Collaboration Community Formation in Open Systems for Agents with Multiple Goals

Fatemeh Golpayegani

A thesis submitted to the University of Dublin, Trinity College in fulfilment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

April 2018
Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University’s open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Fatemeh Golpayegani

Dated: April 17, 2018
Acknowledgements

Working as a Ph.D. student was a magnificent, as well as challenging experience. There were
days that I could not get through without the help of many people to whom I owe a debt of
gratitude.

My sincere appreciation goes to my supervisor, Prof. Siobhán Clarke, for the excellent super-
vision I got in these four years. This work would have not been possible without her invaluable
insight, guidance, expertise and encouragement. I am still amazed that despite her busy sched-
ule, she was able to read the thesis chapter by chapter with comments and suggestions on almost
every page.

My special thanks to my co-supervisor, Dr. Ivana Dusparic, for her expertise and always
being there for me and helping me to get through all those stressful days, by assuring me that I
had LOADS of time.

I also would like to thank Prof. Michael Luck and Prof. Declan O’Sullivan, my viva exam-
iners, for their very helpful comments and suggestions.

Thank you to Adam and Andrei, who were a great help during the early stage of my doctor-
ate, and for setting an example.

Thank you to all 2.14 guys specially Christian for always being so quiet and caring for my
concentration.

To Paul and Jose, thanks for bringing joy back to 2.15, and to Niall for calling me to have
lunch and coffee even when he left 2.15 and got a well-paid job.

Thank you to my family and in-laws, especially to my Mom who always encouraged me to
achieve more, and to my Dad who was always surprised with whatever I achieved.

My special thanks to my Iranian friends Melika, Ramisa, Nima, and Sahar, with whom I could share all the joy and frustration in Persian.

Last but not least, I heartily thank my best friend, my life companion, my husband, Abdollah for being my best pal, and always accompanying me in all ups and downs. In all those frustrated moments, your reassuring look was all I needed to get my confidence back. I enjoyed every single moment of the journey we started together and all the ups and downs we experienced, enjoyed, and survived, during our Ph.D. studies, hand in hand.

Fatemeh Golpayegani
University of Dublin, Trinity College
April 2018
Publications Related to this Ph.D.

Publications directly related to Collaboration Community Formation (published papers):

1. Collaborative, Parallel Monte Carlo Tree Search for Autonomous Electricity Demand Management. Fatemeh Golpayegani, Ivana Dusparic, and Siobhán Clarke. In Sustainable Internet and ICT for Sustainability (SustainIT), 2015, IEEE.


Submitted Paper:

Publications in the broader Multi-agent Collaboration and Learning Algorithm/Smart Grid area (published):


Abstract

Agents frequently coordinate their behaviour and collaborate to achieve a shared goal, share constrained resources, or accomplish a complex task that they cannot do alone. Forming an effective collaboration community in which agents are willing to cooperate, and have no conflict of interests, is the key to any successful collaborative process. Forming such communities has been addressed well in cooperative and closed multi-agent systems. However, it is particularly challenging in open multi-agent systems where agents are self-interested. Such agents are also likely to continuously and unpredictably leave and join the system and have multiple goals to pursue simultaneously.

Existing research has addressed this challenge in open systems with utility-based or complementary-based approaches. Utility-based approaches focus on maximising self-interested agents’ individual pay-off when sharing constrained resources. In complementary-based approaches, agents’ individual skills are composed to accomplish a complex task or achieve a shared goal. In real-world applications, there are 2 main limitations with these approaches. First, it is impractical to assume that an agent will be exclusively either self-interested or cooperative. Agents cannot remain self-interested in open systems where resources are constrained and there is no central coordinator, because they will need to cooperate to sensibly share the resource. Second, it is too limiting to constrain agents, exclusively, to pursue either individual goals or shared goals during collaboration, as in real-world applications agents can simultaneously pursue multiple goals, including shared and individual goals. In open systems, agents need to identify the possible dependencies and conflicts between their individual goals when using constrained re-
sources to pursue multiple goals simultaneously. Such dependencies affect agents’ levels of self-interest and consequently their willingness to form collaboration communities. Given the circumstances, agents need a decentralised mechanism to acquire an understanding of other agents operating in their system, identify their goal dependencies, and adapt their level of self-interest to form effective collaboration communities.

This thesis presents a fully decentralised approach to Collaboration Community FOrmation Model for agents with multiple goals in open systems (CCFOM). CCFOM presents a new social reasoning model and a new distributed community formation algorithm. CCFOM enables agents to pursue their individual and shared goals simultaneously in resource constrained open systems by forming effective collaboration communities. Each agent shares limited privacy-preserving domain knowledge and models its goal dependencies on other agents using social reasoning. The community formation algorithm is used to facilitate agents’ decentralised decision-making process when forming collaboration communities. Using CCFOM, agents adapt their level of self-interest, and adjust their willingness to form effective collaboration communities.

An application-independent simulator is used to evaluate CCFOM under varying levels of agents’ mobility, and systems’ density, and its application effectiveness is evaluated using smart-grid and ride-sharing case studies. Evaluation metrics include measurements of collective and individual rates of success at achieving shared and individual goals. CCFOM is also evaluated against state of the art utility-based and complementary-based approaches to compare the communication and computational cost. Results show that CCFOM out-performs the utility-based and complementary-based approaches when agents pursue both individual and shared goals. It also decreases the computation and communication costs, when agents have multiple goals with varying dependency relations.
## Contents

Acknowledgements iii  
Abstract vi  
List of Tables xiii  
List of Figures xiv  

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Introduction</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1.2 Multi-agent System Model</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1.3 Multi-agent Collaboration Community Formation</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1.3.1 Collaboration Purpose Identification</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1.3.2 Participant Nomination</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1.3.3 Community Formation</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1.4 Collaboration Community Formation Research Gaps</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>1.4.1 Research Questions</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1.5 Thesis Focus and Contribution</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1.6 Evaluation</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1.7 Assumptions</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>1.8 Roadmap</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>
# Chapter 2  Related Work

2.1 Background ................................................. 19
  2.1.1 Open Multi-agent Systems .............................. 20
  2.1.2 Agent Types ........................................... 20
  2.1.3 Multi-agent Collaboration .............................. 23

2.2 Dynamic Coalition Organization ............................ 26
  2.2.1 Utility-based Coalition Organization ................... 27
  2.2.2 Complementary-based .................................. 29

2.3 Dynamic Team Organization ................................. 37
  2.3.1 Organization-based ...................................... 38
  2.3.2 Learning-based .......................................... 38
  2.3.3 Planning-based ......................................... 38
  2.3.4 Ad-hoc team formation ................................. 39
  2.3.5 Analysis of Dynamic Team Organization ............... 39

2.4 Multi-agent Resource Allocation ............................ 40
  2.4.1 Agents’ preferences ...................................... 41
  2.4.2 Social Welfare .......................................... 41
  2.4.3 Negotiation .............................................. 42
  2.4.4 Analysis of Multi-agent Resource Allocation .......... 43

2.5 Multi-Agent Community Formation in Closed Systems ..... 44
  2.5.1 Joint intention-action .................................... 44
  2.5.2 SharedPlans ............................................. 45
  2.5.3 Teamwork ................................................ 45
  2.5.4 Organization-based ...................................... 46
  2.5.5 Analysis of Multi-agent Community Formation in Closed Systems . . . 48

2.6 Summary and Analysis ....................................... 48
  2.6.1 Summary .................................................. 52
# CONTENTS

5.2.1 Experiment Design ............................................. 103  
5.2.2 Results: Application-independent (general) case study .......... 109  
5.3 Smart Grid .................................................. 118  
5.3.1 Experiment Design ............................................. 118  
5.3.2 Results: Smart Grid case study ................................. 123  
5.4 Ride Sharing .................................................... 128  
5.4.1 Experiment Design ............................................. 128  
5.4.2 Results: Ride Sharing case study ............................... 134  
5.5 Summary ......................................................... 142  

**Chapter 6 Conclusion**  
6.1 Overview of Thesis Achievements .................................. 146  
6.1.1 Thesis Contributions .......................................... 149  
6.2 Discussion ....................................................... 150  
6.3 Limitations and Future Work ....................................... 151  

**Bibliography**  
153
List of Tables

2.1 Analysis of Current Collaboration Community Formation Approaches in Open Systems ........................................... 49

3.1 Categories of Agents in Multi-neighbourhood Setting ................................................................. 73

5.1 Evaluation Metrics’ Mapping to Research Questions ................................................................. 101

5.2 Evaluation Metrics in Each Case Study .................................................................................. 102

5.3 Four Experiments Evaluating CCFOM Under Varying Levels of Neighbourhoods’ Density and Agents’ Mobility in Collaborative, Competitive and Combination (CCFOM) Scenarios ................................................................. 104

5.4 Neighbourhoods’ Membership ......................................................................................... 105

5.5 Parameter Settings: 100 Agents Operating in 6 Neighbourhoods ............................................ 107

5.6 Smart Grid Case Study Results Over 3 Days ........................................................................ 125

5.7 Data Used for Parameter Settings in the Ride Sharing Case Study ........................................ 130
# List of Figures

1.1 An Overview of Current System Modelling Used in Collaboration Community Formation Approaches, and the Proposed Model in CCFOM .......................... 6
1.2 An Overview of Current Mechanisms Used to Address the Collaboration Community Formation Process .......................................................... 7
1.3 An Overview of CCFOM’s Contribution to Collaboration Community Formation ......................................................... 15

2.1 A Goal-based Agent’s Internal Structure [145] ........................................ 21
2.2 A Utility-based Agent’s Internal Structure [145] .................................. 22
2.3 Categorization of Major Contributions in Collaboration Community Formation in Dynamic Environments ............................................. 26
2.4 Agents’ Perspectives and the Coalitions They Can Form Using Social Reasoning Theory ..................................................... 36
2.5 System Modelling in Current Collaboration Community Formation Approaches ...................................................... 36

3.1 CCFOM: Collaboration Community Formation Model .......................... 58
3.2 Overview of an Agent’s External Description and a Neighbourhood’s Properties .................................................. 61
3.3 Dependence Relations Identification .................................................. 67

4.1 MAS-ML Meta-model .................................................................... 81
4.2 MAS-ML Meta-model: Relationships .............................................. 81
4.3 The Abstractions and Relationships in MAS-ML Meta-model [165] ....... 82
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4 CCFOM’s Social Dependency Relations: Extended MAS-ML Meta-model</td>
<td>82</td>
</tr>
<tr>
<td>4.5 CCFOM’s Main Components</td>
<td>83</td>
</tr>
<tr>
<td>4.6 CCFOM Class Diagram</td>
<td>84</td>
</tr>
<tr>
<td>4.7 Sequence Diagram of CCFOM, a High Level View</td>
<td>85</td>
</tr>
<tr>
<td>4.8 Neighbourhood Layer: Neighbourhood Membership Update, Agent Layer:</td>
<td>86</td>
</tr>
<tr>
<td>Collaboration Need Identification</td>
<td></td>
</tr>
<tr>
<td>4.9 Social Reasoning Process Implementation</td>
<td>87</td>
</tr>
<tr>
<td>4.10 Self-Evaluation and Adaptation Process Implementation</td>
<td>89</td>
</tr>
<tr>
<td>4.11 Cascade Collaboration Process Implementation</td>
<td>90</td>
</tr>
<tr>
<td>4.12 Decision-Making Process Implementation</td>
<td>90</td>
</tr>
<tr>
<td>4.13 The General Simulator: Simulating Agents’ Operation in Multiple</td>
<td>92</td>
</tr>
<tr>
<td>Overlapping Neighbourhoods</td>
<td></td>
</tr>
<tr>
<td>4.14 A Single Day’s Electricity Usage</td>
<td>93</td>
</tr>
<tr>
<td>4.15 Smart Grid Case Study: GridLAB-D Connected to CCFOM by the General</td>
<td>95</td>
</tr>
<tr>
<td>Simulator</td>
<td></td>
</tr>
<tr>
<td>4.16 Ride Sharing Schematic View</td>
<td>96</td>
</tr>
<tr>
<td>4.17 Ride Sharing Case Study: Ride-Share Component Connected to CCFOM</td>
<td>98</td>
</tr>
<tr>
<td>by the General Simulator</td>
<td></td>
</tr>
<tr>
<td>5.1 An Overview of How the Neighbourhoods Overlap</td>
<td>104</td>
</tr>
<tr>
<td>5.2 Agents’ Neighbourhood Membership Setting</td>
<td>106</td>
</tr>
<tr>
<td>5.3 Individual Goal Achievement, (Experiment 1: Dense Neighbourhoods,</td>
<td>110</td>
</tr>
<tr>
<td>Medium Mobility)</td>
<td></td>
</tr>
<tr>
<td>5.4 Individual Goal Achievement, (Experiment 2: Medium Density Neigh-</td>
<td>111</td>
</tr>
<tr>
<td>bourhood, Medium Mobility)</td>
<td></td>
</tr>
<tr>
<td>5.5 Individual Goal Achievement, (Experiment 3: Dense Neighbourhoods,</td>
<td>112</td>
</tr>
<tr>
<td>Fast Mobility)</td>
<td></td>
</tr>
</tbody>
</table>
5.6 Individual Goal Achievement, (Experiment 4: Medium Density Neighbourhoods, Fast Mobility) .......................................................... 113
5.7 Resource Demand in Experiment 3: Dense Neighbourhoods, Fast Mobility . . . . . . . . . . . . . .(91,215),(763,231) 115
5.8 Collaboration Success Rate in Different Experiments ....................... 116
5.9 Communication Cost Analysis .......................................................... 117
5.10 Smart Grid Residential Neighbourhood ........................................... 119
5.11 Different Dependencies Between EVs and EEVs ............................... 121
5.12 Agents’ Goal Failure Rate ................................................................. 124
5.13 Transformer Load Results for Different approaches ........................... 126
5.14 Conflicts in Formed Collaboration Communities ............................... 127
5.15 Ride Sharing Experiment Design ....................................................... 130
5.16 Ride Sharing- Hired and in Service Example ................................. 132
5.17 Number of Trips per Taxi while in Service, in Different Scenarios ........ 136
5.18 Taxis’ Time Hired while in Service (%) in Different Scenarios .......... 138
5.19 Taxis’ Trip Length in Different Scenarios ........................................ 140
5.20 Passengers’ Individual and Shared Goal Achievement ........................ 141
Chapter 1

Introduction

This thesis presents CCFOM, a decentralised Collaboration Community Formation Model for agents with multiple goals in open systems, enabling agents operating in open systems to form effective collaboration communities to achieve multiple goals simultaneously. Agents use social reasoning techniques, for instance, dependency modelling, context-based reasoning, or social norm descriptions to acquire some knowledge about their neighbouring agents and to understand the dependencies between their goals. Using this knowledge, agents coordinate their behaviour and form effective collaboration communities to achieve multiple goals including shared and individual goals, when sharing constrained resources.

This chapter presents the motivation for this work, and introduces the existing research gaps in collaboration community formation in open multi-agent systems. The chapter also introduces the contribution of this thesis, and finally concludes with a roadmap of the remainder of this thesis.

1.1 Motivation

Multi-agent systems (MAS) consist of multiple autonomous entities that interact within the system to achieve their goals [72]. MAS can be defined as a macro-organization of micro-systems, so called agents, within an environment [153]. In such organizations, the domain knowledge is decentralised and each agent has incomplete. Such information includes the available capacity
Chapter 1: Introduction

of the resources, other agents operating in the system, and their tasks/goals dependencies. In Multi-agent systems, agents are enabled to autonomously operate in large-scale systems without requiring external assistance (e.g., human assistance), using techniques such as learning, or forming teams or coalitions to solve complex problems cooperatively [98]. Such agents continuously interact with other agents in their environment and take actions accordingly, instead of basing their behaviour on a predefined set of actions [170]. Agents can be also adaptive and interleave their activities to achieve multiple goals simultaneously, particularly, in environments where a goal can be achieved to a certain degree, rather than either being fulfilled or not [112]. They require to understand their goals’ dependencies to be able to choose a goal from their set of goals, and adapt their activities accordingly, particularly when they face real world constraints such as constrained resources [89, 112]. Multi-agent collaboration algorithms enable agents operating in large-scale systems to accomplish complex tasks, achieve shared goals, increase their individual payoffs, or share constrained resources [98]. A successful collaboration process requires agents to form an effective collaboration community 1 (i.e., teams or coalitions), which includes agents that are willing to support other community members by providing useful domain information and taking actions towards a shared goal [90].

Traditional approaches define static organizational structures for collaboration communities, where agents’ roles, dependencies, and communication protocols are defined [120]. However, implementing static solutions is infeasible when the variability of the environment and dynamism in the system increases [95]. Different techniques such as coalition formation [141], learning [129] and self-adaptation [33, 118] were introduced to enable agents to form/adapt communities dynamically. In these approaches, agents’ roles and dependencies are defined on the fly. However, such approaches are implemented mostly in closed systems where the numbers of agents operating in a system is constant and agents’ behaviour can be predicted or learned [31, 79, 141]. Therefore, these approaches are not practical when applied where agents’ behaviours is unpredictable, particularly in open systems with the following characteristics [99]:

- **C1**: agents are allowed to leave and join the system frequently and unpredictably;

1In this thesis, the term Collaboration Community is used instead of the literature’s common words such as teams and coalitions to avoid the confusion of concepts they imply such as only shared goal achievement for team and individual utility maximization for coalitions.
• C2: agents may have multiple goals (individual and shared goals) to achieve simultaneously, such goals might be conflicting at some stages.

• C3: agents’ internal architectures are not necessarily known to other agents;

• C4: agents are mostly self-interested, and are not willing to reveal all their information such as their detailed plan or action-selection process;

• C5: agents have incomplete knowledge of the environment and acquiring a complete domain knowledge is either costly or impossible;

• C6: implementing a central coordinator is infeasible.

An illustration, many of the current smart cities applications are implemented in open systems settings. Such systems include smart traffic management systems, where vehicles join and leave systems (e.g., roads of a each region) frequently. On-line betting or e-commerce systems are also examples of open systems, where the users of the system may not be same at each round of betting or transaction. Given the inherent uncertainties, dynamism, and lack of complete domain knowledge or central coordinator, agents need to constantly interact with each other to understand their dependencies and adapt their domain knowledge in a decentralised manner to cooperatively achieve their collaboration purpose (i.e., accomplish a complex task, achieve a shared goal, increase their individual payoffs, or share a constrained resource [10]).

1.2 Multi-agent System Model

In multi-agent systems, the agents’ organizational structure affects the performance and efficiency of the individual agents and their communities [95]. A multi-agent system organization can be defined using agents’ individual and collective behavioural and structural characteristics. These characteristics, such as agents’ dependencies, roles, and goals, determine the system’s and its elements’ behaviour [95]. Such organizational structures define key elements of an interactive community, such as communication protocols, coordination patterns, information flow, and resource allocation models [35, 94]. They specify whether or not an agent can communicate and
collaborate with another agent, how much information should be shared, who has access to the shared information, and how each resource is allocated to agents. For example, hierarchies are tree-shaped organizational structures in which agents only interact with their connected nodes in a specified direction [117]. The information flows from the bottom of the tree to its upper levels and the control flow direction is from the top of the tree to its lower levels. The purpose of a collaboration, the number of agents in the system, their goals, and the characteristics of the operating environment are the key factors that should be considered when an agents’ organizational structure is designed [95]. In closed systems with static aspects such as a constant number of agents, and a fixed set of goals, it is easy and efficient to define static structures such as hierarchies [73].

In open systems, defining static structures is impractical [95], because agents join and leave systems unpredictably, and having a coordinator agent to facilitate the communication between the agents is not the best option, given the fact that agents are not willing to share their information and join and leave a systems frequently. Despite the inherent uncertainty and dynamics of open systems and agents’ self-interest, agents need to interact and communicate with other agents in the system so that they can achieve their collaboration purpose (i.e., accomplish complex tasks, achieve a shared goal, increase their payoffs or share a constrained resource) [10]. In such systems, agent neighbourhoods are defined to establish a communication network structure for agents, facilitate information sharing, and implement the relevant constraint in real world applications, (for example, spatial constraints in wireless sensor networks) [43, 80, 84, 128, 191]. Neighbourhoods can be formed based on geographical locations (e.g., Wireless Sensor Networks, where agents are only neighbours with their next hop sensors), or resource distribution (e.g., smart grid scenarios, where the area served by a single transformer is considered as a neighbourhood). As shown in Fig. 1.1, current approaches have defined different system models, such as single neighbourhood system [191], multiple disjoint neighbourhoods in a system [80], or a system with overlapping neighbourhoods with transitive dependencies between agents [8]. Single neighbourhood systems allow all agents to communicate with each other. Although this solution does not limit agents to communicate with only a subset of operating agents in the system, it is not efficient when the number of agents in the system increases. To limit the number of
communications in large-scale systems, some approaches have defined disjoint neighbourhoods where agents can communicate only with their neighbours [80]. This assumption limits agents to operate in one neighbourhood and use only the resources available in that neighbourhood and collaborate with the neighbours in the same neighbourhood [84]. However, in real world applications, agents may operate alternatively in different neighbourhoods, for example when they could have better access to a resource in another neighbourhood. Some approaches define overlapping neighbourhoods in which agents can have transitive dependencies and can communicate only with the agents on which they have direct or indirect dependencies [82, 191]. However, this assumption is also limiting in real world applications where agents need to coordinate their behaviour and cooperate with all agents operating in overlapping neighbourhoods, when they share constrained resources to achieve their goals. For example in energy management systems, in multiple overlapping neighbourhoods setting, when the consumers want to decrease their energy usage, all consumers need to coordinate their behaviour and collaborate. However, according to existing neighbourhood modelling [82, 191], consumers can only interact only with the one they have transitive relations.

1.3 Multi-agent Collaboration Community Formation

Complex tasks accomplishment [90], sharing constrained resources [38], or achieving individual [140], or shared goals [88], are some of the purposes for agents to collaborate in multi-agent systems. For these purposes, agents need to form effective collaboration communities consisting of agents which are willing to collaborate and do not have any conflicting goals [90]. In such communities, a complex task is broken down to fine grained tasks and each agent will accomplish the assigned task to contribute to accomplish the complex task. However, forming such community is challenging particularly in open systems with self-interested agents, which share constrained resources to achieve multiple often conflicting goals simultaneously. The conflicting goals avoid successful collaboration as self-interested agents prefer their individual goals to the community’s shared goal. In such systems, modelling agents’ organization in the system (systems modelling), the collaboration’s purpose, and agents’ goal dependencies have impact on
The collaboration community formation process and the final community formed. As shown in Fig. 1.2, the collaboration community formation process includes 3 sub-processes [124]: Collaboration Purpose Identification, Participant Nomination, and Community Formation.

1.3.1 Collaboration Purpose Identification

Forming a collaboration community depends on the collaboration purpose. Agents need to know the reason for the collaboration to choose helpful agents. For example, when agents want to accomplish a complex task, they need to form a collaboration community of expert agents that...
are able to contribute towards the complex task accomplishment. The following are the main collaboration purposes addressed in the literature.

- **Complex task**: Accomplishing a complex task requires agents to collaborate when none of the agents can do the task alone. A complex task is broken into sub-tasks and each of them delegated to an expert agent. For instance, in supply chain management, a product...
is produced by delegating sub-products to service-provider agents. The sub-products are then composed to create the final product [127]. The complex task accomplishment purpose is not the focus of this thesis as it has been addressed well in the related work, and it is mostly implemented in fully cooperative environments, where agents work together to accomplish complex tasks.

- **Individual goal:** Self-interested agents may choose to achieve their individual goals in a collaborative manner, to increase their individual pay-off [141]. For example, in an electronic marketplace, autonomous agents combine their needs and buy goods in groups to decrease their own costs [159].

- **Shared goal:** Achieving a shared goal is another collaboration purpose for both self-interested and cooperative agents [60]. For example, in a robot soccer team, all agents (i.e., robots) collaborate to achieve the shared goal (i.e., winning the game) [17].

- **Constrained resources:** In multi-agent systems, sharing constrained resources (i.e., resources which have a limited capacity and can serve a limited number of users) is another collaboration purpose when agents require the constrained resource to achieve their goals or accomplish their tasks. For instance, in a smart grid application, the grid capacity is constrained and smart appliances collaboratively share the grid capacity to accomplish their tasks [87, 106].

### 1.3.2 Participant Nomination

All collaboration purposes need agents to form a collaboration community to accomplish each purpose. Collaboration community formation approaches can be categorized into static or dynamic approaches. In static approaches, a static community that will last for a long-term period is formed at design time. In such communities, the agents’ organization and their dependencies are defined during the formation process and will be preserved while the community exists [108]. In dynamic and open systems settings, dynamic collaboration community formation approaches are used. In these approaches, a collaboration community is formed dynamically when an agent
(i.e., initiator) identifies the collaboration need. The initiator agent needs to acquire some information about the agents currently operating in the environment to nominate a set of qualified agents for collaboration. Such information includes agents’ skills, and their shared and individual goals (selection criteria). To acquire this information, the initiator agent can use the agents’ organizational structure [80], which specifies their skills, roles and dependencies [95]. For example, agents level in a hierarchy can show the agents’ dependency to the higher rank agents in the hierarchy. When such a structure is not available, agents use other techniques such as learning techniques [31], simulation-based approaches, evolutionary algorithms [82], or social reasoning techniques [162] (see Fig. 1.2). Learning-based techniques are used to learn agents’ behaviour and dependencies by observing their interactions in the system over a long-term period. In simulation-based approaches, the initiator agent identifies possible communities, simulating all or a sub-set of possible communities by randomly adding and removing agents to an initial community. Simulation-based and learning-based approaches are used for both complex task accomplishment and individual goal achievement, when working in constrained resource systems [31, 85]. However, when agents do not stay in a system for a long-term period, these techniques are not useful. In such a case, agents use social reasoning approaches to gain the required information [47, 164]. Social reasoning techniques are useful for open environments, where there is no central coordinator and agents join and leave a system unpredictably. Such techniques are based on a social dependence notion, where agents present an external facade (i.e., external description) that includes a number of properties that they are willing to share with other agents in the environment. Agents use these external facades to update their domain knowledge and to adapt their behaviour accordingly. Using social reasoning, agents understand each others’ skills and action/task dependencies, and nominate a set of agents to form a collaboration community [160].

1.3.3 Community Formation

The main collaboration community formation approaches used in dynamic open environments are utility-based and complementary-based approaches [108]. During the collaboration community formation stage, agents decide whether or not to participate in a collaboration. In
complementary-based approaches, agents check their task/action dependencies and agree to collaborate if their actions are needed and no one else can take those actions [108, 133]. In utility-based approaches, agents decide whether or not to collaborate based on their individual pay-off [108]. These approaches either consider the shared goal or individual goal achievement.

In both collaboration community formation approaches, agents need a decision-making mechanism. The common mechanisms used in the literature are structure-based, initiator/coordinator-based, and negotiation-based. In structured-based mechanism [80], the agents’ organizational structure and their dependencies define how the decision-making process works. In the initiator/coordinator-based mechanism [6], the initiator agent coordinates the collaboration process including agents’ communication and information sharing. In the negotiation-based approaches [88], participants negotiate with each other or the initiator to find a mutual acceptance on a conflicting issue.

1.4 Collaboration Community Formation Research Gaps

In this section the research gaps in collaboration community formation are outlined.

- Systems Modelling: In current approaches, multi-agent systems are modelled as either a single neighbourhood, multiple disjoint neighbourhoods, or overlapping neighbourhoods with transitive dependencies [43,80,84,128,191]. The single neighbourhood model is limiting when the number of agents in a system increases. This results in a corresponding increase in computational and communication complexity for any collaboration community formation process (as according to C5 and C6, having a central coordinator is not feasible and acquiring domain knowledge in such systems requires a large amount of communication, which is expensive). Disjoint neighbourhoods and overlapping neighbourhood with transitive dependencies are also limiting as agents can communicate and collaborate only within their neighbourhoods or with the ones with which they have dependencies. This is a restricting assumption, as all agents in a system need to coordinate their behaviour and collaborate when they all share constrained resources to achieve their shared, individual goals. In overlapping neighbourhoods with transitive dependency modelling agents would
be able to interact only with agents on which they have dependencies (according to C5, agents only have incomplete domain knowledge and they may need to interact with agents that are not in their specified neighbourhood, or they may not have any dependencies with the one they need to interact).

To exemplify this limitation in current approaches, consider a town with multiple modes of public transportation: car sharing, city bikes, buses and trams. Citizens may use any of these resources. Some citizens have city bikes and trams monthly memberships, some just use car sharing, or others use all the travel modes alternatively. The limitation happens when disjoint neighbourhoods are formed for each transportation mode. Transportation modes often overlap in some areas of the city, and not allowing inter-neighbourhood collaboration is likely to result in a high rate of collaboration failure in a neighbourhood with an overloaded resource, and subsequent members’ dissatisfaction. For example, consider when there is a shortage of city bikes in a city centre area and the only option is to ask the city bikes’ members to collaborate to decrease demand in that area. Overlapping neighbourhoods could open up alternative choices for passengers if the resource request could be changed to other neighbourhoods, such as trams, and this would improve the performance of the overall transport system.

- Participant Selection Criteria: Current approaches consider either agents’ skills, their individual goals or the shared goal when selecting participants to form a collaboration community. This implies that they are concerned about either complex task accomplishment, shared goal achievement, or individual goal achievement. In these approaches, there are a number of limitations: (1) it is assumed that agents with multiple goals have to choose one of their goals to achieve at a time, (2) agents’ goal dependencies are not considered. However, as mentioned earlier in open systems’ characteristics (C2), agents should be able to achieve multiple goals simultaneously, and they should be able to adapt their activities to achieve multiple goals simultaneously [112]. To exemplify these limitations, consider a city in which all its citizens have a shared goal to decrease the city’s traffic jams, and a single passenger whose individual goal is for fast transportation. In this example, if the
passenger shares a ride with others, it is both contributing to the shared goal and achieving his own goal simultaneously. However, without considering multiple goals and their dependencies, he may end up choosing an option (e.g., taking a taxi) that achieves his own goals at the expense of the shared goal.

- Knowledge Acquiring: Agents’ organizational structure [80], learning methods [177], simulation-based methods [108], or social reasoning techniques [89] are used in the literature to acquire some knowledge about the agents. According to open systems characteristics $C1$, agents leave and join systems frequently, therefore it is not feasible to build and maintain a stable organizational structure for agents. The same characteristic makes learning and simulation-based approaches not useful, because when agents leave, the learned knowledge or simulation results based on their behaviour would not be useful. Therefore, social reasoning techniques are promising techniques in open systems, even though, they have some limitations. Social reasoning techniques enable agents to acquire information and understand their environment, available resources and other operating entities even when they do not have any background information. These techniques are mostly used for complex task accomplishment purposes and allow agents to reason about their action dependencies when breaking a complex task into sub-tasks. However, they do not address reasoning about agents’ goal dependencies when agents do not depend on each others’ actions (when they are not action-dependent), and share constrained resources to achieve their goals. In transportation example, all citizens that use a bus for commuting can take their actions independently from each other. However, they may need to coordinate their actions and collaborate to avoid resource overload (a bus can be considered as a constrained resource).

- Community Formation: The collaboration community formation approaches such as utility-based and complementary-based consider agents to be either self-interested or cooperative. However, agents in real world applications need to coordinate their behaviour and collaborate to advance their limited domain knowledge, particularly when sharing constrained resources to achieve their goals. Agents need to understand their goal depen-
dependencies to form communities that make a trade off between agents’ individual and shared
goals that cannot be achieved simultaneously. In the transportation system example, con-
sider a normal demand on buses in a city, when suddenly there is a terrorist attack in a
stadium, and a large number of buses must be sent to that location as quickly as possible
for evacuation purposes. Most of the main roads to/from the stadium should be cleared to
help deployment of the buses. In this case, if the goal dependency is modelled, individual
passengers whose goals are to travel using the main roads, can make a trade off between
their own individual goals and the shared goal. As a response to the emergency demand
at the stadium, such travellers could choose not to use the main roads, thereby helping to
achieve the shared goal (support the evacuation process).

• Decision-Making: In current approaches, agents either need to negotiate a solution or in-
teract with the initiator during the collaborative process. This means that agents must stay
(make commitments) in the system during the course of collaboration process. However,
in open systems setting (according to C1), agents should be able to leave the system when-
ever they want. They also need a mechanism to make decisions in a decentralised manner,
as having a central coordinator/initiator is not feasible in open systems (according to C5
and C6).

1.4.1 Research Questions

This thesis explores the question of how to enable agents to form collaboration communities to
achieve multiple goals simultaneously in resource-constrained, open systems. This question can
be decomposed into:

• (RQ1: Identify goal dependencies) How can agents acquire knowledge about each other
and identify their goal dependencies on other agents when sharing a constrained resource
in a system?

• (RQ2: Self-interest adaptation) To what extent agents adapt their level of self-interest, and
make decisions, when interacting with other self-interested agents in an open system?
• (RQ3: Operate in overlapping neighbourhoods) To what extent can agents operating in overlapping neighbourhoods be enabled to identify alternative solutions in other neighbourhoods?

1.5 Thesis Focus and Contribution

This thesis proposes a novel Collaboration Community Formation Model for agents with multiple goals in open systems, abbreviated as CCFOM. In particular, this thesis focuses on collaboration community formation when the collaboration purposes are achieving shared and individual goals simultaneously when sharing constrained resources in a system. Using CCFOM, agents model their goal dependencies and adapt their level of self-interest to form effective collaboration communities in overlapping multi-neighbourhood open systems. Figure 1.3 shows the contributions of this thesis in each sub-process of the collaboration community formation process, addressing the research gaps discussed in Section 1.4.

• Contribution 1: Knowledge acquiring and participant selection criteria (A New Social Reasoning Model): Current social reasoning techniques model agents’ actions and task dependencies. Such dependency models are used to compose agents’ skills to accomplish a single complex task. Although these models are useful for task accomplishment, they cannot address other collaboration purposes such as simultaneous shared and individual goal achievement when agents are not action-dependent and share constrained resources to achieve their goals. This thesis proposes a new social reasoning model that enables agents to model their goal dependencies, which is crucial when agents want to simultaneously achieve multiple goals and share constrained resources. To do so, each member of a neighbourhood shares a subset of information with its neighbours. Using this information together with the social reasoning model, each agent identifies its goal dependencies on its neighbours and builds a goal dependency model, which will be used during the participant nomination and collaboration community formation processes. This contribution addresses RQ1: Identify goal dependencies.

• Contribution 2: Collaboration community formation approach (A New Self-
Fig. 1.3: An Overview of CCFOM’s Contribution to Collaboration Community Formation

Evaluation and Adaptation Algorithm): Current approaches assume agents to be, exclusively, either self-interested, when increasing their individual payoffs, or cooperative, when accomplishing a complex task. However, in open systems, when self-interested agents with limited domain knowledge share constrained resources to achieve multiple goals simultaneously, they may need to dynamically adapt their level of self-interest. Using CCFOM, agents can dynamically adapt their self-interest level by evaluating their goal dependencies, and their progress towards their goals, during the course of their operation in a neighbourhood. This contribution addresses RQ2: Self-interest adaptation. The also includes a decentralised Decision-Making algorithm. In current approaches, the decision-making process has been only partially decentralised in negotiation-based and
coordinator-based approaches. In these approaches, an initiator agent initiates a community formation process and coordinates agents information sharing, negotiation and communication processes. It is also assumed that agents stay in the system during the course of collaboration. However, both these assumptions limit agents in open system setting, where having a coordinator agent is not feasible and agents can join and leave the system unpredictably. In CCFOM, a decentralised decision-making algorithm is presented that does not require a coordinator agent and agents make decisions independently and in a distributed manner, and can leave the system at any time.

- Contribution 3: **Support for Operating in Multiple Overlapping neighbourhoods (Cascade Request to an alternative neighbourhood)** In current approaches, a multi-agent system with multiple resources is modelled as a single neighbourhood, multiple neighbourhoods with transitive dependencies, or multiple disjoint neighbourhoods. The first two models are limiting when the number of agents increases. The last one is particularly limiting when agents cannot move between neighbourhoods to find better alternative solutions when the resource in one neighbourhood is overloaded. CCFOM models a multi-agent system as an overlapping multi-neighbourhood system, where agents can operate in more than one neighbourhood. In this model, agents operating in multiple neighbourhoods have the option to change their actions from one neighbourhood to another when they cannot achieve their goals (individual/shared) in the initial neighbourhood. This contribution addresses RQ3: Operate in overlapping neighbourhoods.

1.6 Evaluation

An application-independent simulator (general simulator) was developed to evaluate CCFOM under varying levels of agents’ mobility, and neighbourhoods’ density. CCFOM’s application effectiveness is also evaluated in two real world application case studies, Ride Sharing and Smart Grid. Both case studies implement open systems where agents join and leave systems freely and unpredictably, and use constrained resources to pursue multiple goals simultaneously. In the Smart Grid case study, the grid capacity is the constrained resource for a single neighbourhood
of houses, each with multiple electrical devices and an electrical vehicle. In each neighbourhood, electrical vehicles, represented by agents, collaboratively share the available capacity to achieve their goals simultaneously. Such goals include agents’ individual goals (i.e., get enough charge for their next journey) and their shared goal (i.e., decrease the number of times the transformer is overloaded). The Ride Sharing case study implements a multi-neighbourhood open system in which passengers and taxis, represented by agents, have varying types of dependencies and collaborate to achieve multiple goals simultaneously (e.g., for taxis: increasing their hired time, for passengers: decreasing their rides’ vacant capacity). Evaluation metrics include measurements of agents’ access to shared resources when operating in multiple neighbourhood setting, individual goal achievement, and collaboration success rate (shared goal achievement). CCFOM is also evaluated against state of the art utility-based [191] and complementary-based [8,55] approaches to compare the communication and computational cost.

1.7 Assumptions

This thesis makes a number of common assumptions in multi-agent systems that define the scope of the problem space. Application-specific assumptions are discussed in Chapter 5. Following are the general assumptions made during CCFOM’s design and implementation.

- All agents in the system have internal logic for action-selection and can carry out their actions independently.
- Agents can communicate directly with their neighbours in a neighbourhood.
- Agents are not malicious, and their shared information can be trusted.
- The domain is discrete, and both time and resources can be quantified.

1.8 Roadmap

The remainder of this thesis is organized as follows:
Chapter 1: Introduction

• Chapter 2 presents the background and related work in collaboration community formation in both closed and open multi-agent systems.

• Chapter 3 presents CCFOM, the proposed approach to Collaboration Community Formation for Agents with Multiple Goals in Open Systems.

• Chapter 4 provides the lower level details of CCFOM’s implementation.

• Chapter 5 describes the evaluation scenarios, application areas, and experimental set-up and results obtained.

• Chapter 6 presents thesis conclusion and provides possible directions for future work.
Chapter 2

Related Work

This chapter surveys current research that has addressed collaboration community formation classified by environment types, agent types, collaboration purposes. The survey presented in this chapter categorizes the current research into 3 main classes. (1) Dynamic Coalition Organization, addressing community formation of Self-interested agents to increase their individual utility gain. (2) Dynamic Team Organization, addressing community formation of cooperative agents to accomplish complex tasks. (3) Multi-agent Resource Allocation, addressing resource allocation in constrained resource environments.

The remainder of this chapter is organized as follows. The Background section presents an introduction to open multi-agent systems and multi-agent collaboration in Section 2.1.1 and Section 2.1.3. A survey of the state of the art with particular focus on collaboration community formation is presented in Section 2.2, Section 2.3 and Section 2.4. Collaboration community formation in closed systems is reviewed in Section 2.5. The chapter concludes with a discussion on the state of the art and outlines the requirements of this thesis contribution, Collaboration Community Formation Model for agents with multiple goals in open systems (CCFOM).

2.1 Background

This section provides the background information on multi-agent systems, agents types, and multi-agent collaboration, required for of the questions addressed in this thesis.
2.1.1 Open Multi-agent Systems

Open multi-agent systems consist of multiple autonomous entities, called agents, which use the available resources in the system and interact with each other to achieve their goals. The characteristics of such systems are [99]:

- Agents leave and join the system frequently and unpredictably.
- They have different goals and policies, which are likely to be conflicting.
- They are likely to be self-interested.
- Acquiring complete domain knowledge about the available resources in the system and operating agents is either costly or practically infeasible as agents are not interested in revealing all their information.
- Having a central coordinator to control all the agents is also practically infeasible.

2.1.2 Agent Types

Agents are categorized based on their internal structure as utility-based agents or goal-based agents [145]. They are also categorized as self-interested agents or cooperative agents, based on their behaviour [112].

1. Cooperative agents have autonomous behaviour, an individual domain model, communication and cooperative capacity, and spatial mobility [146]. The communication and cooperative capacity enables intelligent agents to exchange information to build their individual domain model. The possibility of communication enables agents to coordinate their behaviour and cooperatively take actions towards a goal. Cooperative agents are mostly considered to be benevolent agents that try to optimize the overall system’s performance [112]. A cooperative agent can be modelled as a goal-based agent. As shown in Fig.2.1, a goal-based agent evaluates the world’s current state and examines the world’s possible future (e.g., next timestep state) if it takes a specific action. It then uses its limited
knowledge (e.g., the world’s current state and its possible future states), and the information provided by its goals to decide its action sequence that will eventually result in a goal achievement. Planning, searching, and learning are some of the techniques that can be used during the action selection process [145].

Fig. 2.1: A Goal-based Agent’s Internal Structure [145]

2. Self-interested agents are competitive agents trying to optimize their local performance, or increase their individual payoff [112]. Being a self-interested agent does not mean that the agent wants to harm other agents, or that it only cares about its own benefit. It rather means that the agent has its own description of the environment that motivates its actions. This description can be defined by a utility function that quantifies the agent’s degree of preference across alternatives [158]. A self-interested agent can be modelled as a utility-based agent, which uses a utility function that maps its state to a real number that describes the agent’s level of happiness. Utility functions are essential when (1) an agent
has conflicting goals and only one of them can be achieved, (2) an agent has multiple goals to achieve, however, none of them can be achieved with certainty. In the first case, the utility function offers a trade off between the conflicting goals, and in the second case, the utility function provides a likelihood of success that can be weighed against the goal’s importance [145]. As shown in Fig. 2.2, similar to a goal-based agent, a utility-based agent evaluates the environment’s current state and possible future state if it takes a particular action. It then considers its own happiness under different situations that occur when taking different actions. The happiness level is calculated by the utility function and is critical to the action selection process.

Fig. 2.2: A Utility-based Agent’s Internal Structure [145]

3. Cooperative and Self-interested agents: In large and complex open systems, the distinction between self-interested and cooperative agents is blurred, as they have to deal with uncertainty [112]. In such systems, agents’ limited computational and communicational
resources lead to interactions between agents that results in complex interdependencies between agents’ activities. Therefore, self-interested agents tend to become more cooperative to access helpful information when communicating with others and achieve a better performance by cooperating with other agents [112].

2.1.3 Multi-agent Collaboration

Multi-agent collaboration is highly dependent on other multi-agent interaction concepts including cooperation, coordination and negotiation. Cooperation enables agents to work together towards a common benefit [90]. Coordination enables agents to cooperate by regulating their behaviours and actions in a shared environment [77], and negotiation is a mechanism that enables agents to reach a mutually accepted agreement [182]. According to the literature, multi-agent collaboration is a coordinated process in which participants try to mutually solve a problem or contribute to a shared benefit [29, 32, 143]. Alternatively, in a problem solving context, collaboration is defined as a coordinated, synchronous activity that is the result of a continues attempt to construct and maintain a shared conception of a problem [144]. Additionally, collaboration is defined as a coordinated activity in which participants work together to achieve a shared goal [90, 154]. According to all of these definitions, a collaboration process needs a set of agents to work together for a specific purpose. In the following sections, different collaboration purposes and collaboration community formation approaches that are used in the literature are reviewed.

2.1.3.1 Collaboration Purpose Identification

A need for collaboration emerges when an agent has a goal that it is not able to achieve alone, or when it prefers a cooperative solution [185]. Complex task accomplishment, individual goal achievement, individual payoff increase, shared goal achievement, and sharing constrained resources are the main collaboration purposes addressed in the literature. These purposes have been fulfilled using different community formation approaches, and have resulted in different community organizations.

To accomplish a complex task, a responsible agent (to which the task is assigned), needs to
identify possible collaborators, which can provide partial solutions to form a collaboration community to accomplish the task. According to the literature [60, 92, 93, 196], divide and conquer is the most preferred solution, during which the agent that has identified the need for collaboration divides the complex task into smaller and simpler sub-tasks and assigns them to expert agents. Each expert agent is then responsible for the assigned sub-task and can again use divide and conquer to make it even smaller and simpler and ask more agents to cooperate. This purpose has been mostly addressed in the context of team organization and teamwork [60, 92, 93, 196] and it is reviewed in both open and closed systems in Section 2.3 and Section 2.5.3, respectively.

Individual goal achievement and individual payoff increase have been addressed in the context of coalition organization, where agents join coalitions and coordinate their behaviour to increase their individual payoff [23, 100]. For example, in e-commerce a group of customers form a coalition when buying goods to minimize their costs [69]. This purpose has been mostly addressed in the context of utility-based coalition organization that is reviewed in Section 2.2.

Shared goal achievement has been formalised both as a teamwork activity [113, 147], and as a coalition activity [62] depending on agent types. For example, in robot teams, where robots are all cooperative agents, they coordinate their behaviour and carry out their tasks to accomplish a shared goal [78]. The related techniques are reviewed in team organization in Section 2.3. Shared goal achievement has been also studied in the context of coalition formation. For example, in energy grid systems, consumers form coalitions to coordinate their energy usage regarding to the available resource capacity to avoid overloading the grid and decreasing their energy price [81, 115]. The related techniques are reviewed in complementary-based coalition formation approaches, in Section 2.2.2.

Sharing constrained resources is an important issue in dynamic environments. Agents need to coordinate their behaviour and cooperate to get a better use of these resources. This has been implicitly studied in utility-based coalition organization methods as it can be modelled as a shared benefit. There are also other distributed approaches that study the resource allocation in dynamic environments using multi-agent approaches [68, 166] that are reviewed in Section 2.4.
2.1.3.2 Collaboration Community Formation

In multi-agent systems, agents are generally designed to be autonomous. Such agents can operate without any direct intervention or guidance of humans [184]. However, autonomous agents are not necessarily auto-sufficient, which means that they have limited capabilities and domain knowledge and need to communicate and collaborate with other agents for different collaboration purposes [57]. Therefore, autonomous agents need to form communities and work together to successfully fulfil their purposes. Collaboration community formation is a process in which a set of agents form a community organization and cooperate for a specific purpose. Community organizations can be formed once at design-time using agents’ static structures, or at runtime when the need for a collaboration emerges, where agents collaboratively form dynamic organizations that can be changed based on the changes in the environment.

Fig. 2.3 shows the structure of the literature reviewed in this chapter. Section 2.1.3.3 provides the analysis requirements used to analyse these approaches.

2.1.3.3 Analysis Requirements

In the following sections, a review of the current research with a focus on collaboration community formation process is presented and followed by an assessment on their selected process. The assessment requirements that map to research questions discussed in Chapter 1 are as follows:

- **System modelling**, assessing how the approaches model neighbourhoods in a system, particularly if they have addressed multiple overlapping neighbourhood in the system. This requirement is related to \( RQ3: \text{Operate in overlapping neighbourhoods} \).

- **Collaboration purpose**, particularly if agents are able to achieve multiple goals simultaneously, and if their goal dependency is considered. This requirement is related to \( RQ1: \text{Identify goal dependencies} \).

- **Agent types**, particularly if the agents are able to adapt their level of self-interest and cooperation. This requirement is related to \( RQ2: \text{Self-interest adaptation} \).
Resource dependency during the goal achievement process, particularly when the resource is constrained, and assess their ability to coordinate their behaviour and collaboratively share the resource. This requirement is related to RQ1: Identify goal dependencies.

### 2.2 Dynamic Coalition Organization

Dynamic coalition organizations are formed to help autonomous agents operating in open systems and dynamic environments to collaborate. In open/dynamic environments, agents need a mechanism to keep the coalition formation (community formation) and negotiation processes
uninterrupted while reacting to the changes in the systems in real-time. To do so, agents need to adapt their domain knowledge dynamically in these environments [108]. Therefore, the traditional game-theory based coalition organization approaches, which are concerned with coalitions’ stability (e.g., core-stable [192], shaply-stable [192], and kernel-stable [24]), are not applicable in dynamic and open circumstances [141]. Moreover, the optimality, and stability criteria defined in above mentioned approaches are not applicable in frequently changing open environment. To address the coalition organization in open environments two classes of approaches have been introduced [108]. First, utility-based approaches, where an agent’s aim is to increase its individual payoff, or achieve its individual goal by working with or getting help from others. These approaches calculate the coalition value, generate the coalition’s organizational structure and distribute the payoff between participants, which is a reward for every agent in the coalition. Second, complementary-based approaches, where agents’ skills are composed to accomplish a complex tasks. Complementary-based approaches have their roots in social-dependence theory and social reasoning [160]. Agents reason about their dependencies on other agents operating in their environment to form effective communities. The next two sections discuss these two approaches in detail.

2.2.1 Utility-based Coalition Organization

Utility-based approaches include two main steps, coalition organization formation, and utility distribution [105, 142, 150]. This thesis is concerned with coalition organization formation approaches. The main approaches used to form coalition structure are Simulation-based, Negotiation-based, or Adaptation-based techniques.

2.2.1.1 Simulation-based

In these approaches, possible coalitions are simulated and evaluated by a simulating agent. It then chooses a number of coalitions, and negotiates with the involved agents to finally confirm one of the coalitions [64, 86, 101, 187]. The simulation process can be implemented using different techniques such as evolutionary algorithms [87, 188], or genetic algorithms [190]. For example, Klusch et. al. define two types of agents (i.e., World Utility Agent (WUA), and Coali-
tion Leading Agent (CLA)) to address coalition structure formation [108]. WUA receives and maintains information about registered agents in the system. Such information includes agents’ capabilities, their quality of service, and reliability. CLA is a single agent that represents a coalition and is responsible for negotiation and distribution of the payoff. To build the coalition organization, the CLA agent simulates different possibilities for a coalition, selects a subset of simulated coalitions and negotiates with the selected agents to form the final coalition [108].

2.2.1.2 Negotiation-based

In this class of approaches, there is an initiator agent that starts the community formation by negotiating with the agents with the required capabilities to form a coalition [70, 88, 135, 142, 149, 178, 191, 194]. The initiator agent invites a subset of or all the available agents in the system to a negotiation process. Invited agents evaluate their own states and the benefit they may gain if they engage in the collaboration and send back their offers to the initiator agent. The initiator then evaluates all the offers, selects a number of them and forms a collaboration community. The agents, which have agreed to collaborate, need to achieve a mutual agreement on any conflicting issue.

2.2.1.3 Adaptation-based

Using adaptation techniques, agents constantly observe their environment and adapt their actions, decision-making mechanism, and the coalition’s organization according to the changes. Adaptation techniques enable agents to adjust their level of involvement in a coalition [88], and adapt the coalition organization [42, 56, 134, 157, 191]. In these approaches, it is assumed that an initial coalition organization exists and agents adapt the organization over time.

2.2.1.4 Analysis of Utility-based approaches

• **System Modelling**: In utility-based approaches, the system is modelled as a single neighbourhood systems where agents can communicate with all the other agents in the system directly (i.e., in adaptation-based, negotiation-based approaches) or through coordinator
agent (i.e., simulation-based approaches). Although modelling a system as one neighbourhood allows all agents to have access to all possible resources and communicate with all the agents in a system, it causes a large number of communication which is not feasible in large scale real world systems.

- **Agent type and Collaboration purpose**: In utility based approaches, agents are modelled as self-interested agents, which focus on increasing their own individual payoff, or achieving their individual goals by forming effective collaboration communities.

- **Resource dependency**: In these approaches, agents’ dependencies to constrained resources are considered. For example, in adaptation-based approaches agents adapt their behaviour and coordinate their actions according to available resources. Such agents are successful at achieving their individual goals. However, they are not as successful at achieving shared goals as they do not adjust their level of self-interest and cooperation.

- **Decision-making process**: In these approaches, decision-making process is not fully decentralized, as there is either an initiator agent or a central coordinator that facilitates the interactions between agents or makes the final decision. Although having such roles in a system can facilitate the interaction between agents, information sharing and resolving potential conflicts, defining a central coordinator is not feasible in open systems.

### 2.2.2 Complementary-based

Complementary-based approaches dynamically form coalition structures by composing agents’ complementary individual skills to enhance shared goal achievement in a collaborative manner [50]. These approaches use social dependence theory to enable agents to become aware of their social relations to show independent collective behaviour [58, 71, 104]. *Dependence Theory* and *Utility-based and Dependence Theory* approaches are focused on forming coalition of two agents with direct commitments, this idea is advanced further by introducing *Transitive Dependencies* to form coalitions of larger sizes. *Indirect Inference* is then introduced to allow agents to form overlapping coalitions. *Fuzzy Inference* then allows agents to define degrees of dependencies. *Social Dependence and Social Power*, distinguishes between individual power, which
is defined based on agents’ goals, action and resource access, and social power which is defined based on agents’ dependencies on other agents. *Practical Social Reasoning* formalizes a collaborative decision-making process. In the following sections, these approaches are discussed in detail.

### 2.2.2.1 Dependence Theory

In open environments, agents require a mechanism that enables them to adapt to changes in their environment, specifically, when agents join and leave the systems frequently. They also need to identify the agents operating in their system and their capabilities, so that they can collaborate. Social reasoning techniques use dependence theory to enable agents to reason about their dependencies on other agents operating in the same system [57].

Dependence theory is based on social power theory [36], which uses the concept of dependency relations to explicitly define agents dependencies. The dependency relations can be used as a decision criteria in community formation process [160]. The dependence theory and the models derived from this theory enable agents to understand their dependencies on other agents. To build a dependency model, each agent shares a data structure, called *external description*, with all the other agents in the system. An external description includes: Agents’ Goals, a set of goals an agent wants to achieve. Each agent chooses one of its goals to be active at a time. Actions, a set of actions an agent can perform. Resources, a set of resources to which the agent has access. Plans, which define a sequence of actions and the required resources to achieve a goal. From this external description, agents’ autonomy/dependency types are identified and a dependency model is built. Agents’ autonomy types are identified using agents’ plans and other information shared in their external descriptions. Different types of autonomy are defined for agents: agent $a_i$ is *$a$-autonomous* if it can take all the actions required to achieve its goals, $a_i$ is *$r$-autonomous* if it has access to all the resources it needs to achieve its goals. $a_i$ is *$s$-autonomous* if it is both *$a$-autonomous* and *$r$-autonomous*. When an agent is not autonomous, it depends on other agents to achieve its goals. The following are the different dependent agent types:

- Agent $a_i$ is *$a$-dependent* on agent $a_j$ for a specific goal, if it has an action in its plan that
it cannot take itself, and should be delegated to $a_j$.

- **Agent $a_i$ is $r$-dependent on agent $a_j$** for a specific goal, if it needs to access a resource that is controlled by $a_j$.

- **Agent $a_i$ is $s$-dependent on $a_j$**, if it is both $a$-dependent and $r$-dependent on $a_j$.

In these approaches, $a$-dependent agents build a dependency network that captures all the dependency relations between agents operating in an environment, including: **mutual dependence**, which means two agents depend on each other for the same goal, **reciprocal dependence**, which means two agents depend on each other for different goals, and **unilateral dependence**, which means a one way dependence exists and only one of the agents depends on the other at least for one of its goals.

### 2.2.2.2 Utility-based and Dependence Theory

In this approach, in addition to social dependencies, the cost and benefits of engaging in a cooperative process is also considered when forming a community. This work also advances the dependency types by adding **weak dependencies** to increase agents’ payoff and the possibility of collaboration [7]. **Weak dependencies** exists when agents are able to achieve their goals but prefer to cooperate with others to increase their payoff. In this approach, **AND/OR trees** are introduced, which calculate the cost of every action an agent can take for a specific goal using different plans (represented by the tree branches).

To follow this idea, it is argued that every effective collaboration community formation approach should increase both the quality and quantity of a formed community (coalition). Community’s quality can be achieved by using social reasoning approaches as they carefully study agents’ relationships and dependencies, and choose the agents that are most likely to collaborate. Communities’ quantity can be gained by using utility-based approaches, which study the possible options to increase agents’ payoff. This idea was developed by extending the **external description** to include cost and weight factors [54]. The weight factor specifies the importance or priority of a goal and the cost factor specifies the cost of each action in a plan.
2.2.2.3 Indirect Inference

In complementary-based approaches mentioned so far, agents apply social reasoning to understand their dependencies on agents to which they directly relate. They establish bilateral commitments for each of their activities, meaning that if one task is delegated to an agent and it has decided to delegate it to other two agents, the initial agent should have bilateral commitment with those two agents as well. These assumptions limit agents’ options on forming nested collaboration communities. They also do not address how they find a replacement for an agent which has failed to accomplish its assigned task. To address these issues, the indirect inference approach was proposed [122]. In this approach, each agent can adopt two different perspectives during social reasoning: a global perspective, that is acquired by reasoning about all the dependencies between agents that can possibly form a community (including agents involved in indirect delegations), and a local perspective, used when the agent does not have knowledge about all the agents in the system, and it has limited knowledge about how to pursue a goal. Using these perspectives, nested coalitions can be formed and agents have more options when choosing collaborators.

2.2.2.4 Fuzzy Inference

In most social reasoning approaches, dependency relations have exact definitions, whereas in fuzzy inference, it is possible to define varying degrees for each dependency relation [97]. Fuzzy inference-based approaches also argue that most social reasoning approaches consider the immediate benefits they gain from forming a collaboration community, which results in missing the opportunities where there may be benefits at a later time. In fuzzy inference approaches, the agent that wants to form a collaboration community, calculates a possibility value for possible participants, and invites agents with higher possibilities to form collaboration communities. The possibility value is calculated using variables such as the cost of taking an action or providing a resource, agents’ history of assistance in previous collaborations, agents’ willingness to collaborate, and their level of dependency. The same variable that is used for selecting agents to form a community, will be used by invited agents to decide whether or not to accept the request.
2.2.2.5 Social Dependence and Social Power

In this approach, a notion of social power [36] is used alongside social reasoning techniques [160] to form collaboration communities [34]. Agents’ individual power is defined using their goals, actions, resources, and plans, and the social power is defined by their dependency relations. Individual power can be executional (can-do), meaning that an agent knows how to achieve a goal, deontic (entitled-to), which indicates an agent’s permissions, and full power (total), when an agent has both executional and deontic powers. The social power types are: Dependence, when an agent depends on another agent for the execution of an action. Power-over, when agent $a_i$ believes that it depends on agent $a_j$, they say $a_j$ has power over $a_i$. Influencing-power, which implies that agent $a_j$ has influence power over $a_i$, when $a_i$ depends on $a_j$ for one of its goals. There are also another three power types that define agents’ power when they are part of a group, dependencies, authority relationships, roles and norms. Agents use all these power types to understand what would they gain or lose from joining a community.

2.2.2.6 Transitive Dependency

Transitive dependence-based approaches [8, 25, 110, 111] are based on previous social reasoning approaches and introduce a transitive dependence theory. Transitive dependence-based approaches argue that two partner communities are limiting and agents may need a third agent to accomplish a complex task. They clarify this idea with a supply-chain management example, where a consumer agent cannot get its required service from a provider agent unless a delivery agent is considered in this process. In this model, the external description is extended to include action dependencies between two actions to introduce transitive dependency theory. These dependencies are and-action dependency and or-action dependency. The first implies that both actions are required to achieve a goal, and the former implies that only one of the actions is required.
Chapter 2: Related Work

2.2.2.7 Practical Social Reasoning

Practical social reasoning approach formalises a collaborative decision-making process using social reasoning in four stages: Practical Starting Point, where the need for a collaborative behaviour is identified. Group Generation, where a group/community of agents will be formed to make a joint commitment to achieve a goal. Social Practical Reasoning, where each member of the group reasons about the course of actions that the group should take to fulfil its commitment. Negotiation, where agents agree on the actions that should be performed [131, 180].

In this approach, the group generation process deals with finding a number of agents that are helpful and make them to form a joint commitment to start a collaborative process. However, in open systems agents should be free to join and leave a system frequently and unpredictably. They also mention that mutual belief and mutual dependence are necessary to motivate agents to create joint commitment. However, when modelling a multiple neighbourhood system, the agents may not be motivated to make a joint commitment to collaborate, when they can find better alternative options in another neighbourhood and leave. During practical reasoning process, all agents need to find a sequence of actions that is needed to be taken by the whole group to achieve the shared goal. Although this process is successful when all agents have only the shared goal to fulfil, it does not consider agents’ individual goals and such goals and agents’ goal dependencies in general.

2.2.2.8 Analysis of Complementary-based approaches

In summary, the complementary-based approaches have mainly used social reasoning techniques to form two agent coalitions or multiple agent coalitions. Some of the social reasoning first attempts were focused on forming coalitions of two agents that were directly depending on each other to achieve a shared goal [7, 48, 49, 163]. All these approaches develop mechanisms to form communities of size two, by using social dependence theory and combining utility-based approaches and social reasoning. Such approaches use dependence graphs to model the dependencies between all agents in a system [160] and coalitions can be formed only when bilateral commitment exists between agents in a community. For example, as shown in Fig. 2.4,
the possible coalitions in two agent coalitions are the agents with direct link that have the same color. Although these approaches are successful when forming coalitions of two agents, applying these approaches in real world applications is impractical where multiple agents are needed in complex task accomplishment and shared goal achievement processes. Additionally, assuming agents to be committed to a particular course of actions in a coalition organization is not feasible according to open systems’ characteristics discussed in Chapter 1.

Social reasoning techniques are also used to form coalitions of more than two agents, introducing concepts such as group dependencies [14, 27], dependence strength and dependence situations [53], inferencing indirect dependencies [122], and dependence graphs for agents having different goals [59], incorporating agent and group intentions, using chains of exchanges and enforced agreements [26, 152], and transitive dependencies [8, 110]. In transitive dependency models, agents can form communities with all the agents in the system on which they have direct or indirect dependencies (For example, in Fig. 2.4 the green agents). However, in large scale systems, this results in a large amount of communication. In contrast, in indirect inference, agents are allowed to form nested coalitions/collaboration communities to accomplish a complex task. Using a local perspective, agents can access only a few agents (it is not clear in the model which agents are in a local perspective) and if those agents know other agents that can be helpful, they can form separate coalitions with those (for example in Fig. 2.4, in indirect inference, the white coloured agents may not be able to form qualified coalitions). However, this approach also has shortcomings, as using a local perspective in this approach may either result in a large amount communication when there are a lot of options available, or the agent may not be able to find a coalition because of its restricting local view and not having helpful links with agents in its local perspective.

- **System modelling:** Single neighbourhood, multiple disjoint neighbourhood or overlapping neighbourhoods with transitive dependencies are the common ways to model a system in social reasoning techniques for large scale application areas (Fig. 2.5 shows a schematic view of these models). Borrowing the global perspective, some approaches have considered a single neighbourhood for all agents in the system [191], meaning that all agents in a system have a global perspective and can communicate with all the
other agents. This modelling is particularly limiting in large scale systems causing a large amount of communication between agents. Using local perspectives, multiple disjoint neighbourhoods [80] are defined in which only members of a neighbourhood can communicate with each other. This model is also limiting as agents are limited only to their own neighbourhoods. In overlapping neighbourhoods with transitive dependencies, agents in a neighbourhood can communicate with each other and with the one they have a direct or indirect dependencies in other neighbourhoods [8]. Although this model addresses the shortcomings of the previous two models and allows agents to have the possibility of cooperating with agents in the other neighbourhood, it limits their options to the agents on which they have dependencies.

- **Agent type**: In approaches reviewed in this section, self-interested agents share a lot of information in their external description, including their detailed plans, and collaborate to accomplish a complex task. However, self-interested agents are not willing to share all
their information.

- **Collaboration purpose**: These approaches consider only *a- *dependent agents and capture agents’ action dependencies towards accomplishing a complex task, or achieving a shared goal. However, there are other types of dependencies, such as resource dependency and agents’ goal dependencies are not not addressed independently. Such agents may have multiple goals to achieve simultaneously (including shared and individual goals) that might be conflicting with other collaborating agents’ individual goals.

- **Resource dependency and Decision-making**: The assumptions such as agents’ action dependency and the fact that they can choose only one of their goals to achieve at a time, having the shared resource controlled by an agent, and making agents to commit to achieve a goal, made it possible for these approaches to have decentralised collaboration community formation processes. However, these approaches may not be useful in real world applications where the resources cannot be controlled by agents and agents with no action dependencies may have multiple often conflicting goals to achieve simultaneously.

### 2.3 Dynamic Team Organization

Team organization formation is mainly studied in the context of cooperative problem solving, in which finding potential team members, forming a team, designing a plan, and plan execution are the main steps. It is also studied in the context of teamwork where the complementary skills of cooperative agents are composed to accomplish a complex task [91, 124]. Team organization formation has employed methods such as game theory [155], optimisation techniques [107], and approximation and heuristic-based methods [76]. Assumptions made in most of these methods, such as a constant number of cooperative agents and having access to complete domain knowledge, make them inapplicable in open systems settings, and therefore, are not discussed further in this chapter.

Team formation has been also studied in more dynamic environments using different techniques such as organization-based, learning, planning, and ad-hoc teamwork.


2.3.1 Organization-based

Agents’ organizational structure in a system such as hierarchy or holarchy can be used for team formation. This idea has been particularly studied in dynamic team formation when structural dependencies between agents need to be considered to form a successful team [79, 80, 121]. Structure-based adaptation and performance-based adaptation strategies have been studied in this context. In a structure-based adaptation strategy, agents use a preferential attachment notion to adapt their network connectivity [41, 66]. Preferential attachment implies that the probability of connecting to a given node in a network is proportional to the number of nodes that are already connected to that node. A performance-based adaptation strategy, is based on performance and referral concepts [66, 193]. An agent that decides to adapt its connectivity, removes its connection to its most immediate neighbour with lowest performance and asks its neighbour with highest performance for a referral node with a view to making a new connection. This neighbour will refer its neighbour with highest performance to the requesting neighbour.

2.3.2 Learning-based

Several approaches address learning-based team formation, focusing on environments with access to global information [11, 109, 113]. In such approaches, agents can evaluate all the possible options before taking actions. The learning techniques are used for initiating a team formation process or deciding to join a team. Instead of designing specific team formation policies, learning algorithms help agents to learn effective team initiating policies. A learning-based approach has been applied to a social network of self-interested agents to form a team to accomplish a complex task [31]. In such settings, the social network restricts the number of teams an agent may join, as agents could join a team if they have a social binding with one of the current members of the team.

2.3.3 Planning-based

In the planning-based approaches addressing team formation, an initiator agent plays a key role in the whole collaboration process and is responsible for forming a team [4, 123]. The initia-
tor agent initiates the team formation process by assessing potential participants’ capabilities. It then negotiates with them about the conflicting issues or the utilities they may gain and finally plans for the team’s action selection order. The topic of multi-agent team formation has been recently studied in the context of realistic crowd simulation, which includes cooperation, coordination and action-control planning in open environments [195]. Hierarchical planning is used for global and local path planning for agents with limited ability crossing through an environment, aiming to avoid dynamic and static obstacles. Hierarchical planning includes three automatic planning layers, cooperative-task, coordination-behaviour and action-control. Team modelling cannot happen a priori and agents need to reason about their interactions and adjust their behaviour on the fly to form a successful team.

2.3.4 Ad-hoc team formation

Ad-hoc team formation studies the issues of team formation and collaboration amongst agents that have no background information about each other (unknown teammates) [18, 39, 168]. Agents are considered to be heterogeneous with different internal designs and different communication protocols and world models. As stated in [168], "the goal of ad-hoc team work is to create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members". This approach has been used in two-robot collaboration, where one robot is pre-programmed and the other is an adaptive agent that tries to adapt its behaviour according to the other robot. Similarly, this technique is used in the two player game-theory approaches in which the ad-hoc agents learn to maximise their payoff [169]. It has been also used where the ad-hoc agent leads multiple teammates in joint action selection [5]. The learning and planning approaches have been also used to learn the unknown capabilities and to plan a joint action [113, 186].

2.3.5 Analysis of Dynamic Team Organization

- **System modelling:** In dynamic team organization, it is assumed that either agents are able to communicate with all the agents in the system directly (e.g., the learning-based, ad-hoc
approaches) or through a coordinator (e.g., the planning-based, organization-based approaches). These assumptions are limiting when the number of agents increases, resulting in communication and computation costs.

- **Collaboration purpose** and **Agent type**: The main purpose of team formation is to achieve a shared goal. Cooperative agents use different techniques such as learning to understand various aspects of their environment and the behaviour of other agents. Such knowledge helps them to communicate with suitable agents, which can be helpful during shared goal achievement, where agents coordinate their behaviour, and take actions toward a shared goal. However, these approaches are concerned with only a shared goal and do not study the effects of cooperative actions and teamwork on agents’ individual goals. Moreover, they assume agents to be cooperative, whereas in real world applications in open systems, agents are supposed to be self-interested, and they may cooperate only when they can increase their individual payoff.

- **Decision-making**: In current approaches, team formation happens both using a coordinator, when an initiator agent exists, and decentralised, in the learning and ad-hoc approaches. Learning-based and ad-hoc approaches both work when the agents spend enough time in an environment, where they can learn each others’ behaviour, but this is not always possible, particularly in open systems with high frequency of leaving and joining of agents.

## 2.4 Multi-agent Resource Allocation

Multi-agent collaboration in constrained resource environments has been studied in the context of multi-agent resource allocation, in which cooperative agents schedule resource usage in a distributed manner to avoid shared resource overload [10, 40, 63, 176]. Distributed scheduling and finding an allocation requires agents (i.e., consumers) with varying preferences, cooperatively decide how to share a constrained resource. The allocation processes varies depending on the environment, resources’ types, agents’ preferences, roles, authorities, and the allocation objective [40]. The environments’ settings varies from being closed and predictable to open and
unpredictable. Resource types can be continuous or discrete, divisible or indivisible, and shareable or not shareable. Agents express their preferences over a resource allocation. Agents’ roles in different approaches vary from being a controller that controls a resource and schedules the allocations, to a computational entity that wants to use the resource and cooperate with other agents to find an allocation. Finally, the objective of the resource allocation process is either to find an optimal allocation or a feasible allocation (meaning that all the activities or tasks that need the resource to complete, get completed at the expected time). Agents’ preferences social welfare modelling, and negotiation techniques are the main concepts considered in multi-agent resource allocation literature. These concepts are as follows:

2.4.1 Agents’ preferences

Borrowing a game-theoretic view, a preference structure represents agents’ preferences over a number of alternatives, which leads to four families of preference structures [40]:

- Cardinal preference structure, which consists of a utility function that calculates a set of numerical or qualitative values for possible options.

- Ordinal preference structure, which includes a binary relation of alternatives. Prioritised Goals [136, 179], and Ceteris Paribus preferences known as CP-nets [13, 116] are the main approaches.

- Binary preference structure, which partitions a set of alternatives in to good and bad subsets.

- Fuzzy preference structure, which defines the degree to which one alternative is preferred to another.

2.4.2 Social Welfare

Social welfare measure the quality of a resource allocation. Its measuring factors include pareto optimality, collective utility function, leximin ordering, generalisation, envy-freeness and normalised utility [156]. In this section, only collective utility function metrics are reviewed, as this
thesis focus is not on resource allocation problem, but rather it focuses on agents’ goal achievement when sharing constrained resources. Agents’ resource usage and the fairness of their access to the resource in this thesis can be modelled as a collective utility function. The main collective utility functions are as follows:

- **Utilitarian Social Welfare** is the sum of individual utilities. This metric measures how much of the resource is used, and compares it to the overall available resource to measure the unused or idle resource [19].

- **Egalitarian Social Welfare** finds the worst utility, gained in a distribution. This metric is particularly helpful to evaluate the fairness of an allocation [126].

- **Nash Product**, which is the product of agents individual utilities. This metric can measure both overall utility gain and inequality in an allocation [52, 126].

### 2.4.3 Negotiation

Negotiation techniques such as Contract-Net Protocols [88, 167] are needed to support agents when discussing possible allocations in distributed approaches.

Contract Net Protocol (CNP) is a decentralised task assignment or resource allocation protocol widely used in multi-agent systems [167]. CNP was first designed for cooperative agents, however, it evolved over the years to be more flexible, adaptable, and applicable for both cooperative and self-interested agents [148], CNP includes four main stages, *Announcement, Bidding, Assignment, and Confirmation*. It also defines two roles for agents, initiators/managers, and participants/contractors. An initiator is responsible for announcing the available task and monitoring the task execution and evaluating the results. Participants are responsible for task execution. When a request for task execution is sent by the initiator, the participants bid for the task. After receiving all the bids, the initiator sends the request to the participants that offered the best bids. CNP has been extended over time to advance its flexibility, adaptability and performance [20,21,132,189]. Particularly, there are some extensions to CNP that are designed to work with self-interested agents. For example, a pricing mechanism was introduced to enable CNP for both cooperative and self-interested agents. Another extension was developed to allow agents to
leave an ongoing commitment, which is very much likely in open systems [151]. A FIPA standard CNP was later introduced and extended to allow multi-round iterative bidding [137].

2.4.4 Analysis of Multi-agent Resource Allocation

- **System modelling and Agent type**: In multi-agent resource allocation approaches, agents are considered to be controller of the resources in the system. The system is either broken into subsystems each being controlled by a single agent or a group of agents. Agents are collaborative, each have access to the resource provided in their own subsystem and collaborate with other agents in other subsystem to achieve their collaboration purpose.

- **Collaboration purpose**: Decreasing communication and computation cost, optimizing resource’s price and service time, task distribution efficiency, and revenue maximization are the collaboration purposes addressed in 36 distributed resource allocation approaches reviewed in a recent survey [139]. All these objectives are set to enable agents to share constrained resources, but they do not consider agents’ shared or individual goals and their dependencies in resource allocation process.

- **Resource dependency**: In resource allocation approaches, agents’ preferences determine how they categorize and understand the dependency between their own goals, for example by calculating a utility value or grouping the alternative options to good or bad. However, this does not enable them to understand their goal dependency on other agents operating in their systems. This is particularly limiting when agents have both shared and individual goals, and not knowing the dependency between their individual goals may cause taking wrong actions. Moreover, although the social welfare factor evaluates the fairness of the resource distribution, having a fair distribution of a shared resource is not the only concern when agents have multiple goals with different dependencies and priorities.

- **Decision-making**: An optimized solution for constrained resource allocation has been proposed in centralized approaches [12, 114]. However, these approaches are not applicable in large scale open systems as they need access to complete domain knowledge, and also they are NP-hard relative to the number of participants [16]. To address these
limitations, agent-based decentralized approaches are used, where agents make local dec-
cisions using their imperfect knowledge and other agents’ knowledge through negotia-
tion [10, 10, 40, 63, 176]. In these approaches, there is a coordinator agent to facilitate
the negotiation process, which is not feasible in open systems setting. Although these ap-
proaches are successful at distributing constrained resources, they do not consider agents’
goal dependencies and their simultaneous goal achievement.

2.5 Multi-Agent Community Formation in Closed Systems

There are three classic and prominent multi-agent collaboration theories used in closed multi-
agent systems, Joint intention-action, SharedPlans, and Teamwork. These three theories were
selected to be briefly reviewed in this section because of their originality and acceptance by
the multi-agent community [183]. This section also reviews organization-based collaboration
community formation approaches used in closed systems.

2.5.1 Joint intention-action

Joint intention-action theory is formed based on a Belief-Desire-Intention (BDI) model. BDI was
introduced in 1987 [28], where agents’ behaviour is considered controllable using three main
concepts: belief, desire and intention. Belief is an agent’s knowledge about the environment,
desire is its high level goal, and intention is a short term goal [28]. Joint intention-action is a
joint commitment between agents towards a shared goal. In this model, agents take collective
actions and keep a shared mental model [46]. The joint intention-action model is one of the first
formally established theories of multi-agent collaboration [44–46, 102, 143]. This theory is used
for collaborative problem solving in distributed systems, where agents’ skills are composed to
solve a problem (complex task). Joint intention-action highlights the following features for a
collaborative process [46, 103], where agents must:

• agree on a shared goal;

• agree on collaboratively achieving the shared goal;
• agree on a shared plan to achieve the shared goal;

• acknowledge the actions performed by other agents are related;

• have criteria to check the rationality of their commitments;

• and have some specified rules to know how to behave both when joint-action is progressing as planned and when it is not.

Although this theory is very well established, and is used for collaborative problem solving in decentralised systems, its principles such as sharing all intentions, beliefs, and plans, and commitment on taking specific actions are not compatible with open systems’ characteristics specifically with C1, C2, C3.

### 2.5.2 SharedPlans

The SharedPlans [9] theory describes collaboration as a collective activity amongst a group of agents aiming to achieve a specific goal. It requires participating agents to mutually believe in having a shared goal, agree on the action sequence and have an intention to perform the assigned actions [9]. A community of agents have SharedPlans for taking an action when they mutually believe that all members of the community are committed to take the action [125].

Similar to joint intention-action theory, SharedPlans’ principles are not compatible with open systems’ characteristics. These two theories also do not consider agents with multiple goals, their actions’ dependencies, and constrained resources in the system, when designing the plans for a shared goal achievement.

### 2.5.3 Teamwork

A team of agents is a group of cooperating agents, committed to work together to reach a common goal [60,75,173]. In teamwork theory, team members share their goals, and plans, and have access to complete domain knowledge. They inform each other of their own capabilities. They share their intentions to execute a shared plan towards a shared goal. Teamwork starts when an
Chapter 2: Related Work

agent realizes that it cannot achieve a goal by itself [172]. Both joint intention-action and Shared-Plains theories have been used in teamwork. STEAM [173] and TEAMCORE [138, 174, 175] are examples of the Joint intention-action theory’s implementation in teamwork. SharedPlains has been also used in a teamwork framework, RETSINA, as a reasoning mechanism [171].

These teamwork models based their work on collaborative environments, and their aim is to form a group of agents to take actions towards a shared goal. These models do not consider agents with multiple or conflicting goals when forming teams. Although the original teamwork theory is not applicable in open environments, variations of teamwork theory have been developed and applied in open environments, and were discussed in Section 2.3.

2.5.4 Organization-based

A multi-agent system organization can be defined using agents’ individual and collective behavioural and structural characteristics. These characteristics, such as agents’ dependencies, roles, policies, intentions, and goals, determine the behaviour of a system and its organizational structure [95]. Such organizations define key elements of an interactive community, such as communication protocols, coordination patterns, data flow, and resource allocation [35, 94]. It specifies which agents can communicate with each other, how much information should be shared, who has access to the shared information and how the system’s resource is allocated to each agent. Agents’ population, goals, and environment’s characteristics, are the key factors that should be considered when the system’s organizational structure is designed. The main categories of multi-agent organizations are hierarchies, holarchies, coalitions, and teams [95]. In Section 2.2 and Section 2.3, coalition and team structures are reviewed extensively for collaboration community formation in dynamic and open systems. In this section, hierarchies and holarchies that are useful structures in closed and static systems, are briefly reviewed.

1. Hierarchy: A hierarchical organization is a tree-shaped structure in which agents only interact with their connected nodes in a specified direction [117]. Hierarchy structures specify the data flow, control flow, agents’ cooperation partners, and task distribution. Hierarchy structures can implement a global goal tree in which all the agents will be assigned a number of tasks, contributing to the global goal achievement. Such structures
use the notion of task decomposition and divide and conquer to break complex tasks into smaller chunks. The contract net protocol is one of the common approaches to create a long-term hierarchy structure at runtime. In the contract net protocol, an agent decomposes a task into subtasks and contracts it to other agents in the system. Each contractee can itself decompose the assigned task to even smaller chunks and contract it to other agents in the system. This is how a hierarchy of agents is created to serve the initiator agent to accomplish its complex tasks or achieve a shared goal [148].

A hierarchy structure restricts agents’ individual efficiency as they have to keep the hierarchy’s structure including the data and control flow directions for a long-term period. A hierarchy structure also affects both agents’ individual and overall performance in a system. For example, in flat hierarchies agents may get overloaded with too many tasks, and in tall hierarchies there will be a huge delay as the data is passed all the way from the bottom to the top of the hierarchy. These drawbacks are exacerbated in open systems, where agents have their own individual goals to pursue, so there is no reason they would stay in a hierarchy to serve the purpose of the apex agent. Additionally, agents’ frequent joining and leaving make hierarchies a fragile structure and impractical for open systems.

2. Holarchy: Holarchy is a multi-level, group of hierarchies [73]. In a holarchy, agents collaborate within their groups and contribute to the properties of the other groups in the higher levels of the hierarchy. Holarchy is a specific type of hierarchy in which sub-trees can be substituted with a single node and the data and control flows are similar to the hierarchy. Forming the building blocks of holarchies, known as holons and selecting the appropriate agents for each individual holon is challenging. Similar to hierarchies, holons should be formed to serve a shared goal. Defining static holons and a coordinator holon to maintain and adapt the holarchy structure during run time is a common approach to form communities using holarchies. Contract net protocol [96, 130], self-organizing and self-adaptive approaches are also used to form holons [83].

Holarchy organizations, unlike hierarchies, allow more autonomy at individual level, as agents can be member of more than one holarchy. However, the control flow is still from
the top of the hierarchy to its lower levels. Such organizations limit the flexibility and functionality of a community, as agents are allowed to interact only in specified directions. Additionally, they have a central coordinator, as all of the agents depend on the coordinator holon at the top of the structure.

3. Congregations: Congregations are groups of agents with similar or complementary characteristics [30]. A congregation is an organization in which the possible coalitions and teams are formed to remain for a relatively long term period. However, it does not address the type of communities in which agents might have conflicting goals, and are not able to achieve multiple goals simultaneously.

### 2.5.5 Analysis of Multi-agent Community Formation in Closed Systems

Using these theories, agents successfully coordinate their behaviour and cooperate to achieve shared goals in a collaborative process. Both Joint intention-action and SharedPlans theories discuss essential aspects of collaborative processes and how agents should behave when taking actions in a collaboration. Teamwork theory, discusses ways to form a team of agents and uses Joint intention-action and SharedPlans theories to manage the collaboration process in a collaborative environment. However, assumptions made in these theories such as considering agents to be cooperative, sharing plans and intentions, making commitments and having access to domain knowledge make them inapplicable in open environment. In organization-based approaches, static aspects of the structures, such as data and control flow and having a central coordinator are the main reasons that make these organizations inapplicable in open systems.

### 2.6 Summary and Analysis

In this section, a comprehensive analysis is presented for the collaboration community formation approaches implemented in open systems, (reviewed in Section 2.2 and Section 2.3). The analysis criteria in this section are based on the analysis requirements presented in Section 2.1.3.3. Table 2.1 classifies all the collaboration community formation approaches presented in Section 2.2 and Section 2.3.
<table>
<thead>
<tr>
<th>System modelling</th>
<th>Agent type</th>
<th>Collaboration purpose</th>
<th>Dependency modelling</th>
<th>Decision-making</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulation-based</strong></td>
<td>Single neighbourhood</td>
<td>Self-interested</td>
<td>Individual Goal</td>
<td>r-dependent</td>
</tr>
<tr>
<td><strong>Negotiation-based</strong></td>
<td>Single neighbourhood</td>
<td>Self-interested</td>
<td>Individual Goal</td>
<td>r-dependent</td>
</tr>
<tr>
<td><strong>Adaptation-based</strong></td>
<td>Single neighbourhood</td>
<td>Self-interested</td>
<td>Individual Goal</td>
<td>r-dependent</td>
</tr>
<tr>
<td><strong>Utility-based &amp; Dependence</strong></td>
<td>Single neighbourhood</td>
<td>Self-interested</td>
<td>Individual utility &amp; Complex Task</td>
<td>a-dependent</td>
</tr>
<tr>
<td><strong>Transitive Dependence</strong></td>
<td>Single neighbourhood</td>
<td>Self-interested</td>
<td>Complex Task</td>
<td>a-dependent</td>
</tr>
<tr>
<td><strong>Indirect Inference</strong></td>
<td>Multiple overlapping Neighbourhood</td>
<td>Self-interested</td>
<td>Complex Task</td>
<td>a-dependent</td>
</tr>
<tr>
<td><strong>Organization-based</strong></td>
<td>Multiple overlapping Neighbourhood</td>
<td>Cooperative</td>
<td>Complex Task</td>
<td>role dependency</td>
</tr>
<tr>
<td><strong>Learning-based</strong></td>
<td>NA</td>
<td>Cooperative</td>
<td>Complex Task</td>
<td>a-dependent</td>
</tr>
<tr>
<td><strong>Planning-based</strong></td>
<td>Single neighbourhood</td>
<td>Cooperative</td>
<td>Complex Task</td>
<td>a-dependent</td>
</tr>
<tr>
<td><strong>Ad-hoc</strong></td>
<td>Single neighbourhood</td>
<td>Cooperative</td>
<td>Complex Task</td>
<td>a-dependent</td>
</tr>
</tbody>
</table>
• **System modelling:** System modelling studies how agents are partitioned in a system. According to the approaches reviewed in this chapter, a system can be modelled as a single neighbourhood, multiple disjoint neighbourhoods, or multiple overlapping neighbourhoods with transitive dependencies. As summarised in Table 2.1, in utility-based, dependence theory, transitive dependency, planning-based and ad-hoc approaches, the system is modelled as a single neighbourhood. As explained before, single neighbourhood systems results in a large amount of communication between agents. Additionally, using single neighbourhood model for community formation in large scale systems is computationally expensive, as the number of possibilities increases. The indirect inference and organization-based approaches model the systems as multiple overlapping neighbourhoods. The collaboration community in these approaches is formed based on agents roles and dependencies. Organization-based approaches are limiting and not compatible with open systems’ characteristics when there is an agent at the top level of an organization (e.g., a hierarchy) that plays a coordinator role. Indirect inference approach is dynamic in terms of defining overlapping neighbourhoods, however, it allows agents to form intra-neighbourhood communities when a direct or indirect dependency exists between them. This assumption also leads to a large amount of communication in large scale systems. The learning-based approaches do not explicitly model the system.

• **Agent type:** As shown in Table 2.1, agents are considered either being self-interested or cooperative. Although some of the approaches consider individual utility gain as an incentive to encourage agents to show cooperative behaviour, none of these approaches model individual goals and self-interest level adjustment.

• **Collaboration purpose:** Collaboration purpose in utility-based approaches is individual goal achievement or increasing the individual payoff, where as in the complementary-based and team organization approaches, is complex tasks accomplishment or shared goal achievement. There is only one variation of the complementary-based approaches (e.g., utility-based and dependence theory) where both shared goal and individual agent’s utility (payoff) is considered. However, this is only considered as an incentive to motivate the
individual agents to participate in a collaboration. These approaches cannot balance their
goal achievement as they are not designed to capture the dependencies between their goals
or achieve multiple goals simultaneously.

- **Dependency modelling:** Agents’ dependencies in the utility-based approaches include
  resource dependencies (r-dependent), and other types of dependencies are not consid-
ered. In the complementary-based approaches, agents’ action dependencies (a-dependent)
  are considered. In the team organization approaches either role dependencies or action
  dependencies are considered between agents. In the utility-based approaches, agents try
to increase their individual payoff, or achieve their individual goals using the available
resource. On the contrary, in the complementary-based and team formation approaches,
agents collaborate to achieve a shared goal. In none of the approaches simultaneous mul-
tiple goal achievement is addressed, and only individual goal or shared goal is fulfilled.

- **Decision-making:** Different decision-making processes are used in the reviewed ap-
  proaches. This process is implemented using a coordinator or in a decentralised manner.
  In approaches where a coordinator/initiator agent exists, the coordinator initiates, coor-
dinates and monitors the process from the start to the end, all the agents are dependent
  on the initiator. However, having a coordinator agent is not practical in open systems. In
  adaptation-based, learning and planning approaches, the decision-making process is de-
  centralised, where agents use learning techniques to understand the behaviour of other
  agents in the system in a long term period, and make decisions based on the learned
  knowledge. However, approaches such as learning are not helpful when agents frequently
  leave and join a system. In the complementary-based approaches, decision-making is also
decentralised, where agents are required to make commitments when forming a collabo-
ration community, guaranteeing that they take the specified required actions and stay in
the neighbourhood while the collaboration is not terminated. However, these assumptions
limit agents’ frequent leave and join in an open system.
Chapter 2: Related Work

2.6.1 Summary

Systems cannot be always modelled as a single system or distinct subsystems, having overlapping subsystems (overlapping neighbourhoods) is one of the ways that large-scale systems are modelled. Defining the interaction protocols for the agents/entities operating in such overlapping subsystems is important. Such protocols define to which other entities they can interact with, cooperate or collaborate.

Current collaboration community formations approaches model the system as a single neighbourhood, multiple disjoint neighbourhoods or multiple overlapping neighbourhoods with transitive dependencies. Neighbourhoods are defined to decrease the communication complexity. However, single neighbourhood model causes a large amount of communication. Disjoint neighbourhoods limit agents to a small number of options provided in their own neighbourhoods, as in real world applications with multiple constrained resources (e.g., traffic systems) it is not practical to define disjoint neighbourhoods. Overlapping neighbourhoods with transitive dependencies are also limiting as they either have a high communication cost as all agents will be linked to each other in some systems, or agents may not be able to form a community because of their restricting transitive dependencies. Therefore, modelling a system that allows agents to operate in multiple overlapping neighbourhoods is needed, which enables agents to identify alternative solution in other neighbourhoods with a reasonable communication cost is essential. This requirement is related to the \textit{RQ3: Operate in overlapping neighbourhoods}, and will be addressed in the next chapter.

Current approaches form collaboration communities to achieve either: (1) agents’ individual goals (increasing individual payoff) by modelling their resource dependencies, or (2) their shared goals (accomplishing a complex task) by modelling their action dependencies. Current approaches do not consider agents’ goal dependencies, and model agents to be either self-interested or cooperative. However, in real world applications with constrained resources, self-interested agents may have multiple goals (both individual and shared goals) to achieve simultaneously using such resources. Therefore, agents need to understand their goal dependencies, so that they can form effective collaboration communities. This requirement is related to the \textit{RQ1: Identify goal dependencies}, and will be addressed in the next chapter.
Agents also need a mechanism to adapt their level of self-interest and cooperation based on their goal dependencies to achieve their goals simultaneously when sharing constrained resources. This requirement is related to the RQ2: Self-interest adaptation, and will be addressed in the next chapter.

In the collaboration community formation approaches, decision-making process is facilitated through either: (1) a coordinator agent which facilitates the community formation process, information exchanges and final decision-making. However, having a central coordinator is not feasible according to open systems’ characteristics. (2) having simplified assumptions such as single goal achievement and requiring agents to make commitments when forming a collaboration community in a decentralised approach possible. However, agents need a decentralised mechanism that allows them to achieve multiple goals and leave and join the system unpredictably, without making any commitments. This requirement is related to RQ2 and will be addressed in the next chapter.

Next chapter presents the design of CCFOM and how it addresses RQ1, RQ2, RQ3.
Chapter 3

Design

The state of the art in collaboration community formation in open multi-agent systems, presented in previous chapter, has identified and discussed a number of limitations in current approaches. These limitations are: (1) current approaches model the system either as a single neighbourhood system, multiple disjoint neighbourhoods, or overlapping neighbourhoods with transitive dependencies, which either results in high communication cost or limits the possible collaboration communities that can be formed (considering the characteristics of open systems in Section 1.1), (2) current approaches consider either agents’ shared or individual goals when forming collaboration communities, (3) they consider agents to be either self-interested or cooperative. However, in open systems, self-interested agents need to balance their level of self-interest and cooperation to achieve multiple goals simultaneously, particularly because they have access to limited domain knowledge and share constrained resources. (4) agents’ decision-making process is only partially decentralised, and agents need to stay in the system during the course of collaboration, which is also limiting in open systems where agents should be able to leave and join a system unpredictably, at any time.

This chapter introduces the main contribution of this thesis, Collaboration Community Formation Model in open multi-agent systems, CCFOM. It discusses the design objectives of CCFOM, from which design requirements are derived. This chapter then presents the design of CCFOM, and shows how the design addresses the requirements, which can be mapped to re
search questions in Chapter 1.

3.1 Design Objectives and Required Features

As described in Chapter 1, this thesis requirements are to design and build a collaboration community formation model that allows agents to: (1) operate in multiple overlapping neighbourhoods, (2) simultaneously achieve both their individual and shared goals, (3) adapt their self-interest level when forming collaboration communities, and (4) make decisions in a decentralised manner. To meet these requirements, following features (contributions) are required:

- (Contribution 1) A new Social Reasoning Model: In open systems, agents leave and join systems frequently, and so they need a mechanism to acquire information about other agents operating in their neighbourhood to understand their goal dependencies. This is critical when forming collaboration communities of agents that share constrained resources, to achieve multiple goals simultaneously.

- (Contribution 2) Self-evaluation and Adaptation Algorithm: In open systems, self-interested agents have limited domain knowledge and share constrained resources with other agents operating in their neighbourhood. Such agents should be able to adapt their level of self-interest and collaborate with other agents in their neighbourhoods to achieve shared and individual goals. Agents require a decentralised decision-making technique, which does not require a coordinator agent, any commitments, or negotiation during the course of collaboration.

- (Contribution 3) Support for Operating in Multiple Overlapping Neighbourhood: Agents should be able to operate in multiple neighbourhoods and their interactions with agents in each neighbourhood should not be limited to their direct or indirect (transitive) dependencies.
3.2 Collaboration Community Formation Model (CCFOM)

CCFOM addresses collaboration community formation for agents with multiple goals that are allowed to operate in more than one neighbourhood. Agents, which are members of a neighbourhood, share a subset of their information with their neighbours. Using the shared information and the social reasoning model presented in this chapter, agents can build their goal dependency model. A goal dependency model enables agents to determine their dependency relations such as conflicting, competing, or collaborating. Using their dependency models, agents can adapt their level of self-interest and cooperation when operating in an open system with constrained resources to achieve multiple goals simultaneously. For example, they can behave in a neighbourly fashion, by cascading their resource request to another neighbourhood or choosing not to use the resource if they can afford not to, when the shared resource is overloaded, to achieve a better collective result (e.g., decreasing the number of times the shared resource is overloaded).

CCFOM is a distributed process that agents can run autonomously. Agents interact with each other and use each others’ information to coordinate their behaviour. In a CCFOM process, agents with multiple neighbourhood memberships are enabled to find alternative solutions in other neighbourhoods by changing the resource request to alternative neighbourhoods, when the resource in the initial neighbourhood is likely to be overloaded and a shared goal is not achievable. As shown in Fig. 3.1, CCFOM has six stages. In Fig. 3.1, processes and sub-processes are shown by the rectangles and the arrows between these shows the sequence between these processes. CCFOM starts from the Neighbourhood Membership Update process/stage and the arrows then show how this stage is linked to the rest of the stages/processes. Additionally, the clouds on this diagram show the models that can be formed at each stage. The cloud shape is chosen to show the volatility of these models due to agents’ frequent join and leave.

- **Neighbourhood Membership Update:** At this stage, neighbourhoods are formed around resources. Each neighbourhood has a single resource and all agents that use the resource are members of that neighbourhood. Agents announce their joining or leaving, to update other agents operating in the same neighbourhood.

- **Collaboration Need Identification:** Once agents join a neighbourhood, they take actions...
that may require the use of the shared resource (agents’ actions are modelled as boolean values, the action value is true, when agent wants to use the resource and it is false when it does not). However, the resource is constrained and may be overloaded at one or more timesteps. Therefore, all agents need a mechanism to identify when this is likely to occur, and to nominate a qualified set of participants to form collaboration communities. This process is handled at this stage.

- **Neighbouring Agents’ Dependency Model Formation (Contribution 1)**: The nominated agents use a social reasoning technique to acquire an understanding of their neighbourhood and the agents operating in it to build their dependency model.

- **Self-Evaluation (Contribution 2)**: Agents calculate their priority for accessing the resource and evaluate their dependencies, checking whether (1) they can cascade the collaboration to another neighbourhood or (2) change their actions to reduce the demand.

- **Cascade Collaboration (Contribution 3)**: Agents with multiple neighbourhood membership can enter this optional stage, in which they examine all the other alternative neighbourhoods by evaluating their priorities and dependency relations, and decide whether to stay in the current neighbourhood or take action in another neighbourhood.

- **Decision-Making (part of Contribution 2)**: at this stage, depending on the results from previous stages, nominated agents can choose whether or not to collaborate in a decentralised manner and form a collaboration community.

In the following sections these stages are described in detail.

### 3.3 Neighbourhood Membership Update

In open systems, agents join and leave neighbourhoods frequently and unpredictably. Members of a neighbourhood need to be notified if any changes happen in the neighbourhood. A subscriber-publisher design pattern is used for each neighbourhood to capture leave and join events. This pattern is a successful scalable solution for distributed systems and does not limit
agents’ leaving/joining flexibility [22]. In this model, each neighbourhood (representing the resource in the neighbourhood) acts as a publisher and agents subscribe to the resource they want
to use and join the neighbourhood associated with that resource (see Algorithm 1, line 2). All agents subscribed to a particular neighbourhood, will be part of a neighbourhood related to that resource. The agents unsubscribe when they leave the neighbourhood (see Algorithm 1, line 6). When an agent joins or leaves a neighbourhood, the publisher records the changes in the neighbourhood and notifies all the members (see Algorithm 1, line 10).

Formally, $R$ is the set of resources in a multi-agent system. $r_i$ is a single resource around which a neighbourhood $(nc^t_{r_i})$, can be formed, and which has varying $Capacity^t_{r_i}$ and $Demand^t_{r_i}$ at each timestep $t$.

$$
R \equiv \bigcup_{i=1}^{n} \{r_i\}
$$

$$
r_i \equiv \{Capacity^t_{r_i}, Demand^t_{r_i}, nc^t_{r_i}\}
$$

$$
nc^t_{r_i} \equiv \{A_{r_i}, A^t_{\delta}\}
$$

$nc^t_{r_i}$ is the neighbourhood associated with resource $r_i$ at timestep $t$, and includes a set of agents, $A_{r_i}$, which have subscribed to use $r_i$. $A_{r_i}$ is a subset of agents in the system, $A$, which are members of the neighbourhood $nc^t_{r_i}$. $A^t_{\delta}$ is the set of agents in the neighbourhood that take actions, which require the resource $r_i$ at timestep $t$.

$$
A^t_{\delta} \subset A_{r_i} \subset A
$$

Once agents subscribe to a resource, they get access to the neighbourhood’s information, which includes the set of agents operating in the neighbourhood, and the resource’s $Capacity^t_{r_i}$ and $Demand^t_{r_i}$. The neighbourhood is responsible for updating this information. Agents can access this data at any time during their operation in the neighbourhood by requesting them from the neighbourhood, to avoid possible subscriber-publisher miscommunication, which may occur when the publisher is overloaded and fails to update all the operating agents (Fig. 3.2 shows a schematic view of the information about the resource that can be access through neighbourhood).

### 3.3.1 Collaboration Need Identification

The resource in each neighbourhood is constrained and may become overloaded at one or more timesteps. It is to the benefit of all agents in the neighbourhood to collaborate to avoid the
overload before it is likely to occur. All agents in a neighbourhood should be able to identify the need for collaboration, as there is no coordinator or central controller. Agents need to be aware of the resource’s current demand and capacity, every time they take an action (see Algorithm 2). As shown in Fig. 3.1, this process has three steps: the agent first decides its action, increases the resource’s demand, if its action requires resource usage, it checks if the demand meets the available capacity, and nominates a set of agents for collaboration if the demand is higher than the current capacity (see Algorithm 2, Participant-Nomination procedure, line 10). In the Participant-Nomination step, the agent updates $A^t_δ$ by adding itself and any other agents
that have not been added before and want to take an action \((\delta_{t}^{j})\) that needs the shared resource at timestep \(t\) (see Algorithm 2, line 12 and 13).

### 3.4 Neighbouring Agents’ Dependency Model

Given agents’ unpredictable behaviour, and the characteristics of open systems introduced in Section 1.1, learning, planning and simulation-based approaches are not useful for understanding agents’ behaviour and their dependencies. In open systems, agents need a mechanism to acquire knowledge about their neighbourhood (e.g., the current capacity of the shared resource, and the members of their neighbourhood) quickly and in a decentralised way, on the fly. Social reasoning techniques used in current approaches have shown promising results in open environments [161], as they enable agents to communicate directly with each other at any time, and acquire information about each other’s goals, plans and actions. Social reasoning is particularly chosen to be used in this thesis because it enables agents to share information and understand their dependency relations without requiring any help from a central coordinator. The social reasoning used is based on the model introduced and developed by Sichman, et al. [57, 160, 161]. Although the external description introduced in Sichman, et al.’s model facilitates agents’ information sharing process, it requires agents to reveal a lot of their information, which is not desired according to the open systems’ characteristics introduced in Section 1.1. The social reasoning
Algorithm 2: Collaboration Need Identification

1: **procedure** Action Selection

2: $\delta^t_{a_j} = \text{DecideNextAction}()$

3: **if** $\delta^t_{a_j}$ **then**

4:   $\text{Demand}^t_{r_i} +$

5: **if** $\text{Demand}^t_{r_i} \geq \text{Capacity}^t_{r_i}$ **then**

6:   Participant-Nomination()

7: **end if**

8: **end if**

9: **end procedure**

10: **procedure** Participant-Nomination

11:   **for** all $a_j \in nc^t_{r_i}$ **do**

12:     **if** $a_j \notin A^t_\delta \land \delta^t_{a_j}$ **then**

13:       $A^t_\delta \leftarrow a_j$ \> adds $a_j$ to $A^t_\delta$

14: **end if**

15: **end for**

16: **end procedure**
model used in Sichman’s model is also limiting, as it only considers agents’ action dependency, which is useful only when they want to accomplish a complex task. In this thesis, a modified version of the external description, which requires less information to be shared, is presented. Agents build their Neighbouring Agents’ Dependency Model using this modified external description and a social reasoning model that enables agents to understand their goal dependencies while sharing constrained resources.

After Participant-Nomination, once an agent is nominated to collaborate with other agents operating in a neighbourhood, it should be able to identify its neighbours and acquire more information to understand its dependency relations. Agents share some of their information with their neighbours through their external description, and store the information they have acquired from their neighbours. Using the external description, agents can reason about their dependency relations and form their dependency model. In the following two subsections, the external description and the social reasoning technique are introduced.

### 3.4.1 External Description

An agent’s external description is a data structure that is publicly accessible to all the agents in a neighbourhood. Using this structure, agents share some of their information and store the data that is acquired from other agents in their neighbourhood. This data structure creates a level of abstraction for heterogeneous agents and is kept separated from an agent’s internal structure and architecture (see Fig. 3.2). Formally, an external description is defined as:

\[
Ext_{a_i} \overset{\text{def}}{=} \bigcup_{j=1}^{n} Ext_{a_i}(a_j)
\]

This definition is adapted from [57], where \( Ext_{a_i} \) is \( a_i \)’s external description \(^1\) and the \( Ext_{a_i}(a_j) \) is the entry that stores \( a_j \)’s information. Using the proposed external description, agents share less information and they are not required to reveal their plans, or decision-making mechanism. \( Ext_{a_i}(a_j) \) is defined as:

\[
Ext_{a_i}(a_j) \overset{\text{def}}{=} \{G_{a_j}, d_{a_j}, P_{a_j}, N_{a_j}, SP_{a_j}, TP_{a_j}\}
\]

\(^1\)This thesis only uses the definition of external dependencies, and the rest of the dependency model introduced in this chapter is new.
Chapter 3: Design

• **Goals** $G_{a_j}$ is the set of goals $a_j$ wants to achieve. Agents can achieve multiple goals simultaneously.

\[ G_{a_j} = \{g_1, g_2, ..., g_n\} \]

• **Action** $\delta_{a_j}^t$ is the action $a_j$ takes at timestep $t$, which is the next timestep. Particularly, CCFOM is concerned only with the actions that involve using the shared resource. Agents’ actions are modelled using $\Delta_{a_j}(g_k)$, which includes all the actions that agent $a_j$ needs to take to achieve $g_k$.

\[ \Delta_{a_j}(g_k) = \{A\Delta_{a_j}(g_k), \delta_{a_j}^t, P\Delta_{a_j}(g_k)\} \]

$A\Delta_{a_j}(g_k)$ is the set of actions that have been taken for goal $g_k$ so far. $P\Delta_{a_j}(g_k)$ is the set of pending actions that is needed to be taken in the future, and $\delta_{a_j}^t$ is the action that will be taken in the coming timestep $t$. In this thesis, actions are modelled as a boolean variable that can be either true or false. A true value means an agent is taking an action that requires a shared resource and a false value means that an agent is not taking an action that requires a shared resource.

• **Policies** $P_{a_j}$ is the set of policies $a_j$ defines to achieve its goals. Agents’ policies define their level of self-interest and cooperation for each goal by initializing two variables, $\textit{SetPoint}_{g_k}^{a_j}$ and $\textit{Target}_{g_k}^{a_j}$.

\[ P_{a_j} \overset{\text{def}}{=} \bigcup_{k=0}^{n} p_{a_j}(g_k) \]

\[ p_{a_j}(g_k) \overset{\text{def}}{=} \{\textit{SetPoint}_{g_k}^{a_j}, \textit{Target}_{g_k}^{a_j}\} \]

$\textit{SetPoint}_{g_k}^{a_j}$ is a norm defined by the agent, which specifies the minimum number of actions, $\alpha_{g_k}$, that should be achieved for $g_k$ before timestep $t'$. This number is calculated by the agent depending on its preferences and internal states. $\textit{SetPoint}_{g_k}^{a_j}$ specifies the timestep from which an agent can cooperate with other agents in a neighbourhood, while taking actions to achieve $g_k$. Therefore, if $a_j$ has achieved $\alpha_{g_k}$ number of its actions before or at timestep $t'$, it is more likely to be collaborative. For example, when a smart
device gets a minimum battery charge, it is more likely to show a collaborative behaviour compared to when it does not have enough battery (this is explained in Section 3.5). Defining $SetPoint_{g_k}^{a_j}$, also helps agents to understand their dependency relations (explained in Section 3.4.2).

\[ SetPoint_{g_k}^{a_j} = \{\alpha_{g_k}, t_k'\} \]

$Target_{g_k}^{a_j}$ defines the maximum number of actions (that needs to access the resource) an agent wants to take while it has access to the resource $r_i$, before timestep $t''$, which is the last timestep that agent $a_j$ can use resource $r_i$ for $g_k$.

\[ Target_{g_k}^{a_j} \overset{\text{def}}{=} \{(\Lambda_{g_k}, t_k'') : \Lambda_{g_k} \leq |\Delta_{a_j}(g_k)|\} \]

To exemplify the concept of an agent’s policy and its $SetPoint_{g_k}^{a_j}$ and $Target_{g_k}^{a_j}$, consider a smart electrical device that wants to use electricity to charge its battery. It sets its $SetPoint_{g_k}^{a_j}$ value to be the minimum amount of energy it needs in its battery to run normally and sets its $Target_{g_k}^{a_j}$ value to the maximum amount of energy it needs to fully charge its battery. Defining these two values helps the electrical device to understand its needs better and can behave more intelligently.

- **Neighbourhoods’ membership** $N_{e_{a_j}}$ is the number of neighbourhoods of which $a_j$ is a member.
- **Priority** Each agent also calculates a priority $SP_{a_i}$, to access the resource. $TP_{a_i}$ is also a priority value transferred to $a_i$ from other agents when they behave cooperatively (see details in Section 3.5).

### 3.4.2 Dependence Relation Identification

Understanding the nature of the resource dependency, enables agents to identify a qualified group of agents with which they need to interact. It also helps them to detect their own dependency relations, which plays a vital role during the community formation process. In this section, the
definition of a resource-autonomous agent (r-autonomous, introduced in Section 2.2.2.1) is re-
refined\(^2\). The concept of a resource-dependent agent (r-dependent, introduced in Section 2.2.2.1) is also redefined to address agents that need a resource, which is not controlled by agents and is shared between them. Agent \(a_j\) is resource autonomous, \(res - aut\), for goal \(g_k\), when it does not take any action that requires the shared resource. Such agents will not be in any resource neighbourhoods.

\[
res - aut(a_j, g_k) \overset{def}{=} \{|\alpha_{g_k}| = 0\}
\]

Agents in a neighbourhood are defined resource dependent, \(res - dep\), when they have at least one goal in their goal set for which they are not resource autonomous.

\[
res - dep(A_r) \overset{def}{=} \bigcup_{j=0}^{n} \{a_j \mid \exists g_k \in G_{a_j} \land \neg res - aut(a_j, g_k)\}
\]

In CCFOM, action autonomous (a-autonomous) agents share a resource in their neighbourhood. None of the neighbours have control over the resource and they all need to use the resource to achieve their goals, which makes them resource dependent. Each agent uses the information provided by others’ external description and compares them to its own to figure out what kind of dependencies it has with its neighbours.

Current approaches model agents’ behaviour to be either collaborative or competitive, all their life time, when sharing a constrained resource, without considering their goal dependencies [181]. However, more types of dependencies may exist between agents’ goals, and these dependencies may even change during their life time according to their goal achievement progress or the constrained resource availability. For example, in a transportation system, passengers may always behave competitively, but when there is an emergency situation in the city, their behaviour may tend to be more collaborative. Therefore, agents’ should be able to understand their goal dependencies throughout their operation in the system. Fig. 3.3 depicts agents’ different dependence relations and the logic agents use to decide which kind of dependencies they have. This figure shows seven types of dependencies between agents’ goals when sharing a constrained

\(^2\)The definition in the literature defines the resource dependency as if an agent is dependent on another that controls the resource. However, they do not model resources that are not controlled by any of the users in their definitions. In this thesis the new definition specifies this notion.
resource. Current multi-agent resource allocation approaches consider only either cooperative or competitive behaviour. These goal dependency relations define how agents may behave when sharing constrained resources. Agents may have different goals that may affect their behaviour (i.e., be cooperative, self-interested, or have the option to adjust their level of self-interest).

- **Cascade Option (CsO)** Agents, which are members of more than one neighbourhood, can cascade the resource request to another neighbourhood, which has the interchangeable same required resource. Formally, \( a_i \) has Cascade Option relation if it takes an action that requires the shared resource, in timestep \( t \) and it is a member of more than one neighbourhood.

  \[
  CsO(a_i) \overset{def}{=} \{ \exists a_i \in A_t^i : |Nc_{a_i}| > 1 \}
  \]

- **Overlap Dependency (OvD)** Two agents in the same neighbourhood have an Overlap Dependency on the shared resource in that neighbourhood, when they both take actions that use the shared resource and have the same SetPoints for different goals. Having

Fig. 3.3: Dependence Relations Identification
the same SetPoint implies that the two agents want to use the same amount of resource before timestep \( t' \).

\[ \text{OvD}(a_i, a_j) \overset{\text{def}}{=} \{ \exists a_i, a_j \in A^t_{\delta}, p_{a_i}(g_i), p_{a_j}(g_k) : \text{SetPoint}_{gi}^{a_i} = \text{SetPoint}_{gk}^{a_j} \land g_i \neq g_k \} \]

- **Conflict Dependency (CnfD)** This relation exists between two agents that want to use the shared resource for different individual goals, but their policies do not allow collaboration. Formally, \( CnfD(a_i, a_j) \) means that \( a_j \)'s policy is set in a way that it is not able to cooperate with \( a_i \).

\[ CnfD(a_i, a_j) \overset{\text{def}}{=} \{ \exists a_i, a_j \in A^t_{\delta}, p_{a_i}(g_i), p_{a_j}(g_k) : \text{SetPoint}_{gi}^{a_i} \neq \text{SetPoint}_{gk}^{a_j} \land (\alpha_{gi} = \Lambda_{gk}) \land (g_i \neq g_k) \} \]

- **Sequential Dependency (SqD)** This situation exists when agents’ goals are dependent and there needs to be a sequence in goal achievement for such agents. In other words, when agent \( a_i \) and \( a_j \) have \( SqD \), \( a_i \)'s goal cannot be achieved unless \( a_j \) achieves its goal.

\[ SqD(a_i, a_j) \overset{\text{def}}{=} \{ \exists a_i, a_j \in A^t_{\delta}, p_{a_i}(g_i), p_{a_j}(g_k) : \text{SetPoint}_{gi}^{a_i} \neq \text{SetPoint}_{gk}^{a_j} \land \alpha_{gi} = \alpha_{gk} + \beta \land (t'_i > t'_k) \land (\beta > 1) \} \]

- **Concurrent resource use Dependency (CncD)** This situation exists when two agents take the same actions for different goals and have different policies, but do not have Conflict Dependency or Sequential Dependency. Having this relation, the two agents could, but do not have to, use the resource concurrently.

\[ CncD(a_i, a_j) \overset{\text{def}}{=} \{ \exists a_i, a_j \in A^t_{\delta}, p_{a_i}(g_i), p_{a_j}(g_k) : \text{SetPoint}_{gi}^{a_i} \neq \text{SetPoint}_{gk}^{a_j} \land (g_k \neq g_i) \land \neg SqD(a_i, a_j) \land \neg CnfD(a_i, a_j) \} \]

- **Competition Dependency (CmpD)** Two agents have a Competition Dependency when they have the same goals, take the same actions but have different policies. \( CmpD \) is a competing relation where one agent chooses to maximize the shared resource usage in the minimum amount of time.


\[ CmpD(a_i, a_j) \triangleq \{ \exists a_i, a_j \in A_t^\delta, p_{a_i}(g_i), p_{a_j}(g_k) : (t'_i < t'_k) \land (t''_i < t''_k) \land (g_i = g_k) \} \]

- **Friendship Dependency (FD)** This dependency relation exists when agents have the same goals, take the same actions, have the same policies, and behave cooperatively to achieve each others’ goals. \( FD \) defines a relation in which agents behave cooperatively with agents with the same goal and policy. Therefore, when the shared resource is constrained, they cooperate among themselves.

\[
FD(a_i, a_j) \triangleq \{ \exists a_i, a_j \in A_t^\delta, p_{a_i}(g_i), p_{a_j}(g_k) : (SetPoint_{a_i}^{g_i} = SetPoint_{a_j}^{g_k}) \\
\land (Target_{a_i}^{g_i} = Target_{a_j}^{g_k}) \land (g_i = g_k) \}\]

### 3.5 Self-Evaluation and Adaptation Algorithm

Nominated agents from the Participant-Nomination procedure (see Algorithm 2), re-evaluate their selected actions before starting to collaborate. In this process, agents try to find a way to decrease their demand before starting a collaboration process. As shown in Fig. 3.1, in this stage, agents take two steps, Priority Calculation and External Description Evaluation.

#### 3.5.1 Priority Calculation

In this step, agents calculate their priority to access the shared resource, as compared to other agents’ priorities. In this model, each agent has two priority values, Self-Priority, \( SP_{a_i} \), and Transferred-Priority, \( TP_{a_i} \).

**Self-Priority**: \( SP_{a_i} \) is the summation of priorities of all \( a_i \)’s goals that are dependent on the action \( \delta_{a_i}^t \) that will be taken in timestep \( t \). Each single priority is calculated using the number of pending actions \( |P_{\Delta a_i}(g_k)| \) and the duration that the agent will spend in the neighbourhood, \( d_{a_i} \) (see Equation 3.1). A higher priority means that an agent has more pending actions to take in the remaining time. In a neighbourhood, an agent with a higher priority may get a better chance to access the shared resource compared to others with lower priorities, if the agents consider their
dependencies and behave collaboratively.

\[ S P_{a_i} = \sum_{k=1}^{n} \frac{|P \Delta_{a_i}(g_k)|}{d_{a_i}} \]  \hspace{1cm} (3.1)

**Transferred-Priority:** \( a_i \)’s Transferred-Priority, \( TP_{a_i} \), stores other agents’ Self-Priorities that are transferred to \( a_i \). Agents may transfer their priorities for two reasons: First, they have Sequential Dependency (\( SqD \)), so if the agent (on which other agents depend) fails to achieve its goals, it means they (the dependent agents) cannot achieve their goals either. Second, they are cooperative agents, so they transfer their priorities to the agent with a higher priority in their group to increase its chance to access the resource. At this stage, \( TP_{a_i} \) is set to zero (see Algorithm 3, Priority Calculation Procedure). The Transferred-Priority should have a range between 0 and a maximum value \( MaxTP \) that can be defined based on the application. \( MaxTP \) is the maximum priority that can be assigned to Transferred-Priority. This value is set to avoid the accumulation of transferred priorities in to a single agent. It helps to distribute the Transferred-Priority between agents with high priorities.

**Algorithm 3** Self-Evaluation

1: **procedure** PRIORITY CALCULATION

2: \[ SP_{a_i} = \sum_{k=1}^{n} \frac{|P \Delta_{a_i}(g_k)|}{d_{a_i}} \]

3: \( TP_{a_i} = 0 \)

4: **end procedure**

5: **procedure** EXTERNAL DESCRIPTION EVALUATION

6: for all \( a_j \in A^i_s \) do

7: if \( Demand^i_{r_i} > Capacity^i_{r_i} \) then

8: \( \text{EvaluateSqD}(a_i) \)

9: \( \text{EvaluateFD}(a_i) \)

10: \( \text{EvaluateCsO}(a_i) \)

11: end if

12: end for

13: **end procedure**
3.5.2 External Description Evaluation

After calculating the priorities, each nominated agent \( a_i \) searches its \( Ext_{a_i} \) for \( SqD, FD, \) and \( CsO \) dependency relations, to find a potential demand decrease before starting the collaboration (see Algorithm 3, procedure Self-Evaluation). As explained, \( SqD(a_i, a_j) \) means that \( a_i \)'s goal is sequentially dependent on \( a_j \)'s goal. Therefore, \( a_i \) prefers \( a_j \) achieves its goal sooner. \( a_i \) transfers its \( SP_{a_i} \) and \( TP_{a_i} \), to \( a_j \) to increase its chance to access the resource. \( a_i \) then changes its action and decreases the demand. When an agent has multiple \( SqD \) relations with multiple agents it picks one of them randomly if all their conditions are the same. When there is a chain of \( SqD \) between agents, \( EvaluateSqD(a_i) \) is called recursively (see Algorithm 4, procedure \( EvaluateSqD(a_i) \), line 2). In presence of a \( FD(a_i, a_j) \) relation, \( a_i \) transfers its priorities to \( a_j \) when: (1) there is no other agent \( a_k \) in \( A^1_\delta \) for which \( FD(a_i, a_k) \) is true, and (2) \( a_i \) has lower priority than \( a_j \) and (3) \( TP_{a_j} \) is less than \( MaxTP \) (see Algorithm 4, procedure \( EvaluateFD(a_i) \), line 14). When \( CsO(a_i, a_j) \) exists, \( a_i \) asks \( a_j \) to cascade the resource request to an alternative neighbourhood. \( a_j \) evaluates those neighbourhoods and decides whether or not to change its action in the current neighbourhood and take action in the new neighbourhood (see Section 3.5.3). If the agent decides to stay in the current neighbourhood, it starts Algorithm 2, and initiates a collaboration in the new neighbourhood. If the collaboration is successful, it changes its action in the initial neighbourhood and decreases the demand.

The other dependency relations such as \( OvD, CmpD, CnfD, \) and \( CncD \), will be required during decision-making process.
Algorithm 4 Self-Evaluation

1: procedure EVALUATESQD(a_i)
2:   if (∃a_j ∈ A^i_δ : SqD(a_i, a_j)) ∧ (∃a_k ∈ A^i_δ : (SqD(a_j, a_k) ∨ SqD(a_k, a_i))) then
3:     TPa_j+ = SPA_i + TPa_i
4:     δ_{a_i} = False
5:   end if
6: end procedure

7: procedure EVALUATEFD(a_i)
8:   if ∃a_j ∈ A^i_δ : (FD(a_i, a_j) ∧ (SPA_i < SPA_j)) ∧ (∃a_k ∈ A^i_δ : (FD(a_j, a_k) ∧ (SPA_k > SPA_j)) ∧ TP_{a_j} < MaxTP) then
9:     TPa_j+ = SPA_i + TPa_i
10: return
11: end procedure

12: procedure EVALUATECSO(a_i)
13:   if ∃a_j ∈ A^i_δ : CsO(a_i, a_j) then
14:     if Cascade Collaboration()==True then
15:       δ_{a_j} = False
16:     end if
17:   end if
18: end procedure
### Table 3.1: Categories of Agents in Multi-neighbourhood Setting

<table>
<thead>
<tr>
<th>Categories</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
<th>Group 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristics</td>
<td>( \delta_{a_j} = 1 )</td>
<td>( \delta_{a_j} = 1 )</td>
<td>( \delta_{a_j} = 0 )</td>
<td>( \delta_{a_j} = 0 )</td>
</tr>
<tr>
<td>( N_{c_{a_j}} = 1 )</td>
<td>( N_{c_{a_j}} &gt; 1 )</td>
<td>( N_{c_{a_j}} = 1 )</td>
<td>( N_{c_{a_j}} &gt; 1 )</td>
<td></td>
</tr>
</tbody>
</table>

#### 3.5.3 Cascade Collaboration

In this stage, the agents with multiple neighbourhood memberships evaluate their priorities and dependencies in other neighbourhoods and decide whether to change their actions and cascade the resource request to another neighbourhood (see Algorithm 5).

As mentioned earlier, each agent can be a member of more than one neighbourhood and this results in overlapping neighbourhoods in a system. During their action selection, agents choose the resource and consequently the neighbourhood in which they want to take action. The agents operating in each neighbourhood in a multi-neighbourhood setting can be grouped into 4 different groups (see Table 3.1): **Group 1** is the group of agents that have decided to take actions in the next timestep \( t \), and are members of only one neighbourhood. **Group 2** is the group of agents that have decided to take actions in the next timestep \( t \), and are members of more than one neighbourhood. **Group 3** is the group of agents that do not want to take actions in the next timestep \( t \) and are members of only one neighbourhood. **Group 4** is the group of agents that are not taking actions in the next timestep \( t \) in this neighbourhood, and are members of more than one neighbourhood. The agents in the last group may or may not be taking actions in other neighbourhoods, and if they are taking actions they may change them and decide to use the resource in this neighbourhood and cascade the collaboration request. These groups are helpful concepts when agents want to evaluate their alternative neighbourhoods and compare them to their current neighbourhood (Groups are used in the following proposed states).

When an agent wants to decide whether or not to cascade the collaboration request (see Multi-neigh Pri-Dep-Eval procedure in Algorithm 5, line 9), it considers the following factors in each individual neighbourhood of which it is a member: (1) The neighbourhood’s resource demand and capacity for the next timestep \( t \), (2) the population of agents in each of the above
Algorithm 5 Cascade Collaboration

1: procedure CASCADE COLLABORATION
2:    for all nc<sub>r_k</sub> : a<sub>j</sub> ∈ nc<sub>r_k</sub> do
3:        if δ<sub>tr</sub><sup>j</sup> can be taken by r<sub>k</sub> then
4:        Multi-neigh Pri-Dep-Eval (nc<sub>r_k</sub>)
5:    end if
6:  end for
7:  Neighbourhood Selection(nc<sub>r_k</sub>)
8: end procedure

9: procedure MULTI-NEIGH PRI-DEP-EVAL(nc<sub>r_k</sub>)
10:      if State 1 then
11:          Selected = nc<sub>r_k</sub>
12:      else if State 2 || State 3 then
13:          if (Capacity<sup>r_k</sup> − (HE(SP<sub>a</sub>) + |Group(4)nc<sub>r_k</sub>|) ≥ Capacity<sup>err</sup> − (HE(SP<sub>a</sub>))) then
14:              Selected = nc<sub>r_k</sub>
15:          end if
16:          if |FD|<sub>nc<sub>r_k</sub></sub> > |FD|<sub>nc<sub>err</sub></sub> V |CmpD|<sub>nc<sub>r_k</sub></sub> < |CmpD|<sub>nc<sub>err</sub></sub> then
17:              Selected = nc<sub>r_k</sub>
18:          end if
19:      end if
20: end procedure

21: procedure NEIGHBOURHOOD SELECTION
22:      if Selected! = nc<sub>err</sub> then
23:          ActionSelection()
24:          return True
25:      end if
26: end procedure
mentioned groups, (3) its Self-Priority, and (4) its dependency relations, particularly the number of agents with \( FD \) and \( CmpD \). Using this information, agents are able to estimate their chances to access the resource in each neighbourhood. Considering these factors, an agent can be in one of the following states, when deciding to cascade its resource request:

**State 1:** when an agent will have access to the resource in \( nc_{r_{err}} \) (agents’ current neighbourhood), even if all its neighbours that are currently taking actions in other neighbourhoods, change their mind and take actions in this neighbourhood. Formally, \( a_i \) is in State 1, when the resource capacity of \( nc_{r_{err}} \), \( Capacity_{r_{err}}^t \), can serve \( a_i \), all the agents in Group 4, and agents with higher or equal priority to \( a_i \), \( HE(SP_{a_i}) \).

\[
Capacity_{r_{err}}^t \geq HE(SP_{a_i}) + |Group(4)_{nc_{r_{err}}}|
\]

When \( a_i \) is in this state, it changes its action in the current neighbourhood only when a \( nc_{r_{k}} \) exists, where \( a_i \) can have a better or the same priority to access its resource.

\[
Capacity_{r_{k}}^t - (HE(SP_{a_i}) + |Group(4)_{nc_{r_{k}}}|) \geq Capacity_{r_{err}}^t - (HE(SP_{a_i}) + |Group(4)_{nc_{r_{err}}}|)
\]

**State 2:** when the agent will have access to the resource only if the current conditions of the neighbourhood do not change. Formally, \( a_i \) is in this state, when \( Capacity_{r_{err}}^t \) can serve a subset of agents including \( a_i \).

\[
Capacity_{r_{err}}^t - (HE(SP_{a_i})) \geq 0
\]

\( a_i \) assesses whether it could have a higher priority to access a resource in another neighbourhood or it has better dependency relations in \( nc_{r_{k}} \).

\[
Capacity_{r_{k}}^t - (HE(SP_{a_i}) + |Group(4)_{nc_{r_{k}}}|) \geq Capacity_{r_{err}}^t - (HE(SP_{a_i}))
\]

If the priority to access the resource in any other alternative neighbourhoods is similar to the current neighbourhood, the agent checks its dependency relations and chooses the neighbourhood in which it has less competition and/or conflict dependencies and/or more friendship dependency (see Algorithm 5).
Chapter 3: Design

\[ |FD|_{nec_k} > |FD|_{neccerr} \lor |CmpD|_{nec_k} < |CmpD|_{neccerr} \]

**State 3:** when the agent cannot access the resource according to the current conditions of the neighbourhood.

\[ Capacity^k_{neccerr} - \langle HE(SP_{a_i}) \rangle < 0 \]

In this state, similar to State 2, the agent chooses another neighbourhood if it has a higher priority or dependency relations.

### 3.6 Decentralised Decision-Making

Agents’ priorities and their dependence relations are the main factors that help them to decide whether to act cooperatively during this process. This is a sequential process and in each round, the agent with the lowest priority among the nominated agents, runs the decision-making process and finalizes its action. This stage ends when: (1) the demand becomes less than or equal to the available capacity, or (2) all the agents in \( A'_{t} \) have run the process and finalized their actions. In this algorithm, the agents with lower priorities are expected to change their actions to decrease the demand. However, they need to check their dependency relations to decide. For example, if there is a competing agent, they will not change their actions.

Agent \( a_j \), which has the lowest priority, starts the process (Algorithm 6, line 3). \( a_j \) finalizes its action when it has Competition Dependency with another agent and does not have any Friendship Dependency with any other agents with a higher priority (see Algorithm 6, line 9-11). However, it transfers its priority, changes its action, and behaves cooperatively, when there is a Friendship Dependency between \( a_j \) and \( a_l \) (see Algorithm 6, line 5-7). Finally, when \( a_j \) does not have either of these dependency relations, it changes its action and decreases the demand (see Algorithm 6, line 14-15).

Given the varying dependency relations between agents, there might be more agents than the available capacity of the shared resource decide to use the resource, which means the collaboration has failed and results in shared resource overload.

Using the six stages of CCFOM, agents are enabled to freely join and leave multiple neighbourhoods, when pursuing their individual goals. They use the social reasoning techniques to
Algorithm 6 Collaboration Decision-Making

1: procedure DECIDE
2:   for all $a_j \in A^t_\delta$ do
3:     if $\exists a_k \in A^t_\delta : \delta_{a_k} \land SPA_k + TPa_k < SPA_j + TPa_j \land (Demand^t_{ri} > Capacity^t_{ri})$ then
4:       if $\exists a_k \in A^t_\delta : CompD(a_j, a_k)$ then
5:         if $\exists a_l \in A^t_\delta : FD(a_j, a_l)$ then
6:           $\delta^t_{a_j} = False$
7:           $Demand^t_{ri} -=$
8:           Update($A^t_\delta, a_j$)
9:         else
10:           $TakeAction()$
11:           $Demand^t_{ri} -=$
12:           $Capacity^t_{ri} -=$
13:           Update($A^t_\delta, a_j$)
14:       end if
15:   else
16:     $\delta^t_{a_j} = False$
17:     $Demand^t_{ri} -=$
18:     Update($A^t_\delta, a_j$)
19:   end if
20: end if
21: end for
22: end procedure

build their own dependency model, which helps them to collaboratively achieve multiple goals simultaneously, while sharing a constrained resource.
Chapter 3: Design

3.7 Summary

This chapter first presented a set of design requirements and features for a collaboration community formation model that enables agents to form collaboration communities in open system, when sharing constrained resources to achieve multiple goals simultaneously. To meet these requirements, it presented CCFOM, which is a decentralised collaboration community formation model that enables agents to operate in multiple overlapping neighbourhoods and form collaboration communities using the social reasoning technique presented and decide to form collaboration communities and collaborate in a decentralised manner.

CCFOM models the open system as a multi-neighbourhood system, where neighbourhoods are overlapping, in that agents may operate in more than one neighbourhoods. The feature enables agents to operate in more than one neighbourhood and take full advantage of interacting with all their neighbours, without limiting agents to have any specific dependency relations. In addition, CCFOM presents its own knowledge acquiring method, and a new social reasoning model, which enables agents to advance their selection criteria from considering either exclusively shared or individual goal to consider both goals when sharing constrained resources. Moreover, CCFOM also defines a new collaboration community formation model, which allows agents to adapt their level of self-interest and cooperation, depending on their own states and dependency relations when achieving multiple goals simultaneously. CCFOM also defines a decentralised decision-making algorithm, which allows agents to make decisions in a decentralised manner and does not require agents to make commitments and stay in the neighbourhood during the course of a collaboration process.

In summary, agents can use CCFOM to form multiple overlapping neighbourhoods, identify other agents operating in their neighbourhood, understand their dependencies, and adapt their level of self-interest and cooperation to form collaboration communities in a decentralised manner.
Chapter 4

Implementation

The previous chapter presented the design of the Collaboration Community Formation Model for agents with multiple goals operating in open systems. The design presented allows agents to operate in multiple overlapping neighbourhoods to pursue multiple often conflicting goals simultaneously.

This chapter presents the implementation of CCFOM’s main components, including Neighbourhood Membership, Social Reasoning, Self-Evaluation and Adaptation, and Decision-Making. For implementation and modelling purposes, one of the available agent-oriented modelling languages called MAS-ML is used [51]. This thesis has extended this model in a few directions, particularly to model the social reasoning component. CCFOM is written in C++ for ease of integration with an existing Smart Grid simulator. This implementation is coupled with a new general simulator, which was developed for this thesis to evaluate CCFOM in application-independent scenarios. This simulator is also used in the Smart Grid and Ride Sharing case studies. The remainder of this chapter is organised as follows. Section 4.1 briefly reviews MAS-ML, a modelling language for multi-agent systems. It also presents an extended version of MAS-ML, which is designed to capture the social dependencies between agents operating in multiple overlapping neighbourhoods. This extended model is the basis for CCFOM implementation. Section 4.2 presents the details on CCFOM implementation. Finally, Section 4.3, discusses the implementation details of the three different simulators that are used in this thesis.
4.1 UML Meta-model for Multi-agent Systems

MAS-ML provides an extension to UML to represent agents, organizations, environments, and agents’ roles [51]. However, MAS-ML does not model the social behaviour of agents. This chapter describes an extension to MAS-ML, to address social behaviour concepts. The color coding in Fig. 4.2 shows the existing classes of UML, the classes added in MAS-ML and the new classes added by this thesis. In this section, the existing meta classes that were needed to be modified in the MAS-ML are described.

- The AgentClass meta class that extends the Classifier meta class to represent agents and their structural and behavioural features.

- The OrganizationClass meta class extends the AgentClass to have goals, plans, beliefs, and actions.

- The Property meta class is also modified by adding three stereotypes as Goal, Belief and Axiom. These stereotypes are instantiated as an attribute for each agent.

These meta classes are modified/extended by adding the new NeighbourhoodClass meta class and the Policy stereotype.

- The NeighbourhoodClass extends the OrganizationClass by adding the features defined in Chapter 3, Section 3.3.

- Agents’ policies formalised in Chapter 3, Section 3.4.1 are defined with the Policy stereotype on the property class, instantiated as an attribute.

As shown in Fig. 4.2, MAS-ML has added some meta classes to the UML model, defining new relationships to link elements such as objects, agents, organizations, environments, agents’ roles, and objects’ roles. These relationships are Ownership, Inhabit, Control, and Play. Fig. 4.3 shows how these elements are linked using the new relations defined in MAS-ML.

This thesis adds the Social Engage meta class and the classes that extend it to model the social dependencies defined and formalised in Chapter 3, Section 3.4.2 (see Fig. 4.2). As depicted
in Fig. 4.4, which is an extended version of Fig. 4.3, Social Engage links the Agent class to the Neighbourhood class. It also shows the dependency relation between the agents on the Agent class.
Chapter 4: Implementation

4.2 CCFOM Implementation

CCFOM enables agents to operate in an open system consisting of multiple neighbourhoods. Fig 4.5 shows the main components of CCFOM, with a Neighbourhood layer including Neighbourhood Manager, and an Agent layer including External Description Manager, Communication, Social Reasoning, Self-Evaluation and Adaptation, and Decision-Making modules. It is important to note that all the processes needed to form a collaboration community happen at the Agent layer and each individual agent can complete these processes without any help from a central coordinator.

CCFOM is implemented in a way that each agent contains all the processes needed to perform in an open system to form collaboration communities. The modules mentioned at both Neighbourhood and Agent layers have corresponding classes in the class diagram shown in Fig.
4.6. The interaction between the instantiated objects from these classes are also shown in Fig. 4.7. The sequence diagram pictured in Fig. 4.7, captures a high level view of the processes that happen when an agent operates in a multi-neighbourhood open system. When an agent wants to join a neighbourhood, it sends a subscribe message to the neighbourhood object, and this object notifies the whole neighbourhood, $A_{ri}$. When the agents in a neighbourhood take actions they check whether or not a collaboration process is needed. When this is identified agents are added to a nominated collaboration community, $A_{tδ}$. Each agent then initiates CCFOM process individually and parallel to other agents in the nominated community. This process includes Social Reasoning, Self-Evaluation and Adaptation, Cascade Collaboration and Decision-Making processes. The details on the execution sequence of these processes are depicted in Fig. 4.7.

![Fig. 4.5: CCFOM’s Main Components](image)

**4.2.1 Neighbourhood Layer**

As shown in Fig. 4.5, the Neighbourhood layer has a neighbourhood manager component that takes care of two main elements $nc_{ri}$ and $A_{ri}$. This component is implemented to manage the joining and leaving of agents and keep record of the current capacity of the neighbourhood’s resource. This component stores the data formalised in Chapter 3, Section 3.3, notifies the agents operating in the neighbourhood of any changes in these data and facilitates agents’ access to these data.

Fig. 4.8 shows a detailed sequence diagram of the Neighbourhood Membership Update (sub-
scribe/unsubscribe), and Collaboration Need Identification processes. These processes engage the agent object (i.e., a joining or leaving agent), the neighbourhood object, and the object that keeps information about the agents operating in the neighbourhood formalised as $A_{r_1}$. When a new agent wants to join a neighbourhood, it sends a subscribe message to the neighbourhood object. The neighbourhood object then updates $nc_{r_1}$, by adding the new agent to the list of agents kept by $A_{r_1}$ object, and asks it to notify all members. The notify message includes the updated information stored in $nc_{r_1}$, and is sent to all the members of the neighbourhood ($A_{r_1}$). In the Collaboration Need Identification process, each agent decides its action and asks the neighbourhood object to increase the demand and asks for the current demand and capacity of the resource in return. It then evaluates the state of the resource using these data and initiates the Participant-Nomination process, by updating $A_{r_1}^d$.

### 4.2.2 Agent Layer

The Agent layer includes five modules including Communication, External Description Manager, Social Reasoning, Self-Evaluation and Adaptation, and Decision-Making (see Fig. 4.5). The design and implementation of this layer equip each agent with all the required skills to communicate with other agents to acquire and share information (Communication), store a subset of their own information that are willing to share and store the acquired information from others (External Description Manager), interpret the information and understand their goal dependen-
cies on other agents in the neighbourhood (Social Reasoning), evaluate their state, and adapt and adjust their level of self-interest and cooperation (Self-Evaluation and Adaptation), and to decide whether or not to participate in a collaboration process (Decision-Making).

The Communication module is responsible for agent-to-agent, as well as agent-to-neighbourhood object communications. This module handles the sending and receiving of mes-
 Chapter 4: Implementation

Fig. 4.8: Neighbourhood Layer: Neighbourhood Membership Update, Agent Layer: Collaboration Need Identification

Messages, and stores them in a buffer until the agent is ready to process them. The messages agents send and receive in a neighbourhood include request to access each others’ external descriptions. The messages between the Agent and Neighbourhood layers are about accessing the $nc_{r_i}$ and $A_{r_i}$ data.

The **External Description Manager** is responsible for sharing and updating the external description data structure. It receives messages through the Communication module and either shares information or updates the data structure with a new or updated entry about another agent.

The **Social Reasoning** module is responsible for understanding agents’ goal dependencies.
It requests data from the External Description and neighbourhood objects, and identifies the agent’s dependencies on each neighbour using the method introduced in Chapter 3, Section 3.4. It then stores these dependencies in a data structure (i.e., Neighbouring Agents’ Dependency Model) to be used during the Self-Evaluation and Adaptation, and Decision-Making processes. Fig. 4.9 shows a detailed sequence diagram of this process including all of the method calls and data passing.

Fig. 4.9: Social Reasoning Process Implementation
The **Self-Evaluation and Adaptation** module is responsible for adjusting the agent’s level of self-interest and cooperation. It uses the dependencies identified in the Social Reasoning module, (i.e., stored in the Neighbouring Agents’ Dependency Model). In this process, as shown in Fig. 4.10, it first calculates agent’s priority using the data acquired from External Description object and then evaluates agents’ dependency relations. In this process, the priorities are transferred to another agent according to the procedure explained in Chapter 3, Section 3.5.2. An optional process that an agent, which has multiple neighbourhoods membership, may choose to run during Self-Evaluation and Adaptation process is Cascade Collaboration. This happens when an agent searches for an alternative neighbourhood in which to take action. To do so, the Cascade Collaboration object communicates with the neighbourhood object of all the other neighbourhoods of which it is a member, to get access to the $ne^1_{rk}$ and $Ark$ data. Based on these data and using the procedure defined in Chapter 3, Section 3.5.3, it decides whether or not to cascade the resource request to another neighbourhood. Doing so, it asks the neighbourhood object of its current neighbourhood to decrease the demand on its resource, and then takes action in the alternative neighbourhood. The detail of this process is shown in Fig. 4.11.

The **Decision-Making** module is responsible for finalizing the agent’s action. It can either decide to be cooperative and change its action or be self-interested and take its initial action, regardless of its neighbours’ priorities or the resource’s capacity. As shown in Fig. 4.12, the Decision-Making object asks for the neighbourhood’s data and requests the information from the Neighbouring Agents’ Dependency Model about the CmpD and FD relations, it updates its action and asks the neighbourhood to update its information accordingly. The decision criteria is discussed in detail in the Decision-Making procedure presented in Chapter 3, Section 3.6.

### 4.3 Case Studies

In the following sections, the implementations of the case studies and their relevant simulators used in this thesis are described. First a general case study and its general simulator is introduced. The general simulator allows agents to form and operate in multiple overlapping neighbourhoods and does not require them to stay in the collaboration, unlike the common agent-based simulator.
that agents are required to make a commitment if they agree to collaborate. The general simulator allows agents to leave and join a system freely.

The Smart Grid and Ride Sharing case studies are also introduced and their implementation details are presented. These case studies are used to evaluate CCFOM’s performance in real world application areas.
Chapter 4: Implementation

Fig. 4.11: Cascade Collaboration Process Implementation

Fig. 4.12: Decision-Making Process Implementation
4.3.1 General Case Study

Current agent-based simulators such as JADE and Jason, manage interactions using contract-net protocols, which limit the agents in open system settings. Moreover, they either do not address agents’ social dependence or consider a commitment-based approach, which requires agents to stay in the system during the course of a collaboration process [14, 15]. Therefore, for the purpose of this thesis, the common concepts in current agent-based simulators were used to develop a new simulator that is more compatible with open systems’ characteristics introduced in Chapter 1. This simulator is used to implement a multi-agent, multi-neighbourhood open system, allowing agents to leave and join the system at any timestep. The simulator allows agents to interact without forcing agents to make commitments when taking actions during a collaboration process. The following are the main aspects of the simulator as shown in Fig. 4.13:

- Time is discrete and divided into a number of timesteps. The Clock class in Fig.4.13 simulates time.
- Resources are constrained and divisible. The simulator can simulate multiple resources in a system. The capacity of the resources at each timestep is less than the number of agents in each neighbourhood. This number is calculated by the calculation entity that implements the neighbourhood.
- A neighbourhood compromise a set of agents that subscribe to use a resource. Each neighbourhood has a single resource. Neighbourhoods are allowed to overlap, when there exists at least one agent that operates in more than one neighbourhood.
- Agents have multiple goals that need the shared resource to complete.

This simulator is later used in Chapter 5, evaluating CCFOM's performance considering different levels of agents’ mobility and neighbourhoods’ density. This simulator is also used in the Smart Grid and Ride Sharing case studies as an interpreter to link the agent, neighbourhood and resource classes to their relating classes in the case studies implementations (details in Section 4.3.2, and Section 4.3.3).
4.3.2 Smart Grid Case Study

The conventional operation in the first generation of electrical grids, was a one way direction operation from the large central power plants to end-users. In these grids, the demand was reasonably stable and the grid operators’ aim was to offer inexpensive and reliable power. The conventional approaches are not efficient when facing the new properties of new generation of power grids [65]. These properties are: (1) new distributed power generation that allows the consumers to become producers, which changes the one-directional power flow to bi-directional power flow, (2) the emergence of new sensor technologies to predict the future load on the grid, (3) the use of new electrical devices that can be both energy consumers and energy storage (e.g., electrical vehicles), and (4) renewable energy generation, which is not as flexible as generating energy from fossil power plants, for instance. To take full advantage of these properties, the Smart Grid concept was introduced [65, 74]. SG is a new flexible model of electrical grid, which supports techniques such as Residential Demand Response to improve the efficiency of
the grid [67]. Residential Demand Response allows multiple stakeholders (e.g., consumers) to interact and collaboratively control and plan the electricity usage by adjusting their demand according to the available capacity of the grid. The following are a few reasons why Residential Demand Response is particularly successful [119].

- It can manage uncontrollable supply, such as renewable energy.
- It decreases energy generation cost, by shifting the unnecessary demand to off peak times.
- It increases the grid resilience, by demand reduction, which can avoid grid instability when an unpredicted event increases the load.
- It helps decrease energy bills, by cooperating with the grid’s users to shift demand to times where low cost energy is provided.

Fig. 4.14: A Single Day’s Electricity Usage

In Residential Demand Response, the demand is adjusted to the available capacity. When the demand is lower than the capacity, extra demand is created on the grid to use the available capacity, for example by turning on storage devices. When the demand is higher than the available capacity, devices can shift their demand to other times when more capacity is available. To react to these situations, devices must be able to control their own operation and decision-making process. Residential Demand Response is interested in devices with storage capability and large
power draw that can be flexible when using electricity (for example, Electrical Vehicles (EVs),
water heaters, and heaters, which have either thermal or electrical storage capabilities and are
flexible enough to reschedule their operation when there is not enough capacity available on the
grid). Fig. 4.14 shows a normal energy usage created by non-reschedulable demand (base load)
in a residential area over a day. It shows the energy usage pattern with low energy usage dur-
ing night, increased usage in the afternoon and the peak in the evening. In Residential Demand
Response, the reschedulable demand can be shifted to reduce the peak or fill the valleys. This
will smooth the demand line which allows to have a stable power generation on the grid and
increases the use of idle available capacity of the grid.

Smart Grid is particularly an interesting real world application area for CCFOM, where there
are many electrical devices sharing a constrained resource to achieve their goals. Such devices
can control their operation and intelligently use electricity to both achieve shared goal (avoid
overloading the transformer to decrease the risk of blackout), and individual goal (use low cost
energy to do their activities) simultaneously.

In our previous work on Residential Demand Response [61, 87, 88, 177], we used GridLAB-
D, which is a power network simulator that simulates an electrical grid and is produced by
the U.S. Department of Energy or Pacific Northwest National Laboratory [37]. This simulator is
open source, which makes it easy to be integrated with CCFOM implementation. Fig. 4.15 shows
how CCFOM is integrated to GridLAB-D. The Core component is responsible for controlling
the simulation and synchronizing of all the other components in GridLAB-D. The Residential
component simulates different electrical devices such as electrical vehicles, washing machine,
water heater. From these devices, particularly Electrical Vehicles (EVs) are selected, which are
implemented by EVCharger in GridLAB-D. EVs leave and return to a house, which is a good
model of an open system. Using CCFOM, EVs handle their collaboration process and pass their
finalized actions to the Residential component. As shown in Fig. 4.15, the transition and inter-
pretation of data between the Residential component and CCFOM happens in the Interpreter
component of the general simulator, which connects CCFOM to GridLAB-D’s components. At
each timestep, Interpreter converts the data acquired from transformer’s current load and its max-
imum capacity to create the resource object’s properties (e.g., capacity). The selected action and
battery charge from each electrical vehicle is also translated to agent object’s properties (e.g., action, pending actions, and achieved actions, defined in Chapter 3, Section 3.4.1). During the Decision-Making process in CCFOM, agents finalize their actions and send them to Interpreter to be sent to the Residential component.

![Diagram](image)

**Fig. 4.15**: Smart Grid Case Study: GridLAB-D Connected to CCFOM by the General Simulator

### 4.3.3 Ride Sharing Case Study

Ride Sharing is a real-time on-demand service that enables a number of passengers to share a ride on a very short notice [3]. Passengers and drivers may prefer to share a ride for different goals such as decreasing their travel cost, act in an environmentally friendly manner, or reduce journey time by using high-occupancy vehicle lanes. Passengers may have different destinations and may share their rides for some part of their journey. The implementation of a ride sharing system requires a matching mechanism that enables passengers to team up and share a ride with other passengers who do not have conflicting goals.

Ride Sharing is an interesting application area for CCFOM for several reasons. First, it has
all the characteristics of an open system, where passengers and vehicles join and leave the system frequently and unpredictably. Second, passengers and drivers have different goals (sometimes even conflicting). Third, both passengers and vehicles can be modelled as a constrained resource. Passengers are a constrained resource for vehicles, as the number of passengers asking for vehicles varies over time, and at some hours of the day, it may be less than the number of available vehicles. Vehicles are also a constrained resource at some hours of the day when there are more requests for vehicles than the actual number of available vehicles.

![Fig. 4.16: Ride Sharing Schematic View](image)

The Ride Sharing simulator developed for this thesis simulates a system of available vehicles and passengers who want to share a vehicle from the same start-point (i.e., taxi rank stations) to different destinations (see Fig 4.16). In this simulator, both passengers and drivers have multiple goals. When sharing a ride, both passengers and drivers consider their goal dependencies. To make routing easier, the simulator uses a shortest path algorithm and a grid-shape map for mapping the destinations instead of a real map of a city, as routing is not in the scope of this thesis. Fig.4.17, shows the Ride Share case study class diagram.

- Taxi ranks, represent the neighbourhoods in this application area. Each taxi rank has a limited number of taxis available to serve the passengers. This number may vary during the day.
• Taxis can be member of more than one taxi rank (i.e., neighbourhood). The taxis from different taxi ranks can interact, identify their goal dependencies, and collaborate to achieve their goals.

• Passenger Queue keeps record of the passengers (agents which are members of a neighbourhood), which want to use the available resources (taxis) in their neighbourhood.

• Passengers can only be member of one neighbourhood. They interact with other passengers in their queue, identify their goal dependencies, and collaborate to form groups to share rides using CCFOM.

• Ride-share Request, passengers can only request for a ride share from their own taxi rank. A ride share request includes the number of passengers sharing a ride and their destinations.

4.4 Summary

This chapter presented CCFOM implementation details, describing its major modules at Agent and Neighbourhood layers. It also presented a general simulator, which allows agents to form multiple overlapping neighbourhoods, and leave and join system unpredictably. This simulator will be used in the general case study in the next chapter, evaluating CCFOM considering different levels of agents’ mobility, and neighbourhoods’ density. It also introduced the simulator used for Smart Grid case study, GridLAB-D. The implementation details on how the general simulator links CCFOM to GridLAB-D is also explained. Finally, the implementation details for Ride Sharing case study is presented. These case studies evaluate CCFOM’s performance in real world application areas.
Fig. 4.17: Ride Sharing Case Study: Ride-Share Component Connected to CCFOM by the General Simulator
Chapter 5

Evaluation

This chapter presents the evaluation of Collaboration Community Formation Model (CCFOM), for agents with multiple goals operating in open systems. The evaluation includes one application-independent (general) case study to extensively evaluate CCFOM’s performance under different levels of neighbourhoods’ density and agents’ mobility in an open system. CCFOM’s ability to operate in complex real world applications is evaluated using a Smart Grid and a Ride Sharing case study.

The reminder of this chapter is organized as follows. Section 5.1 presents the evaluation metrics used to evaluated CCFOM in different case studies. Section 5.2 presents the results obtained from the general case study for each of the evaluation metrics. Section 5.3 introduces the Smart Grid case study and presents the results. Section 5.4 introduces the Ride Sharing case study and presents the results obtained. Finally, Section 5.5 summarizes the results.

5.1 Requirements and Evaluation Metrics

This section presents the requirements for evaluating the claims made about CCFOM throughout the previous chapters. Table 5.1 shows the evaluation metrics mapped to CCFOM’s design requirements outlined in Chapter 3, which were derived from the research questions presented in Chapter 1. The evaluation metrics are explained as follows and are summarized in Table 5.2.
1. Agents’ proportion of access to the constrained shared resource: Measured by comparing the results obtained when agents operate in a single neighbourhood membership setting and a multiple overlapping neighbourhood membership setting. Agents’ access to the shared resource should be increased when they operate in multiple overlapping neighbourhoods. This metric evaluates Contribution 3, which is concerned with agents’ operation in multiple overlapping neighbourhoods. The hypothesis being tested here is that by allowing agents to operate in multiple overlapping neighbourhoods, they can both increase their own access to shared resource by alternating between neighbourhoods and also allow agents with single neighbourhood membership to have a higher chance to access the shared resource.

2. Agents’ individual goal achievement: Measured by evaluating their access to the shared resources, the closer they get to their Target value the better result is achieved.

3. Collaboration success: This metric measures the shared goal achievement. For example, the lower the number of shared resource overload, the better result is achieved for shared goal in the general and Smart Grid case studies. Agents’ individual goal achievement and collaboration success metrics address Contribution 1 and Contribution 2, where agents’ ability to identify their goal dependency and adaptability is tested. The hypothesis being tested in metric 2 and metric 3 is that by enabling agents to understand their goals’ dependencies, they can achieve a better balance of shared and individual goal achievement compared to when they do not understand their goal dependencies.

4. Computation and communication costs: A measure of the communication and computation costs of successfully finding a collaboration community. The lower the number of messages (sent/received) to form a collaboration community and the lower the number of conflicts in a collaboration community, the lower communication and computation costs are achieved, respectively.
Table 5.1: Evaluation Metrics’ Mapping to Research Questions

<table>
<thead>
<tr>
<th>Research Questions</th>
<th>Component Addressed</th>
<th>Evaluation Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RQ1</strong>: Identify goal dependencies</td>
<td>Social Reasoning, Self-Evaluation and Adaptation</td>
<td>Agents’ individual goal achievement</td>
</tr>
<tr>
<td><strong>RQ2</strong>: Self-interest adaptation</td>
<td></td>
<td>Agents’ shared goal achievement (Collaboration success rate)</td>
</tr>
<tr>
<td><strong>RQ3</strong>: Operate in overlapping neighbourhoods</td>
<td>Multiple Neighbourhood Membership</td>
<td>Agents’ proportion of access to the constrained resource</td>
</tr>
<tr>
<td>Decentralised decision-making</td>
<td>Decentralised Decision-Making</td>
<td>Communication and Computation cost</td>
</tr>
<tr>
<td>Evaluation Metrics</td>
<td>General case study</td>
<td>Smart Grid case study</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Agents' proportion of access to the constrained resource</td>
<td>Section 5.2.2.1, Fig. 5.3, 5.4, 5.5, 5.6</td>
<td>Section 5.2.2.3, Fig. 5.7, 5.8</td>
</tr>
<tr>
<td>Number of conflicts in the formed collaboration</td>
<td>Section 5.2.2.1, Fig. 5.3, 5.4, 5.5, 5.6</td>
<td>Section 5.2.2.3, Fig. 5.7, 5.8</td>
</tr>
<tr>
<td>EEVs and EVs' SoC</td>
<td>Section 5.3.2.1, Table 5.6</td>
<td>Section 5.3.2.1, Table 5.6</td>
</tr>
<tr>
<td>Passengers reaching their destinations</td>
<td>Section 5.4.2.1, Figs. 5.18, 5.19, 5.20</td>
<td>Section 5.4.2.1, Figs. 5.18, 5.19, 5.20</td>
</tr>
<tr>
<td>Transformer load</td>
<td>Section 5.3.2.2, Figs. 5.13</td>
<td>Section 5.3.2.2, Figs. 5.13</td>
</tr>
<tr>
<td>Decreased CO(_2) emission</td>
<td>Section 5.3.2.2, Figs. 5.3, 5.4, 5.5, 5.6</td>
<td>Section 5.3.2.2, Figs. 5.3, 5.4, 5.5, 5.6</td>
</tr>
<tr>
<td>Agents' individual goal achievement</td>
<td>Section 5.4.2.2, Fig. 5.20</td>
<td>Section 6, Table 5.6</td>
</tr>
<tr>
<td>Agents' shared goal achievement</td>
<td>Section 5.4.2.2, Fig. 5.20</td>
<td>Section 6, Table 5.6</td>
</tr>
<tr>
<td>Number of conflicts in the formed collaboration</td>
<td>Section 5.2.2.2, Fig. 5.7, 5.8</td>
<td>Section 5.2.2.2, Fig. 5.7, 5.8</td>
</tr>
<tr>
<td>Agents' individual goal achievement</td>
<td>Section 5.4.2.2, Fig. 5.20</td>
<td>Section 6, Table 5.6</td>
</tr>
<tr>
<td>Increased communication and computation costs</td>
<td>Section 5.2.2.3, Fig. 5.9</td>
<td>Section 5.2.2.3, Fig. 5.9</td>
</tr>
<tr>
<td>Number of conflicts in the formed collaboration</td>
<td>Section 5.2.2.2, Fig. 5.7, 5.8</td>
<td>Section 5.2.2.2, Fig. 5.7, 5.8</td>
</tr>
<tr>
<td>Agents' individual goal achievement</td>
<td>Section 5.4.2.2, Fig. 5.20</td>
<td>Section 6, Table 5.6</td>
</tr>
<tr>
<td>Communication and computation costs</td>
<td>Section 5.2.2.3, Fig. 5.9</td>
<td>Section 5.2.2.3, Fig. 5.9</td>
</tr>
</tbody>
</table>

**Table 5.2**: Evaluation Metrics in Each Case Study
5.2 Application-Independent case study (General case study)

For the purpose of this thesis, an application-independent case study was developed to allow flexibility with experiment design and to model multiple overlapping neighbourhoods. Using this case study, a system of six neighbourhoods was modelled, as illustrated in Fig. 5.1. Agents use the resources available in each neighbourhood to achieve their goals.

This case study allows CCFOM to be evaluated under circumstances not encountered in the other two case studies, in particular, different levels of neighbourhoods’ density, and agents’ mobility.

5.2.1 Experiment Design

12 different combinations of parameters are used to evaluate the first three metrics, presented as four sets of experiments. In each experiment, there are 3 scenarios (i.e., Collaborative, Combination (CCFOM), Competitive). Each experiment has the same density and mobility parameter setting for the three scenarios (see Table 5.3).

- **Experiment 1** covers dense neighbourhoods with medium mobility agents.
- **Experiment 2** covers medium density neighbourhoods with medium mobility agents.
- **Experiment 3** covers dense neighbourhoods with fast mobility agents.
- **Experiment 4** covers medium density neighbourhoods with fast mobility agents.

5.2.1.1 Parameter Settings

Table 5.5 shows the parameter settings for agents’ goals, mobility, dependency and the neighbourhoods’ density. These parameter are discussed below.

1. **General Settings**: All experiments implement an open multi-agent multi-neighbourhood system with 100 agents operating in 6 overlapping neighbourhoods. Agents leave and join the system unpredictably. Fig. 5.2 and Table 5.4 show the number of neighbourhood memberships for each agent, which is the same for all experiments (for example agents 25
Table 5.3: Four Experiments Evaluating CCFOM Under Varying Levels of Neighbourhoods’ Density and Agents’ Mobility in Collaborative, Competitive and Combination (CCFOM) Scenarios

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Neighbourhoods’ Density</th>
<th>Agents’ Mobility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>Dense</td>
<td>Medium</td>
</tr>
<tr>
<td>Experiment 2</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Experiment 3</td>
<td>Dense</td>
<td>Fast</td>
</tr>
<tr>
<td>Experiment 4</td>
<td>Medium</td>
<td>Fast</td>
</tr>
</tbody>
</table>

to 30 are members of two neighbourhoods). Members of more than one neighbourhood can use the resources interchangeably, with no additional costs or benefits. Agents decide the SetPoint, and Target for their individual goals by setting the values of $\alpha_{g_k}$, (i) specifying the minimum number of actions, which will use the shared resource, and $\Lambda_{g_k}$, specifying the maximum number of actions, which will use the shared resource, each choosing a random number within the below ranges:

\[
30 < \alpha_{g_k} < 80
\]

\[
t_{\text{arrive}} + 30 < t_{k}^{t} < t_{\text{arrive}} + 80
\]

\[
50 < \Lambda_{g_k} < 80
\]

\[
t_{k}^{\prime\prime} = t_{\text{leave}}
\]

Fig. 5.1: An Overview of How the Neighbourhoods Overlap
Table 5.4: Neighbourhoods’ Membership

<table>
<thead>
<tr>
<th>Agent ID</th>
<th>Neighbourhood ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁ to a₂₅</td>
<td>nₙr₁</td>
</tr>
<tr>
<td>a₂₆ to a₃₀</td>
<td>nₙr₂, nₙr₃</td>
</tr>
<tr>
<td>a₃₁ to a₃₅</td>
<td>nₙr₂, nₙr₃, nₙr₅</td>
</tr>
<tr>
<td>a₃₆ to a₅₃</td>
<td>nₙr₃, nₙr₆</td>
</tr>
<tr>
<td>a₅₄ to a₆₄</td>
<td>nₙr₂, nₙr₃, nₙr₄, nₙr₆</td>
</tr>
<tr>
<td>a₆₅ to a₁₀₀</td>
<td>nₙr₆</td>
</tr>
</tbody>
</table>

2. **Resource Settings**: Resources are divisible and their capacity varies in each timestep based on the population of the neighbourhoods. Each agent can use only one unit of the resource at each timestep. The resource in each neighbourhood is modelled implicitly by defining the neighbourhood’s density.

3. **Neighbourhood’s Density**: The density of a neighbourhood is sparse when its resource capacity is equal to 100% of the population in each timestep (in neighbourhoods with sparse density, all agents have access to their required resource at all the time, and so there is no need to adjust their self-interest level. Therefore, neighbourhoods with sparse density are not addressed in the following experiments, because the focus of the thesis is on constrained resource environments). The density of a neighbourhood is medium, when its resource capacity is 65% of the population and it is dense, when its capacity is 50% of the population. By running 100 rounds of experiments the variables for defining the density of a neighbourhood are chosen in way to create situations where there is not enough resource for all the agents. This allows the scenarios to test agents behaviour when they are required to adapt their goals and adjust their level of self-interest and cooperation. In the following experiments medium and dense density is considered, as operating in sparse environment means that all agents have access to the resource they want and they would need to cooperate or adapt their behaviour.

4. **Agents’ Mobility**: In open systems it is important to consider agents’ frequent leaving
Chapter 5: Evaluation

Fig. 5.2: Agents’ Neighbourhood Membership Setting

and joining as this impacts their behaviour including their willingness to behave collaboratively. This concept is tested here by considering different mobility pace for agents. Agents’ Mobility is Slow, when agents’ stay duration \(d\) in a neighbourhood is longer than the maximum number of timesteps they need to access the resource. It is Medium, when \(d\) is equal to 80\% of the number of their pending actions \(d = 0.8 \times |P \Delta_{a_j}(g_k)|\), and it is Fast, when \(d = 0.5 \times |P \Delta_{a_j}(g_k)|\). In the following experiments only medium and fast mobility is considered for agents as when their mobility is slow, they spend enough time in a system to use the resource they want, and even a very simple scheduling algorithm can coordinate their behaviour. On the other hand, for choosing the values for defining medium and fast mobility, rounds of experiments are run. The medium value is chosen at the pace when the have enough time to behave collaborative and strict enough to choose their actions intelligently. The Fast is also set to show when they do not spend enough time in a system, how would they behave.

5. Agents’ Dependency: In the Collaborative scenario, FD, SqD, OvD, and CncD are equally distributed amongst agents. In the Competitive scenario, CnfD and CmpD dependencies exist amongst agents, and in the Combination (CCFOM) scenario, there is an equal distribution of all types of dependencies between agents, (i.e., 10 FD, 10 SqD, 20 OvD, 20 CncD, 20 CmpD, and 20 Cnfd). These dependencies were explained in Chapter 3, Section 3.4.
Chapter 5: Evaluation

Table 5.5: Parameter Settings: 100 Agents Operating in 6 Neighbourhoods

<table>
<thead>
<tr>
<th>Parameter Settings</th>
<th>General Settings</th>
<th>Neighbourhoods’ Density</th>
<th>Agent’s Mobility</th>
<th>Agent’s Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual Goals (SetPoint and Target)</td>
<td>Sparse (Capacity_{rk} = 100%)</td>
<td>Medium (Capacity_{rk} = 65%)</td>
<td>Collaborative (FD, SqD, OvD, CncD)</td>
</tr>
<tr>
<td></td>
<td>30 &lt; α_{gk} &lt; 80 and t_{arrive} + 30 &lt; t'<em>k &lt; t</em>{arrive} + 80</td>
<td>Medium (Capacity_{rk} = 65%)</td>
<td>Medium (d = 0.8 *</td>
<td>P Δ_{aj}(g_k)</td>
</tr>
<tr>
<td></td>
<td>50 &lt; Δ_{gk} &lt; 80 and t''<em>k = t</em>{leave}</td>
<td>Dense (Capacity_{rk} = 50%)</td>
<td>Fast (d = 0.5 *</td>
<td>P Δ_{aj}(g_k)</td>
</tr>
</tbody>
</table>

5.2.1.2 Baseline Methods

CCFOM was compared against a number of different baseline approaches.

- For the first three evaluation metrics, a Collaborative approach and a Competitive approach are considered as baselines. In the Collaborative approach, agents’ shared goal (decreasing the number of overloads) has a higher priority than their individual goals. When the need for collaboration is identified, agents which have lower Self-Priority to use the shared resource will change their actions, to avoid resource overload. The priority calculation formula, introduced in Chapter 3, Section 3.5.1 is used in this approach. In the Competitive approach, agents are self-interested and do not collaborate. They take their actions without considering their neighbours’ priorities, or the shared goal. In these sets...
of experiments, a Combination (CCFOM) scenario is used, which considers a combination of goal dependencies and agents use CCFOM during their operation for collaboration community formation. The Combination (CCFOM) scenario is compared against the baselines.

• To evaluate the communication and computation costs, two sets of baselines, which implement the state of the art functionality, were used. For the communication cost metric, CCFOM is compared against dynamic coalition formation algorithms, in particular, Simulation-based Dynamic Coalition Formation (DCF-S) and Transitive Dependence-based Dynamic Coalition Formation (DCF-TD).

  – DCF-S [82, 191], is an algorithm where agents operate in open settings and cooperate to achieve a set of goals. Each agent can initiate a collaborative process with the following steps: preparation, choosing a goal to achieve, simulation, in which the initiator agent simulates any possible coalition by randomly adding and removing candidates and calculates the contribution of each individual added agent, and negotiation, in which agents may or may not agree to cooperate in the determined coalition structure in the previous step.

  – DCF-TD [8, 55], is an algorithm in which agents use social reasoning to form coalitions and includes information gathering, transitive dependence-based reasoning, negotiation, and coalition resolution. For the computation cost, DCF-TD’s social reasoning processing is compared to CCFOM’s social reasoning.

5.2.1.3 Performance Criteria

The performance criteria for the general case study, which map the obtained results to the initial research questions (summarized in Table 5.1) are as follows:

• Agents’ proportion of access to the constrained shared resource which is discussed in Section 5.2.2.1, point 4.
• Agents’ individual goal achievement, which is calculated using the amount of the shared resources used by each individual agent, discussed in Section 5.2.2.1.

• Collaboration success (shared goal achievement), which is calculated by the ratio of the successful collaboration processes (i.e., the number of times the demand on resources that could make them overloaded was decreased by collaboration) to the number of all the collaboration processes over the simulation cycle. The demand on the shared resource is also reported in Section 5.2.2.2.

• Computation and communication costs, which is measured by calculating the big O complexity of two state of the art baselines in Section 5.2.2.3.

5.2.2 Results: Application-independent (general) case study

The results obtained from the general case study map to the evaluation metrics discussed in Section 5.1, as follows:

5.2.2.1 Agents’ Individual goal achievement

Resource usage for each agent over 10 runs of simulation is reported in Fig. 5.3, Fig. 5.4, Fig. 5.5, and Fig. 5.6. Each run’s duration is 80 timesteps (the maximum number of timesteps agents needed to access the resource). The bar on top of each figure indicates the number of neighbourhood memberships for each agent. To explain the results, the three different scenarios, (i.e., Collaborative, Combination (CCFOM), and Competitive) are compared in all four experiments. The following discussion considers the effect of agents’ dependencies on their goal achievement when the density and mobility properties are constant (Comparison 1-Dependency), the effect of density when the dependency and mobility are constant (Comparison 2-Density), and the effect of mobility when the other variables are constant (Comparison 3-Mobility). The result obtained by agents with single neighbourhood membership is compared against agents with multiple neighbourhood memberships (Comparison 4). Finally, there is a discussion on how the dependency model helped agents to adjust their level of self-interest and behave in a neighbourly fashion to acquire a better individual and shared goal achievement (Comparison 5).
Chapter 5: Evaluation

Fig. 5.3: Individual Goal Achievement, (Experiment 1: Dense Neighbourhoods, Medium Mobility)

1. **Comparison 1-Dependency**: As shown in Fig. 5.3, Fig. 5.4, Fig. 5.5 and Fig. 5.6, in the Collaborative scenarios, agents’ resource usage does not deviate as much as other two approaches, for the agents operating in single neighbourhood. However, the resource usage deviates more for the agents with multiple neighbourhood memberships and it is also greater than the resource usage of agents with single membership. In the Combination (CCFOM) scenarios, agents’ resource usage deviates more as they are not as collaborative as agents in the Collaborative scenarios. However, as the median value (shown by a red dash on each box-and-whisker bar for each agent) and distribution of the data (shown on the bars) shows, they have still achieved a fair access to the resource. In each set, the resource usage in the Competitive scenarios, is higher than the Collaborative and Combination (CCFOM) scenarios. This is because agents are self-interested and have not collaborated.

In the Collaborative scenarios, the resource usage is less deviated compared to the Com-
Fig. 5.4: Individual Goal Achievement, (Experiment 2: Medium Density Neighbourhood, Medium Mobility)

Combination (CCFOM) and Competitive scenarios, and the mean values are very close to each other, which means agents had a fair access to the shared resources. In all experiments, the highest resource usage has happened in the Competitive scenarios. In the Combination (CCFOM) scenarios, agents’ different dependency types result in both competition and collaboration between the agents. In these scenarios, the collaboration decreases agents’ resource usage compared to the Competitive scenarios, and the competition between the agents results in higher resource usage compared to the Collaborative scenario.

2. **Comparison 2-Density**: Different scenarios in Experiment 1 are compared to their counterparts in Experiment 2, where agents’ mobility is medium, and scenarios of Experiment 3 are compared to their counterparts in Experiment 4, where agents’ mobility is fast. In all comparisons, agents in neighbourhoods with higher density, have proportionally less access to the resource (e.g., the resource usage in all scenarios in Experiment 1 is lower
Chapter 5: Evaluation

Fig. 5.5: Individual Goal Achievement, (Experiment 3: Dense Neighbourhoods, Fast Mobility)

than their counterparts in Experiment 2). Comparing the Collaborative, Combination (CCFOM), and Competitive scenarios of Experiment 1 to Experiment 2, and Experiment 3 to Experiment 4, it can be concluded that the denser the neighbourhood, the less resource there is for each agent, as there are more agents to use the resources in each neighbourhood.

3. **Comparison 3-Mobility**: Different scenarios in Experiment 1 are compared to their counterparts in Experiment 3, where the neighbourhoods are dense, and the scenarios of Experiment 2 are compared to their counterparts in Experiment 4, where the neighbourhoods’ density is medium. As shown in Fig. 5.3 and Fig. 5.5, agents with medium mobility in Experiment 1, have a higher resource usage and lower deviation compared to agents with fast mobility in Experiment 3, as they spend more time in the neighbourhood. The same results can be concluded by comparing Experiment 2, Fig. 5.4 to Experiment 4, Fig. 5.6. From comparing the agents’ performance under varying levels of dependency, density and
mobility, it can be concluded that agents in the Collaborative scenario always achieve the fairest resource usage. Agents in the Combination (CCFOM) scenario always achieve a fairer access to the resource when compared to the Competitive scenario. In all scenarios, the amount of resource used by each agent decreases when they move faster and the neighbourhood is denser.

4. **Comparison 4-Single membership/ multiple membership**: In all four experiments, the agents with multiple neighbourhood memberships (e.g., agents’ ID 26 to 65) have achieved proportionally higher resource usage, which means they have either found better dependency types or higher priorities to use the resource in another neighbourhoods.

5. **Comparison 5-Dependency model effect on neighbourly behaviour**: In each experiment, the Collaborative and Competitive scenarios are baselines, and the Combination (CCFOM) scenario can be considered as a normal environment with all sorts of depen-
dencies between agents. In all 4 experiments, the amount of resource usage in the Collaborative scenario has the lowest deviation, as agents have accessed a fair proportion of shared resource through collaboration. The Combination (CCFOM) scenario is also less deviated than the Competitive scenario. This shows that some degree of collaboration has also happened in this scenario, which is possible only when agents use their dependency model to identify their goal dependencies and and adjust their level of self-interest.

5.2.2.2 Collaboration success

1. **Shared Resource Utilization**: Agents operating in a neighbourhood use a shared resource to achieve their individual goals, and collaborate to adjust the demand on the shared resource to decrease its overload range. In this section, the resource demand of different scenarios in Experiment 3, which has the most extreme parameter settings (where agents’ mobility is fast and neighbourhoods are dense,) is reported. The same pattern was obtained in the other experiments. Fig. 5.7 shows the remaining resource capacity when agents finalized their action in each timestep. The resource is allowed to go below zero to model the overload. As shown in Fig. 5.7, the mean value is above 0 in 98.2% of timesteps in the Collaborative scenario and it is above 0 in 97.6% of timesteps in the Combination (CCFOM) scenario. This means that, on average, the resources were not overloaded in 98.2% and 97.6% of timesteps. However, the results in the Competitive scenario are not promising as the resource is overloaded in 64.2% of timesteps on average, with the resources overload by 2 times of their capacity in 60% of timesteps.

2. **Collaboration Success Rate**: Fig. 5.8 shows the success rate of all four experiments and in each three scenarios. These results are obtained from a run with overall 80 timesteps, using the total number of times the collaboration need was identified and the number of collaborations that were successful (i.e., the resource is not overloaded). As shown in this figure, the collaboration success rate in the Competitive scenario is under 0.15, which means there has been a low number of successful collaborations amongst self-
interested agents. On the other hand, the Collaborative scenario has achieved a better success rate, varying between 0.6 to 1. This shows that collaborative agents were successful when they were spending enough time in the neighbourhood. The most interesting results are achieved from the Combination (CCFOM) scenario, in which a combination of goal dependencies was considered and agents could adjust their level of self-interest and cooperation (using their goal dependencies and Self-Priority). The results show that they achieved a success rate between 0.39 to 0.89 depending on the amount of time they spend in the neighbourhood. The results obtained from this scenario are only possible when agents can understand their goal dependencies and can form effective collaboration communities.

5.2.2.3 Computation and Communication costs

To understand the commutation and communication costs, CCFOM is compared to two other state of the art methods considered as baselines (introduced in Section 5.2.1.2).

- Computational complexity: To calculate the computation cost big O notation is used to evaluate the performance of CCFOM compared to DCF-S and DCF-TD. The big O for
these two methods is calculated using their algorithm in [82, 191], [8, 55], respectively. Both DCF-S’s and DCF-TD’s main algorithms include three nested loops, to consider all combinations of the possible coalitions. Therefore their big $O$ is $n^3$. On the other hand, CCFOM’s main algorithm has only one loop which makes the the complexity to be $n$. The computational complexity of CCFOM is linear, and considerably lower than the baseline approaches. This is because the invited agents separately evaluate their dependencies and priorities and decide whether or not to engage. CCFOM’s distributed decision-making algorithm is particularly applicable in time-constrained real time applications. The other two approaches, similar to many other approaches in the literature [139], run a pre-calculation, in which they consider a sub-set of possible communities that can be formed. This calculation is practically costly and not efficient in time-constrained environments. In some existing approaches, the complexity issue is avoided by omitting the pre-calculation phase and inviting all the available agents to a negotiation process. However, this only moves the computation complexity to the negotiation phase [55]. In contrast, using the CCFOM’s dependency model, agents continue the process independently in a distributed manner, without any negotiation.

Fig. 5.9 shows the number of candidates considered to form collaboration communities (possible collaborators), the maximum number of agents that can accept to form collaboration communities, and the number of agents that enter a negotiation process in CC-
FOM and DCF-TD in 3 different dependency settings, (1) Collaborative Dependencies, where only CncD, SqD, OvD and FD exist, (2) All types of dependencies exist, and (3) Competitive Dependencies, where only CnfD and CmpD exist. In Fig.5.9, where only collaborative dependencies exist, all the agents in the neighbourhood can be a possible candidate to form a collaboration community, in both approaches. In CCFOM, the number of possible candidates is decreased, when all types of dependencies exist between agents, and it is zero, when only competitive dependencies (i.e., CnfD and CmpD) exist between agents. Using CCFOM, agents can identify their goal dependencies and exclude the agents on which they have conflicting dependencies from the set of possible candidates. However, this number is constant in negotiation-based approaches such as DCF-TD. The communication cost in baseline approaches is not limited to the number possible candidates, as agents start a negotiation process afterwards which requires multiple rounds of message passing, whereas in CCFOM, agents act independently and decide in a distributed manner (see Algorithm 6).

![Fig. 5.9: Communication Cost Analysis](image)

### 5.2.2.4 Summary of the results from the application-independent case study

In summary, from the results obtained in Section 5.2.2.1, it is shown that agents operating in multiple neighbourhoods had a better access to resource themselves, and had decreased the de-
mand on their initial neighbourhoods by changing their resource request to an alternative neighbourhood (addressing RQ3: Operate in overlapping neighbourhoods). The results also showed that agents in the Combination (CCFOM) scenario achieved a fairer resource usage compared to agents in the Competitive scenario in all experiments, which shows agents were able to understand their goal dependencies and adjusted their level of self-interest and cooperation, and had successful collaborations (addressing RQ1: Identify goal dependencies). However, when the agents’ mobility and neighbourhoods’ density is increased, agents’ resource usage is decreased and it is more deviated, as agents do not spend enough time in the system to increase their resource usage. Moreover, agents achieved a better balance at achieving multiple goals (both individual and shared goals) in the Combination (CCFOM) scenario, when comparing the results from agents’ resource usage and the shared resource demand shown in Fig. 5.5 and Fig5.7, respectively (addressing RQ2: Self-interest adaptation). Finally, the results show that CCFOM has a lower computation and communication cost, because of its decentralised implementation.

5.3 Smart Grid

Energy demand is unevenly distributed over a day, depending on households’ energy consumption. Demand usually increases in the morning and peaks in the evening (when people get ready for a working day in the morning and when they get back home in the evening), and the off-peak hours starts from mid-night. The maximum demand determines the grid capacity for the whole day. If demand increases during peak hours, the utility companies have to turn on more generators, which is costly. Shifting the reschedulable demand from peak hours to off-peak hours is an approach to use the available capacity of the grid in off-peak hours and decreasing cost for companies and end-users [61, 67, 119].

5.3.1 Experiment Design

As depicted in Fig. 5.10, a neighbourhood in this Smart Grid scenario includes a number of houses that are served with a single transformer. In this thesis, the neighbourhood includes a res-
idential area of 90 houses. Each house uses different electrical devices regularly, which creates a base load (non-schedulable demand) on the grid. Each household also has an Electrical Vehicle (EV) that needs to be charged for its next journey. The scenario includes a maximum number of 80 normal Electrical Vehicles (EVs), and 10 Emergency Electrical Vehicles (EEVs) (e.g., for local doctors in the community), which are controlled by agents and create reschedulable load on the grid. Each EV/EEV has its own daily plan which includes the time they leave/arrive home, and the distance they travel each day.

EVs and EEVs have varying individual goals and shared goal.

- **EV’s Individual Goal:** Each EV aims to achieve a certain amount of charge for the next journey, depending on its travelling time.

- **EEV’s Individual Goal:** EEVs’ aim is to achieve 100% battery charge as soon as possible.

- **Shared Goal:** Both EVs and EEVs share the same constrained resource (available capacity on the grid), and their shared goal is to avoid overloading the transformer as much as possible while simultaneously achieving their individual goals.

EVs and EEVs can choose ON or OFF as their actions in each timestep (e.g., 15 minutes).
Chapter 5: Evaluation

Each EV or EEV individually decides its actions. An EEV starts charging (action=ON) as soon as it gets back home. The normal EVs try to decrease their cost by shifting their charging to off-peak hours, when low cost energy is provided. In CCFOM’s implementation of the Smart Grid scenario, a single neighbourhood which is served by one transformer is considered. In this scenario, each EV and EEV share their external description which includes their goals, decided action for the next timestep, policy, the number of neighbourhoods of which they are a member, Self-Priority and transferred priority. When the need for collaboration emerges the EV or the EEV that has identified a possible overload on the transformer for the coming timestep, nominates all the EVs and EEVs that have chosen ON as their action for the next timestep. These EVs and EEVs then form their Neighbouring Agent Dependency Model using each others’ external description. They run the Self-evaluation and Adaptation process and decide whether to change their actions to OFF and transfer their priority to another EV or EEV. In the Decision-Making process, EVs and EEVs finalize their final actions according to their priorities and goal dependencies.

5.3.1.1 Parameter Settings

1. **General Settings:** EVs and EEVs need to determine their *SetPoint* and *Target* for their individual goals by setting the values of $\alpha_{g_k}$, specifying the minimum number of actions, which will use the shared resource, and $\Lambda_{g_k}$, specifying the maximum number of actions, which will use the shared resource. Each EV decides these numbers based on its daily travel plan.

   \[
   30 < \alpha_{g_k} < 50 \quad t_{\text{arrive}} < t'_{k} < t_{\text{arrive}} + 50 \\
   60 < \Lambda_{g_k} < 100 \\
   t''_{k} = t_{\text{leave}}
   \]

   EEV’s $\alpha_{g_k}$ and $\Lambda_{g_k}$ values for determining their *Setpoint* and *Target* are as follows:

   \[
   \alpha_{g_k} = 100 \quad \text{and} \quad t'_{k} = t_{\text{leave}} \\
   \Lambda_{g_k} = 100 \quad \text{and} \quad t''_{k} = t_{\text{leave}}
   \]
2. **Resource Settings**: The resource’s available capacity is set to be 350 KW at each timestep, which is 85% of actual grid capacity. In a real setting, experts adapt it according to predicted consumptions.

3. **Neighbourhood’s Density**: Smart Grid scenario includes a normal density neighbourhood based on the real data obtained from a smart-meter trial in Ireland [2].

4. **Agents’ Mobility**: EEVs’ mobility level is considered to be fast, as they only spend enough hours at home to charge the vehicles. EVs’ mobility is set to be a continual 12 hours home and 12 hours away plan.

5. **Agents’ Dependency**: In the Collaborative scenario, FD, SqD, OvD, and CncD are equally distributed amongst EVs. In the Competitive scenario, CnfD and CmpD dependencies exist amongst EEVs and EVs, and in the Combination scenario there is an equal distribution of all types of dependencies between EVs. As shown in Fig. 5.11, there are only FD and CmpD dependencies between EVs and EEVs. An EEV has CnfD dependencies with other EEVs, and EVs can have OvD, CncD, CmpD, and CnfD dependencies.

### 5.3.1.2 Base Line Methods

CCFOM’s performance is compared to a set of approaches: **Greedy, Collaborative, Competitive**. In the Greedy approach, EVs and EEVs do not use any artificial intelligence to achieve their goals and start using electricity to charge their batteries as soon as they come back home,
without considering the demand on the grid or their neighbours’ priorities. In the Competitive approach, EVs and EEVs are self-interested, their individual goals have more priority than the shared goal, they take their individually decided actions independently, and do not collaborate. In the Collaborative approach, EVs and EEVs collaborate to achieve the shared goal. Variation of demand on the grid results in variations of available capacity of the grid. If the collective demand from EVs and EEVs cannot be handled by the available transformer capacity at each timestep, there is a need for collaboration. In the Collaborative approach, EVs and EEVs choose not to use the electricity, using their calculated priorities when the demand is high and the transformer is going to be overloaded.

5.3.1.3 Performance Criteria

The performance of CCFOM in Smart Grid is measured by the following metrics.

1. Agents’ individual goal achievement, which is measured by EEVs’ and EVs’ individual goal failure rate, EVs’ State of Charge (SoC) standard deviation (STDEV). Individual goal failure rate reports the percentage of EVs that run out of charge, and EEVs that could not get the required battery charge at their specified time, and STDEV shows the fairness of the approaches (Section 5.3.2.1).

2. Collaboration success (shared goal achievement) is measured by, (1) Peak-to-Average Ratio (PAR), which shows the smoothness of the demand distribution over time, the lower the PAR, the better demand distribution is achieved, (2) Transformer load graph also shows the demand distribution and load shifting of different approaches over time, and (3) shared goal failure rate, which reports the percentage of transformer overload (Section 5.3.2.2).

3. Communication cost, which is measured by the number of conflicts in the formed collaboration communities. The results are obtained from the first 2 hours of the evening time, which includes 9 timesteps. Note, collaboration is most likely to happen at the peak hours, as only a limited amount of grid capacity is available to be used by EVs and EEVs, which is why it is shown only for those 2 hours (Section 5.3.2.3).
5.3.2 Results: Smart Grid case study

5.3.2.1 Agents’ individual goal achievement

1. **Individual goal failure rate**, reported in Fig. 5.12 (a), show that the Competitive approach has achieved the best results with a failure rate of 0%, compared to the Collaborative approach and CCFOM. This implies that, in the Competitive approach, EVs and EEVs only considered their individual goals and did not consider the transformer’s constraints, achieving 100% of their individual goals. This is also recorded in Table 5.6, where the number of EVs with \( \text{SoC} \leq 0 \), and the number of EEVs with \( \text{SoC} < 100 \) are both recorded as 0. As shown in Fig. 5.12(a), the Collaborative approach has the highest individual goal failure rate, as all the EVs and EEVs in this approach have collaborated to achieve the shared goal. The results for CCFOM shows 0% individual goal failure rate for day 2 and day 3 and 3.3% for day 1. The figures recorded in Table 5.6 show that three EVs run out of charge in day 1.

2. **EVs’ SoC standard deviation (STDEV)**, reported in Table 5.6 shows that the EVs using the Competitive approach and CCFOM have achieved a fairer distribution of SoC. In the Competitive approach all the EVs and EEVs take their actions individually and do not consider the constraints of the transformer. In CCFOM, EVs’ have considered their goal dependencies and their priorities when forming a collaboration communities. However, in the Collaborative approach, EVs and EEVs goal dependency is not considered and they collaborate only to achieve the shared goal.

5.3.2.2 Collaboration Success

1. **PAR** results recorded in Table 5.6 show that the Collaborative approach has achieved the smoothest transformer load. This implies that it has achieved the shared goal better than the Combination (CCFOM) and Competitive approaches. This is because in the Collaborative approach, all EEVs and EVs are collaborative and collaborate to decrease the demand in peak hours to achieve their shared goal. Table 5.6 also shows that the CCFOM has achieved a lower PAR than the Competitive approach.
Fig. 5.12: Agents’ Goal Failure Rate

The CCFOM approach has achieved better PAR results compared to the Competitive approach. This is because in the Competitive approach, EVs and EEVs do not adapt their level of self-interest and their individual goal (e.g., having certain amount of charge) has priority over the shared goal (e.g., minimizing the transformer’s overload times).

2. The transformer load graph depicted in Fig. 5.13 shows the demand on the transformer over three days. The Greedy approach increases the demand during peak hours as both EEVs and EVs start charging as soon as they get back home to achieve their individual goals. However, their shared goal is not achieved as the transformer load has passed the 85% of the grid capacity in all three days. The Combination (CCFOM), Competitive and Collaborative approaches, have successfully shifted the load to the off-peak hours and decreased the demand compared to the Greedy approach. However, the Competitive and Combination (CCFOM) approach have passed the 85% of the grid capacity in some of the timesteps. As shown in Table 5.6, the demand from the Combination (CCFOM) approach is higher than the Collaborative approach and lower than the Competitive approach during peak hours (see Fig. 5.13). This is because, unlike the Collaborative approach, in the Combination (CCFOM) approach, EVs consider their goal dependencies on EEVs and shift their demand to off-peak hours to avoid overloading the transformer and allow EEVs
Table 5.6: Smart Grid Case Study Results Over 3 Days

<table>
<thead>
<tr>
<th></th>
<th>Statistical Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PAR</td>
</tr>
<tr>
<td>Day 1</td>
<td></td>
</tr>
<tr>
<td>Collaborative</td>
<td>1.66</td>
</tr>
<tr>
<td>Competitive</td>
<td>1.45</td>
</tr>
<tr>
<td>Combination (CCFOM)</td>
<td>1.59</td>
</tr>
<tr>
<td>Day 2</td>
<td></td>
</tr>
<tr>
<td>Collaborative</td>
<td>1.78</td>
</tr>
<tr>
<td>Competitive</td>
<td>1.51</td>
</tr>
<tr>
<td>Combination (CCFOM)</td>
<td>1.55</td>
</tr>
<tr>
<td>Day 3</td>
<td></td>
</tr>
<tr>
<td>Collaborative</td>
<td>1.76</td>
</tr>
<tr>
<td>Competitive</td>
<td>1.49</td>
</tr>
<tr>
<td>Combination (CCFOM)</td>
<td>1.63</td>
</tr>
</tbody>
</table>

to achieve their individual goals.

3. **The shared goal failure rate** recorded in Fig. 5.12 (b), shows that the Collaborative approach has achieved the best result (100% shared goal achievement, 0% failure rate). The Competitive approach has the highest failure rate and the Combination (CCFOM) approach has less than 6.2% failure rate on average.

From the Fig. 5.12 (a) and (b), it can be concluded that EVs have balanced their **level of cooperation and self interest** in the Combination (CCFOM) approach compared to the other two approaches by lowering the individual and shared goal failure rate from 9% to 1%, and from 12.5% to 6.2% on average, respectively.

**5.3.2.3 Communication Cost**

The number of **conflicts in selected collaboration communities** is a good indicator of communication cost when agents want to collaborate. This number shows that agents with conflicts are invited to form a collaboration community but never take any helpful actions (wasted communication cost) or need an extra negotiation step (extra communication cost).

Fig. 5.14 shows the number of conflicts within a formed collaboration community for the Collaborative and the Combination (CCFOM) approach. It also shows the total number of EVs...
and EEVs in the system at each timestep, the number of collaborators in each collaboration community in each approach. The results show that a large number of conflicts exists in the communities formed in the Collaborative approach, as it does not consider EEVs’ and EVs’ goal dependencies. Fig. 5.14 also shows that no conflicts exist in any of the collaboration communities formed using the Combination (CCFOM) approach. This is because EEVs and EVs’ goal dependencies are considered (for example, EEVs are not in any of the collaboration communities).

Even though the Collaborative approach has achieved better results at achieving the shared goal, and the Competitive approach has achieved better results at individual goal achievement, the Combination (CCFOM) approach has achieved better results at balancing the shared and individual goal achievement. Overall, CCFOM has failed to achieve the goals the least number of times, 7.7% of overall goals on average, while the Collaborative approach has failed 10.3%, and the Competitive approach 13.3%.

5.3.2.4 Summary of results from Smart Grid case study

This section summarizes the results obtained from the Smart Grid case study in which two types of electrical vehicles (EVs and EEVs) were operating in a single neighbourhood setting (as it
Fig. 5.14: Conflicts in Formed Collaboration Communities
is not practical to consider a multiple neighbourhood setting at the end-user level in the Smart Grid case study). The results showed that EVs have identified their goal dependencies on each other and on EEVs, and managed to adjust their self-interest and cooperation level better than the two baseline approaches (addressing RQ1: Identify goal dependencies). As shown in Table 5.6, in the Combination (CCFOM) approach, all EEVs have achieved their individual goals and the number of EVs that have not achieved their individual goals is less than the Collaborative approach. The results also showed that the Combination (CCFOM) approach has achieved a better balance of shared and individual goals (Fig. 5.3.2.2) (RQ2: Self-interest adaptation). Moreover, by reporting the number of conflicts in the formed collaboration communities using the Combination (CCFOM) approach, and the two baselines, it is shown that the number of conflicts in the Combination (CCFOM) approach is lower than the other two approaches.

5.4 Ride Sharing

The Ride Sharing case study simulates a system of passengers that share their ride from specific starting points (taxi ranks) to reach their destinations. The Ride Sharing case study includes two different types of agents (i.e., passengers and taxis). Each of these agents are considered to be a resource for the other agent type. The passengers use the available taxis in a taxi rank as a resource to achieve their goals (e.g., to arrive at their destination). Taxis serve passengers (constrained resources) to achieve their goals (e.g., do a certain number of trips per working day).

Taxis can operate in multiple taxi ranks (i.e., neighbourhoods), and passengers can only operate in a single neighbourhood.

5.4.1 Experiment Design

The data that is used in the experiment design is acquired from the Annual Taxi Statistics for Ireland [1], some of the data is summarized in Table 5.7. The Ride Sharing experiment includes 5 taxi ranks, 288 passengers and 12 taxis in each taxi rank to serve the passengers (Note, these numbers are calculated using the data from Annual Taxi Statistics for Ireland [1], the calculation is explained in Section 5.4.1.1). Passengers entering a taxi rank: (a) may join a group and share
a ride right away, (b) if there is no grouping possible, they may wait for a limited amount of time for a possible option, or (c) they may prefer not to share a ride and hire a single occupant taxi. Taxis operate for a number of hours a day, and have a number of goals to achieve. Taxis can serve passengers in other taxi ranks if some of the taxis in other taxi ranks consider collaboration. The experiment is run for a duration of 10 standard days and 4.8 hours in each day. A standard day can be any day from Monday to Saturday excluding holidays and the working hours are from 8 a.m. to 8 p.m.. Each day’s result is independent from the other days, as the passengers and taxis have daily goals. To evaluate the performance of CCFOM in the Ride Sharing case study, 3 different scenarios are implemented. The Ride Sharing case study is particularly interesting as it includes two subsystems that can not be evaluated separately, as each one provides the shared resource for the other to run. Fig. 5.15 shows the experiment design for this case study.

• Scenario 1: Passengers use the Collaborative approach, and taxis use the Collaborative, Competitive and Combination (CCFOM) approaches.

• Scenario 2: Passengers use the Combination (CCFOM) approach, and taxis use the Collaborative, Competitive and Combination (CCFOM) approaches.

• Scenario 3: Passengers use the Competitive approach, and taxis use the Collaborative, Competitive and Combination (CCFOM) approaches.

5.4.1.1 Parameter Setting

Based on the data obtained from Dublin’s Taximeter Survey [1], the average distance travelled in a hiring cycle is 14.9 km, and the average speed of a Dublin taxi was 27 km/hr. Therefore, each taxi spends around 33 minutes on average, for each trip. This data is used to calculate the number of taxis in a taxi rank and the gap between each taxi dispatch.

1. General settings include individual and shared goal settings for passengers and taxis:

   • **Passengers’ Shared Goal**: Passengers may have different reasons to choose a ride-share option when hiring a taxi. They may choose this option to decrease their travel
Table 5.7: Data Used for Parameter Settings in the Ride Sharing Case Study

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Measured Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of days worked per week</td>
<td>5.3</td>
</tr>
<tr>
<td>Hours in service during a working day</td>
<td>4.8 hrs</td>
</tr>
<tr>
<td>Proportion of time hired while in service</td>
<td>42%</td>
</tr>
<tr>
<td>Fare per trip</td>
<td>14.97 €</td>
</tr>
<tr>
<td>Metered revenue per day for one taxi</td>
<td>117 €</td>
</tr>
</tbody>
</table>
cost or to be environment friendly (by decreasing the vacant capacity of each ride which consequently results in decreasing carbon dioxide ($CO_2$) emission).

- **Passengers’ Individual Goal:** Passengers’ individual goal is to reach their destination within a certain time. They set the $\alpha_{g_k}$ and $\Lambda_{g_k}$ values to define their *SetPoint* and *Target*.

\[
\alpha_{g_k} = 0 \\
t'_{k} = t_{\text{arrive}} \\
\Lambda_{g_k} = 1 \\
t_{\text{reach}} < t''_{k} < t_{\text{reach}} + \lambda \\
20\% \cdot t_{\text{reach}} < \lambda < 50\% \cdot t_{\text{reach}}
\]

- **Taxis’ Shared Goal:** Using the data reported in Table 5.7, the average number of trips per taxi is 7.8 per standard day. Taxis are hired for only 42% of their time in service, which is around 2 hours per day, out of 4.8 hours. The remaining 58% (2.8 hours) is spent to return back to the taxi rank to do the next trip. To exemplify this, as shown in Fig. 5.16, the yellow taxi that has dropped its passenger/s to the destination, has to travel back to a taxi rank. One option is to return back to its original taxi rank, or go to a nearer one to decrease the amount of in-service-not-hired travelled distance. The drivers’ shared goal is to decrease this in-service-not-hired time, by collaborating between the taxi ranks (operating in multiple neighbourhoods) and consequently decrease the overall $CO_2$ emission rate.

- **Taxis’ Individual Goal:** Taxis’ individual goal is to have 5 to 8 trips per working day. They set $\alpha_{g_k}$ to be a number between 3 and 5 and a number between 7 and 10 for $\Lambda_{g_k}$ to define their *SetPoint* and *Target*.

\[
2 < \alpha_{g_k} < 4 \\
1.5 < t'_{k} < 2.5
\]
Fig. 5.16: Ride Sharing- Hired and in Service Example

\[ 7 < A_{gk} < 10 \]
\[ 2.5 < t''_k < 4.8 \]

2. Initially, each taxi rank has 12 taxis and there is a minimum of 2.5 minutes gap between each taxi dispatch. The passengers who arrive at the taxi rank are aware of this routine. This gap ensures that the taxi rank has always one or more taxis ready to serve passengers. This gap can be more than 2.5 minutes if there is not enough passengers, or they prefer to wait for new passengers to arrive to have more possibilities to share a ride.

3. Passengers’ population in each taxi rank is calculated based on the survey [1]. A random number of 1 to 5 passengers will arrive at each taxi rank every 2.5 minutes. Equation 5.1 calculates \( \text{Passengers}_{\text{population}} \), which is the number of passengers arriving at each taxi rank, where \( \text{Taxi}_{\text{service}} \) is set to 8, which is the average number of services per day. \( \text{Taxi}_{\text{capacity}} \) is the average taxi capacity which is set to 3, and \( \text{Taxi}_{\text{population}} \) is the taxis’ population in each taxi rank which is set to 12. (Note, each passenger travels alone and has its own goal and destination, for example couples, families, and groups of friends are
excluded from the experiments).

\[ \text{Passengers}_{\text{population}} = \text{Taxi}_{\text{service}} \times \text{Taxi}_{\text{capacity}} \times \text{Taxi}_{\text{population}} \]  
\[ \text{Passengers}_{\text{population}} = 8 \times 3 \times 12 = 288 \]

4. Passengers’ and Taxis’ Dependencies: When they use the Collaborative approach, they have FD, CncD, and OvD. In the Competitive approach, they have CnfD and CmpD, and in the Combination approach, where CCFOM is used, they have FD, CnfD, OvD, CmpD and CncD dependency relations.

5.4.1.2 Baseline Method

Similar to the previous evaluation case studies, Ride Sharing is also evaluated in the Collaborative, Competitive and Combination approaches, both for passengers and taxis. The Collaborative and Competitive approaches are considered as baselines and the Combination approach (CCFOM) is compared against these two.

**Passengers’ baseline:** In the Collaborative approach, passengers queueing in a taxi rank, interact with each other to find possibilities for sharing their rides to be environment friendly and minimize the vacant capacity of each single trip. In the Competitive approach passengers’ individual goal has a higher priority than the shared goal, therefore they prefer not to share a ride to reach their destination on time if there is not any options for a ride-share.

**Taxis’ Baseline:** In the Collaborative approach, taxis can operate in all of the taxi ranks. Taxis calculate their priorities and the taxis with higher priority will do the next service. In the Competitive approach, all the taxis need to return to their initial taxi rank and they are not allowed to operate in multiple taxi ranks, as they are not collaborative. Taxis do not consider the other taxis priorities and they do the services in a first come first serve order, meaning that the taxi that is the first in the taxi rank queue will do the next service.
5.4.1.3 Performance Criteria

The performance criteria for the Ride Sharing case study, which map the obtained results to the initial research questions (summarized in Table 5.1) are as follows:

- Agents’ proportion of access to the constrained shared resource, which is addressed in Section 5.4.2.1, comparing the results from the Competitive, where all taxis only operate in their initial taxi rank, to the Collaborative and Combination (CCFOM), where all or some of the taxis operate in multiple taxi ranks.

- Agents’ individual goal achievement, which is addressed for both taxis’ and passengers’ individual goals by calculating the number of hours taxis were hired per service (Section 5.4.2.1), taxis’ number of services per day, and the number of passengers that reached their destinations on time (Section 5.4.2.2), respectively.

- Collaboration success (the shared goal achievement), which is reported for both taxis and passengers, by reporting taxis’ trip length in Section 5.4.2.1, and the vacant capacity of each ride in Section 5.4.2.2.

5.4.2 Results: Ride Sharing case study

5.4.2.1 Taxis’ Goal Achievement

- Taxis’ shared goal achievement (Taxis’ time hired per trip): Fig. 5.18 shows the amount of time that taxis were hired per trip while they were in service. The results obtained in Scenario 1, where taxis use the Collaborative approach, show a higher percentage of hired hours for taxis compared to the other two scenarios. In Scenario 1, taxis behave collaboratively, and instead of going back to their original taxi ranks from which they were dispatched, they collaborate with other taxis in closer taxi ranks and go there for their next service. Therefore, the amount of time taxis needed to travel back to their taxi ranks is decreased and consequently taxis’ hired hours is increased. In Scenario 2, where taxis have used the Combination (CCFOM) approach, and have considered their
goal dependencies, the hired hours is increased compared to Scenario 3, but not as high as Scenario 1.

Another trend that is interesting to note is in each of the three groups depicted in Fig. 5.18. In each group taxis’ chosen approach is fixed and passengers use different approaches. Group 1 shows the results when taxis use the Collaborative approach and passengers use the Collaborative, Combination (CCFOM) and Competitive approaches. Group 2 shows the results when taxis use the Combination (CCFOM) approach and passengers use the Collaborative, Combination (CCFOM) and Competitive approaches. Group 3 reports the results when taxis use the Competitive approach and passengers use the Collaborative, Combination (CCFOM) and Competitive approaches.

In each group, the highest hired hours per trip is achieved when passengers use the Collaborative approach. This is because they tend to use the vacant capacity of a taxi as much as possible, which results in longer journeys and more hired hours. Moreover, the second highest hired hours per trip is achieved when the passengers use the Combinations (CCFOM) approach which allows them to consider their goal dependencies. Finally, the lowest hired hours is achieved when passengers use the Competitive approach, where they mostly prefer not to share rides.

From the results shown in Fig. 5.18, it can be also concluded that taxis’ hired hours is increased in all settings, except when both passengers and taxis are competitive (i.e., the average hired hours was 48% of a working day [1]). It is assumed that this increase in hired hours, results in an increase in taxis’ income. According to Fig. 5.18, the highest income can be achieved when both passengers and taxis use the Collaborative approach, which results in longer journeys with more passengers in a car and shorter not-hired-in-service journeys back to a taxi rank.

- **Taxis’ shared goal achievement (Trip length):** Fig. 5.19 shows the trip length in each of the three scenarios. A trip includes both service trip (from a rank to the final destination) and the trip back to a rank. The trip length in all three scenarios follows the same pattern. In each scenario, the shortest trip length is achieved when taxis use the Collaborative
Fig. 5.17: Number of Trips per Taxi while in Service, in Different Scenarios.

The diagram illustrates the number of trips per taxi in different service scenarios. The x-axis represents the number of trips, ranging from 5 to 10. The y-axis shows the frequency of trips. The different colored boxes represent the number of trips for each group and scenario.
approach, a longer trip length is achieved when they use the Combination approach, and the longest trip length is achieved when the taxis use the Competitive approach. Moreover, comparing the results within each group specified on Fig. 5.19, the shortest trip has taken place when passengers use the Competitive approach and the longest trip length is achieved when passengers use the Collaborative approach. In general, the shortest trip length (the best results achieved for the shared goal) is achieved when passengers use the Competitive approach and taxis use the Collaborative approach, and the longest trip length has taken place when passengers use the Collaborative approach and taxis use the Competitive approach, this is because each taxi has served more than one passenger (which means a longer journey) and had to travel back to its original taxi rank.

- **Taxis’ individual goal achievement (Taxis’ number of trips per day):** Fig. 5.17 shows the number of trips per taxi within a working day (over a 10 day run). The results obtained show that taxis in scenario 3, where passengers are competitive have achieved a higher number of trips per day (a better individual goal achievement) compared to results obtained in other scenarios. In this scenario, taxis travel a shorter distance compared to the other two scenarios and are ready for their next trip in a shorter time (as passengers do not share their rides). In each scenario, the variation of number of trips obtained in each approach (Collaborative, Combination, and Competitive) is different. The results obtained from the Collaborative approach have less variation compared to the other two approaches, which means agents (taxis) have achieved a fairer access to the shared resource (the passengers). This is because in the collaborative approach, taxis’ priorities are considered. In the Combination (CCFOM) approach, the results have less variation, as taxis consider their goal dependencies, which results in a fairer distribution of the trips between taxis compared to the Competitive approach, where the taxis do not consider their goal dependencies and do the trips on a first in first out order (the taxi that returns to the taxi rank sooner, will do the next service). Moreover, in Fig. 5.17, within each group shown by different colors, taxis have achieved a higher number of trips per day, when passengers have used the Competitive approach. The highest number of trips (8.5 trips on average) is achieved in Group 3, when both passengers and taxis use the Competitive approach.
Fig. 5.18: Taxis’ Time Hired while in Service (%) in Different Scenarios
• The impact of operating in multiple taxi ranks on taxis’ goal achievement: Fig. 5.18 and Fig. 5.19 show that when taxis use the Collaborative or Combination (CCFOM) approaches, they achieve better results for both individual and shared goals. This is because taxis are not limited to their initial taxi rank (i.e., neighbourhood) and have collaborated with other taxis in different taxi ranks to achieve a better performance.

5.4.2.2 Passengers’ Goal Achievement

• Passengers’ shared goal achievement (Decreasing the vacant capacity of each ride): Fig. 5.20 (b) shows the vacant capacity of each ride over 10 working days. According to these results, when passengers use the Collaborative approach, they achieve the lowest vacant capacity compared to the Combination (CCFOM) and Competitive approaches. When passengers use the Combination (CCFOM) approach the vacant capacity at 50% of times is between 0 and 2 empty seats, and at 50% of times it is between 2 and 4 empty seats. However, in the Competitive approach, the vacant capacity is between 1 and 3 empty seats at 50% of times, and between 3 and 4 empty seats at other times.

• Passengers’ individual goal (Failed to reach their destinations): Fig. 5.20 (a) shows the percentage of passengers that failed to reach their destinations at the desired time in rides over 10 working days. The results show that the highest failure rate has occurred in the Collaborative approach. This is because the passengers’ shared goal (i.e., decreasing the vacant capacity of each ride) had a higher priority to passengers’ individual goal (i.e., reaching their destinations on time). The lowest failure rate is achieved when passengers use the Competitive approach, as they do not wait for others to arrive to share a ride, and also their trips are not long as they are the only passenger in the taxi.

As shown in Fig. 5.20 (a) and (b), using the Collaborative approach, passengers have achieved their shared goal better than the other two approaches, however, they have not achieved their individual goals as good as the other two approaches. Using the Competitive approach, passengers have achieved 100% of their individual goals, while they have
Fig. 5.19: Taxis’ Trip Length in Different Scenarios

[Box plot diagram showing trip length comparisons across different scenarios and groups.]
failed to achieve their shared goal. Finally, when the Combination approach is used, where passengers’ goal dependencies are considered, a balance of shared and individual goals is achieved (the vacant capacity is decreased compared to the Competitive approach and less number of passengers had to adapt their desired arrival time, compared to the Collaborative approach).

![Passengers’ Individual and Shared Goal Achievement](image)

**Fig. 5.20**: Passengers’ Individual and Shared Goal Achievement

### 5.4.2.3 Summary of the results from the Ride Sharing case study

In summary, the Ride Sharing case study, presents the results obtained from the operation of two types of agents (i.e., passengers and taxis), where passengers operate in a single neighbourhood setting and taxis operate in a multi-neighbourhood setting. The Ride Sharing case study shows that when the taxies have used the Collaborative approach, they have increased their time hired per trip and also decreased their trip length. However, they were not as successful as the Combination (CCFOM), and Competitive approaches at achieving their individual goals (the number
of trips). When comparing the results in Fig. 5.17, Fig. 5.18 and Fig. 5.19, it can be concluded that the Combination (CCFOM) approach (Group 2 coloured in green in all figures) has achieved a better balance of individual and shared goals. On the other hand, as shown in Fig. 5.20, passengers have achieved the best results for the shared goal (decreasing vacant capacity of each ride) when they have used the Collaborative approach, however, they have not achieved good results for their individual goals, as for 50% of times more than 25% of passengers could not reach their destinations. Analysing the results obtained from both passengers and taxis in Fig. 5.20, Fig. 5.18, Fig. 5.17 and Fig. 5.19, it can be concluded that the better balance of both taxis’ and passengers’ shared and individual goals have achieved when they have used the Combination (CCFOM) approach. This is because in the Combination (CCFOM) approach agents’ goal dependencies is considered during collaboration community formation, and agents have balanced their level of self-interest and cooperation to achieve multiple goals simultaneously (addressing RQ1: Identify goal dependencies and RQ2: Self-interest adaptation).

5.5 Summary

This chapter evaluated CCFOM according to the research questions and the relating design requirements discussed in Chapter 1 and Chapter 3. The evaluation metrics as summarized in Table 5.1 are as follows: (1) Agents’ proportion of access to constrained resources when operating in multiple overlapping neighbourhoods, (2) Agents’ individual goal achievement, (3) Agents’ shared goal achievement, and (4) CCFOM’s communication and computation cost. The three evaluation case studies presented in this chapter include, the application-independent (general) case study, which has evaluated CCFOM in a multiple neighbourhood setting in four application-independent experiments under varying levels of agents’ mobility and neighbourhoods’ density. CCFOM is also evaluated in two real world application case studies, Smart Grid and Ride Sharing.

Agents which have used the Combination (CCFOM) approach, have balanced their shared and individual goal achievement. Comparing the results in the Collaborative, Combination (CCFOM), and Competitive approaches in all case studies, it is shown that the Combination (CC-
FOM) approach has obtained better results at achieving multiple goals simultaneously, a lower percentage of shared goal failure compared to the Competitive approach, and a lower percentage of individual goal failure compared to the Collaborative approach. For example, in the general case study, in Experiment 3, where agents’ mobility is fast and neighbourhoods are dense, agents in the Combination (CCFOM) approach have used 43.2 units of the shared resource on average and overloaded the shared resource 2.4% of the timesteps. In the Collaborative approach, agents have used 40.8 units of the shared resource and overloaded the shared resource 1.8% of the timesteps. However, in the Competitive approach, agents have used 58.4% of the shared resource, and overloaded the shared resource 64.2% of the timesteps. In the Smart Grid case study, EVs have balanced their individual and shared goal achievement by decreasing the transformer overload rate (shared goal failure) from 12.5% to 6.2% compared to the Competitive approach and the percentage of EVs, which have run out of battery (individual goal failure) from 9% to 1%, when compared to the Collaborative approach. In the Ride Sharing case study, passengers, which have used the Combination (CCFOM) approach, have balanced their shared and individual goal achievement by decreasing the percentage of passengers not reaching their destination on time (individual goal) from 26% to 9% compared to the Collaborative approach and have decreased the range of the number of vacant capacity of each ride (shared goal), which is between 2 to 3 empty seats in 50% of times to 1 to 3 empty seats in 50% of times (see Fig. 5.20).

Lower communication and computation cost is also achieved in CCFOM when compared to state of the art approaches, as CCFOM does not employ a coordinator agent, and does not require agents to make commitments, or negotiate their course of actions. Although the distributed Decision-Making process in CCFOM, allows agents to leave and join the system at any time, it does not guarantee a 100% collaboration success rate (as shown in Fig. 5.8), as some of the agents may decide not to collaborate and there is no mechanism considered to convince them or force them to collaborate.

Agents operating in multiple neighbourhoods have achieved a better access to the shared resource. They have also contributed to the shared goal achievement in their initial neighbourhoods by decreasing the demand on the shared constrained resources. This is shown in the general case study for agents operating in more than one neighbourhoods, where they have achieved
13.8% higher resource usage on average, compared to agents operating in single neighbourhoods. Moreover, in the Ride Sharing case study, taxis in the Collaborative and Combination (CCFOM) approaches, which were allowed to operate in multiple taxi ranks, achieved higher percentages of time hired compared to the Competitive approach, where taxis could operate only in their initial taxi rank. For example, in Fig. 5.18, Scenario 2, where passengers use the Combination (CCFOM) approach, taxis’ average hired time per trip was 57.08% in the Collaborative approach, 49.7% in the Combination (CCFOM) approach, and 46.4% in the Competitive approach.

In summary, in the general and Ride Sharing case studies, agents/taxis operating in multiple overlapping neighbourhoods, achieved better results at their individual goals when compared to agents operating in single neighbourhood setting, and also decreased the resource demand in their initial neighbourhoods, allowing agents with single neighbourhood membership to use the constrained resource.

In summary, agents’ proportion of access to shared constrained resources is addressed in both single neighbourhood and multiple neighbourhood settings in the general and Ride Sharing case studies. The results obtained from these case studies have shown that agents have performed better at achieving individual goals, and helped their neighbourhoods to achieve the shared goal by decreasing the demand on the shared resource, when operating in multiple overlapping neighbourhoods (higher resource usage, Fig. 5.3, and increased time hired, Fig. 5.18 and decreased trip length Fig. 5.19). These results address RQ3: Operate in overlapping neighbourhoods.

The results relating to the shared goal and individual goal achievement in all case studies have shown that a better balance of individual and shared goal is achieved when agents have used the Combination (CCFOM) approach, comparing the results in Fig. 5.5 and Fig. 5.7, in the general case study, comparing the results in Table 5.6 in the Smart Grid case study, and comparing the results in Fig. 5.18, Fig. 5.17 and Fig. 5.20, in the Ride Sharing case study. It is interesting that in the Ride Sharing Case study, this balance is achieved when both taxis and passengers use the Combination (CCFOM) approach. Moreover, the results have also shown that agents which have used CCFOM have achieved a fairer access to the shared resource compared to the Competitive approach in the general and Smart Grid case studies, less deviation in agents’
resource usage, Fig. 5.3 and smaller STDEV Table 5.6, respectively. These results address **RQ1**: Identify goal dependencies and **RQ2**: Self-interest adaptation.

Analysing the communication cost in the general and Smart Grid case studies has shown that CCFOM decreases the amount of communication compared to the baseline approaches, as there is no central coordinator and agents do not need to negotiate their actions. Additionally, the computation complexity which has compared the big O of CCFOM to the state of the art approaches, has shown that CCFOM’s decentralised implementation, and not employing a central coordinator decreases the computation cost compared to the baseline approaches.
Chapter 6

Conclusion

This thesis presents CCFOM, a Collaboration Community Formation Model for agents with multiple goals operating in open systems. CCFOM allows agents to operate in multiple overlapping neighbourhoods, and simultaneously achieve multiple goals when sharing constrained resources. This chapter summarizes the thesis contributions and explores the potential avenues for future work.

6.1 Overview of Thesis Achievements

The main aim of this research was to enable agents operating in open systems to form effective collaboration communities to achieve multiple goals (both shared and individual goals) simultaneously, while sharing constrained resources.

Introduction Chapter 1 provided the motivation for this work, which arose from new challenges relating to collaboration community formation in open systems with constrained resources, particularly when agents have multiple goals. The chapter argued that multi-agent community formation algorithms, particularly utility-based and complementary-based ones, can be used to form a collaboration community when agents want to achieve a single goal (individual goal, shared goal, or utilising a constrained resource). However, these approaches are not sufficient when agents want to achieve multiple goals simultaneously while sharing constrained resources. The chapter analysed current solutions and claimed that the following
assumptions make them insufficient: (1) their system modelling (i.e., single neighbourhood, multiple disjoint neighbourhood, multiple overlapping neighbourhoods with transitive dependencies), either causes enormous communication and computation cost, or limits agents to operate in the specified neighbourhood or with the specified agents, (2) the collaboration community members’ selection criteria is limited to agents’ skills, their individual goals (e.g., payoff), or a shared goal, (3) agents are considered to be either cooperative or self-interested and cannot adapt their level of self-interest and cooperation, and (4) the presence of a coordinator agent during the decision-making process, which is not feasible in open systems. Based on this analysis, a hypothesis was proposed, postulating that a new Collaboration Community Formation Model can enable agents operating in multiple neighbourhoods in open systems with constrained resources, to achieve multiple goals simultaneously by modelling their goal dependencies, adapting their level of self-interest and cooperation, and making decisions in a decentralised manner.

**Related Work** Chapter 2 presented the state of the art in collaboration community formation and multi-agent resource allocation approaches in open systems. This chapter outlined two main classes of approaches, which form collaboration communities for different purposes: Utility-based approaches, which use utility functions to increase agents’ individual payoffs (i.e., achieving individual goals), and Complementary-based approaches, which use social reasoning to compose agents’ skills and determine their action sequence to accomplish complex tasks. Of these, social reasoning was determined to be a useful technique to enable agents, which leave and join the system frequently, to identify their neighbours and their goal dependencies, to be used during a collaboration community formation process.

**Design** The new Collaboration Community Formation Model proposed in this thesis, CCFOM, is presented in Chapter 3. CCFOM enables agents to: operate in multiple overlapping systems; identify their neighbours and their goal dependencies; adapt their level of self-interest and cooperation; and finally form collaboration communities to achieve multiple goals simultaneously, while sharing constrained resources in a decentralised manner. CCFOM has six stages:
Chapter 6: Conclusion

In the Neighbourhood Membership Update stage, neighbourhoods are formed around a single resource, and agents’ joining and leaving are managed. A neighbourhood contains a resource which is shared with all the neighbours in the same neighbourhood, and it keeps the record of its members and the capacity of the shared resource at each timestep. In the Collaboration Need Identification stage, agents are enabled to identify if there is a need for collaboration according to their decided actions, and the resource capacity, and inform the relevant agents to form a collaboration community. In the Neighbouring Agents’ Dependency Model stage, each agent shares an external description with all the other agents in their neighbourhood. The shared information is then used during the social reasoning process, allowing agents to identify and build their goal dependency model. The Neighbouring Agents’ Dependency Model is then used during the Self-Evaluation and Adaptation stage. The Self-Evaluation and Adaptation stage allows agents to calculate their own priority and adapt their level of self-interest and cooperation, considering their goal dependencies and priorities. In the Cascade Collaboration stage, agents which are members of more than one neighbourhood evaluate the alternative neighbourhoods to cascade their resource request, when the resource in their initial neighbourhood is likely to be overloaded. Finally, in the Decision-Making stage, agents decide whether or not to collaborate and finalise their actions in a decentralised manner.

Implementation Chapter 4 provided CCFOM implementation details, an application-independent simulator, and two application simulators, into which CCFOM is integrated. The application-independent simulator (general simulator), simulates a system containing multiple overlapping neighbourhoods, in which agents can join and leave the system freely. This general simulator is later used as an Interpreter to connect CCFOM to the Smart Grid and the Ride Sharing simulators.

Evaluation Chapter 5 evaluated CCFOM using the general, Smart Grid and Ride Sharing case studies. In all these case studies, CCFOM is used in an approach called Combination (CCFOM), in which a combination of goal dependencies exist between agents and it is compared against the Collaborative and Competitive baseline approaches. In the Collaborative
approach, agents place a higher priority on the shared goal, while in the Competitive approach only agents’ individual goals are considered. The general case study is a multiple neighbourhood application-independent case study, evaluating CCFOM under varying levels of agents’ mobility and neighbourhoods’ density. The Smart Grid case study evaluated the performance of CCFOM in a single neighbourhood system with two types of electrical vehicles. The Ride Sharing case study includes a multiple neighbourhood system for taxis and a single neighbourhood (multiple disjoint neighbourhoods) for passengers.

6.1.1 Thesis Contributions

In summary, the main contribution of the thesis, CCFOM, is a Collaboration Community Formation Model, which enables agents to form effective collaboration communities, to simultaneously achieve multiple goals, while sharing constrained resources. CCFOM includes the following contributions:

- Neighbouring Agents’ Dependency Model, which enables agents to capture their goal dependencies, using social reasoning. Using the shared information in agents’ external description, agents can reason about their dependencies and use this dependency model later when they want to form collaboration communities.

- A Self-Evaluation and Adaptation process, which enables agents to adapt their level of self-interest and cooperation considering their priorities for accessing the shared resource, their options for cascading their resource request to an alternative neighbourhood and their dependencies on their neighbours. This process also has a decentralised decision-making process, which enables agents to make decisions in a decentralised manner without requiring a coordinator agent or a negotiation process.

- Support for multiple overlapping neighbourhoods operation, which enables agents to be a member of more than one neighbourhood. Such agents have the option to change their actions from one neighbourhood to another when the resource in the initial neighbourhood is likely to be overloaded or when they have a better chance to access a resource in another neighbourhood.
6.2 Discussion

Using CCFOM, agents can operate in multiple overlapping neighbourhoods, understand their goal dependencies, and form qualified communities including agents which do not have conflicting goals to achieve multiple goals simultaneously. They can better balance and adapt their level of self-interest and cooperation. They can prioritize their goals and their policies and decide whether or not to fulfil them according to state of their neighbourhood.

CCFOM is particularly designed for agents with multiple goals operating in open systems with constrained resources. Such agents join and leave systems frequently and collaborate to achieve their shared and individual goals simultaneously. In environments where agents do not need to achieve multiple goals simultaneously, their goals are not dependent, their mobility is slow (the time they spend in a neighbourhood is not limited), or the resource is not constrained, using CCFOM neither improves nor hinders the performance.

Allowing agents to operate in multiple overlapping neighbourhoods without limiting them to transitive dependencies, enabled them to seek alternative solutions in other neighbourhoods when the shared resource in the initial neighbourhood was overloaded. Doing so, they could increase their individual resource usage and increase the resource accessibility of their neighbours with single neighbourhood membership.

The multiple overlapping neighbourhoods modelled in CCFOM and the relating Cascade Collaboration process have helped agents to operate in multiple neighbourhoods setting, enabling them to cascade their resource request to an alternative neighbourhood, when the resources in both neighbourhoods can be used interchangeably. In CCFOM, agents with multiple neighbourhood membership would not gain or lose any benefits when the resources in such neighbourhoods are not interchangeable.

CCFOM can have ideal results when agents’ mobility and the neighbourhoods’ density is medium. By increasing their mobility speed to fast and the neighbourhoods density to dense, agents’ performance will decline. Therefore, this approach is helpful in applications where agents’ mobility and the density of the neighbourhood is not fast and dense. So that agents have some time and also resource to be more adaptive and balance their level of self-interest and
cooperation. One the other hand, using CCFOM is applications that agents mobility is slow and the density if the neighbourhoods are sparse would not be very effective, as agents have enough time to use the available shared resources or the shared resources are not scarce.

In this work, it was assumed that agents are not malicious, and their shared information can be trusted, however if this is not the case, then agents need a trust mechanism to identify the trusted neighbours first and then use CCFOM. In this case agents who are not trusted will still share the same resource in the same neighbourhood and would not be entered in the collaboration process. As a result there would be lower capacity to share and agents’ would become more competitive. It is also assumed that the shared resources are quantifiable, which means that multiple users can use it at a same time, if this is not the case and resources can be used only by one agents at each timestep then the CCFOM is not a solution, as there is no need to form any communities or identify goal dependencies.

6.3 Limitations and Future Work

CCFOM, like all the other social reasoning approaches, has assumed that agents’ external description is understandable for all the other agents operating in their neighbourhood. However, in open systems, agents are heterogeneous, which may affect their interpretation of the environment. In the presence of such heterogeneity, expecting all agents to represent their data in the same form (external description) is not practical. To address this issue, an ontology-based layer could be added to help agents to map the acquired information from each others’ external description to an understandable form of information for themselves. This would not affect the performance of agents’ goal achievement process, since it happens when agents want to share their external descriptions, but may affect general load on the agents. This needs to be evaluated.

CCFOM was implemented to consider only one individual goal and one shared goal for each agent. However, there are systems such as electronic market place, where buyers may have multiple individual goals (multiple goods to buy) and multiple shared goals with different/overlapping sets of buyers. To address these types of systems’ requirements, the Neighbouring Agents’ Dependency Model can be extended to accommodate the dependency types of agents with mul-
multiple individual goals and multiple shared goals. This does not change the conceptual model for CCFOM, and just requires each agent to balance their own individual goals, in addition to other agents’, using the same approach.

CCFOM addresses single community formation at each timestep for each neighbourhood. However, when agents have multiple shared goals in a neighbourhood, multiple neighbourhoods might be needed to be formed simultaneously. To do so, the agents’ decision-making process needs to employ some approximation and heuristic-based methods to enable agents to decide in which collaboration community to operate, based on their goal dependencies and priorities. This may impact the performance of agents’ at achieving shared goals, and needs to be evaluated.

CCFOM is a decentralised algorithm, which does not require agents to make commitments at any stage of their operation in the system. Although this feature does not limit agents’ frequent joining and leaving in open systems, it may cause a higher rate of collaboration failure compared to approaches which require commitments. To address this issue, agents could have a portable reputation profile which records their performance in the previous collaboration communities so that they can adjust their behaviour accordingly. Although this profile can help agents to understand each others’ performance history, it requires more information sharing compared to CCFOM.

Additionally, agents can use some learning techniques to understand the risk and benefits of involving in a collaboration process. This will help them to better balance their individual and shared goal achievement, if they are staying for a long time in a community.
Bibliography


Bibliography


