a mine without being informed of its position by a UUV. In this case, the crawler identifies the target, adds it to its task list as an achieved task, and broadcasts achievement information to maintain the system consistency. Crawlers determine the bid values by using the MTRP cost evaluations to model the MLOs as MTRP targets.

In the identification task, when crawlers are within an area close to an MLO location, they begin keeping time while surveying the MLO location. Whenever the time limit is reached, they set the task status as achieved and broadcast this information. If a detection event occurs during this time period, the MLO location is considered to be an actual mine and the task is assumed to be achieved; otherwise, it is determined as a false alarm after deadline. In either case, the task is marked as achieved.

A conceptual flowchart summarizing operations of UUVs and crawlers, and the general operations implemented by both types of AUVs is given in Figure 6.1.

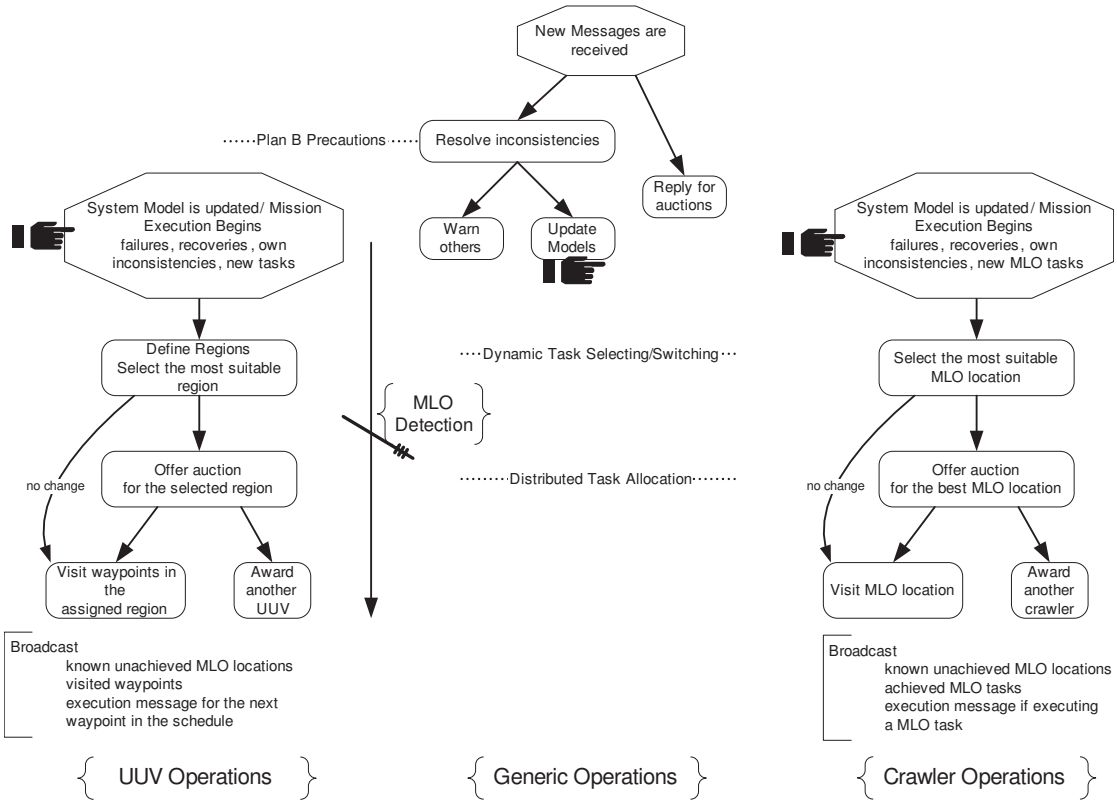


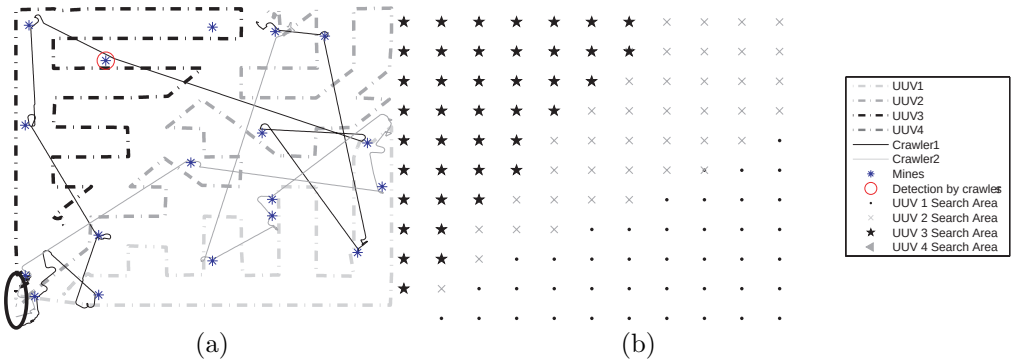
Figure 6.1: Conceptual flowchart related to the AUV operations

6.3. Experimental Results on the Naval MCM

The performance of our framework and the precaution routines is evaluated in ALWSE-MC. Three sample scenarios in the simulator are given to illustrate the performance for Naval MCM missions. UUVs are equipped with sensors capable of detecting mines within 30 feet from the skin of a target in the simulator. However, they are not able to correctly identify them. The crawlers are equipped

with cameras which can both detect and identify mines within 20 feet. None of the AUVs have predefined search patterns. UUVs have internal navigation errors, therefore, their estimated location values are different from actual locations in most cases. Two AUVs can communicate with each other whenever the receiver AUV is in the sender AUV's transmitter range, within its transmitter beam width, and the sender AUV is within the transmitter AUV's receiver beam width.

All UUVs and crawlers begin execution from a deployment area. There is no *a priori* information about mine locations. 121 waypoint locations (environment size: 200x200) are known but are not assigned initially. UUVs begin negotiations and divide the overall mission area into three (known number of UUVs) regions. Since they are within line of sight, they can communicate their location information. Therefore, initially defined regions are almost the same for all UUVs. Figure 6.2 illustrates a successful mission scenario with three UUVs and two crawlers. The legend for all simulation scenarios are also provided in the figure. Allocations of waypoints after negotiations can be seen in Figure 6.2(b). Since there are no failures, the waypoint assignments do not change during run time. However, the crawlers sometimes switch among tasks if they are not informed about tasks that are being executed and sometimes parallel executions occur. Whenever they are in communication range, they can resolve the conflicts efficiently by means of the precaution routines. As in Figure 6.2(a), the crawlers can also detect mines without being informed (red circled in the figure). The routes of the crawlers may seem somewhat random. However, it should be noted that the tasks related to the MLO locations appear during run time when they are discovered, and the communication range is limited.



(a) (b)

Figure 6.2: Scenario 1: (a) The UUVs cover the area; the crawlers visit MLO locations.

In the second scenario, one of the UUVs (UUV3) fails in the same setting of scenario 1 (Figure 6.3). Initially all UUVs begin execution (Figure 6.3(a)).

When UUV3 fails, the other UUVs take responsibility of all unvisited waypoints (The location of the failure is indicated with a red arrow in the figure.). Initial regions for all UUVs change after UUV3 fails (Figure 6.3(b)). The other UUVs revise their region definitions and, after negotiations, they share the full area as indicated in the figure. The visited waypoints are not in their region coverage. Due to the uncertainties, some waypoints may remain uncovered in the schedules (indicated with the red diamond in the figure). However, this uncertainty related problem is resolved by UUV2 and the mission is completed.

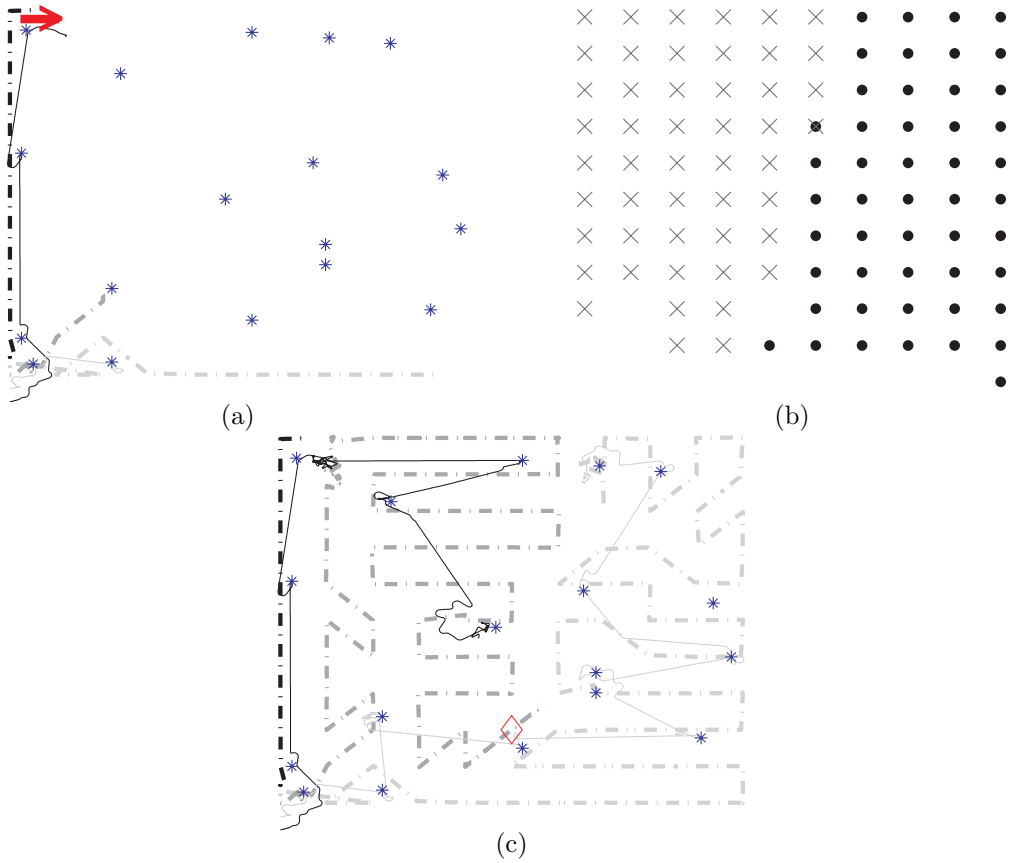


Figure 6.3: Scenario 2: UUV failure is handled by the other UUVs in the system

In the third scenario (Figure 6.4), UUV3 fails and the other UUVs detect the failure and they negotiate the remaining unvisited waypoints and new schedules are determined as in Figure 6.4(b). While these UUVs execute their tasks, another UUV (4) is released from the deployment area (Figure 6.4(c)). Detecting the arrival of a new UUV, the other UUVs change their region definitions accordingly (Figure 6.4(d)) and offer auctions for these areas. Initially UUV4 is not informed about the visited waypoints and defines its regions with this incomplete knowledge. After negotiations, the regions are assigned and the schedules are formed. Entering into the the communication

range, UUV4 redefines its regions by considering incoming information for the visited waypoints. Videos of these scenarios are available at the website-ref: SarielMCMMov.

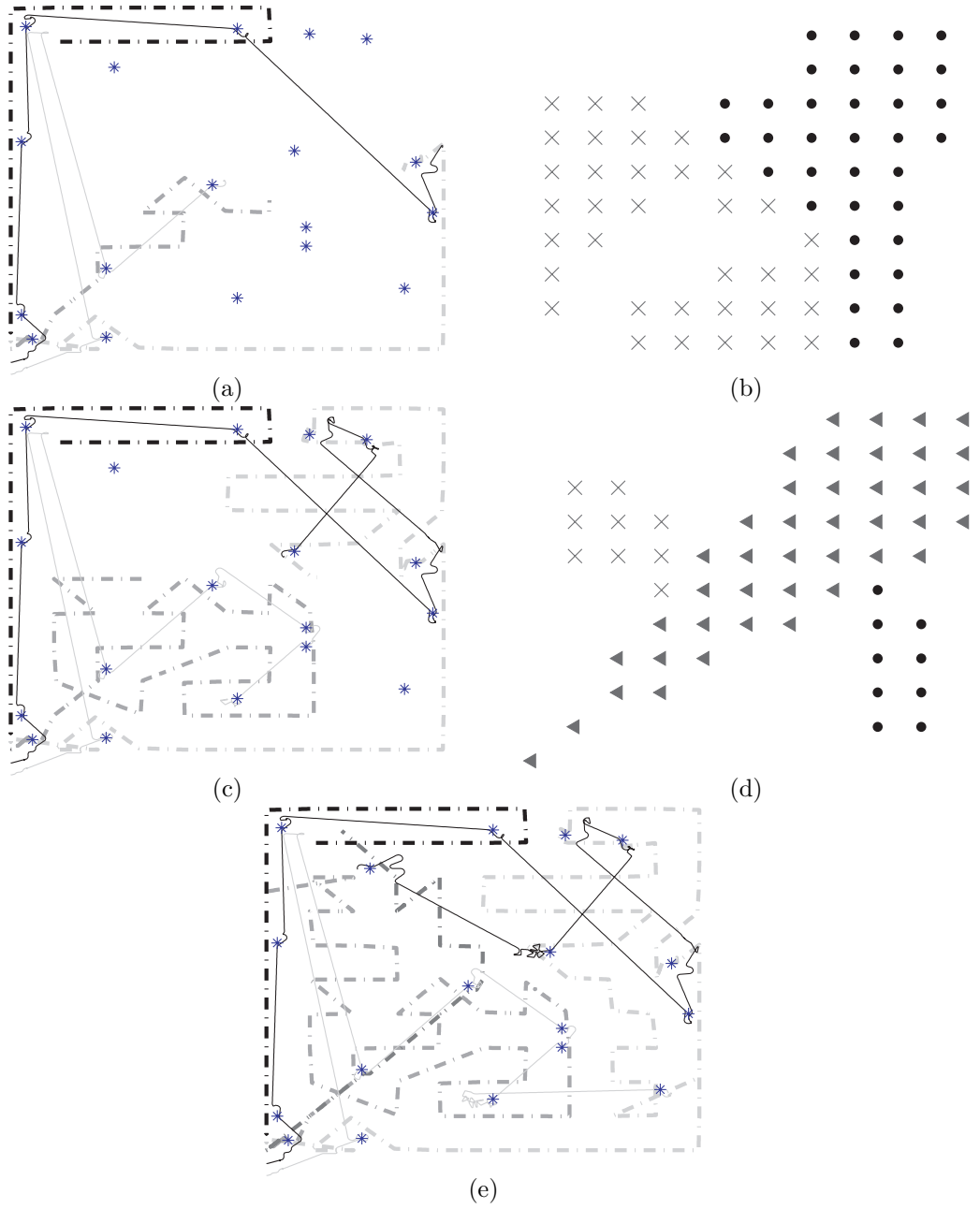


Figure 6.4: Scenario 3: UUV failure is handled and new robot arrival is also used to improve the system utility in a distributed manner

In the same settings, another experiment is conducted to evaluate the effect of message loss rate on the mission completion success. Table 6.1 illustrates the results ($\mu \mid \sigma$) averaged over 10 runs. When the message loss rate is different from 0, as expected, the mission completion time performance of the system

degrades but linearly. It should be noted that even for a rate of 0.75, the overall mission (M_C and M_I) by the final identification of the mines is completed. The average of the first visit times of the waypoints increases linearly due to the delays occurring by redundant visits of the targets. The number of waypoint (w) visits increases for high message loss rates. When the message loss rate is 1, there is no communication among AUVs and they cannot correctly reason about the region portions. Therefore, each UUV searches the full area completely. The crawlers detect and identify 12.8% of mines by their local detection in a small area (MLO target information can not be communicated in this case). Since the identification mission is not complete, the overall mission is not completed. This table illustrates the performance of our framework against message losses. As a final remark, auction generation and clearing in an environment with communication delays desires special attention. Especially auction deadlines should be determined by considering communication delays which may vary during the run. *Plan B Precautions* could resolve these kinds of problems. Precautions for delayed messages on out-of-date situations prevent the system from getting stuck in further inconsistencies and deadlocks.

Table 6.1: The performance results for different message loss rates given with the average and standard deviation values (μ, σ)

Mssg Loss Rate	0		0.25		0.5		0.75		1	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
M_C Comp. (%)	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
M_I Comp. (%)	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	12.8	4.1
M_C Comp. time	3349.4	60.5	3683.2	167.1	4909.0	430.1	5141.2	938.1	6304.2	139.0
M_I Comp. time	2852.8	35.3	3227.6	205.3	4205.0	836.9	5021.2	692.7	N/A	N/A
(w) first visit	1380.1	6.1	1390.0	16.3	1922.0	92.8	2256.6	334.5	2936.0	104.5
(w) #of visits	1.0	0.0	1.0	0.0	1.01	0.01	1.09	0.04	3.0	0.0

6.4. A NAVY Homeland Security Application

We have evaluated DEMiR-CF in ALWSE. In this experiment, the mission consists of online tasks whose generation times are not known in advance by the robots (AUVs). The overall mission is searching a predefined area in order to protect the deployment ship from any hostile intents.

The initial mission graph for the extended MCM mission is given in Figure 6.5. Initially the mission consists of only the Search Task. Although $reqno = 1$ for this task, since there are no other tasks and the robots have enough fuel capacities, they execute the task as a coalition and divide the area to be searched. The Search Task execution with three robots and the corresponding search areas are illustrated in Figure 6.6. The overall area is divided into regions related to

the generated task instances (Figure 6.6(a)). The robots patrol the subareas which are determined after the negotiations (Figure 6.6(b)). Therefore, although there is only one task on the higher level, the robots create instances of the Search Task (Search 1-3) as if each instance is another separate task. If there are no hostile intentions, the robots only search the area.

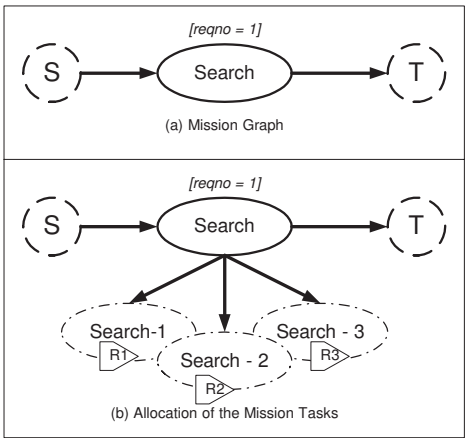
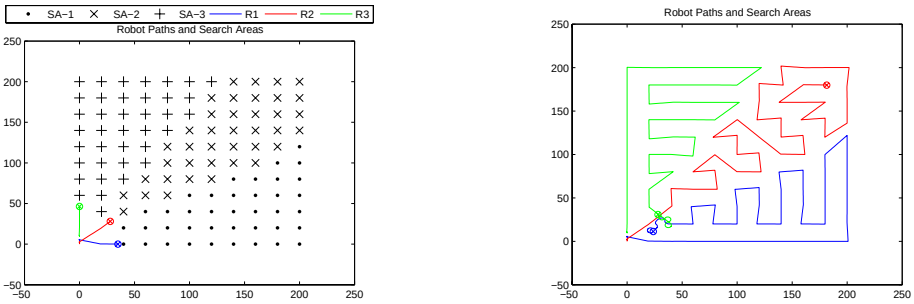


Figure 6.5: Initial mission graph consists of only the search task



(a) (b)

Figure 6.6: Robots patrol the area in the corresponding regions

Whenever a hostile diver is detected by the robots, a related interception task is generated. The scenario is illustrated in Figure 6.7. The robots begin searching the area (Figure 6.7(a)). R2 recognizes the hostile intent (Figure 6.7(b)). After detection, the hostile vehicle attacks R2. R2 returns to the deployment ship. R1 takes control of the intercept task. The hostile intention disappears (Figure 6.7(c)). R1 and R3 continue searching the area (Figure 6.7(d)).

The updated mission graph for the sample scenario is illustrated in Figure 6.8. The hostile diver may be destructive by using missiles. Therefore, task execution

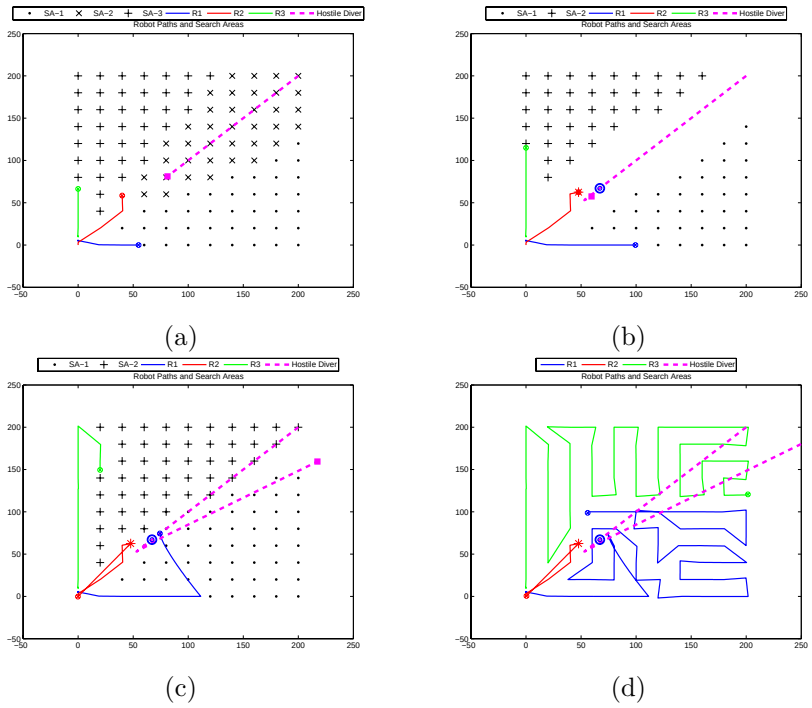


Figure 6.7: A sample execution trace under highly dynamic task situations in which failures occur after shots by the hostile diver

may need to be preempted and the task execution authority is exchanged during run time. The robots may need to generate local tasks (e.g., Repair/Refuel Task, which is generated by R2 after being shot by the hostile diver unexpectedly) as in Figure 6.8(d).Therefore, the mission graphs may be different for the robots even when they work cooperatively (Figure 6.8(c-d)). In Figure 6.8(c), although executing the Intercept Task, R1 can make a coalition commitment assuming it will succeed in a predefined time period (described as TBD). At this time R2 cannot make any coalition commitment for the search task because its future operations depend on its recovery.

Cost evaluation for the tasks are computed by considering the task facilitating composite (multi) objective missions. While the robots try to optimize the fuel levels for the search task, the intercept task requires immediate response and time minimization. Therefore, different cost evaluations are carried out for different tasks. We provide the cost evaluations for the task types used in the experiments in Table 6.2. Cost evaluation for the search task is implemented by first dividing the search area into regions (with corresponding interest points) and comparing the distance values for these interest points. The same cost evaluation used in the MCM mission is adopted for the search task. For the intercept task, the expected time to achieve (intercept the diver) the task is taken as the cost value.

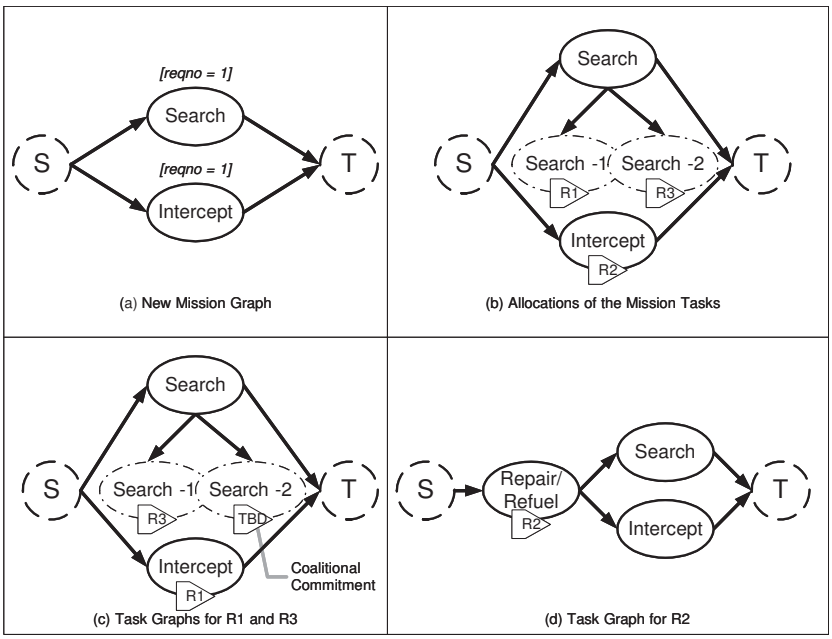


Figure 6.8: Mission graph and allocations evolving through time accordingly

Actions taken to execute the tasks are defined before mission execution. In our

Table 6.2: The cost evaluations for the application domain

Task Type	Cost Function	Taken Action
Search Task	Distance to the region interest points as in MCM Mission	In depth analysis is needed. Standard auction is applied.
Intercept Task	Expected time to achieve the task: $t_E = E[dist(r_j, t_i)]/E[speed\_diff(r_j, t_i)]$ where $speed\_diff$ function returns the estimated speed difference of the vehicles	Immediate response is needed. One step auction or direct execution is applied.

approach, the auction announcements are used both to maintain the models of the other robots in the system and to announce clues for the intentions. Emergency tasks (e.g., Intercept Task) require immediate action. We do not suggest the standard auction steps for these types of tasks. Instead, either a one-step auction is performed or the task is directly executed, which is the approach adopted in the experiments. However, in this case parallel executions may occur and should be resolved. This facility is provided in our framework by the precaution routines. We allow the parallel executions to handle the emergencies to be resolved when recognized. The intercept task is assumed to be achieved whenever the hostile threat is believed to have disappeared. In a sample scenario for a limited communication range, the parallel executions arise for the emergency tasks such as the intercept task as in Figure 6.9. However, these inconsistencies

are resolved by the activation of the corresponding precaution routines whenever the robots enter into the communication range. In this scenario, R3 switches to the search task after detecting the parallel execution. R1 continues to execute the intercept task.

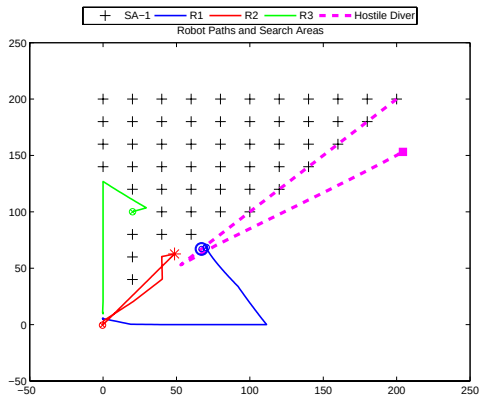


Figure 6.9: Under limited communication ranges, parallel executions may occur and are resolved when detected

6.5. Summary and Discussion

We have implemented and tested DEMiR-CF in the NAVY domains, where the environments are highly dynamic. The rough schedule generation scheme implemented by each under water vehicle to define regions offers an efficient way of considering the problem from a global perspective. The incremental task assignment approach has also proved to be successful in handling the environmental dynamism.

As the experiments we have performed illustrate, DEMiR-CF ensures robust execution and efficient completion of missions against several different types of failures. Results for MCM missions are promising in the sense of mission completion, and AUV paths are close to optimal in the presence of uncertainties. Evaluations also reveal the high performance of DEMiR-CF on online task and situation handling. Since the framework is a single-item auction method, it can be used for the environments with limited, delayed or unreliable communication.

It should be noted that the selected application domain, objectives and limitations are similar to the Search and Rescue (SR) domain. Therefore, we believe this research can also be useful for similar application domains such as SR.

7. EMPIRICAL EVALUATION OF DEMiR-CF ON RESOURCE CONSTRAINED AND INTERRELATED TASKS

In this chapter, we investigate the performance of DEMiR-CF on complex missions with resource constrained and interrelated tasks. Different from previous chapters, the robots take part in more complex tasks where they interact with the objects in the environment (Sariel et al., 2007a). The objective is not only optimizing cost functions but also obeying rules and resolving constraints on task execution during runtime. When the tasks of a mission are interrelated and subject to several resource constraints, more efforts are needed to coordinate robots towards achieving the mission than during independent tasks.

As in the previous experimental evaluations, the DEMiR-CF framework is evaluated for complex domains. The incremental task selection and allocation mechanisms of DEMiR-CF also eliminate redundant considerations in this domain. The base mechanisms of DEMiR-CF are used for designing the solution. Rough schedule formation and cost evaluations are designed according to the complex mission requirements.

7.1. Complex Multi-Robot Mission Problem Statement

The multi-robot task allocation problem is better viewed as a scheduling problem if there are interrelations among tasks, suggesting the use of Operation Research methods. However, when the problem solving time is limited and/or reallocations are frequently required at runtime, Operation Research (OR) solutions such as Branch and Bound (Brucker et al., 1998) or Integer Programming methods may not be directly applicable. In this chapter, our focus is on complex missions (project tasks) with interrelated tasks whose requirements on task execution may vary. These interrelations may correspond to shared resources, producer/consumer, simultaneity and task-subtask dependencies (Ossowski, 1999). The Pick-Up/Delivery domain tasks can be classified in this class because of the producer/consumer type of dependency relation for the pick-up and delivery tasks. More complicated interrelations may be placed in mission representations. Simultaneous execution requirements imply tightly coupled task execution where the actions implemented by each robot are highly dependent on

the actions of others. Furthermore, a group of robots executing a task may be either homogeneous or heterogeneous. The heterogeneity may be in the possessed capabilities or in the task execution performance. For example, robots may have the same equipments capable of achieving all the tasks of the mission but may differ in abilities such as speed. According to the classification of multi-agent organizations given in Horling and Lesser (2005), coalitions (agent groups) are formed to perform tasks in cooperation. From our perspective, coalitions are suitable for meeting the simultaneous resource requirements of executing tasks with a subteam of robots.

7.2. Applying DEMiR-CF to Complex Missions

The dynamic and incremental task selection, distributed allocation and contingency handling mechanisms of DEMiR-CF are used in the design of the solution for complex mission achievement. This domain forms a platform to apply the full functionality of DEMiR-CF.

Although the base mechanisms are the same, the rough schedule generation scheme and the cost functions are designed accordingly to meet the interrelation and resource constraints.

7.2.1. Dynamic Priority-based Task Selection Scheme and Online Scheduling

As the core principle of DEMiR-CF, the robots make instantaneous decisions (from their local perspectives) which are both precedence and resource feasible in the context of the global-time extended view of the problem. While the completion of the mission is the highest priority objective, performance related objectives can additionally be targeted. Each robot initially forms a rough schedule of its activities for an overall time extended resolution of the mission. Since these schedules are highly probable to change in dynamic environments and the robots also have the real-time burdens of path planning, mapping etc., the rough schedules formed are tentative and constructed by computationally cheap methods (explained in subsection 7.2.3.). Therefore, the robots in our framework come up with their rough schedules and refine their plans during actual fast execution when information available in the current context enables them to make specific, detailed decisions.

Instead of scheduling all tasks in one step, we propose a Dynamic Priority-based Task Selection Scheme (DPTSS) to allocate tasks to robots incrementally,

considering the global solution quality. The main objective of the proposed scheme is the incremental allocation of tasks by taking into account the precedence and resource constraints whenever a new task needs to be assigned, instead of scheduling all tasks from scratch.

The $CTSP$, introduced earlier, is an optimization problem as in ScP and it is desirable to find a solution by considering the problem from a global perspective. Therefore, the instantaneous task selection scheme needs to be strengthened by considering the problem as a whole, with the designed cost evaluation functions. Depending on the objective function, either priorities or penalties can be applied to find an efficient solution, ensuring a time-extended view of the problem.

Each robot r_j generates its rough schedule as a dynamic priority queue similar to runqueues by considering its critical task list (L_{Cj}), the eligible task set (T_{Ej}), the conjunctive arcs (if any) and the requirements. If there are no new online tasks or invalidations, the order of the tasks which are connected by the conjunctive arcs remains the same in the priority queue, even though there may be additional intermediate entries into the queue at runtime.

The critical tasks may be determined by either negotiations or beliefs. To eliminate intractable communication overhead, we use a rough belief update approach to form the critical tasks. Each critical task is assigned a probability value to indicate its criticality. Critical task information is used for determining the task requirements such as power, fuel etc.

Algorithm 7 GeneratePriorityList for robot r_j

input: Eligible task set (T_{Ej}), active task set (T_{Aj})
output: Topologically ordered and prioritized schedule list: S_{Rj}

$S_{Rj} = \phi$, $S_{Temp} = \phi$
$S_{Temp} = DFS(T_{E_j})$ /*List generated by a depth-first search, the tasks are ordered by ascending order of estimated task completion times*/
for all $t_i \in S_{Temp}$ **do**
  if $t_i \in T_{Aj}$ **then**
    insert t_i in S_{Rj} as ordered by the cost value and the precedence
  else
    insert t_i to the front of S_{Rj}
  end if
end for

The rough schedule of a robot constitutes a topological order of the directed

acyclic graph of the eligible mission tasks. While generating the rough schedules, both precedence constraints and cost values are considered. Basically each rough schedule is a priority list (T_o, topological order) determined by Algorithm 7. While forming the topologically ordered prioritized schedule list, a depth first search (DFS) is performed to topologically order the tasks by using the estimated task completion times. Next, the tasks are inserted into the list according to their completion times. If a task is an active task, its priority key is computed as a combination of the precedence and the cost value. Tasks with equal precedence are ordered according to their cost values.

The rough schedule of a robot is generated by execution of Algorithm 8. $curcs_j$ represents the remaining capacity of robot r_j, and $reqcs(t_i)$ represents the required capacity for task t_i in terms of the consumable resources (e.g., fuel).

Algorithm 8 GenerateRoughSchedule for robot r_j

input: Eligible task set (T_{Ej}), active task set (T_{Aj}), critical task list (L_{Cj}), remaining capacity ($curcs_j$) of robot r_j
output: Rough schedule (S_{Rj}) of tasks, the top most suitable active task t_s

$t_s = \phi$; $R = curcs_j$; $achievable = true$;
S_{Rj} = GeneratePriorityList(T_{Ej}, T_{Aj})
/*Determines if the mission is achievable*/
for each $t_i \in L_{Cj}$ **do**
  $R = R - reqcs(t_i)$
  if $R < 0$ **then**
    $achievable = false$
    $R = curcs_j$
    break
  end if
end for
if $S_{Rj} \neq \phi$ and $(top(S_{Rj}) \in L_{Cj} \parallel R - reqcs(top(S_{Rj})) \geq 0)$ **then**
  $t_s = top(S_{Rj})$
end if

In the rough schedule generation algorithm, while forming the rough schedule, the remaining capacity of the robot is also monitored. If the capacity of the robot is not sufficient for executing all of its critical tasks and the mission is believed to be unachievable as a result, then the robot may select an active task to execute even if it is not a critical task for the robot in case new robots can be deployed. However, if the mission is believed to be achievable, the robot may select to stay idle until its critical tasks become active. This selection is done after forming the rough schedule. The active task on top of the rough schedule that can be executable is the most suitable task to be executed for the robot. Sometimes the

rough schedule of the robot may be empty. In this case, the robot selects to stay idle as determined in the DPTSS algorithm.

7.2.1.1. DPTSS Algorithm

In our incremental allocation approach, the fundamental decision that each robot must make is the selection of the most suitable task from the active task set (T_A) by considering the eligible task set (T_E). Algorithm 9 presents the DPTSS in which a rough schedule is generated before making a decision. The four different decisions made by robots after performing the DPTSS are:

- continue to execute the current task (if any),
- join a coalition,
- form a new coalition to perform an available task or
- stay idle.

Algorithm 9 DPTSS Algorithm for robot r_j

input: Mission (M) task descriptions
output: Action to be performed depending on the selected task

Determine the T_{Ej}, $T_{Aj} \subseteq T_{Ej}$ and $L_{Cj} \subseteq T_{Ej}$
/*GenerateListOfCriticalTasks*/
$L_{Cj} = \phi$
for each $t_i \in T_{Ej}$ **do**
  $P_{ct}(t_i) = \frac{reqno}{\#ofsuitablerobots}$
  if $P_{ct}(t_i) \geq 0.5$ **then**
    Insert t_i in L_{Cj} prioritized by the $P_{ct}(t_i)$
  end if
end for

$[S_{Rj}, t_s]$ = GenerateRoughSchedule (T_{Ej}, T_{Aj}, L_{Cj}, $curcs_j$)

if $t_s \neq \phi$ **then**
  if t_s is the current task **then**
    Continue with the current execution
  else
    Offer an auction to form a new coalition or directly begin execution
  end if
else
  if $t_s \in T_{ie}$ and it is profitable to join the coalition **then**
    Join the coalition
  else
    Stay idle
  end if
end if

The dynamic task switching scheme is used by robots to dynamically switch between tasks if updates in the world knowledge compel. Therefore, issues related to both online scheduling and scheduling under uncertainty are addressed.

The DPTSS process is repeated whenever a robot completes its current task execution or detects a change in its world knowledge. Instead of regenerating the rough schedule at each call of the DPTSS, the rough schedule may be repaired whenever it is desirable.

7.2.2. Distributed Task Allocation Scheme

Standard auction procedures of our distributed allocation procedures are applied. For task executions with multiple robot requirements (for which $reqno > 1$), coalitions are formed. For such a task, a coalition leader and the required number of coalition members are selected. These roles are assigned to ensure synchronous task execution among coalition members although tight coordination routines are beyond the scope of this research. Additionally, precaution routines are added to check validity, consistency and coherence in these negotiation steps. Each robot intending to execute a task announces an auction after determining its rough schedule and performing the DPTSS. Basically, auction announcements are ways to illustrate intentions to execute tasks for which $reqno = 1$ or to select members of coalitions to execute tasks for which $reqno > 1$. The succeeding steps of the distributed task allocation scheme are applied as in the general design of the framework.

7.2.3. Cost/Bid Evaluation and Tie Breaking Rules

The cost evaluation has a tremendous impact on the solution quality. Each task type as a part of the mission requires a different cost evaluation to efficiently solve the problem. As an example, while estimating the cost value for picking up an item, the distance between the robot location and the estimated location of the item may be considered. However, to form a globally efficient allocation, the locations of the other items must be considered as well. We have validated this statement in the MTRP domain. Cost evaluation is performed by using the corresponding functions given in Table 7.1. If a robot is executing a task when it receives an auction message, it sends the bid value by considering the final destination of the current task as the location of itself.

Another important design criteria is determining the bid values to be sent to the

Table 7.1: Cost evaluations for the tasks

Task Type(s)	Cost Function
Locate/Pick-up	Estimated time to reach at the location of the object.
Deliver/Push	Estimated time to carry/push the object from the initial location to the final destination.
Clean	Estimated time to cover the whole environment.

other robots. Furthermore, robots may need to cancel or postpone their offers for auctions if there is synergy between tasks when announcements of offers from others are received. Intuitively, robots only send bid values for the active tasks for themselves.

A common situation appears when the auctions are offered at the same time by different robots either for the same task or for different tasks. To decide on the selection of the robots to execute specific tasks in a distributed setting is a challenging issue. In our approach, if there are conflicting auctions for the same task, only the robot offering an auction with the smallest cost value continues with the auction negotiation process. In the case of the conflicting auctions for different tasks, a resource-based rule (related to the *reqno* of the tasks) borrowed from Operations Research, Greatest Resource Requirements (GRR), is used (Brucker and Knust, 2006).

7.2.4. Analysis of the Approach

The approach we have proposed for this particular problem domain offers a polynomial time solution. The critical task list generation takes $O(nlog(n))$ time for all n number of tasks. Achievability of the mission is determined in $O(n)$. The complexity of the rough schedule generation is bounded by the topological list generation algorithm which is in the order of $O(n+e)$ (where e is the number of conjunctive arcs, i.e., hard dependencies). Therefore, the total complexity becomes $O(n(e+nlog(n)))$. If $(e << n)$, the complexity of the proposed approach reduces to $O(n^2log(n))$.

7.3. Complex Mission Experiments

We have conducted real-world experiments and real-time dynamic simulation experiments on WEBOTS, the professional mobile robot simulation software. It contains a rapid prototyping tool to create 3D virtual worlds with robots and objects possessing physics properties (WEBOTS).

In our simulation experiments, each environment is represented as a 5m by 5m 3D virtual world where 70mm-size simulated Khepera II robots and objects are located. The environments are randomly generated VRML files containing the robots and the objects. Each Khepera II robot is mainly equipped with a 25MHz MC68331 micro-controller, 512K Flash and 512K RAM memories and 8 infra-red sensors with a limited obstacle detection range as simulated in WEBOTS. Communication is achieved through wireless links in both simulations and in real-world experiments. Real Kheperas have standard radio turrets mounted on them to communicate through the selected radio frequency. Emitters in the simulator are configured to have a baud rate of 9600 and a buffer size of 1024B as in the receiver modules.

A multi-process controller implementation is performed on the real robots to achieve the proposed system. Different modules on the task allocation layer are integrated with the lower-level Sensory Interface, Motor Interface, Motion Model and Mapping modules in the multi-threaded structure.

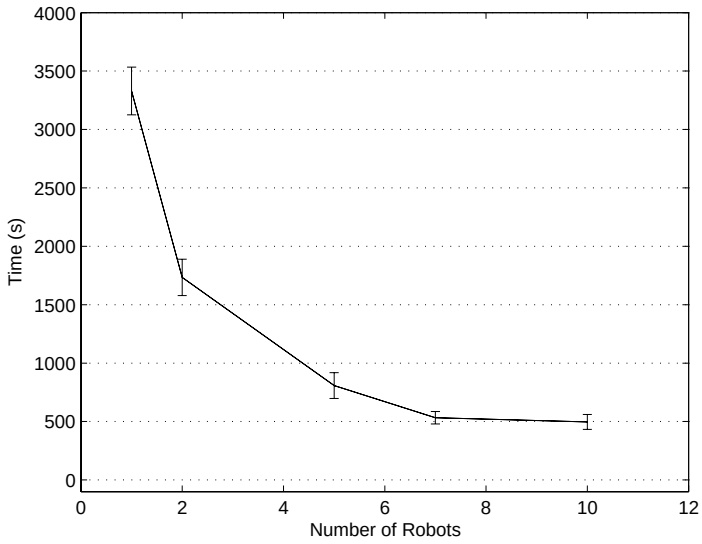


Figure 7.1: Mission completion time(s) for the pick-up/delivery mission, with task number fixed at 20

The first set of experiments is targeted to analyze the scalability of the proposed approach on the pick-up/delivery mission in which the tasks are interrelated by picking up and delivery constraints. All picked up items are collected

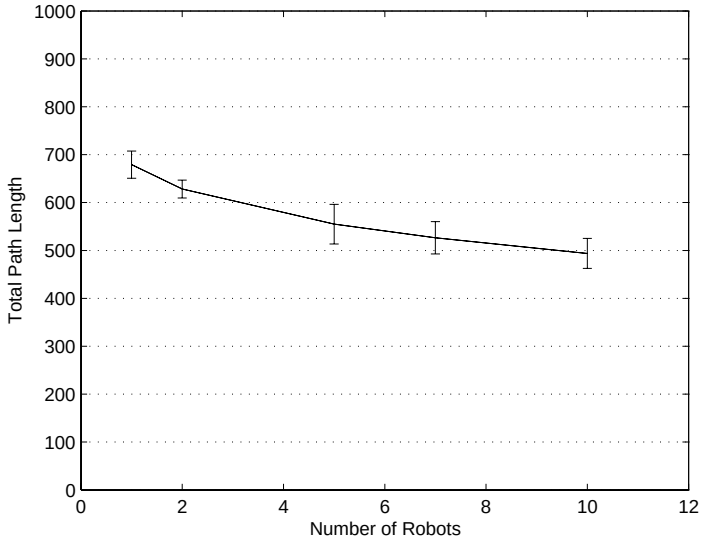


Figure 7.2: Total path length traversed by the robots(mm) for the pick-up/delivery mission, with task number fixed at 20

in the center of the environment. In the fully flexible version, while there are precedence constraints between pick-up and delivery tasks, there are no interrelations between pick-up tasks for different items. The items are distributed in the environment at fixed locations for each run. The robot locations are randomly determined. Figure 7.1 illustrates the mission completion times for sets with different numbers of robots. As expected from this approach, the time to complete the overall mission is greatly reduced with increasing numbers of robots, validating the scalability of the approach. Figure 7.2 plots the total path length traversed by the robots. Since the items are delivered to the center of the environment, an extreme variation for the expected utility in the total path length traversed by robots is not expected, as illustrated in the graph.

A sample scenario for a complex mission which includes tasks for pushing boxes and picking-up and delivering items to a desired location is given in Figure 7.3 with five participating robots. Locations of the objects are indicated with the red arrows in the figure. In the first scenario, two items are picked up and delivered to the destinations by the robots possessing grippers. The two robots simultaneously and independently push the two boxes. One of the robots stays idle during mission execution. In the second scenario, since the minimum required number of robots to push one of the boxes is two, the two robots form a coalition and push the heavy box synchronously.

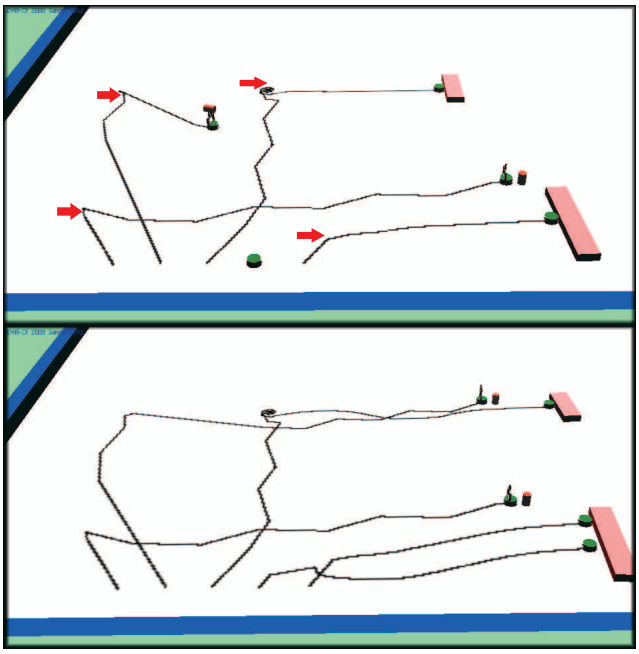


Figure 7.3: Scenario 1 and 2: Robots push and carry boxes to a final destination. Some tasks may require simultaneous and tightly coordinated task execution.

Another complex mission allocation scenario which includes tasks for pushing a box, carrying a cylindrical object to a final destination and then inspecting the environment is implemented by three Khepera II robots and the execution scenario is illustrated in Figure 7.4. There are interrelations between push, carry and inspection tasks respectively as in the graph depicted in Figure 7.5. While the objects can only be carried by the robots with grippers, the inspection task requires possessing a camera. The box can be pushed by any of the three robots. However, due to the cost evaluations and the critical task list consideration, allocations are implemented accordingly. Robots obey the interrelation constraints and each robot takes part in a suitable task execution for itself in which the decision is made in a completely distributed manner. The video of this scenario is available at the website-ref: SarielKh2Mov.

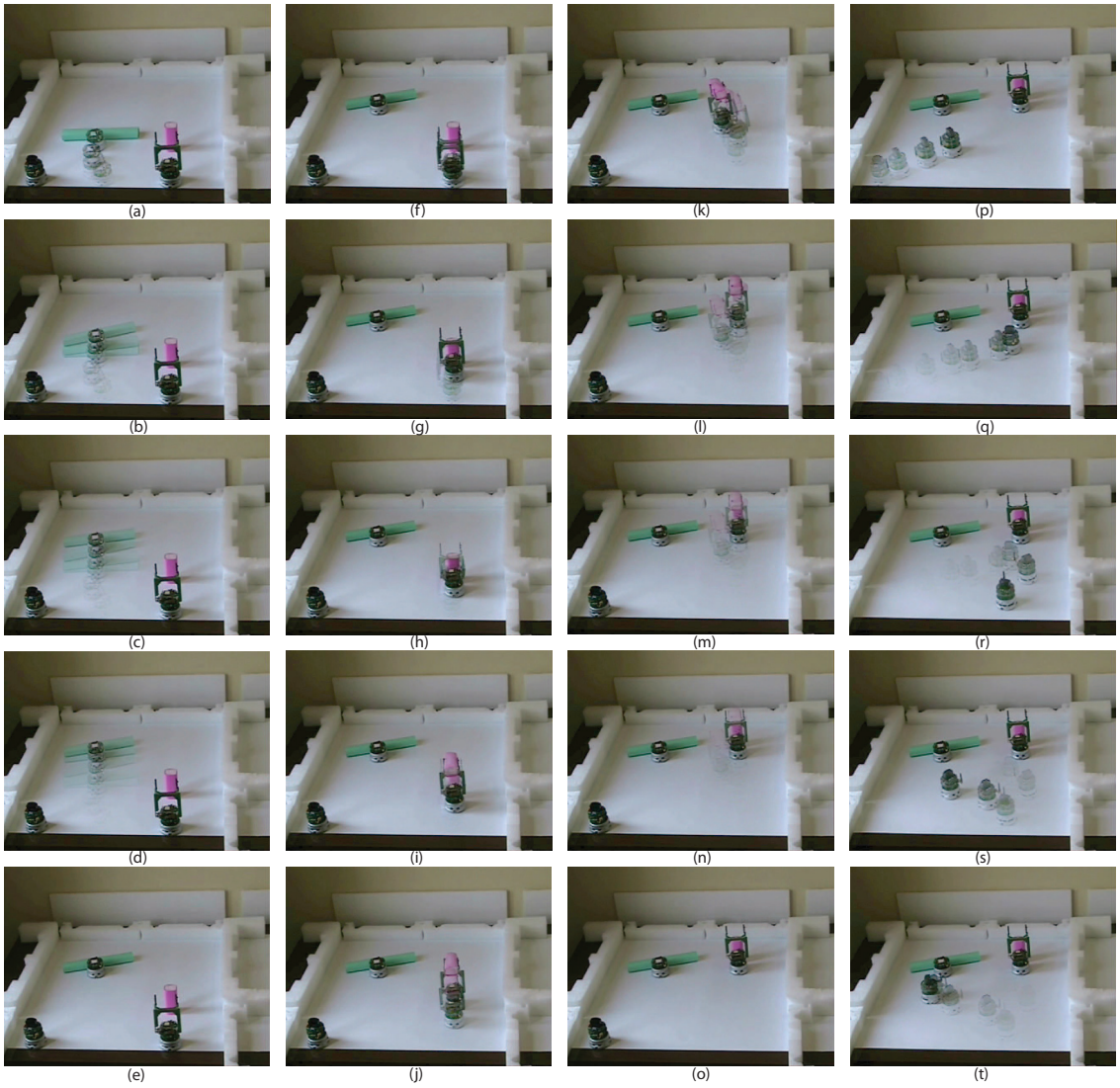


Figure 7.4: Khepera II robots achieve the overall complex mission of pushing/carrying the objects to the final destinations and inspecting the area.

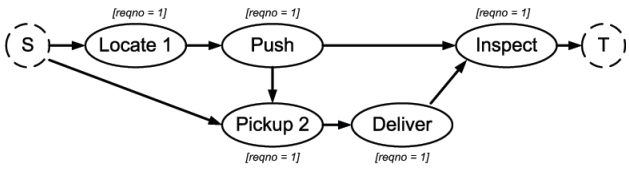


Figure 7.5: Real scenario mission graph with interrelated tasks

7.4. Summary and Discussion

We have described the details of the implementation of DEMiR-CF on interrelated, resource constrained tasks of a mission. The full functionality of the proposed approach is validated on multi-robot complex mission execution. The scalability of the approach is validated through experiments. Real robot implementations are given and scenarios with interrelated tasks for which robots interact with objects in the environment are presented. The rough schedule generation scheme by calculating a topologically ordered task list is shown to be an efficient approach that can be applied to robots with limited computational capabilities. The $CTSP$ is solved by each robot implementing incremental task selection, distributed task allocation and contingency handling mechanisms by DEMiR-CF to achieve the overall $CMAP$ for complex missions.

8. DISCUSSION AND CONCLUSION

DEMiR-CF is a multi-robot cooperation framework to solve the Cooperative Mission Achievement Problem ($CMAP$). The $CMAP$ asks for achieving a complex mission of either independent or interrelated tasks with multi-robot requirements for their execution. The problem should be solved in a cost efficient manner while simultaneously handling unplanned events and contingencies. The Coordinated Task Selection Problem ($CTSP$) that we formulate is a task selection problem for a robot participating in a multi-robot team running to solve the $CMAP$. Each robot involved in the $CMAP$ proceeds by generating incremental solutions to the $CTSP$ at runtime until the mission is completed.

DEMiR-CF is designed for complex missions including interrelated tasks that require diverse (heterogeneous) capabilities and simultaneous execution. The framework combines *The Dynamic Priority Based Task Selection Scheme*, *Distributed Task Allocation* and *Coalition Maintenance Schemes* as cooperation components and *Plan B Precaution Routines*. These components are integrated into a single framework to provide an overall system that finds efficient solutions for real-time task execution.

The Dynamic Priority Based Task Selection Scheme forms the basis of incremental task selection for each robot. Each robot maintains a rough schedule of future tasks before deciding on an active task to execute. Rough schedules ensure ways to consider the problem as a whole although the decisions are made locally by each robot. *The Distributed Task Allocation Scheme* ensures the selection of appropriate robots for corresponding tasks in a decentralized way. *Coalition Maintenance and The Dynamic Task Switching Scheme* ensures dynamic reconfiguration of allocations. *Plan B Precaution Routines* ensure system consistency, coherence and robustness in a decentralized way. Robots model system tasks and the states of other robots to maintain an up-to-date representation of the current status of the execution environment. *Plan B Precaution Routines* include both recovery routines that failing robots can execute and warning mechanisms that aim to correct behaviors of other robots in the system.

DEMiR-CF is different from earlier work by ensuring both instantaneous assignment procedures incrementally and forming rough schedules to consider the problem as a whole from global perspectives. The rough schedule generation process uses polynomial time even for complex tasks. Combinatorial task exchange mechanisms as in market-based approaches are not used in DEMiR-CF. Auctions are used by robots to announce intentions about task execution and to select the appropriate task executers to deal with world information incompleteness. Only single items are allocated incrementally during task execution. Contingency handling mechanisms are directly integrated into the dynamic task selection mechanisms, which in turn facilitate recovering from failures dynamically and efficiently, reconfiguring robots during runtime, and maintaining system consistency. These utilities are ensured by autonomous robots implementing DEMiR-CF in a completely distributed manner without central authorities and/or complete knowledge injected manually.

DEMiR-CF has been evaluated in different domains in both simulations and in real environments with robots. WEBOTS simulator and US NAVY's ALWSE-MC simulator are used as simulation environments and Khepera II robots are used as physical real hardware to carry out experiments.

The Multiple Traveling Robot Problem (MTRP) is a generalization of the well known Traveling Salesman Problem (TSP) where each target should be visited by at least one robot, optimizing an objective which could be minimization of time, minimization of path length traversed by robots, etc. This domain forms a basis for several types of complex missions, such as search and rescue operations. Another sample domain involves space exploration operations, where the total path length of robots needs to be minimized. The integrated components that make up the DEMiR-CF framework are successfully implemented for the MTRP domain.

MTRP experiments are performed in four sets. In the first set of experiments, new heuristic cost functions are proposed and evaluated for single robot route construction. The performance of the proposed heuristic cost functions are compared with optimal results that are generated by using an Integer Programming formulation running on a commercial IP solver, CPLEX. It has been observed that the DEMiR-CF results generated with the use of the designed heuristic cost function deviate from the optimal solutions by at most 15.24% for a large TSP instance.

In the second set of experiments, the task allocation approach of DEMiR-CF is compared with both the Prim Allocation approach and the Integer Programming approach. As expected, both DEMiR-CF and the Prim Allocation methods have tractable computational complexities compared to the Integer Programming approach. However, as the results presented in Section 5. reveal, DEMiR-CF integrated with new heuristic cost functions produces results that are close to the optima for the multi-robot case of the problem.

In the third set of experiments, the performance of DEMiR-CF under real-time, dynamic conditions is evaluated. Online task handling scenarios in environments where map information is not available are presented. It has been shown that, even when the map information is not available, robots can efficiently reconfigure themselves according to changes in the environment and incrementally select tasks suitable for themselves through decisions that take into account the most recent information.

In the fourth set of the experiments, real robot scenarios are presented against robot failures to validate the robustness of DEMiR-CF.

The results of the experiments support our thesis that in a dynamic execution environment an incremental task selection approach eliminates redundant efforts that are introduced by allocating all tasks from scratch if there is an unexpected change. The rough schedule generation scheme forms loose commitments, which if needed, can be canceled in the future. Thus, it offers a way to reconsider the problem globally when it is appropriate. The approach is efficient with its polynomial computational and communication complexities. *Plan B precautions* ensure that the $CMAP$ is successfully solved at the end of mission execution. If real resources permit, failures are handled to maintain system consistency. Even though a certain amount of additional communication overhead is injected into the system by the *Plan B Precaution Routines*, communication efficiency is also achieved as experimental results illustrate.

DEMiR-CF is also evaluated in NAVY domains where the cooperation of underwater vehicles is required for homeland security missions, such as the mine countermeasure mission. This domain has a challenging structure since it is performed underwater and communication is achieved through acoustic modems. In this domain, the robot team is modeled as a heterogeneous team and the mission consists of different types of tasks, requiring different vehicles. The domain is modeled to include both the coverage problem and the MTRP.

The coverage problem is solved by having robots generate rough schedules for the regions to be covered and negotiate over these regions. While covering the environment, robots simultaneously sense "mine like objects" to generate new online tasks. They transmit this information to different types of robots that can visit targets and perform correct classification. With this mode of operation, the problem turns into the MTRP and is solved in a decentralized way for continually generated online tasks. Depending on the types of robots, different types of rough schedules are generated by either coverage or classification robots. Regions are represented as rough schedules for the coverage tasks, whereas target sets are formed as rough schedules for the classification tasks.

The online task handling performance of DEMiR-CF is validated through experiments. The robustness of the framework against both communication and robot failures and the efficiency with which it responds to dynamic changes in the environment are tested in simulations.

As a final domain, object construction domain is selected to use and validate the full functionality of DEMiR-CF. Complex missions investigated in this domain involve tasks with resource constraints and interrelations. Multi-robot task allocation problem is better viewed as a scheduling problem if there are interrelations among tasks. Therefore, this domain forms the basis of the design objective of DEMiR-CF. A topological list generation approach for forming rough schedules is used to solve the $CTSP$ and to resolve the constraints on tasks. Applicable cost functions are proposed for separate tasks for robots to achieve an overall complex mission. The scalability of DEMiR-CF is validated through experiments. Efficiency is analyzed and validated both in theory and in experiments. As results illustrate, the $CTSP$ is solved by each robot implementing the incremental task selection, distributed task allocation and contingency handling mechanisms of DEMiR-CF to achieve the overall $CMAP$ for complex missions.

In conclusion, in this PhD thesis, DEMiR-CF has been designed and implemented as a generalized framework for cooperative multi-robot mission achievement. Several performance tests are carried out for different domain implementations of the framework. It is demonstrated that the framework is an efficient, complete, scalable and robust decentralized framework for a multi-robot system. Furthermore, DEMiR-CF is shown to be applicable on even very small and simple robots with cheap computational capabilities, such as Khepera II.

8.1. Future Work

If embedding a commercial program such as CPLEX IP solver on robots is possible and the requirements of the decision frequency and change in task allocations do not affect the response time of the system, an Integer Programming method may be used to generate rough schedules of robots. In this case, redundant allocations can be implemented to reach globally optimum results for the current situation. This approach can also be successfully integrated to the incremental assignment approach that we propose. However, as we mentioned earlier, if there are computational limitations on the robot hardware, a heuristic cost evaluation is inevitable for the system.

In the current design of the DEMiR-CF, planning activity is performed for the MTRP domain. Route construction is a high-level path planning problem and is solved by DEMiR-CF by integrating it with task allocation. For more complex tasks, a global plan with interrelations between tasks is given to the robots initially. DEMiR-CF is capable of changing the structure of given plans during runtime in a decentralized way. Online tasks are integrated into the task graph. However, this research area desires more investigation for both constructing a global plan by the robots and satisfying and resolving conflicts in the global plan as investigated in MPCP.

There are no previously designed test-beds for multi-robot systems research. Although comparisons and results are provided, usually it is hard to evaluate and compare implemented systems with the insufficient implementation details presented in the publications. Even though RoboCup leagues present test-beds for the comparison of the architectures, research papers can only present ad-hoc implementations without comparisons for real robots. Therefore, a formalism and designs of test-beds are greatly needed to improve the field and find better ways to make the robots do the right thing.

BIBLIOGRAPHY

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N. and Hull, D., 2002. Morse Decompositions for Coverage Tasks. *The International Journal of Robotics Research*, **21**(4), 331–344.

Alami, R. and Botelho, S. C., 2001. Plan-Based Multi-robot Cooperation. *In Advances in Plan-Based Control of Robotic Agents*.

Alami, R., Ingrand, F. and Qutub, S., 1998. A Scheme for Coordinating Multi-Robot Planning Activities and Plans Execution. *In Thirteenth European Conference On Artificial Intellingence*.

ALWSE. http://nswcpc.navsea.navy.mil/analysis/capabilities.asp, 2006.

Balch, T. and Arkin, R. C., 1994. Communication in Reactive Multiagent Systems. *Autonomous Robots*, **1**(1), 1–25.

Balch, T. and Arkin, R. C., 1998. Behavior-based Formation Control for Multi-robot Teams. *IEEE Transactions on Robotics and Automation*.

Balch, T. and Parker, L., 2002. Robot Teams: From Diversity to Polymorphism. AK Peters.

Berhault, M., Huang, H., Keskinocak, P., Elmaghraby, W., Griffin, P. and Kleywegt, A., 2003. Robot Exploration with Combinatorial Auctions. *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Boddy, M. S., Bennett, B. H., Isle, B. A. and Isle, R. A., 2004. NASA Planning and Scheduling Applications: Emerging Technologies and Mission Trends. Technical Report **NASA Grant NAG-2-1631**, Adventium Labs.

Botelho, S. and Alami, R., 1999. M+: a Scheme for Multi-robot Cooperation Through Negotiated Task Allocation and Achievement. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Brucker, P., 2001. Scheduling Algorithms. Springer Verlag.

Brucker, P., 2002. Scheduling and Constraint Propagation. *Discrete Applied Mathematics*, **123**, 227–256.

Brucker, P. and Knust, S., 2006. Complex Scheduling. Springer Verlag.

Brucker, P., Knust, S., Schoo, A. and Thiele, O., 1998. A branch bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, **107**, 272–288.

Brumitt, B. L. and Stentz, A., 1998. GRAMMPS: A Generalized Mission Planner for Multiple Robots in Unstructured Environments. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Burgard, W., Moors, M., Stachniss, C. and Schneider, F. E., 2005. Coordinated Multi-Robot Exploration. *IEEE Transactions on Robotics and Automation*, **21**(3), 376–386.

Chaimowicz, L., Campos, M. F. M. and Kumar, R. V., 2002. Dynamic Role Assignment for Cooperative Robots. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 293–298.

Cox, J. S., Durfee, E. H. and Bartold, T., 2005. A distributed framework for solving the Multiagent Plan Coordination Problem. *In AAMAS*, pages 821–827.

Dahl, T. S., Mataric, M. J. and Sukhatme, G. S., 2004. Emergent Robot Differentiation for Distributed Multi-Robot Task Allocation. *In Distributed Autonomous Robotic Systems (DARS)*.

Davids, A., 2002. Urban Search and Rescue robots: From Tragedy to Technology. *IEEE Intelligent Systems*, **17**(2), 81–83.

Decker, K., 1996. Foundations of Distributed Artificial Intelligence, chapter TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms, pages 429–448. John Wiley and Sons.

desJardins, M., Durfee, E., Ortiz, C. and Wolverton, M. J., 1999. Survey of Research in Distributed, Continual Planning. *AI Magazine*, **20** (4), 13–22.

Dias, M. B. and Stentz, A., 2002. Opportunistic Optimization for Market-Based Multirobot Control. *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Dias, M. B., Zlot, R. M., Kalra, N. and Stentz, A., 2005. Market-Based Multirobot Coordination: A Survey and Analysis. Technical Report **CMU-RI-TR-05-13**, Carnegie Mellon University, Robotics Institute.

Dias, M., 2004. TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. Phd thesis, Robotics Institute, Carnegie Mellon University.

Dias, M., Zinck, M., Zlot, R. M. and Stentz, A., 2004. Robust Multirobot Coordination in Dynamic Environments. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Dudek, G., Jenkin, M., Milios, E. and Wilkes, D., 1996. A Taxonomy for Multi-Agent Robotics. *Autonomous Robots*, **3**(4), 375–397.

Durfee, E. H., 2001. Scaling Up Agent Coordination Strategies. *IEEE Computer*, **34**(7), 39–46.

Finin, T., Labrou, Y. and Mayfield, J., 1997. KQML as an agent communication language, chapter Software Agents. The MIT press, Cambridge, MA.

FIPA ACL. http://www.fipa.org/specs/fipa00061/SC00061G.pdf, 2002.

Gancet, J., Hattanberger, G., Alami, R. and Lacroix, S., 2005. Task Planning and Control for a Multi-UAV System: Architecture and Algorithms. *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Gerkey, B. and Mataric, M. J., 2002. Sold!: Auction Methods for Multirobot Coordination. *IEEE Trans. Robot. Automat.*, **18**(5), 758–768.

Gerkey, B. and Mataric, M. J., 2004. A Formal Analysis and Taxonomy of Task Allocation. *Intl. J. of Robotics Research*, **23**(9), 939–954.

Ghiani, G., Guerriero, F., Laporte, G. and Musmanno, R., 2003. Real-Time Vehicle Routing: Solution Concepts, Algorithms and Parallel Computing Strategies. *European Journal of Operational Research*, **151**, 1–11.

Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., Smith, T. and Stentz, A., 2002. A Distributed Layered Architecture for Mobile Robot Coordination: Application to Space Exploration. *In 3rd Intl. NASA Workshop on Planning and Scheduling for Space*.

Goldberg, D., Cicirello, V., Dias, M. B., Simmons, R., Smith, S., Smith, T. and Stentz, A., 2003. Market-Based Multi-Robot Planning in a Distributed Layered Architecture. *In Multi-Robot Systems: Swarms to Intelligent Automata, Proc. Of Intl. Workshop on Multi-Robot Systems*.

Hazon, N. and Kaminka, G. A., 2005. Redundancy, Efficiency, and Robustness in Multi-Robot Coverage. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Horling, B. and Lesser, V., 2005. A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review*, **19**(4), 281–316.

Huhns, M. N. and Stephens, L. M., 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chapter Multiagent Systems and Societies of Agents, pages 80–120. The MIT Press.

ILOG-CPLEX-9.0-UserMan. http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsp/index.html, 2007.

Jarnik, V., 1930. O jistem problemu minimalnim (About a certain minimal problem). *Prace Moravske Prirodovedecke Spolecnosti*, **6**, 57–63.

Jennings, J. J., Whelan, G. and Evans, W. F., 1997. Cooperative Search and Rescue with a Team of Mobile Robots. *In International Conference on Advanced Robotics (ICAR)*.

Jennings, N. R., 1996. Foundations of Distributed Artificial Intelligence, chapter Coordination Techniques for Distributed Artificial Intelligence, pages 187–210. John Wiley and Sons.

Kalra, N., Ferguson, D. and Stentz, A., 2005. Hoptiles: A Market-Based Framework for Planned Tight Coordination in MultiRobot Teams. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Kaminka, G. A. and Tambe, M., 2000. Robust Multi-Agent Teams via Socially-Attentive Monitoring. *Journal of Artificial Intelligence Research*, **12**, 105–147.

Kitano, H., 2000. Robocup rescue: A grand challenge for multi-agent systems. *In Proceedings of ICMAS*.

Koes, M., Nourbakhsh, I. and Sycara, K., 2005. Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints. *In AAAI National Conference on Artificial Intelligence*.

Kose, H., Kaplan, K., Mericli, C., Tatlidede, U. and Akin, L., 2005. Market-Driven Multi-Agent Collaboration in Robot Soccer Domain, chapter Cutting Edge Robotics, pages 407–416. Pro Literatur Verlag.

Lagoudakis, M. G., Berhault, M., Koenig, S., Keskinocak, P. and Kleywegt, A., 2004. Simple Auctions with Performance Guarantees for Multi-Robot Task Allocation. *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Lagoudakis, M. G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A. and Jain, S., 2005. Auction-Based Multi-Robot Routing. *In Robotics: Science and Systems (RSS)*.

Lawler, E. L., Lenstra, J. K., Kan, R. and Shmoys, D. B., 1985. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley & Sons, New York, NY.

Lemaire, T., Alami, R. and Lacroix, S., 2004. A Distributed Task Allocation Scheme in Multi-UAV Context. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Malone, T. W. and Crowston, K., 1994. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, **26**(1), 87–119.

Michel, O., 1998. Webots: Symbiosis Between Virtual and Real Mobile Robots. *In Proceedings of the First International Conference on Virtual Worlds*.

Moravec, H. and Elfes, A. E., 1985. High Resolution Maps from Wide Angle Sonar. *In Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 116 – 121.

Muller, H. J., 1996. Negotiation principles, pages 211–229. John Wiley & Sons, New York, NY. ISBN 0-471-006750.

Nilsson, N. J., 1986. Principles of Artificial Intelligence. Morgan Kaufmann Publishers.

Ossowski, S., 1999. Co-ordination in Artificial Agent Societies, Social Structure and Its Implications for Autonomous Problem-Solving Agents. Springer Verlag.

Paquet, S., 2006. Distributed Decision-Making and Task Coordination in Dynamic, Uncertain and Real-Time Multiagent Environments. Phd thesis, Laval University, Quebec.

Parker, L. E., 1998. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Trans. Robot. Automat.*, **14**(2), 220–240.

Parker, L. E., 2004. Current Research in Multi-Robot Systems. *Journal of Artificial Life and Robotics 7*.

Parker, L. E. and Tang, F., 2006. Building Multi-Robot Coalitions through Automated Task Solution Synthesis. *Proceedings of the IEEE, Special Issue on Multi-Robot Systems*.

Pinedo, M. L., 2005. Planning and Scheduling in Manufacturing and Services. Springer Verlag.

Prim, R. C., 1957. Shortest connection networks and some generalisations. *Bell System Technical Journal*, **36**.

Randall and Smith, R. G., 1983. Negotiaton as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, **20**(1), 63–109.

Reinelt, G., 1991. TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing 3*, pages 376–384.

Reinelt, G., 1994. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag.

Rekleitis, I. M., Lee-Shue, V., New, A. P. and Choset, H., 2004. Limited Communication Multi-Robot Team Based Coverage. *In IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Sandholm, T. W., 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, chapter Distributed Rational Decision Making, pages 202–258. The MIT Press.

Sariel, S. and Akin, H. L., 2005. A Novel Search Strategy for Autonomous Search and Rescue Robots. *In RoboCup*, volume 3276 of *Lecture Notes in Computer Science*, pages 459–466. ISBN 3-540-25046-8.

Sariel, S. and Balch, T., 2005a. Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments. *In Integrating Planning into Scheduling: Papers from the 2005 AAAI Workshop, WS-05-06*, pages 27–33.

Sariel, S. and Balch, T., 2005b. Robust Multi-Robot Coordination in Noisy and Dangerous Environments. Technical Report **GIT-GVU-05-17**, GVU Center, Georgia Institute of Technology.

Sariel, S. and Balch, T., 2006a. Distributed Autonomous Robotic Systems (DARS) 7, chapter A Distributed Multi-Robot Cooperation Framework for Real Time Task Achievement, pages 187–196. Springer Verlag.

Sariel, S. and Balch, T., 2006b. Dynamic and Distributed Allocation of Resource Constrained Project Tasks to Robots. *In Multi-Agent Robotic Systems (MARS) Workshop at the Third International Conference on Informatics in Control, Automation and Robotics*.

Sariel, S. and Balch, T., 2006c. Efficient Bids on Task Allocation for Multi-Robot Exploration. *In The 19th International The Florida Artificial Intelligence Research Society (FLAIRS) Conference*.

Sariel, S., Balch, T. and Erdogan, N., 2006a. Robust Multi-Robot Cooperation Through Dynamic Task Allocation and Precaution Routines. *In The International Conference on Informatics in Control, Automation and Robotics (ICINCO)*.

Sariel, S., Balch, T. and Erdogan, N., 2007a. Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks. *In IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Sariel, S., Balch, T. and Stack, J. R., 2006b. Distributed Autonomous Robotic Systems (DARS) 7, chapter Empirical Evaluation of Auction-Based Coordination of AUVs in a Realistic Simulated Mine Countermeasure Task, pages 197–206. Springer Verlag.

Sariel, S., Balch, T. and Stack, J. R., 2006c. Distributed Multi-AUV Coordination in Naval Mine Countermeasure Missions. Technical Report **GIT-GVU-06-04**, GVU Center, Georgia Institute of Technology.

Sariel, S., Erdogan, N. and Balch, T., 2007b. An Integrated Approach To Solving The Real-World Multiple Traveling Robot Problem. *In The 5th International Conference on Electrical and Electronics Engineering*.

SarielKh2Mov. http://www2.itu.edu.tr/∼sariel/videos/KheperaII-Movies.html, 2007.

SarielMCMMov. http://www2.itu.edu.tr/∼sariel/videos/MCM-Movies.html, 2007.

Shehory, O. and Kraus, S., 1998. Methods for Task Allocation via Agent Coalition Formation. *Artificial Intelligence*, **101**, 165–200.

Simmons, R. and Apfelbaum, D., 1998. A Task Description Language for Robot Control. *In Conference on Intelligent Robotics and Systems*.

Smith, R. G., 1980. The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver. *IEEE Transaction on Computers C-*, **29**(12), 1104–1113.

Stack, J. and Manning, R., 2004. Increased autonomy and Cooperation in Multi-AUV Naval Mine Countermeasures. *In Proceedings of Undersea Defence Technology*.

Tews, A., 2001. Adaptive Multi-robot Coordination for Highly Dynamic Environments. *In International Conference On Computational Intelligence for Modelling (CIMCA)*.

Toth, P. and Vigo, D., 2001. The Vehicle Routing Problem. Society for Industrial and Applied Mathematics.

Vail, D. and Veloso, M., 2003. Dynamic Multi-robot Coordination. *Multi-Robot Systems: From Swarms to Intelligent Automata*, **2**, 87–98.

Vig, L. and Adams, J. A., 2005. Issues in Multi-robot Coalition Formation. *In Multi-Robot Systems. From Swarms to Intelligent Automata. Volume III*, pages 15–26.

WEBOTS. Webots User Guide 5.1.11, 2006.

Weglarz, J., 1999. Project Scheduling: Recent Models, Algorithms and Applications. Kluwer.

Wikipedia-Cooperation. http://en.wikipedia.org/wiki/Cooperation, 2007.

Zlot, R. and Stentz, A., 2006. Market-based Multirobot Coordination for Complex Tasks. *Intl. J. of Robotics Research*, **25**(1).

APPENDIX

A. TSP HEURISTICS

In the following heuristic function definitions, T_{TSP} is a partial tour and k is a city not on T_{TSP}, and $\{i, j\}$ is one of the edges of T_{TSP}.

Minimum Spanning Tree Heuristic (MST): In this method, either the Prim's algorithm or the Kruskal's algorithm may be used to generate the MST for the given set of the cities. A depth first search of T is constructed. After introducing shortcuts into the depth first search, it is ensured that cities are visited only once.

Nearest Merger Heuristic (NMH): This method corresponds to the Kruskal's minimum spanning tree algorithm. The algorithm starts with n partial tours, each of which consists of a single city, then, successively merges the tours until a single tour containing all cities is obtained. Each time trees to be merged are chosen so that $minc_{ij} : i \in T_{TSP}$ and $j \in T'_{TSP}$ is as small as possible.

Nearest Neighborhood Heuristic (NNH): The algorithm starts with an empty tour T. Cities k and j, for which $c(j, k)$ minimized are found. i, j is replaced by i, k and k, j to obtain a new tour including k. The algorithm continues until all cities are added to T_{TSP}.

Nearest Insertion Heuristic (NIH): The algorithm starts with an empty tour T_{TSP}. Cities k and j, for which $c(j, k)$ minimized are found. i, j is the edge of T_{TSP} which minimizes $c(i, k) + c(k, j) - c(i, j)$, and it is replaced by i, k and k, j to obtain a new tour including k. The algorithm continues until all cities are added to T_{TSP}.

Cheapest Insertion Heuristic (CIH): The algorithm starts with an empty tour T_{TSP}. If T_{TSP} does not include all cities, for each k, the edge i, j of T_{TSP} which minimizes $c(T_{TSP}, k) = c(i, k) + c(k, j) - c(i, j)$ is found. Then, the city k minimizing $c(T_{TSP}, k)$ is found. If i, j is the edge of T_{TSP} for which $c(T_{TSP}, k)$ is minimized, it is replaced by i, k and k, j to obtain a new tour including k. The algorithm continues until all cities are added to the T_{TSP}.

Farthest Insertion Heuristic (FIH): The algorithm starts with an empty tour T_{TSP}. City k, which is the farthest of all the cities out of T_{TSP}, is inserted to T_{TSP}. The algorithm continues until all cities are added to the T_{TSP}.

Christofides Algorithm (CA): MST of the given set of cities is constructed. Minimum matching $M*$ for the set of all odd-degree vertices in T_{TSP} is constructed. An Eulerian tour for the Eulerian Graph that is the union of the T_{TSP} and $M*$ is found, and it is converted into a tour using the shortcuts.

Except from the FIH and the NNH, all the algorithms presented above have worst case solutions bounded by $2 * OPT$. The NNH is bounded by (log n)*OPT, whereas the CA is bounded by 1.5*OPT.

B. TSPLIB INSTANCES

TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types (Reinelt, 1991). Each instance is given with the coordinates of the cities to be visited. In our experiments, these instances are used for the performance analysis, and the first node is taken as the initial location of the robot in each instance. While the cities are marked with red x markers, the initial nodes and the robot locations are indicated with blue circles in the following graphs.

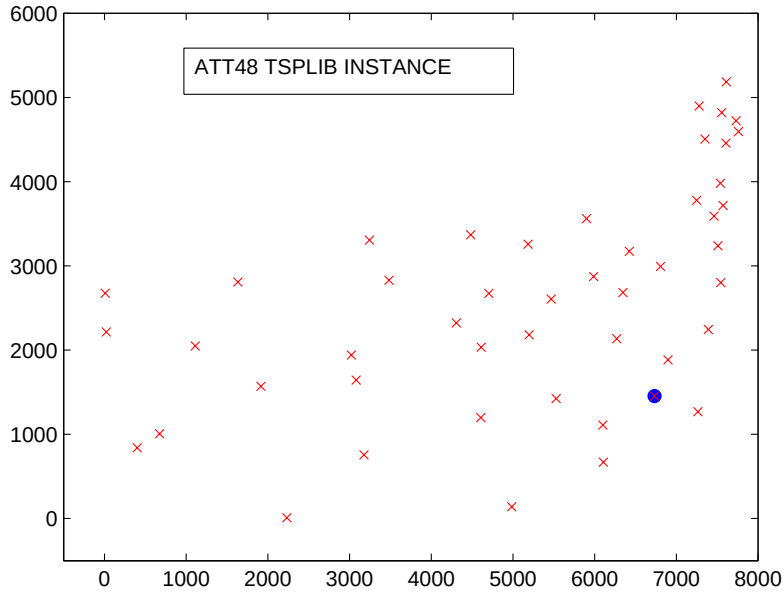


Figure B.1: ATT48 TSPLIB instance with 48 nodes

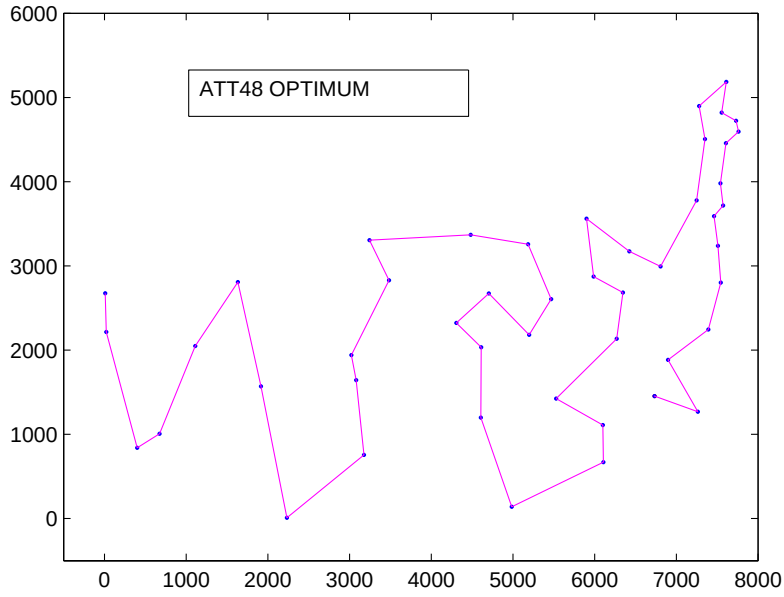


Figure B.2: Optimal open-loop route for the ATT48 TSPLIB instance

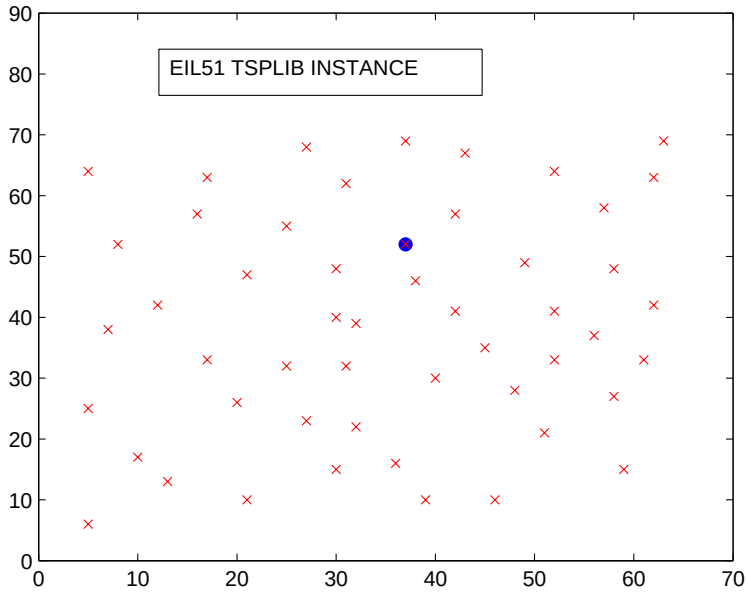


Figure B.3: EIL51 TSPLIB instance with 51 nodes

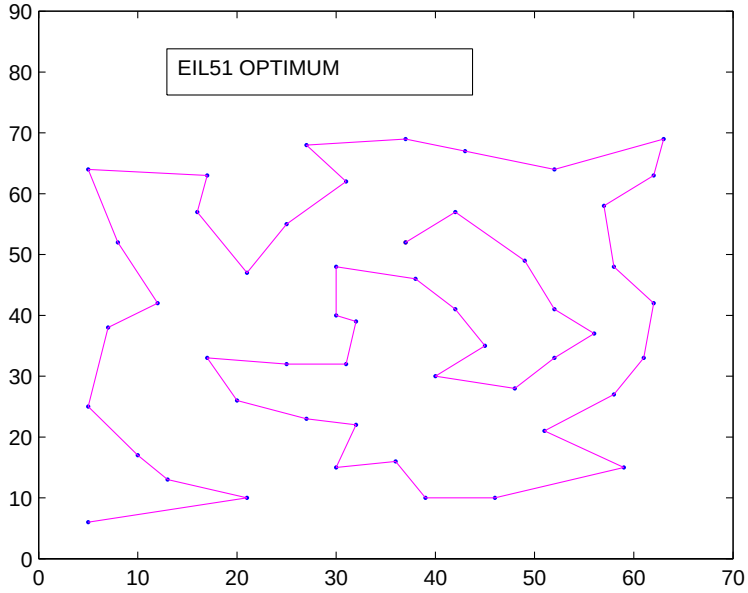


Figure B.4: Optimal open-loop route for the EIL51 TSPLIB instance

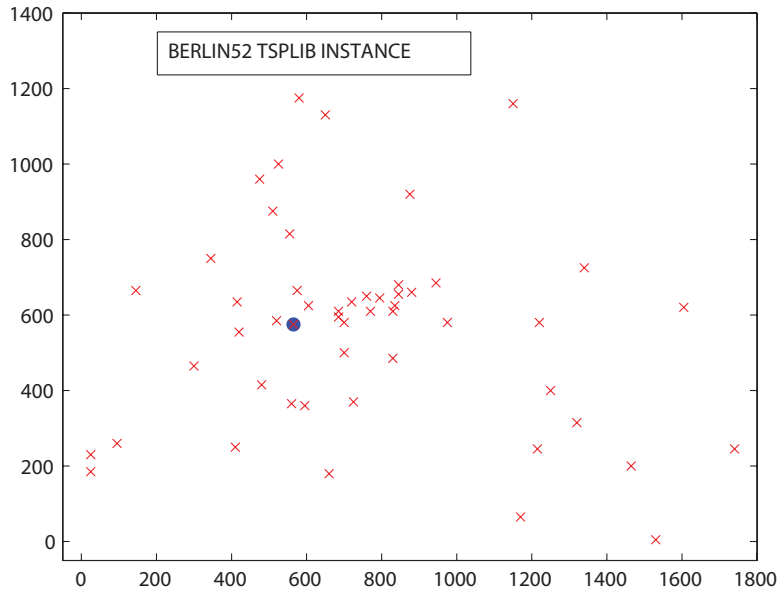


Figure B.5: BERLIN52 TSPLIB instance with 52 nodes

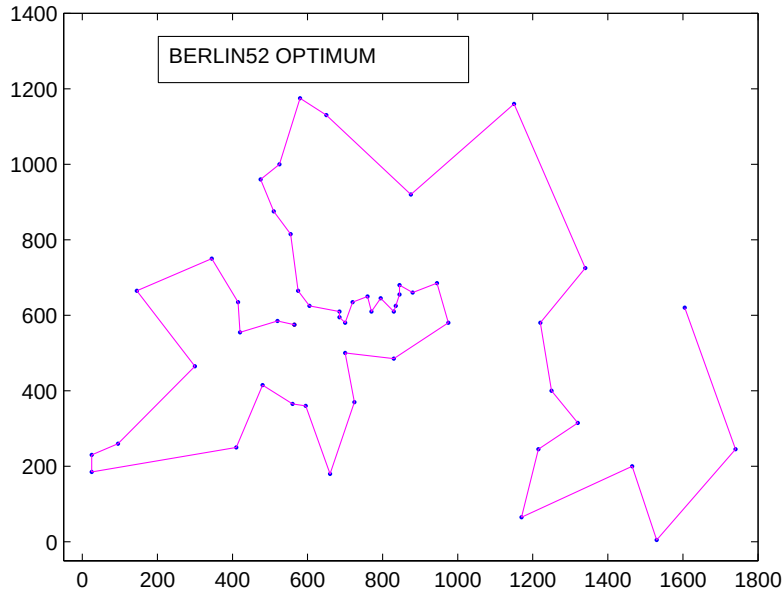


Figure B.6: Optimal open-loop route for the BERLIN52 TSPLIB instance

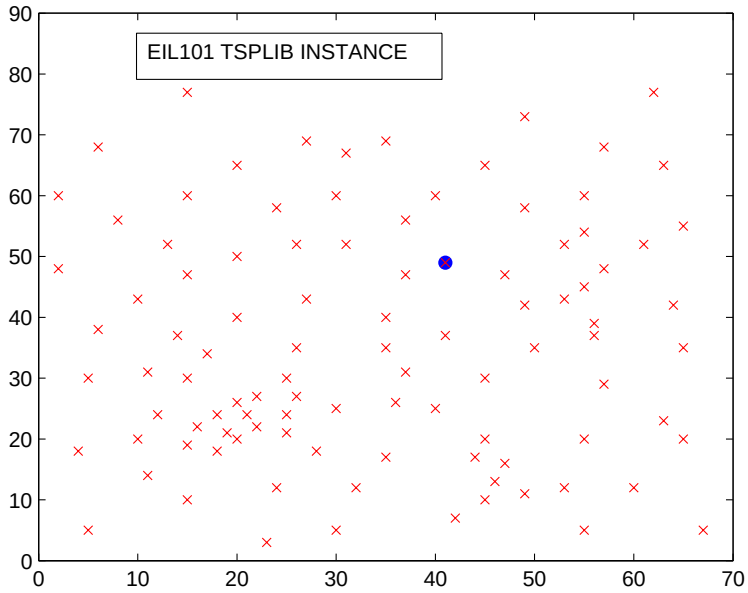


Figure B.7: EIL101 TSPLIB instance with 101 nodes

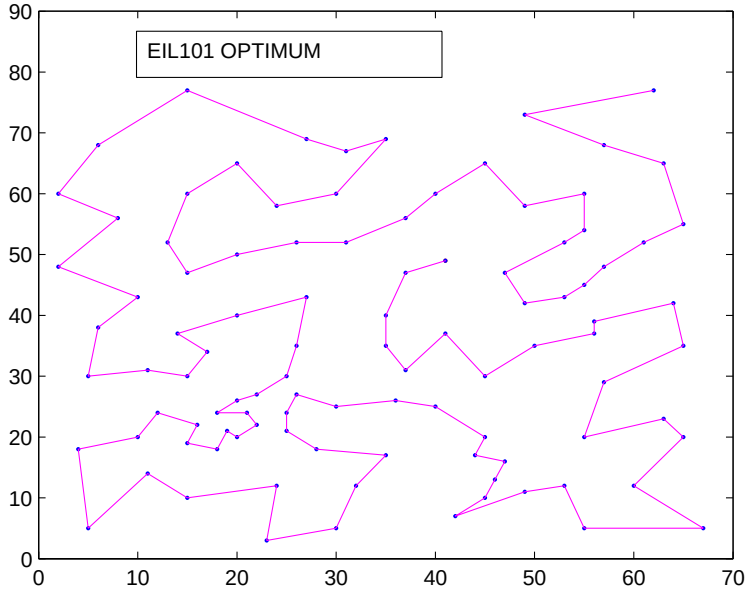


Figure B.8: Optimal open-loop route for the EIL101 TSPLIB instance

BIOGRAPHY

Sanem Sariel received her B.Sc. in Computer and Control Engineering and M.Sc. in Computer Engineering from Istanbul Technical University in 1999 and 2002 respectively. She has been serving as a research and teaching assistant at ITU since 1999. She was awarded with a research scholarship to do research for her Ph.D. studies in the United States in 2004 where she worked with Prof. Tucker Balch in the BORG Laboratory at Georgia Institute of Technology on multi-robot coordination. Sanem Sariel's research interests lie in the area of Distributed Problem Solving, Multi-Robot Coordination and Intelligent Agents. The following research papers were published during her PhD research:

Book Chapters:

- S. Sariel and T. Balch, "A Distributed Multi-Robot Cooperation Framework for Real Time Task Achievement", Distributed Autonomous Robotic Systems (DARS) 7, Springer Verlag ISBN:4431358781, 2006, pp. 187-196.
- S. Sariel, T. Balch and J. Stack, "Empirical Evaluation of Auction-Based Coordination of AUVs in a Realistic Simulated Mine Countermeasure Task", Distributed Autonomous Robotic Systems (DARS) 7, Springer Verlag ISBN:4431358781, 2006, pp. 197-206.

Conference Papers:

- S. Sariel, T. Balch and N. Erdogan, "Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2007.
- S. Sariel, N. Erdogan and T. Balch, "An Integrated Approach To Solving The Real-World Multiple Traveling Robot Problem", The 5th International Conference on Electrical and Electronics Engineering, 2007.
- S. Sariel, T. Balch and N. Erdogan, "Robust Multi Robot Cooperation through Dynamic Task Allocation and Precaution Routines", The Third International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2006.
- S. Sariel and T. Balch, "Efficient Bids on Task Allocation for Multi Robot Exploration", The Nineteenth International Florida Artificial Intelligence Research Society (FLAIRS) Conference, 2006.
- S. Sariel and T. Balch, "Dynamic and Distributed Allocation of Resource Constrained Project Tasks to Robots", Multi-Agent Robotic Systems (MARS) Workshop at the Third International Conference on Informatics in Control, Automation and Robotics, 2006. (Also presented at the AAAI Workshop on Auction Mechanisms for Robot Coordination, 2006).
- S. Sariel and T. Balch, "Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments", In Integrating Planning into Scheduling: AAAI Workshop 2005, pp.27-33.

Technical Reports:

- S. Sariel, T. Balch and J. Stack, "Distributed Multi-AUV Coordination in Naval Mine Countermeasure Missions", GVU Tech Report GIT-GVU-06-04, 2006.
- S. Sariel, T. Balch, "Robust Multi-Robot Coordination in Noisy and Dangerous Environments", GVU Tech Report GIT-GVU-05-17, 2005.
- S. Sariel, "A Framework for Multi-Robot Coordination", The International Conference on Automated Planning and Scheduling (ICAPS), Doctoral Consortium, 2005.