universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Multi-Agent Negotiation and Optimization in Decentralized Logistics"

verfasst von / submitted by

Patrycja Pultowicz, BSc MA

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 920
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Quantitative Economics, Management and Finance
Betreut von / Supervisor:	o. Univ. Prof. Dipl.-Ing. Dr. Richard Hartl
Mitbetreut von / Co-Supervisor:	Dr. Margaretha Gansterer

LIST OF FIGURES

Introduction

The fast pace of technological development has enabled the creation of software systems in which data is exchanged by global computer networks faster, safer and at a reduced cost. At the same time, globalisation and the shifting structure of modern supply chains have increased the need for intelligent autonomous agents in a multi-issue bargaining process to adopt and expand traditional trading and production operations with the goal of becoming more decentralized and, thus, robust to environmental changes. The objective of such intelligent agent settings is a fully automated system working independently of human interaction or assisting decision-making processes. The idea to involve autonomous agents in day-to-day applications in the supply chain has been a focus of research since two decades. In practice, automated negotiation processes include, for instance, automated storage and retrieval systems, industrial robots (e.g. Amazon Kiva robotic) and Google's self-driving cars. The different objectives, constraints and capabilities of decentralized agents make the negotiation processes, however, a complex problem to be solved, with currently many limitations on their implementation in real-world settings. Thus, most of the modern automation processes still involve human intervention in decision-making and control.

This thesis focuses on decentralized automated negotiation solutions implemented as a multi-agent system (MAS), motivated by the developments of the so-called *Industry 4.0* and the *internet-of-things*. The paper explores research literature and real-life projects in which autonomous agents are implemented largely de-centrally to facilitate dynamic and complex workflows. The aim of the paper is to explore current approaches to automated MAS with a focus on logistics, their functionality and efficiency, as well as underlying technological requirements.

The paper is organized as follows: the first chapter serves as an introduction to MAS in comparison to single-agent and centralized systems to an extend which facilitates the understanding of the chapters thereafter. In the second section, MA-frameworks and autonomous agents are further explored in their functionalities, capabilities and technical nature. The third and fourth chapter introduce different negotiation protocols and optimization techniques used in theoretical MA-frameworks, followed by a literature exploration to automated negotiation systems in logistics and a brief introduction to real-life MA-projects. The last section of the paper provides a computational implementation of a multi-agent flow shop example, giving a solution comparison to a centralized approach.

Einführung

Die rasanten Entwicklungen der technischen Möglichkeiten erweitern traditionelle Produktions- und Logistik-Bereiche immer stärker um den Einsatz komplexer Systeme, in denen Daten schneller, sicherer und kosteneffizienter über Computersysteme fließen. Gleichzeitig zwingen Globalisierung und die sich stetig verändernde Struktur der Beschaffungskette den Einsatz von intelligenten Softwareagenten, die untereinander teilweise oder gänzlich autonom verhandeln und dadurch die traditionellen Arbeitsabläufe erweitern und automatisieren. Ihr Einsatz kann dabei dezentral ohne Menschenzutun gesteuert werden oder zentralistisch als vorletzte Instanz der Entscheidungsprozesse erfolgen. Die Idee, autonome Softwareagenten in den täglichen Prozessen einer Vertriebskette einzusetzen, ist Fokus von zwei Jahrzehnten an Forschung. In der Praxis sind autonome Verhandlungsagenten vor allem im Bereich der automatisierten Lagerung, der Industrieroboter (z.B. Kiva Roboter von Amazon) und bei selbst-gesteuerten Alltagsprojekten (z.B. Google's Autos ohne Fahrer) eingesetzt. Die wohl größten Herausforderungen im Umgang mit automatisierten Verhandlungsagenten sind die unterschiedlichen Zielfunktionen, Beschränkungen und Kapazitäten, die den Einsatz solcher Agenten sehr komplex gestalten. Immer noch werden autonome Agenten aus diesem Grund- ihr Potential nicht ausschöpfend- überwiegend in Entscheidungsprozessen als Ergänzung, aber nicht als Entscheidungsträger mit einbezogen.

Diese Arbeit konzentriert sich auf dezentral-automatisierte Verhandlungsansätze, umgesetzt als Multi-Agenten Systeme (MAS), motiviert von den Entwicklungen der sog. *Industrie 4.0* und der *Internet-der-Dinge*. Die Arbeit analysiert die Literatur und praktische Einsätze von autonomen Agenten, die überwiegend dezentral agieren um komplexe Arbeitsprozesse zu erleichtern. Ziel der Arbeit ist, mit einem Fokus auf logistische Systeme gegenwärtige Methoden zu MAS, deren Funktionalität und Effektivität sowie technische Einschränkungen und Voraussetzungen zu untersuchen.

Die Arbeit ist wie folgt unterteilt: der erste Abschnitt dient als Einführung in die Terminologie von MAS und Single-Agenten Systemen, gefolgt von einer vertiefenden Einsicht in MA-Strukturen und der Natur von autonomen Agenten. In den darauffolgenden Kapiteln sind unterschiedliche Verhandlungsprotokolle und Optimierungstechniken von MAS vorgestellt. Die vorletzten Abschnitte stellen die in der Literatur erwähnten MA-Ansätze in der Logistik vor, gefolgt von einer kurzen Übersicht über praktische Agenten-Erfahrungsprojekte. Im letzten Abschnitt der Arbeit ist ein dezentrales Multi-Agenten Flow Shop Problem vorgestellt und anschließend mit einem zentralen Beispiel verglichen.

1 Global Logistics under the Internet-of-Things

Today's digital world allows for more flexibility in manufacturing, improved productivity, faster quality assessments, as well as a stronger participation and influence of customers. The broadly integrated wireless interaction between people and the internet has created services and communication networks that were unthinkable two decades ago. People get real-time access to events happening thousand kilometres away, they communicate faster and through a broader distance, they can transfer information and money within seconds to locations far away, and they can communicate with pictures and videos. Most importantly, people have obtained global access to any kind of services and information, gaining significantly more purchasing power, a wider choice set, and more complex terms and conditions for building a long-term business relationship and brand loyalty. The developments in computer software and hardware, coupled with globalisation, has additionally made it increasingly attractive for companies to use intelligent non-human software agents, each being equipped with different (possibly conflicting) goals to assist in their operations and decision-making processes.

Such significant influences have also shaped the nature of modern logistics. For example, production firms increasingly outsource the transportation orders under a set of pre-negotiated terms to other companies (intermediary logistic companies) that negotiate the distribution of these orders with other smaller companies that have the actual transportation capacity, forming so-called *multi-party logistics*. The **Internet-of-Things** has additionally increased the complexity and dynamics of modern logistics and SCM (supply chain management) through relocating the demand and supply market into the internet (e.g. Amazon, Ebay). TechTarget defines the Internet-of-Things (IoT) as a *system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network wirelessly*[1]. The data transfer thereby takes place through the use of four distinct area networks: BAN (body area network like a smart glass or piece of clothing with integrated sensors, so-called wearables), LAN (local area network like smart meters at home), WAN (wide area network like connected cars and Telematics) and VWAN (very wide area network like e-services that are accessible from everywhere with no physical locations)[2]. The increasing digitalization and interconnection of products, value chains and processes through the use of sensors, wireless networks, intelligent robots and machines as well as the increased computing power and the development of 'big data' analytics in the industrial sector has led to the so-called *fourth industrial revolution*, also known as **Industry 4.0** (synonymous terms are *Smart factories*, or

[1] IoT Agenda (2016): online
[2] council(2016): online

Advanced manufacturing). Industry 4.0 uses advancements from five main areas: *information and communication technology* (ICT) to digitise and integrate information at all process stages; *cyber-physical systems* that use ICTs to monitor and control processes through the use of sensors, robots or additive manufacturing devices (e.g. 3D printing); *network communication* that links machines, products, systems and people across the supply chain through wireless and internet technologies; *simulation* that models and virtualises processes in their development stage; *big data analysis* and *cloud computing* that evaluates inputs and outputs virtually; and *augmented reality* or other *intelligent tools* for greater design and production support.[3]

Creating intelligent and largely independent interaction processes between different entities is a study area in *Artificial Intelligence* (AI). Following the technological advancements and globalization, **Distributed Artificial Intelligence** (DAI) has evolved as a subfield of AI about three decades ago. DAI deals with learning, planning, communication and decision-making processes occurring in the interaction between multiple independent entities (both human and artificial entities) in a domain. Traditionally, DAI was divided into two sub-disciplines: **Distribution Problem Solving** (DPS) that focuses on the information processing between several branches working together towards a common goal (e.g. task decomposition), and **Multi-Agent System** (MAS) that deals with communication processes between several independent entities, so-called *agents* (e.g. subcontracting the decomposed tasks to different problem-solving agents).[4] Many definitions of agents have been proposed in the fields of MAS, the most common and suitable definition is derived from Wooldridge (1997): *an agent is defined as a computer system that is capable of autonomous [independent] actions in the environment [domain] in which it is situated in order to meet its design objectives*[5].

While DAI divides a particular problem into smaller problems of common knowledge, MAS aims for establishing a *negotiation system* among artificial agents in which they cooperate or compete following their individual knowledge and action set. The literature suggests that MA-systems deal with both the construction of complex systems of multiple independent competing or cooperating entities and the mechanisms for coordinating the individual agents' behaviour and, thus, embedding important aspects of DPS into MAS in the search for optimal negotiation processing between multiple agents. This converged definition of MAS is used in the further context of this paper.

Automated systems in logistics, also referred to as *automated logistics* or *logistics automation*, are strongly centrally coordinated intelligent interactions between independent

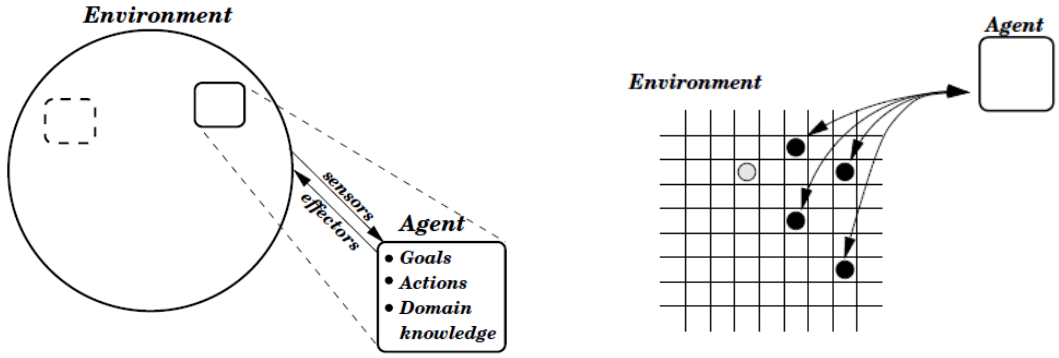
[3] Davies R. (2015): pp.1-3
[4] Stone P. and Veloso M. (2000): pp.1-2
[5] Wooldridge M. (1997): pp.28-29; Kraus S. (2001): p.152

entities, either via a human interface or via a central computer system that makes all decisions and assigns the individual tasks and resources to other entities. **Automated logistics** describes centralized automated workflow solutions in which a predefined wireless negotiation process between several objects is used to improve the efficiency of logistic operations. Logistics automation is used in warehousing and storage through AS/RS, in production planning and quality management through automated conveyors that are adjusted to check for deviations in product type, weight, size or other quality features, as well as in product sorting through barcode or radio-frequency identification tag scanners to check for product type and destination (e.g. mail sorting line). In the last twenty years, the evolution of retail logistics has largely evolved into a so-called *e-commerce logistics* involving internet-only retailers and increasingly automated distribution networks including 24/7 operating fulfilment centres in which the items are stocked and picked at item level, sorting centres in which orders are being sorted automatically by post code, and integrated technology where shopping carts connect via API or other connection to a transportation management system.[6] Such e-commerce logistics allow, for instance, a large transparency in the supply chain, high cost-reduction due to a faster human-independent processing (less human-related errors) and an online synchronous overview of the current delivery status. While automated logistics can be traditionally considered a **single-agent system**, that is a centralized system with the full decision-control over multiple machines and resources, the technological development has made it increasingly possible to employ *multi-agent* platforms in which several autonomous software agents coordinate themselves to perform certain processes.

Figure 1 illustrates a general single-agent framework according to P. Stone and M. Veloso (2000) on two examples. In a single-agent system, the agent models itself (i.e. its own objectives and constraints), the environment in which it operates and their interaction processes (left illustration). The central agent interacts with its environment through sensors (getting information from the environment) and effectors (communicating actions and decisions to the environment). In case other agents are part of this system, they are considered part of the environment but, differently to the central agent, they do not have their own goals, actions and knowledge. In the example of a predator/prey-game (right illustration), the agent controls all predators (black circles) and the (grey) prey is part of the environment and not under the control of the agent. The central agent aims to coordinate the predators in a way that the prey can be caught.[7]

[6] Cerasis (2014): online
[7] Stone P. and Veloso M. (2000): pp.6-7

Figure 1: a general single-agent framework as proposed by Stone P. and Veloso M. (2000) given two illustrative examples

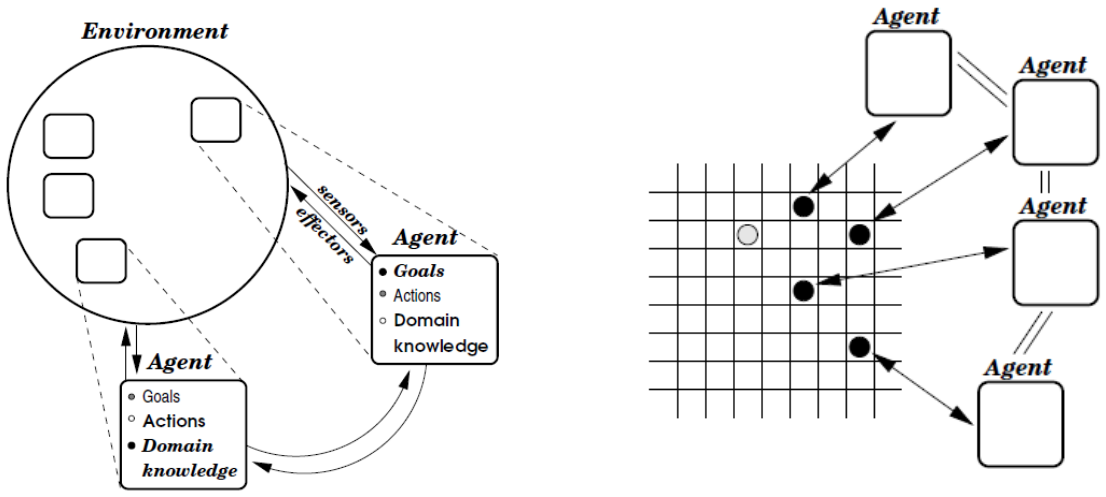


Differently to a single-agent system, a **multi-agent system** consists of several agents, whereby each models its own objectives, constraints, the potential interaction with other agents and their environment (see Figure 2). The environment is established by the individual agents, which themselves can differ in objectives and constraints, and the environment can change frequently depending on the agents' actions and decisions. Thus, multi-agent systems typically have dynamic environments. As Figure 2 illustrates, each agent is both part of the environment and, thus, needs to obey the rules that govern the system, and a separate entity that influences the environment's dynamics. Given the example of a predator/prey game, the predators would each be modelled as an individual agent with own objectives, constraints and an action set. The agents may have limited information about the other agents and they need to coordinate each other to haunt the prey. In a competitive scenario, each agent would try to maximize its own utility in the presence of environmental constraints by haunting the prey alone outside any cooperation approaches.[8]

In a homogeneous environment, agents can have an identical structure (sensors, effectors, domain knowledge, and decision functions) but communicate differently with the environment, because they are situated differently and make their own decisions. This is a necessary condition for MAS and the main difference to a single-agent system: each agent acts on its own behalf and through sensor inputs and effectors that differ from other agents, or else they act identically in case the agents' objectives, constraints, actions and domain knowledge are all identical.[9]

[8] Stone P. and Veloso M. (2000): pp.8-13
[9] Stone P. and Veloso M. (2000): p.14

Figure 2: a general multi-agent framework as proposed by Stone P. and Veloso M. (2000) given two illustrative examples



Kumar S. (2012) describes MAS with the following characteristics[10]:

- there is an arbitrary number of agents in the system which are capable of interacting with each other by exchanging messages through some computer network infrastructure
- the individual agents are acting autonomously, mostly on behalf of users, and can have very different goals and motivations. Therefore, they make independent decisions and aim to maximize the utilities for their owners
- each agent has incomplete information or capabilities for solving the problem. Thus, they must cooperate with other agents to fulfil their own objectives
- there is no system global control
- data is decentralized and accessible by anyone in the system

There are numerous **advantages** in using MAS: multiple agents could work parallel in case a domain is easily broken into several independent components that can be handled by separate agents. The use of MAS can be also more robust against failure than having a single entity- processor or agent who controls everything. Another advantage of MAS is their scalability. Some authors claim that it is easier to add new agents to a MAS than new capabilities to a monolithic system, making MAS attractive for systems with dynamic parameters. Additionally, the division of tasks can facilitate programming and computation time, and it is particularly practical to use MAS simulation for problems in social and life sciences. Another benefit of MAS is that geographically distributed operations can be better served by multiple agents acting locally than with the use of a central agent. Finally, MAS

[10] Kumar S. (2012): p.11; Radu S. (2013): pp.48-49

can have a higher cost efficiency than single-agents, because capabilities bundled together can make a system more expensive to engineer.[11]

To obtain a desirable MAS-technology, Kumar S. (2012) proposed the following criteria to be met[12]:

a) Simultaneous computation and communication must be kept minimal (*computational efficiency*). While the number of agents in a system can be arbitrary, each agent can only communicate with a small subset of other agents in the system. Thereby, communication must not need to be a one-to-one way between agents, but each agent can simultaneously negotiate with multiple other agents (parallel negotiation). A system that involves less or faster communication is preferred in a real-world setting as it is less time consuming.

b) The system must be stable. Stability, on the one hand, refers to the system being able to recover fast from component failures when it is able to dynamically find agents with redundant capabilities (*reliability*). On the other hand, a mechanism is stable if the agents involved do not have an incentive to deviate from their agreement (*stability*).

c) System reliability to some extend already implies the necessity for MAS to be responsive, hence, reacting locally to anomalies (*responsiveness*).

d) The system must be flexible, in terms of both the number and capabilities of agents working on a problem (*extensibility*) and of agents with different abilities being able to adaptively organize to solve the problem (*flexibility*).

e) The system must be robust, both in terms of tolerating uncertainty (*robustness*) and in terms of remaining unaffected by a changing number of agents in the system (*scalability*).

f) The agents involved should be reusable in different agent teams (*reuse*).

[11] Stone P. and Veloso M. (2000): p.4
[12] Kumar S. (2012): p.11 adopted from Rosenschein and Zlotkin (1994)

2 Basic Principles of MAS

Multi-agent systems are developed around the following main set of components:
1. agents: their functionalities and internal state
2. environment: the system and its dynamics in which the agents operate
3. interaction and negotiation setting: the communication style and message semantics the agents use in pursuing individual and joint objectives - this topic is more widely discussed in the next chapter

2.1. The Nature of Agents

An agent is characterized by two important criteria: (i) it is a discrete entity with its own goals (design objectives), constraints (resources) and rules of behaviour, and (ii) it is autonomous and, therefore, acts outside the control of a central authority, whereby it is capable of adapting its behaviour to its environment and the domain rules.

In particular, agents are limited to two main behaviours: (1) they autonomously act on their owner's behalf (pursuing predefined objectives), and (2) they interact with other agents by exchanging data or by coordination, cooperation, negotiation, etc. to achieve the objective under a set of constraints and domain knowledge[13]. Figure 3 represents two agents interacting with each other and their environment. In order to fulfil an individual objective, agents must be able to assess the different states of the environment by processing their interaction and envisioning possible outcomes from potential decisions.[14]

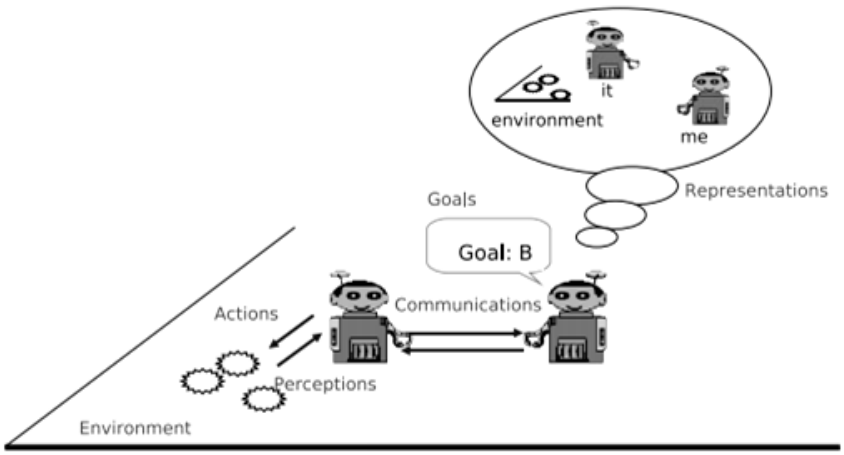
In order to fulfil tasks autonomously, agents must meet following criteria[15]:

- they must be clearly identifiable from other objects in their environment
- they follow one or several individual objective functions (goals), usually defined in advance and not available to their environment (private information)
- they operate under well-defined, mostly private boundaries (constraints) and interfaces, and they possess their own set of resources
- they rely on inputs through sensors and act on the environment through effectors which results in a direct or indirect communication with other agents
- they have control over their internal state and, thus, over their own behaviour
- they are reactive to changes in their environment; usually those changes are being addressed within a feasible set of possible actions
- they act proactively towards fulfilling their goals

[13] Kumar S. (2012): pp.9-10
[14] Uhrmacher A. and Weyns D. (2009): p.15
[15] Uhrmacher A. and Weyns D. (2009): pp.14-15; Jennings N. R. (2000b): p.280

Usually, MAS consists of heterogeneous agent types that are designed to implement some functionality of a process, so-called *functional agents*. Broadly, functional agents can be further divided into two groups: *information agents*, who provide system information to other group agents, and *agents who are related to a certain good or role* of the system. For instance, *seller agents* are functional agents aiming to sell a negotiation object, while *buyer agents* are designed with the goal to buy a particular good, task or information.[16]

Figure 3: a multi-agent world according to Uhrmacher A. and Weyns D. (2009)



The internal organization of an agent is generally referred to as *architecture*. Ferber et al. distinguish between two main approaches for analysing agent architectures, which is (1) the ***stimuli-response approach*** and (2) the ***cognitivist approach*** that relies on the explicit representation of the environment by the agents involved. The approaches can be also combined to a hybrid approach. In the stimuli-response approach, the agent lacks an explicit representation of its environment and the agents therein and, thus, behaves only in terms of predefined operations in reaction to what it perceives. For instance, agents could react along predefined priorities, or they focus on the needs of the negotiation partner in a cooperative behaviour, or they act given a set of tasks based on neural nets. In a cognitive architecture, agents are supposed to be rational operators who reason from knowledge explicitly representing their environment and the agents therein (its object dynamics, states and properties). Therefore, an agent acts upon its beliefs about the world's states and its knowledge to satisfy a goal. The Belief-Desire-Intention architecture of this type, for instance, assumes that its agents possess a library of plans, each aiming to achieve a particular objective. When an agent selects a specific goal to pursue (e.g. given a certain message that the agent decides to act upon), it follows the strictly predefined environmental actions and communications by using a particular plan or set of plans compatible with the agent's beliefs. The beliefs, on the other hand, are formed by external perceptions and messages (incl.

[16] Chen Y. et al. (1999): p.2

learning effects). If no plan can be found, a problem can be decomposed into sub-goals that the agent aims to satisfy.[17]

In the forefront of modelling an agent's architecture, the behaviour of agents can be largely summarized into the three steps: (1) perception of inputs, (2) deliberation of intentions, and (3) action to produce outputs. Expressed mathematically, we can thus model an agent's behaviour by the function $Behaviour_a: \Sigma \mapsto A_a$ representing the set of possible actions that can be taken by agent a based on the following steps, as defined by Uhrmacher A. and Weyns D.:[18]

(1) $Perception_a: \Sigma \mapsto P_a$- the agent obtains a perception $p_a \epsilon P_a$ given an input from a current state $\sigma \epsilon \Sigma$.

(2) $Deliberation_a: \mathrm{P_a} \times S_a \mapsto S_a$- this function calculates a new internal state s_a of the agent a based on the previous perception and the current internal state. The result is used to update the agent's action plan.

(3) $Action_a: \mathrm{P_a} \times S_a \mapsto A_a$- this function computes an agent's output (decision-making), based on its new internal state and current perception from its environment.

2.2. Environment

The environment defines the conditions and infrastructure in which the agents can exist together. In a typical setting, an environment is a system with the following differentiated properties (originally selected from Russell and Norvig, 2003): [19]

- an environment is accessible if agents can perceive all the information from it, else it is inaccessible
- an environment is deterministic if its next state entirely depends on the agents' actions and the current state, else it's nondeterministic
- similarly, an environment is static if its state is influenced by the agents only and not, like in a dynamic setting, endogenously
- an environment is discrete if it has a limited number of possible inputs and outputs (actions). In a dynamic environment, the range of perceptions and actions typically depend on the individual agents and the nature of the reaction and not on the environment (agent-centred)

[17] Ferber J. et al. (2001): pp.13-14; Uhrmacher A. and Weyns D. (2009): pp.14-16
[18] Uhrmacher A. and Weyns D. (2009): pp.16-17
[19] Uhrmacher A. and Weyns D. (2009): pp.18-20

2.3. Interaction and Negotiation Setting

Agent interaction can take place in form of a *cooperation* (multiple agents (e.g. in a team) trying to maximize a global utility), an operation *request* (handing over tasks to an agent with more adequate or higher capacity) or in form of an *information exchange* to follow a self-interest (each agent trying to maximize its local utility)[20]. All those interactions involve **negotiation** - the process by which the involved autonomous agents are being convinced to a mutually acceptable agreement on a task, or they disagree and depart from the negotiation process[21].

There are mainly the following three approaches to automated negotiation, which are discussed in greater detail in the next chapter:[22]

1) **Game-Theoretic approach**: refers to negotiation among self-interested, strategically reasoning agents that aim to maximize their own utilities by taking into account the decisions of the other agents in the system. The basic assumption for agents following a game-theoretic negotiation is that all agents act to optimize their individual welfare.

Subchapters: game-theoretic bargaining, coalitional games, auctions

2) **Heuristic- based negotiation**: is a cooperative negotiation approach by which the negotiation space is searched in a non-exhaustive manner in order to produce good, rather than optimal solutions (proposals), to save computational time and, thus, costs associated with strategic decision-making.

Subchapters: concession-based negotiation, service-oriented negotiation

3) **Argumentation-based negotiation**: is an approach by which additional information are exchanged among the agents involved in the negotiation setting. Such additional information can include, for instance, preferences of the agent who is proposing a counteroffer, or a critique of a proposal following a rejection.

Subchapters: argumentation-based negotiation

2.3.2. Agent Language

Agents interact using a certain agent language that follows a predefined negotiation protocol, i.e. logics and rules of interaction. Each agent has some knowledge about the system

[20] Castelfranchi C. (1998): pp.158ff
[21] Jennings N. R. et al. (2001): p.201
[22] Kumar S. (2012): pp.23-26; Jennings N. R. et al. (2001): pp.11-24

ontology, that is the nature of the system and the interaction primitives employed by the system that is has to adhere to.

The vision of automated trading settings, in which software agents bargain on behalf of an end-user, largely depends on the quality of the agent communication languages and protocols used. Areas of foci are the design of interaction protocols among agents, the dynamic management of agents among themselves, and the [semantic] style and transportation of messages among the agents involved. Several initiatives with the goal of developing such MA protocols have emerged about two decades ago, of which the most widely known is the *Foundation for Intelligent Physical Agents* (FIPA), a committee founded in 1996 and consisting of different companies and colleges worldwide that emphasize in multi-agent technologies. The FIPA Community continuously works on developing a range of agent-related standards and performatives to support MAS-negotiation, of which the most widely used negotiation language are the *Agent Communication Language* (ACL) and the *Knowledge Query and Manipulation Language* (KQML)[23]. Agent languages like KQML, however, only give frameworks for negotiation and have to be adapted to the type of negotiation used.

In most of the MAS literature, *proposal-based* communication primitives have been observed, which is popularly illustrated as a so-called *contract net interaction protocol* (CNP; see Figure 4 as example) introduced by R.G. Smith (1980) and later adopted by FIPA as an alternative FIPA agent communication language (FIPA ACL). Such a negotiation starts with one agent (initiator agent) making the first offer (typically a *call-for-proposal message*; CFP) to another agent or to a group of recipient agents who can accept (i.e. send a *propose* message or accept a proposal), reject (i.e. refuse to participate in the negotiation process or reject a proposal) or make a counter-offer on a previous offer (in case the first offer is a proposal), which the receiving agent can react to in the same manner until a negotiation is closed (e.g. agreement or deadline is reached). The *propose* message send by the participating agents (e.g. bid) is, thereby, computed using the agent's local algorithm which is private and, thus, unknown to the other agents.[24]

There have been numerous extensions to the basic CNP that take into consideration collaborative and competitive environments with multiple parallel negotiations[25] or more flexible contracting in competitive environments[26].

[23] Wooldridge M. and Parsons S. (2000): p.1; Chen Y. et al. (1999): pp.2-3
[24] Kalina P. (2014): pp.34-35; Carabelea C. (2002): pp.3-4
[25] Aknine S. et al. (2004): p.7
[26] Vokrinek J. et al. (2007): p.660

3 Components of a Negotiation Setting

Negotiation is one of the main research areas in MAS, largely because agents operate in environments with scarce resources which share they have to optimally allocate among the agent population. Negotiation is a communication process in which a group of agents interact to come to a mutually acceptable agreement about a matter of possibly conflicting interests. The negotiation process usually starts with contradictory demands from which the parties involved move towards an agreement by making concessions or searching alternative proposals[27]. In a mathematical framework, *negotiation can be seen as a distributed search through the space of potential agreements*[28]. The outcome of the agreement is typically represented in terms of the allocation of resources such as commodities, services (i.e. outsourcing tasks), time or money. Negotiation happens either in a *one-to-many* setting (e.g. in an auction where auctioneer and bidders reach an agreement typically on the price of an item on sale) or in a *one-to-one* setting (e.g. in task allocation where a task allocation agent searches for the most suitable task agent to perform a task on the client's behalf) and it can be used to foster coordination (e.g. outsourcing) and cooperation (coalition) or to resolve conflicts in a multi-agent setting (e.g. conflicts over the usage of joint resources or tasks). In multi-agent systems, we assume that the agents are self-interested but are able to cooperate if they get a benefit from the cooperation. While auctions typically adopt a game-theoretic approach with possibly some cooperation intention (coalition games), one-to-one negotiation is often heuristic-based due to limited information about the opponent's preferences.[29]

This chapter details the different components of a negotiation setting, including the object under negotiation, the strategy used that guides the behaviour of the individual agents and the negotiation protocol as the public set of rules all agents must adhere to. As the last component is the most important part in a negotiation setting, this chapter focuses on the different literature-based negotiation protocols applicable in MAS.

When designing a multi-agent framework, several important components of the automated negotiation setting, as proposed by Kumar S., should be predefined:[30]

i. ***Negotiation objects***: the range of issues over which an agreement must be reached. A negotiation object may contain a single issue (e.g. price) or multiple issues (e.g. price & quality & terms and conditions, etc.), whereby we assume a preference independence between the individual attributes in a multi-issue setting (e.g. price is independent of quality). In some cases, the content of the agreement is fixed and can

[27] Faratin P. et al. (1998): p.159
[28] Jennings N. R. et al. (2001): p.6
[29] Sadri F. et al. (2002): p. 405
[30] Kumar S. (2012): pp.12-13

be only accepted or rejected by the participants. In other scenarios, the value of the issues in the negotiation object can change following a counter-proposal. Participants may also alter the structure of the negotiation object by adding or removing issues (e.g. offering additional benefits to clinch the deal).

ii. ***Negotiation protocol***: the set of rules that govern the interaction such as the types of participants, the negotiation states (when to bid, when to propose), the events that alter the negotiation states and the participants' valid actions in a particular state[31].

iii. ***Negotiation strategy***: is derived from the choice of the negotiation setting and protocol and defines which message can be send by whom, to whom, at what stage, and how the tactics' weights should change over the course of the negotiation rounds. The negotiation strategy, thus, specifies the sequence of actions (offers, responses) that an agent plans to make during the negotiation process. While a protocol dictates the rules of negotiation all the involved parties must abide by and is, thus, necessarily public and open, a strategy is a private guideline within a single agent or multiple cooperative agents to achieve the best outcome (e.g. utility maximization)[32].

iv. ***Reasoning model or decision-making strategies***: usually includes a description of the potential agents to be contacted, the communication style to be implemented (parallel with all agents or sequential), the initial proposals at which to start the negotiation process, the development of counter-proposals, and the decision space (when to accept, reject or opt out from the negotiation).

v. ***Domain knowledge***: In an open environment, agents frequently deal with new negotiation partners, while in a closed setting, the number of agents involved is closed or only expandable if a new agent introduces itself to the other agents.

vi. ***Social context and inter-relationships***: if negotiation takes place between agents in the same organisation, the agents usually follow a cooperative negotiation style. Bargaining involving the vertical line of the supply chain can be strongly competitive instead, involving utility maximization objectives of all agents involved.

[31] Jennings N. et al. (2001): pp.13-14
[32] Bartolini C. et al. (2001): online

3.1 Negotiation object: issues, tactics and offers

A negotiation object is an issue (or attribute) bundle that agents negotiate about. *Single-issue* negotiation usually involves negotiating about price. Seller agents, for instance, want to sell an item at the highest price achievable, while buyer agents negotiate for the lowest price achievable. In real-world applications, negotiations are based on multi-attribute or *multi-issue* objects with attributes like price, quality, quantity, etc. In most approaches, multi-attribute offers are formed as a combination of specific *tactics*. A ***tactic*** can be defined as a function representing a single component of the negotiation object (issue, attribute)[33]. The result of a negotiation round is referred to as an ***offer***, which is a linear combination of differently weighted tactics whereby the weights represent the relative importance of a respective attribute. Likewise, counter-offers represent differently weighted tactics with the goal to increase the negotiation partners' utility in order to reach an agreement.[34]

3.2 Negotiation Strategy: From Bargaining to IBN

A negotiation strategy reflects an agent's behaviour, that is a sequence of actions consisting of offers and counter-offers (responses) and it is guided by the rules of the underlying protocol. The goal of a strategy is to win a preferred share of the negotiation object. Traditionally, a negotiation strategy was **position-based (bargaining)**, also referred to as distributive or competitive bargaining, in which the parties aimed to maximize their own expected utility regardless of their negotiation partner's interests, economically also referred to as a "zero-sum game". In such a negotiation, the utility cake is fixed and divided proportional to their share won among the parties, whereby a better deal for one agent means a worse deal for another. In position-based bargaining, an agent aims to win the negotiation by imposing a preferred solution on the opposing negotiators through the use of, for instance, *time pressure* (e.g. imposing a deadline or increasing negotiation costs), *threats* (e.g. threatening to change partners), (future) *promises* or by *appearing firm* (firm agents typically start with a very high demand and concede only slowly).[35]

Contrary to the monotonic bargaining, in an **interest-based negotiation** (IBN), also referred to as *integrative* or *cooperative bargaining* or ***problem solving***, the negotiators' interests are more important than the position to be won. Hence, the agents search the space of negotiation objects (rather than the space of deals for a particular object) in order to reach a "win-win" outcome. Problem solving behaviour could include arrangements that expand the "pie" *(non-zero sum game)*, or the agreement comes with a *nonspecific compensation* (i.e.

[33] Lopes F. et al. (2009): p.20
[34] Faratin P. et al. (1998): p.162; Vetschera R. et al. (2012): p.77
[35] Shi Z. et Sadananda R. (2006): pp.327-330; Lopes F. et al. (2009): pp.18-19

paying off the opponents for accommodating an interest). Other collaborative behaviours are *reframing* (i.e. an agreement is reached from changing strategic issues only such that no concession must be made with respect to the underlying goal), *log-rolling* (i.e. one party trades off a low priority issue that is of high priority for the counter-party), *cost cutting* (i.e. reducing or eliminating negotiation costs if an agreement is reached), or *bridging* (i.e. searching for new alternatives to reach a mutually beneficial agreement).[36]

3.3 Negotiation Protocol

An effective strategy largely depends on an appropriate negotiation protocol. As previously discussed, a ***negotiation protocol*** are rules governing the interaction between the negotiating agents, including the available negotiation states (e.g. accepting, rejecting, counter-offering proposals), the valid state actions and the state transition events. The individual agent's decision, on the other hand, is derived from the internal (private) strategy. Usually, a negotiation protocol does not indicate which action to take but rather only restricts the possible action set and marks process phases at which the agents must make a decision according to their strategy.[37]

The evaluation of the negotiation results of multi-agent protocols is in so far not easy, because each agent acts upon its self-interests (i.e. objectives and constraints). Thus, determining the success of a negotiation typically requires the evaluation of the benefits or losses for each individual agent in the multi-agent setting or of the overall system outcome.

In the course of this subchapter, the main negotiation protocols from MAS literature are being discussed. These protocols broadly differentiate in their strategic positioning (interest-based, position-based, a mixture of both) and approach to offer generation. Latter is being described here mostly in its mathematical logics and is computationally often implemented with some heuristic methods. The protocols discussed range from self-interested game theoretic frameworks and argumentation-based approaches to more cooperative [interest-based] adaptations such as, for instance, coalition games and concession-based frameworks.

3.3.1 Game-Theoretic Bargaining: Non-cooperative Bargaining

Standard economic theory deals with the question of how people bargain over a pie of a certain size. One approach is to specify a fair division such as dividing a pie in its half. Another approach is to divide the pie in a way that makes the solution stable (in a Nash

[36] Shi Z. et Sadananda R. (2006): pp.327-330; Lopes F. et al. (2009): pp.18-19
[37] Jennings N.R. et al. (2001): p.18

equilibrium (NE), see definition in Appendix A) following the consideration that a purely fair division might not equally satisfy the agents involved.[38]

Game-theoretic bargaining (most prominently called *Nash bargaining*) is most often adopted in one-to-many negotiation research areas[39]. The basic idea behind the concept of game theoretic bargaining is that self-interested agents (i.e. agents who select a strategy that maximizes their expected payoff) must take into account the decisions other agents may make assuming that the agents will act in a way that maximizes their own outcome (utility). Two well-known examples of such normal-form games are *the Prisoner's Dilemma* (constant-sum game, most commonly known as zero-sum game) and *common pay-off games* (same-sum game). Latter example is also referred to as pure coordination games or team games in which the agents have no conflicting goals but aim to decide on an outcome that is maximally beneficial to every agent.[40]

In a standard negotiation approach that is being followed in the course of this paper, an agent whose turn it is to make an offer at time t makes a proposal that can be either accepted or rejected by the other agents. Alternatively, a responding agent can also opt out of the negotiation, ending the bargaining in a conflicting result. If all the agents accept an offer, the negotiation ends in the offer being implemented. If at least one agent has rejected the offer, the negotiation continues with the next agent making a counter-offer to the other agents at time $t + 1$. Thereby, the agents are not bound to any previous offers that have been made before. This strategy can be also called the *simultaneous response protocol*.[41]

Finding equilibria in dynamic MAS typically always involves such repeated games with offers and counter-offers until an equilibrium is reached. A prominent example for repeated games is the method of ***Rubinstein's alternating offers*** which employ the notion of *subgame-perfect equilibrium* (i.e. every subgame is in a NE) in repeated games among agents with an infinite time horizon (i.e. unlimited offers and counter-offers). In the Rubinstein's repeated games' model, agents differ in their preferences but have complete information over the other preferences, and time is valuable (i.e. time discounts through fixed bargaining costs and a fixed discounting factor on future utility). The discounting and cost factor in Rubinstein's alternating offers is crucial to keeping the equilibrium reached stable. Therefore, the main difference between Nash's and Rubinstein's equilibrium is that Nash equilibrium is in some scenarios a weak one in which one agent might have yield a higher pay-off in a first negotiation round but has settled for a lower pay-off in the next round to reach equilibrium, and it might, thus, have an incentive to deviate from the final outcome in order to reach a

[38] Levin J. (2002): p.1
[39] Sadri F. et al. (2002): p.405
[40] Shoham Y. and Leyton-Brown K. (2009): pp.56-58; Kraus S. (1997): pp.82-83
[41] Goel A. K. et al. (2011): p.377; Chen Y. et al. (1999): p.3

higher pay-off. Rubinstein's subgame perfect equilibrium, on the other hand, requires that an agent's strategy is optimal (in NE) at *every* negotiation round rather than only at the initial stage.[42]

Self-interested intelligent agents can choose between two forms of strategies: (1) they select a single action (*pure strategy*), or (2) they can randomly select over a set of available actions according to some probability distribution (*mixed strategy*)[43]. In case of a multi-issue negotiation object, parts of the whole object can be separate subjects of the bargaining process.

Game theoretical bargaining is strongly related to classical *decision theory* that describes mechanisms for making decisions in systems in which the outcomes of the various actions are unknown. In a typical MAS setting, intelligent agents operate in an open environment with incomplete information. Thus, for some aspect of the current state of the environment X_i, the agents only know that each possible value x_{i_j} of X_i has some probability, formally[44]:

$P: x \in X \mapsto [0,1]$ and $\sum_j P\left(x_{i_j}\right) = 1$

If an agent knows that the current state has a value x_{i_j}, then $P\left(x_{i_j}\right) = 1$, else $P\left(x_{i_j}\right) = 0$. If two aspects of the current state of the environment, X_1, X_2, are related to each other, they have dependent probabilities (conditional probabilities), hence: $P\left(x_{1_i} \wedge x_{2_j}\right) = P\left(x_{1_i} | x_{2_j}\right) * P(x_{2_j})$, else they are independent with a probability of $P\left(x_{1_i} \wedge x_{2_j}\right) = P\left(x_{1_i}\right) * P(x_{2_j})$.[45]

Typically, each agent places a different value to a current (or future) state of the environment depending on how well it is connected to the agent's goal, economically referred to as *utility*. As Von Neumann and Morgenstern showed, utility can be easily defined in terms of preference relations such that an outcome that is preferred has a higher utility[46], formally:

$$u(s_i) \geq u\left(s_j\right) \; iif \; s_i \succcurlyeq s_j \qquad (1)$$

$$u([p_1: s_1, .., p_k: s_k]) = \sum_{i=1}^{k} p_i u(s_i) \qquad (2)$$

In words: (1) the utility under a state i is at least as high as under another state j if and only if the outcome under the state i is at least as preferred as the outcome under state j, and (2) the overall utility of a negotiation equals the respective utility from the current state's aspects considering their probability to occur.

---

42 Rubinstein A. (1982): pp.98-108
43 Shoham Y. and Leyton-Brown K. (2009): p.59
44 Parsons S. and Wooldridge M. (2000): p.2
45 Parsons S. and Wooldridge M. (2000): pp.2-3
46 EconPort (2006): online; Levin J. (2006): pp.8-10; Shoham Y. and Leyton-Brown K. (2009): pp.52-53

By combining the probability for a state with the intention to maximize utility, intelligent agents work with a so-called *expected utility function* (also called the *Neumann-Morgenstern utility function*), that is the linear and continuous weighted average utility of each possible outcome with the weight being the probability of an outcome under a certain action performed, mathematically:

$U(A_i) = \sum_{S_j \epsilon S} P(S_j | A_i) * U(S_j)$, with S being the set of all possible states and $U(S_j)$ being the different utility values for each of the state outcomes. The agent selects A^* such that $A^* = \arg\max_{A_i \in A} \sum_{S_j \epsilon S} P(S_j | A_i) * U(S_j)$.[47]

In the practical context, the game-theoretic approach in bargaining has several drawbacks: first, considering the necessity to create a stable negotiation protocol, agents do not deviate and must strictly follow the strategies defined for them in the strategy profile of the NE. However, as a negotiation can involve counter-proposals and concessions made by other agents, the use of Nash Equilibrium creates unstable agreements (a strategy could be in equilibrium at the beginning of the negotiation, but may become unstable if an agent must deviate from its optimal payoff to reach an agreement). To overcome the limitations of Nash equilibrium strategies, subgame perfect equilibrium could be used in which a NE can be obtained after every negotiation round. Under incomplete information, however, a subgame cannot exist, as it would require information sharing between the competitive agents. A second drawback to the stable game-theoretic bargaining is the general lack of cooperation possibilities as the agents follow a purely payoff-maximization strategy. Thus, a MAS with game-theoretic bargaining among intelligent agents would be very likely always unstable.[48]

3.3.2 Argumentation-based Negotiation

The typical negotiation process is largely based on exchanging proposals only. Such bargaining starts with one agent making the first offer to another agent who can accept, reject or make a counter-offer on the first offer, which the receiving agent can react to in the same manner until a negotiation fails or ends. Argumentation-based negotiation (ABN) enriches the negotiation process by offering justifications for offers and critics on proposals that were rejected (see Figure 4, represented as "attack proposal" in the right graph). ABN is widely used in dialogue games and is a tool particularly used in decision-making under uncertainty. The fundamental approach in ABN is the ability of agents to exchange

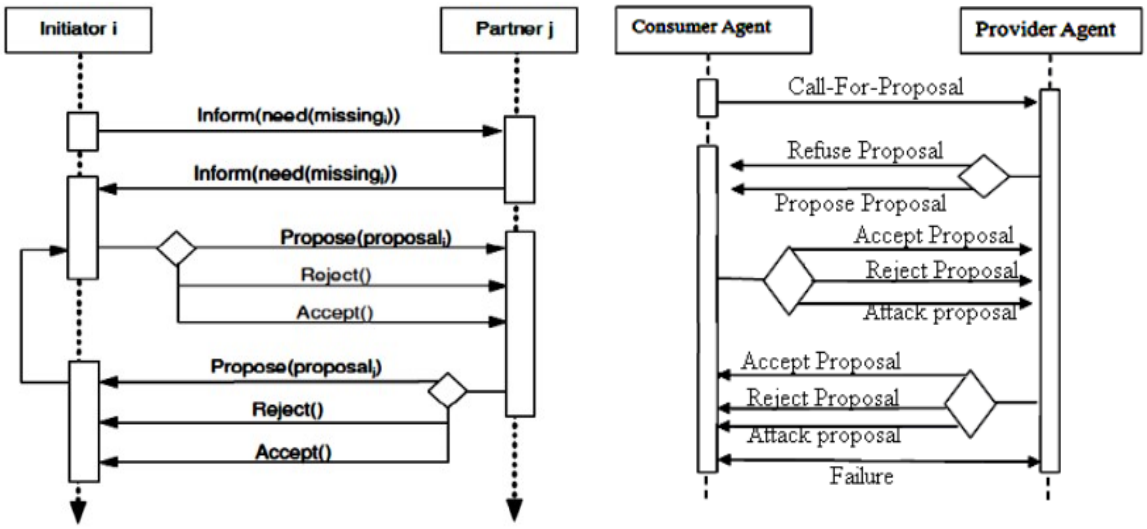
[47] Parsons S. and Wooldridge M. (2000): p.5; Levin J. (2006): p.6
[48] Kraus S. (2001): pp.153-156

information about their internal state in order to convince the other of an offer or communicate their preference relations on the set of offers.[49]

Convincing or explaining a proposal or withdrawal to the negotiation partner is an element crucial in an interest-based strategy as arguments can help reaching an agreement more efficiently if it lies in the interest constraints of the counterpart. As such, argumentation-based negotiation can be considered a suitable negotiation framework for interest-based negotiation if the arguments included are not of non-interest-based nature (e.g. using threats or authority).[50]

Existing literature on ABN focuses on two main approaches: (1) designing negotiation concepts for logically defending arguments, and (2) designing bargaining-frameworks with exchange mechanisms of rhetorical arguments (e.g. rewards, promises, appeals).[51]

Figure 4: UML 2.0 specification of a traditional bargaining protocol (left graph) and of an ABN (right graph), illustrated in a so-called contract net interaction protocol (CNP)[52]



In a typical negotiation setting, intelligent agents are utility maximizers and they interact with the external environment to find supporting information for their offer's argumentation. In ABN, an offer is transmitted with an argument, created from both internal and external information that supports the conditions under which the offer was made. If an agent is unable to produce a new offer with arguments for its goal, it searches for new arguments under which it can accept the (counter) proposal of the other agent. Figure 5 illustrates an ABN protocol as proposed by Kakas A. and Moraitis P. (2006) from the view of an agent (here, agent X), assuming that the opposing agent (here, agent Y) performs the exact same functions (which is normative for protocols). In their protocol, the authors use argumentative

[49] Marey O. et al. (2014): p.61; El-Sisi A. and Mousa H. (2014): pp.157-158; Rovatsos M. et al. (2005): p.7
[50] Rahwan I. (2004): p.64
[51] Rahwan I. et al. (2004): p.2; Kraus S. et al. (1998): p.1; Ramchurn S. et al. (2003): p.1
[52] El-Sisi A. and Mousa H. (2014): pp.157-158

functions (deliberate, accept) and three evaluation functions used by the sender agent to determine which information to send ($propose(Offer, e_j, s_i)$). They differentiate between three types of proposals (offers): in the first type, denoted as $propose(O^Y, e^{Y \to X}, s_n^{Y,Y})$, agent Y sends a new offer to agent X, O^Y, with new supporting information $s_n^{Y,Y}$. From the previous step, agent Y has evaluated the information for the offer sent by agent X (Note: the offer send by agent X is denoted as O^X), which [the evaluation] is denoted as $e^{Y \to X}$. In the first step (agent X has not sent an offer yet), $e^{Y \to X}$ equals 0. In the second proposal type, $propose(O^Y, {e_m}^{Y \to X}, \otimes)$, agent Y sends an answer to agent X's previous offer to accept O^Y under given terms or conditions [proposed by agent X], ${e_m}^{Y \to X}$ (agent X has entered the so-called *conciliation phase* when it is willing to accept an offer under certain conditions). In the third proposal type, $propose(O^X, \otimes, s_n^{Y,X})$, agent Y sends an answer to agent X with conditions $s_n^{Y,X}$ under which it would accept the offer from agent X, O^X (agent Y has entered the conciliation phase).[53]

Carabelea C. (2002) suggests an argumentation-based negotiation approach between two agents in which a so-called *negotiation balance* (NB), as the ratio between the value under negotiation and the value being traded for (ideally being 1), is being compared by each agent. While the server agent typically tries to raise the balance so that the value obtained is higher than the value that is traded in exchange, the client agent aims to lower the balance in its own favour. The argument chosen by each agent is, thereby evaluated based on an expected effect of that argument on the opponent, and hence their evaluation of the NB. Typically, each agent creates a list of possible arguments from which it chooses the argument that improves the NB for the opposing agent while still being in the acceptance range of its own NB.[54]

A similar approach is proposed by Rahwan I. et al. (2004) in which the value of items is computed as a utility of a contract or plan, and the utility is influenced by the *worth of states* which is proportional to the set of desires that are satisfied by the respective state. The state is influenced, for instance, by the cost of carrying out a contract or plan (if the cost increase, utility decreases) or by additional [free] benefits that are directly linked to the offer.[55]

[53] Kakas A. and Moraitis P. (2006): pp.386-387
[54] Carabelea C. (2002): pp.5-6
[55] Rahwan I. et al. (2004): p.9

Figure 5: Argumentation-based negotiation protocol based on Kakas A. and Moraitis P. (2006)

Begin
S={}; n=0
1. agent X receives a proposal O from an agent Y

2. **if** O is of the form propose(O^Y, $e_{n-1}^{Y\to X}$, $s_n^{Y,Y}$) **then**
2.1. $e_n^{X\to Y}$←evaluate(X, $s_n^{Y,Y}$);
2.2. S←update(S, $e_{n-1}^{Y\to X} \cup e_n^{X\to Y}$)
2.3. **if** $e_n^{X\to Y} = s_n^{Y,Y}$ and accept(X, G^Y, S) **then**
END(agreement, O^Y)
else
2.4. n ←n+1; find $s_n^{X,X}$ s.t. deliberate(X, G^X, S; $s_n^{X,X}$)
and $s_n^{X,X} \neq s_i^{X,X}$ ∀ i < n
2.5 **if** s_n^X exists **then** propose(O^X, $e_{n-1}^{X\to Y}$, $s_n^{X,X}$) to Y
else (***Enter Conciliation Phase***) m = 0
2.6. S←update(S, $e_n^{Y\to X}$)
2.7. m ←m+1; find $s_m^{X,Y}$ s.t. deliberate(X, G^Y, S; $s_m^{X,Y}$)
and $s_m^{X,Y} \neq s_j^{X,Y}$ ∀ j < m
2.8. **if** $s_m^{X,Y}$ exists **then** propose(O^Y, ⊗, $s_m^{X,Y}$) to Y
else END(Failure)
2.9. endif
2.10. endif
2.11. endif
2.12. endif

3. **if** O is of the form propose(O^Y, $e_m^{Y\to X}$,⊗) **then**
3.1. goto 2.6; endif

4. **if** O is of the form propose(O^X, ⊗, $s_k^{Y,X}$) **then**
4.1. $e_k^{X\to Y}$←evaluate(X, $s_k^{Y,X}$)
4.2. S←update(S, $e_k^{X\to Y}$)
4.3. **if** $e_k^{X\to Y} = s_k^{Y,X}$ and accept(X,G^X,S) **then**
END(agreement, O^X)
4.4. **else** propose(O^X, $e_k^{X\to Y}$, ⊗)
4.5. endif
4.6. endif
end

Because arguments are always selected based on knowledge, beliefs and preferences that are usually unknown to the addresser, ABN remains a negotiation approach that has to deal with a high degree of uncertainty. Marey O. et al. (2014) discussed ABN in relation to two main types of uncertainties and they proposed techniques to handle those uncertainties: *Type I* is the agent's uncertainty about selecting the right moves during the dialogue, and *Type II* is the agent's uncertainty that the opposing party will accept the move. Their uncertainty-aware techniques classify arguments into several classes depending on their *risk of failure* (*note:* failure is meant as not being chosen). Arguments of the same risk class are considered to be equally favourable and preferable (thus, have a higher acceptance rate by the opponent), and more favourable and preferable than any other argument of another class

with a higher risk rate. The authors further classified the risk classes into smaller disjoint subclasses in case of several arguments. If two arguments are considered equally preferable, they are put into the same low subclass (higher acceptance rate). If an argument is less preferable than an existing argument, it enters a higher subclass, etc.[56]

ABN is said to also work well with multiple negotiation partners (e.g. in an auction), whereby each agent follows the same protocol rules and the negotiation runs parallel between several sender agents (e.g. seller agents) and one receiving agent (e.g. buyer agent). In practice, however, ABN can come with considerably low scalability and extensibility in a dynamic system in which the number of agents can continuously change, making the MAS computationally extensive and complex. Particularly in a highly competitive setting in which competing agents start with extreme positions and only gradually concede, ABN could turn out to include much redundancy and be computationally more expensive than beneficial to be implemented.

3.3.3 Cooperative Game Theory: Coalitional Games

Negotiation is not only about winning a position or creating profits from a business trade agreement, but it is likely also about creating successful synergies and profitable cooperations. Coalition refers to agents grouping together to cooperate on a joint task, using joint resources with individual competencies and abilities. Coalition can be formed between parties who offer complementary services (e.g. for allocating a task to more suitable partners), or between similar parties (e.g. outsourcing). Sometimes it is also beneficial to join with strongly diverse partners in order to create a wider market share and save costs on SCM processes due to committed long-term collaborations. Coalitions are typically used in task allocation, sensor networks and electronic marketplaces.[57]

Coalition formation deals with the following three sub-problems: first, *selecting an adequate coalition partner* in the agent population such that agents within each coalition self-organise their activities to achieve their goals while the cooperation of the coalition at large is rewarded and not the individual agents; and second, *sharing the cooperation value* between the coalition partners; and third, *solving a joint optimization problem* with shared and combined resources from that coalition. As the optimal coalition structure is NP-complete, finding an optimal coalition from the agent population can get computationally very exhaustive.[58]

---

[56] Marey O. et al. (2014): pp.62-65
[57] Airiau S. (2010): p.1
[58] Sandholm T. et al. (1999): pp.209-211; Airiau S. (2010): p.3

The fundamental questions behind coalitional game theory are[59]:

1. **Which coalition will form? (i.e. how to partition the set of all agent population into coalitions)**
2. **How should the coalition divide its payoff among its members?**

The most likely coalition structure to be formed is the so-called *grand coalition* that is the coalition of all the agents in the agent population. As a grand coalition yields a higher payoff than other coalition structures and is, thus, more likely to form, the focus in this subchapter lies in payoff division only.[60]

While existing literature focuses on single coalition structure formation, several coalitions can also co-exist within a coalition system, which usually places additional physical and organizational constraints. Skibski O. et al. (2016) assume that while there are typically low restrictions on the feasibility of a coalition structure, hence, any coalition structure may form, the number of feasible coalitions within a coalition structure is limited. Thus, a feasible coalition structures can still contain infeasible coalitions[61], which can make coalition formation a practically hard task to be implemented in real-life MAS projects. In this subchapter, feasibility constraints are, due to their complexity, not further considered.

Coalition games are one branch of game theory studies in which agents can cooperate and, thus, do not act position-based only. The literature of cooperative game theory differentiates between two main models: the transferable utility games (TU games) in which utility can be compared between two agents and utility can be distributed freely among the coalition members, and the non-transferable utility games (NTU games). While in the TU game, the value created from the coalition can or has to be shared among the members, thus a comparison and utility transfer must be allowed, coalition members in an NTU game cannot compensate any agent while each agent still profits individually from the coalition.[62]

Typically, analysing the payoff division lies in the core of any coalition forming, as rational agents are more likely to cooperate in more profitable coalitions than acting in less profitable coalitions or alone. Unfortunately, dividing the coalition payoff equally among its members is rarely a stable solution. This subchapter suggests the most common solution concepts from literature to divide a coalition payoff along with important characteristics for stable solutions.

[59] Shoham Y. and Leyton-Brown K. (2010): p.384
[60] Shoham Y. and Leyton-Brown K. (2010): p.387
[61] Skibski O. et al. (2016): pp.177-178
[62] Airiau S. (2010): p.2

Transferable utility (TU) games

A system with a universal currency is a classic example for a TU coalition, with the coalition's payoff being a single value. A fundamental criterion for coalition formation is that each agent is guaranteed a payoff that is at least as high as a one-agent coalition.[63]

We denote the finite set of agents N, indexed by i, and let $v: 2^N \mapsto \mathbb{R}$ (characteristic or valuation function) be the real-valued payoff $v(S)$ for a coalition $S \subseteq N$ that the members of the coalition can distribute among themselves ($v(\emptyset) = 0$). In TU games, coalitions follow at least one of the following characteristics: they can be ***additive***, i.e. the worth of each coalition is always the same regardless of its degree of cooperation with other coalitions (formally, for any additive game $G(N, v)$: $\forall S, T \subset N$, if $S \cap T = \emptyset$, then $v(S \cup T) = v(S) + v(T)$); they can be ***super-additive***, i.e. the payoff of the sum of two non-overlapping coalitions is at least as large as the individual payoffs, implying additional inequality to additivity (formally, $v(S \cup T) \geq v(S) + v(T)$); they can be ***sub-additive*** when the coalition is less profitable than remaining along (i.e. $v(S \cup T) \leq v(S) + v(T)$); and they can be ***convex***, i.e. the marginal contribution of an agent i to a coalition S, $v(S \cup \{i\}) - v(S)$, increases with the size of the coalition the agent joins (formally, for any convex game $G(N, v)$: $\forall S, B \subset N, S \subseteq B$ and $i \notin B$, $v(S \cup \{i\}) - v(S) \leq v(B \cup \{i\}) - v(B)$).[64]

The question to how the coalition's payoff should be divided among the members can be answered using the following most widely known methods:[65]

1. ***The Shapley value*** divides the coalition's payoff among its members based on the *fairness* principle under the formula:
   $\phi_i(N, v) = \frac{1}{|N|!} \sum_{S \subseteq N \{i\}} |S|! \, (|N| - |S| - 1)! \, [v(S \cup \{i\}) - v(S)]$ (Average marginal contribution of agent i, taking into consideration all possible sequences S of the agents entering the coalition, starting from the empty set and adding uniformly at random one agent at a time; differently put, the average of an agent i's marginal contribution at the time the agent is added considering all its possible sequences of additions).[66]
2. ***The core*** divides a coalition's payoff based on what division makes the agents want to form the coalition ("*strong equilibrium"*), considering that not all agents are willing to accept their fair share from the coalition and have, thus, an incentive to deviate. A linear feasibility problem is used to derive to a payoff vector for each member that

[63] Shoham Y. and Leyton-Brown K. (2010): p.388
[64] Airiau S. (2010): p.3; Shoham Y. and Leyton-Brown K. (2010): pp.386-387
[65] Shoham Y. and Leyton-Brown K. (2010): p.384
[66] Shoham Y. and Leyton-Brown K. (2010): pp.389-390

makes the coalition attractive to be formed. Thereby, we say that the *payoff vector must be in the core of a coalitional game* (N, v) which is the case iif:[67]

$$\forall S \subseteq N, \sum_{i \in S} x_i \geq v(S) \ ,$$

whereby x_i refers to the agent i's share (real value) of the coalition's payoff. Differently speaking, the sum of payoffs to any group of agents $S \subseteq N$ must be at least as high as the payoffs these agents would share by forming a sub-coalition. To fulfil this straightforward criterion, the core must be both non-empty and unique which is, however, not always the case. The possibly strongest condition to allow a valid payoff division is convexity. However, Shoham Y. and Leyton-Brown K. (2010) proposed several other theorems that allow for checking the core's validity. They additionally suggested several refinements to the formula mentioned above that allow a non-existing and non-unique core.[68]

Non-transferable utility (NTU) games

Differently to TU games, non-transferable utility games additionally have different outcomes depending on the coalition formed. Thus, while in TU games agents can have allocation preferences (i.e. preferences over the share of the coalition's payoff), agents in NTU games make their cooperation choice based on the outcome from a coalition.

The set of outcomes is denoted with X whereby x_S denotes the unique outcome from a coalition $S \subseteq N$. The valuation function is $V: 2^N \rightarrow 2^X$ that describes the payoff outcome $V(S) \subseteq X$ ($V(S) = \{x_S\}$) that can be achieved from a coalition S. Additionally, each agent i has preferences over the set of outcomes given as the preference relation $\succ_i$, which is considered transitive and complete. For any two outcomes x_S and x_T from coalitions S and T that agent i could enter, $x_S \succcurlyeq_i x_T$ (in words, agent i prefers the outcome from coalition S at least as much as the outcome from coalition T) iif $S \succcurlyeq_i T$.[69]

As agents in NTU games cannot measure the common worth of a coalition, they cannot aim for a fair share (computed with the Shapley value). Instead, an agent chooses a coalition based on a stable outcome from the coalition, measured by ***the core*** value:

$$core(V) = \{x \epsilon V(N) | \nexists S \subset N, \nexists y \in V(S), \forall i \in S \ y \succ_i x\}$$[70]

i.e. a coalition is in the core if there is no other outcome $y \epsilon X$ that is strictly preferred by any other member of the coalition. Otherwise (i.e. for any outcome $x \preccurlyeq_i y$), agents have an

[67] Shoham Y. and Leyton-Brown K. (2010): p.391
[68] Shoham Y. and Leyton-Brown K. (2010): pp.391-394
[69] Airiau S. (2010): pp.21-22
[70] Airiau S. (2010): pp.21-22

incentive to deviate from the coalition to achieve that outcome and, thus, no coalition is formed.

Constructive approaches to generating coalitions in TU and NTU games have been rarely studied yet in multi-agent applications. Thereby, coalition games research focuses on the complexity of assessing the stability of coalitional structures, such as determining the non-emptiness of the core. Dunne P. E. et al. (2010) proposed a Coalition Resource Game (CRG) framework in which the stability of a coalition can be determined in feasible amount of time with small number of agents (exponentially) and agent objectives (polynomially) while the outcome proves to meet subgame perfect equilibrium, alongside other stability properties[71].

3.3.4 Auctions

An auction is a *one-to-many negotiation* style in which one client aims to find a business partner (e.g. buyer agent) out of a group of several potential servers. Auctions are not only one of the most ancient negotiation styles, they are also the bargaining type most dominant in e-commerce (e.g. EBay, AuctionBot). There are several types of auctions with each modelling a specific protocol. The most prominent auction types are the *English* auction in which the auctioneer raises the price of the negotiation object until only one bidder remains; the *Dutch* auction which works opposite to the English version with the highest possible price and gradual price decreases until the first bidder accepts the price; the *First-Price Sealed Bid* auction in which bids are made simultaneously and the highest bidder wins; and the *Vickrey* auction which is similar to the First-Price Sealed Bid but with the highest bidder (winner) paying the second highest price.[72]

When considering an auction protocol, the negotiation follows some basic assumption: (i) agents aim to maximize their expected utility from an outcome, and (ii) the opponents' valuations are held private and are, thus, not known to the opposing negotiators. In contrast to argumentation-based negotiation, agents do not have a direct reasoning for why an offer has been rejected by another party. This typically leads to suboptimal solutions (local optima). The uncertainty about the opponent's preferences can be handled by either modelling a likely belief of the opponent (e.g. learning from experience following an interest-based strategy) or by constructing a strategy that is independent of the opponent's beliefs (position-based strategy). In order to reach a stable deal, the agents must follow a protocol that is based on the agents' dominant strategy which yields an expected payoff that is at least as high as the other's expected payoff regardless of the opponent's behaviour. A

[71] Dunne P. E. et al. (2010): p.20
[72] Jennings N. (2000a): p.30

dominant strategy prevents the partners to deviate after closing the contract. In this subchapter, a protocol framework for English based auctions is formulated. Literature argues that both English and Vickrey protocols use dominant strategies, but that the English protocol works better in more complex negotiation settings including time, issue preferences and offer (change) dynamics.[73]

As auction-based protocols are quite common in MAS literature, this subchapter offers a generic mathematical model for implementing auction protocols in multi-issue negotiations as suggested by Benameur H. et al. (2002). An actual protocol implementation is offered by Bartolini C. et al. (2002), see link in the references, which described in detail the actions and messages of a single-issue English-based auction in JESS according to the General Negotiation Protocol[74].

For simplicity reasons, we assume identical and fixed number of items Q under auction, as well as private evaluations and one vendor agent for a finite number n buyer agents $A_1, \dots, A_n$. An auction framework for variable item quantity is proposed by Lengwiler Y. (1999).

Each buyer A_i aims to get a quantity q_i whereby $\sum_{i=1}^{n} q_i > Q$, so that the buyers become competitive bidders for their desired quantity. The money available for each bidder is denoted as $V_i, i = 1, \dots, n$ with $V_i \sim unif(V_{min}, V_{max})$, the continuous uniform distribution, where $V_{max} - V_{min} > 1$, so that the valuations are independent and private. As all items are identical, the buyer agents are willing to bid the same amount of money v_i for all the desired items:

$$V_i = v_i \times q_i$$

Let b_i be the function of the offers submitted that describes the auction strategy of buyer A_i depending on the budget V_i and desired quantity q_i of item i:

$$b_i = b(V_i, q_i), \quad i = 1, \dots, n$$

In order to increase the price bid, a buyer agent could decrease the desired quantity during an auction according to $v'_i = \frac{v_i}{q'_i}$, with v'_i being the new price bid and q'_i the new quantity desired ($q'_i < q_i$). The quantity $\bar{q}_i$ buyer agent A_i receives at the end of the auction directly depends on the offers b_j of the other buyer agents A_j. Thus, the following auction result holds with respect to the opponents' bids:[75]

[73] Jennings N. (2000a): pp.31-35
[74] Bartolini C. et al. (2002): pp.8-9
[75] Benameur H. et al. (2002): pp.7-8

$$\bar{q}_i = q_i \qquad if\ Q - \sum_{j:b_j > b_i} q_j \geq q_i$$
$$\bar{q}_i = Q - \sum_{j:b_j > b_i} q_j \qquad if\ 0 < Q - \sum_{j:b_j > b_i} q_j < q_i$$
$$\bar{q}_i = 0 \qquad if\ Q - \sum_{j:b_j > b_i} q_j \leq 0$$

whereby $\sum_{j:b_j > b_i} q_j$ denotes the quantity of the opponent bidders. The utility received from the auction is thus: $U_i = (\bar{v}_i - b_i) \times \bar{q}_i = V_i - b_i \bar{q}_i$ with $U_i \geq 0$ to be maximized.

While the auction protocol as proposed by Benameur M. et al. (2002) is straightforward, the difficult part lies in determining the bid b_i, such that the utility is maximized under a small, yet comparatively best bid. The actual strategy that determines the bid should consider those two contradicting constraints. Benameur M. et al. further state that- in order to obtain the best possible bid b_i- the strategy considers the expected probabilities for the $n-1$ opponent bidders' offers b_j such that the buyer agent's bid b_i is superior to what is expected from the opponents. Mathematically, this can be described with the conditional probability function:

$P(b_j < b_i | q_j) = \int_{V_{min}}^{q_i b_i} F(V) dV$ with $F(x) = \frac{1}{V_{max} - V_{min}}$, that results in

$P(b_j < b_i | q_j) = \frac{q_i b_i - V_{min}}{V_{max} - V_{min}}$

The probability that all offers of the other $n-1$ opponent bidders are inferior to b_i is computed using the maximum-likelihood method (considering independent bids):

$$\prod_{i=1}^{n-1} \frac{q_i b_i - V_{min}}{V_{max} - V_{min}} = \left[\frac{q_i b_i - V_{min}}{V_{max} - V_{min}}\right]^{n-1}$$

As every bidder aims to maximize their utility from the auction, each bidder maximizes the following expression (the own bid multiplied by the expected probability of all other bidders):

$$\frac{\partial (V_i - q_i b_i) \left[\frac{q_i b_i - V_{min}}{V_{max} - V_{min}}\right]^{n-1}}{\partial b_i} = 0$$

After differentiating above maximization problem under b_i, the result to compute the optimal offer $\hat{b}_i$ of the buyer agent A_i is:

$\hat{b}_i = \frac{V_{min} + (n-1) V_i}{n q_i}$,

which represents the minimum offer that maximizes the probability to win the auction at a uniformly distributed budget. In case another budget distribution is used, the above mentioned $F(x)$ must be replaced by the respective cumulative distribution choice. As V_{min} is a value unknown to the respective bidder, the bids submitted are usually not optimal.

However, this auction protocol should give a good baseline to compute bids as close as possible to their optimal value.[76]

3.3.5 Concession-Based Negotiation

In a concession-based bargaining, the parties begin the negotiation with inconsistent and often extreme positions in their favour and then make concessions in each negotiation round, relaxing their demand, in order to reach an agreement. Thereby, concessions can be changes in the issue space or in the utility space, or both. Concessions are typically based on the opponent's preferences, e.g. changing an issue to a more preferred option for the opponent. In case the negotiator lacks information about the opponent's preferences, though, concessions could move from a more preferred option to a less preferred option in at least one issue for the negotiator, in each negotiation round.[77]

Vetschera R. et al. (2012) proposed a concession-based approach to bargaining with a focus on offer generation, looking at both the utility space and the issue space. This framework proposed is being introduced in this subchapter. Similar to Lai G. and Sycara K. (2009), who established a multi-attribute offer generation approach to negotiation that uses Rubinstein's alternating offer game in a time-based strategy to reach an agreement, Vetschera R. et al.'s framework offers solutions along the Pareto-frontier and considers incomplete information to concession-based bargaining which, in contrast to similar studies, better resembles real-world negotiation scenarios. Both the alternating-offer game by Rubinstein as proposed by Lai G. and Sycara K., and the concession-based approach by Vetschera R. et al. generate offers based on the utility level of the opposing party's preferences (considering some minimum information are available) and make concessions by minimizing the Euclidean (shortest) distance between the offer and the last offer of the opponent. In case no information about preferences is available, a mediator agent is used to collect some information about both parties' preferences and suggests an improved offer close to the Pareto-frontier. While the alternating-offer framework generates offers based on the final outcome (preferences), the concession-based framework by Vetschera R. et al. focuses on preferences and properties in each concession step.[78]

[76] Benameur H. et al. (2002): pp.7-11
[77] Vetschera R. et al. (2012): p.78
[78] Vetschera R. et al. (2012): pp.77-78; Lai G. and Sycara K. (2009): pp.11-27

In the framework proposed, three criteria must hold in each negotiation round, thus for each offer being proposed[79]:

(i) *real concession-making*: refers to offers that are of higher utility to the opponent and lower utility to the negotiator

(ii) *reciprocity*: in order to generate stable offers (avoiding too small or too large concessions that could lead to unfavourable agreements), the concession made should reciprocate the concessions received from the opponent to compensate any utility losses the opponent made in his last offer with utility gains in the current offer of the negotiator

(iii) *value creation*: in order to create Pareto-optimal solutions, each negotiation step should allow the joint utility to increase in value. Value creation, thus, compares the utility given up by the negotiator to the utility gained by the opponent in the same negotiation round, and requires the gains to exceed utility loss

In order to generate offers successfully, the negotiation process considers the time interval and negotiation history. At each negotiation step, both parties make a concession until two offers are compatible and close the deal. In their own offers, the agent demands a higher utility for itself and proposes a lower utility to the opponent than in the opponent's offers. The concession made by the opposing party is denoted as $\delta > 0$ and describes the utility decrease the opponent accepted in his last round. The utility from a previous offer is denoted as u_{own}^0 respectively u_{opp}^0 for the negotiator and the opponent, and $(u_{own}(x), u_{opp}(x))$ are the utility values achieved by the new offer x.[80]

As mentioned above, any new offer $x \in X$ (utility vector with issue values) must fulfil the criteria of real concession-making, reciprocity and value creation, besides being feasible. Mathematically, new offers must, thus, meet feasibility and the following conditions:[81]

(i) $u_{own}(x) < u_{own}^0$

(ii) $u_{opp}(x) \geq u_{opp}^0 + \delta$

(iii) $u_{own}(x) + u_{opp}(x) > u_{own}^0 + u_{opp}^0$.

The framework proposed can be briefly described as follows: from the starting extreme positions (u_{own}^0, u_{opp}^0), each agent makes a concession along the path

$u_{own}(x) + \alpha\, u_{opp}(x) = u_{own}^0 + \alpha\, u_{opp}^0$, with $\alpha > 0$, describing the trade-off rate between both parties' utility, (how much one agent gives up in utility for the gain of the other agent). In case

[79] Vetschera R. et al. (2012): pp.78-79
[80] Vetschera R. et al. (2012): p.80
[81] Vetschera R. et al. (2012): p.80

those issues are not continuous but discrete values only, the concession path above might not reach a feasible solution, and the authors proposed an alternative concession cone that contains solutions (new offers) close to the concession path:[82]

$$u_{own}(x) + (\alpha - \tau)\, u_{opp}(x) \leq u^0_{own} + (\alpha - \tau)\, u^0_{opp}$$

$$u_{own}(x) + (\alpha + \tau)\, u_{opp}(x) \geq u^0_{own} + (\alpha + \tau)\, u^0_{opp}$$

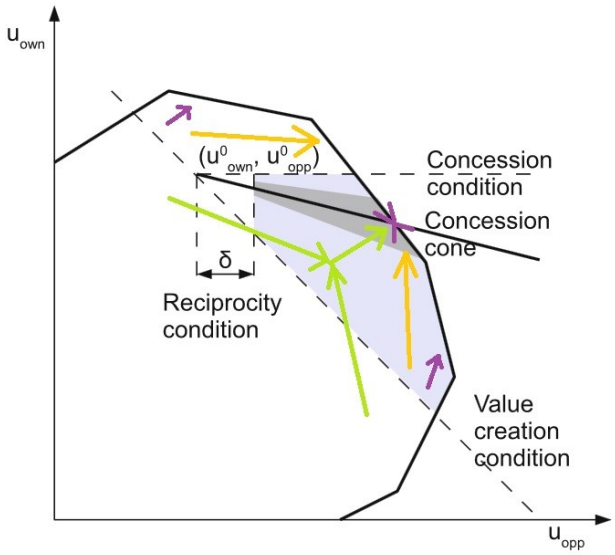
with τ being a constant indicating how far the concession can deviate from the concession path.

Both α and τ are parameters individually set by the agents who must not necessarily select similar values. Figure 6 shows the discrepancy between continuous and discrete issues as suggested by Vetschera R. et al. In case issues are discrete, the feasible set contains only discrete points and is, thus, no longer convex. The new formulas for the concession cone allow the agents to search for offers close to the concession path that imitates continuous issues under a convex set. Also visible in the graph is the importance of offers being on the Pareto-frontier. Let us assume that one agent has selected a ***negative α-value*** and, thus, aims for Pareto-improving offers (selfish step that would contradict the initially established condition of real concession making). In Figure 6 this situation is presented with *violet* arrows, the selfish agent is the leftmost violet arrow. This agent would reach the Pareto-frontier very fast but far away from the opponent's utility position. In order to find an agreement, the agent must then move along the frontier, giving up a larger utility in comparison to the opponent's gains and losing all its initial utility gains from the selfish step. Let us assume now that both negotiating agents choose a ***positive but low α-value***, in Figure 6 they are displayed as the *yellow* arrows. Both reach the Pareto-frontier close to one another and would come to an agreement faster than in the first scenario. Now let us assume, both agents choose a relatively large ***α-value*** (*green* arrows) which leads to an inefficient agreement (both offers match at inefficient points below the frontier) from which they both look for mutual improvements.[83]

[82] In case of $\alpha \leq 0$, the agent would make Pareto-improving steps instead of real concessions. Vetschera R. et al. proposed an $\alpha \leq \min(1, \frac{u^0_{own} - u^p_{own}}{u^*_{opp} - u^0_{opp}})$ with p indicating the previous offer and $u^*_{opp}$is the optimal objective value for the linear maximization problem $\rho\, u_{own}(x) + u_{opp}(x) \rightarrow max,\ s.t.: u_{own}(x) \geq u^p_{own}\ and\ x \in X$. ρ is a constant that serves to eliminate weakly dominated solutions in case of multiple optima (Vetschera R. et al. (2012): pp.81-82)

[83] Vetschera R. et al. (2012): pp.81-83

Figure 6: Concession cone based on Vetschera R. et al. (2012)



Besides that agents aim to maximize their utility by making offers based on the most optimal agreement along the Pareto-frontier, further constraints include the issue values themselves which define the feasible set of offers. Vetschera R. et al. proposed a connection between utility values and issue values assuming an additive utility function $u(x) = \sum_j w_j v_j(x_j)$ with x_j being a vector of issue values describing an offer x, w_j is the issue j's weight, and $v_j(.)$ is the partial utility function for issue j. Typically, the preferences of the opponent are in the opposite direction, for which reason an offer x_j for the negotiator can be replaced by $1 - x_j$ for the opponent's preferences. The issue value x_j can be expressed as a linear combination of two neighbouring values $x_{j,k-1}$ and $x_{j,k}$ and only two neighbouring factors greater than zero $s_{j,k-1}$ and $s_{j,k}$ with $x_j = \sum_{k=0}^{n} s_{j,k} x_{j,k}$, whereby $\sum_k s_{j,k} = 1$. Assuming a linear utility function, the same weights for the utility $v_{j,k}$ and the issue value $x_{j,k}$, we can express the utility function with $v_k = \sum_{k=0}^{n} s_{j.k} v_{j,k}$. The weights are left out in the formula in order for the utility function to be linear and the authors argue that the partial utility functions $v_j(x_j)$ for the respective issue in offer can be scaled between 0 and the respective issue j's weight w_j. The entire utility function for the offer is then $u = \sum_j v_j$. The authors additionally embedded a binary variable to restrict the neighbouring factors $s_{j,k}$ and $s_{j,k-1}$ to be greater than zero. In case of discrete issue values, this additional constraint can be omitted. Further optimization models that include predicting the concession path and distance computations for generating counter-offers similar to the opponent's offers using augmented Tschebyscheff norm are detailed in the paper of Vetschera R. et al.[84]

[84] Vetschera R. et al. (2012): pp.83-84

3.3.6 Service-oriented (Heuristic) Negotiation: a bilateral model

In a service-oriented negotiation, an agent (allocation agent) aims to find an agreement with another agent (task agent) about the conditions under which a particular service should be executed on the client's behalf. A service can be, thereby, defined as a problem solving activity with clear definitions of start and end points[85]. Typically, a service can be provided by more than one agent in the negotiation setting, some service offers might be identical, some others might vary in one or several issues. A service-oriented negotiation is very likely a multi-party, multi-issue, one-to-many negotiation in which several characteristics of a service have to be negotiated about and trade-offs among the differently important issues might be necessary to come to an agreement.

This subchapter introduces a negotiation model for service-oriented negotiation as proposed by Faratin P. et al. (1998).

Let $i \epsilon \{a, b\}$ be the negotiation agents and $j \epsilon \{1, \dots, n\}$ the issues of a negotiation object (with n being finite). Let $x_j \in [min_j^i, max_j^i]$ be a value for issue j that is in the *acceptable range* of agent i. Each agent has a scoring function that ranks an issue's value for each agent i in the interval [0,1]: $V_j^i: [min_j^i, max_j^i] \rightarrow [0,1]$. The value of an offer x for an issue j, $V_j^i(x_j)$, can be modelled as a linear function with respect to the issue's acceptable value range:[86]

$$V_{issue1}^{agent\ a}(x_{issue1}) = \frac{x_{issue1} - min_{issue1}^{agent\ a}}{max_{issue1}^{agent\ a} - min_{issue1}^{agent\ a}}$$

$$V_{issue2}^{agent\ a}(x_{issue2}) = 1 - \frac{x_{issue2} - min_{issue2}^{agent\ a}}{max_{issue2}^{agent\ a} - min_{issue2}^{agent\ a}}$$

Those two value formulas are just a computational example for how two issue values could be computed in a linear way; non-linear approaches could be used here too.

The relative importance (weight) of each issue j for agent i is defined as w_j^i with $\sum_{1 \leq j \leq n} w_j^i = 1 \quad \forall i \in \{a, b\}$ (normalized). The offer valuation for agent i can be computed as a linear combination of each issue's value considering their respective importance (weight):

$$V^i(x) = \sum_{1 \leq j \leq n} w_j^i V_j^i(x_j)$$

As an example, let us consider two functional agents, a seller agent s and two buyer agents b_1, b_2 and a multi-issue object under negotiation. The seller agent s wants to delegate a task

[85] Faratin P. et al. (1998): p.160
[86] Faratin P. et al. (1998): pp.163

(e.g. production of a certain good) under a certain price (x_{price}) and at a certain deadline ($x_{prod.days}$). The reservation prices for the agent with respect to the issues is $[min^s_{price}, max^s_{price}]$ = [10, 20] and $[min^s_{prod.days}, max^s_{prod.days}]$ = [1, 3] . We also take into consideration different issue weights assuming that the number of days for completing the task is more important than the price with the weight range $[w^s_{price}, w^s_{prod.days}]$ = [0.3, 0.7]. The agent s now receives from the two buyer agents the offers [18, 3] and [15, 2]. Thus, the value for the price offer from agent b_1 is $(\frac{18-10}{20-10})$ = 0.8 and the value of the price offer from b_2, respectively, is $(\frac{15-10}{20-10})$ = 0.5. Computing the value for the production time offers, b_1's offer yields $1-(\frac{3-1}{3-1})$ = 0 and b_2's offer yields $1-(\frac{2-1}{3-1})$ = 0.5. The *total value of the offered contract* from b_1 is hence $V^{b_1}(x) = 0.3 * 0.8 + 0.7 * 0 = 0.24$ and from b_2 $V^{b_2}(x) = 0.3 * 0.5 + 0.7 * 0.5 = 0.5$. Assuming that the seller agent wants to maximize its utility, it therefore chooses b_2's offer and rejects b_1's offer.[87]

Typically, the search for an agreement can last several rounds and does not necessarily need to end in an agreement. At the end of each round, the proposing agents give their final offer to the receiving agent who weights the best alternative offer and either accepts an offer ending the negotiation process or rejects both offers and asks for new proposals. Depending on the negotiation protocol, the receiving agent could additionally make a counter-offer to facilitate the search for an agreement. In a service-oriented negotiation, counter-offers could be changes in issue weights, so that the proposing agents can find an offer that is in the acceptable range of the receiving agent much faster.

By introducing a time indicator, we can make the service-oriented approach time dependent and, thus, implement several negotiation rounds in the search process. The vector of values a proposing agent a sends to the receiving agent b at time $t_k \; with \; k \in 1 \dots T$ (number of days; $t_1 = 0$) is defined as $x^t_{a \to b}$. The negotiation process remains active if the current proposal is neither accepted nor rejected at the time t' when the evaluation is done, $x^{t'}_{a \leftrightarrow b} \notin \{accept, reject\}$.[88]

When agent a receives an offer at time t from agent b, $x^t_{b \to a}$, it rates the respective offer using its scoring function. If the value of $V^a(x^t_{b \to a})$ is greater than the value of the *counter-offer* agent a could offer at time t', $x^{t'}_{a \to b}$ with $t' > t$, then agent a accepts. Otherwise, agent a

87 Faratin P. et al. (1998): pp.163
88 Faratin P. et al. (1998): pp.165

makes a counter-proposal. The interpretation of the proposing agents' offer at time t' can be defined more formally:[89]

$$I^a(t', x^t_{b\to a}) = \begin{cases} reject & if\ t' > t^a_{max} \\ accept & if\ V^a(x^t_{b\to a}) \geq V^a(x^{t'}_{a\to b}) \\ x^{t'}_{a\to b} & otherwise. \end{cases}$$, where t^a_{max} is a constant representing the time by which agent a wants the negotiation to end, and $x^{t'}_{a\to b}$ is the counter-offer of agent a to agent b at time of evaluation. In case of two proposing agents, the receiving agent a first evaluates both offers and, if both offers are unacceptable, makes a counter-proposal to each agent, which they can accept, reject or counter-propose to.

In order to create counter-offers, agents use tactics that generate new values for each issue of the object under negotiation. Faratin P. et al. (1998) proposed mathematical definitions for a number of tactics that are suitable for service-oriented negotiation, e.g. *time dependent* tactics in which the remaining negotiation time directly influences the next offer's value, *behaviour dependent* tactics in which the next offer depends on the previous offer made by the opponent negotiator (thereby, different imitation computations can be performed), *dynamic deadline* tactics in which an increasing number of agents entering the negotiation rounds increase the time interval available to reach an agreement (i.e. the more proposing agents there are, the lower the pressure for the receiving agent to reach an agreement) or *resource dependent* tactics which generate counter-offers based on how a particular resource (e.g. money) is being used up.[90]

Finding the right negotiation behaviour with respect to the rate of approach to the reservation price of an opponent is crucial to the success of the multi-agent setting. Faratin P. et al. conclude that time dependent tactics are most useful in environments with long deadlines when the approach rate to reservation is generally low, while resource dependent tactics are overall more successful in environments with high communication costs and perform generally better than time dependent negotiation. Their findings also show that a faster rate of approach from the initial offer to the reservation price generally yields worse solutions, because a lower range of offers is being selected until a deadline is reached.[91]

[89] Faratin P. et al. (1998): pp.165-166
[90] Faratin P. et al. (1998): pp.168-170
[91] Faratin P. et al. (1998): pp.171-180

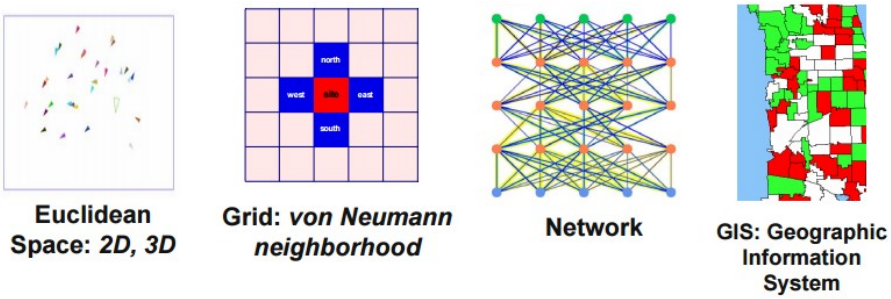
4 Agent-based Optimization

Different from a centralized optimization problem in which a sole central agent coordinates the actions of a group of (non-autonomous) agents in an environment, multi-agent problems are typically distributed (decentralized) problems in which both inter-agent and intra-agent optimization methods are applied. Inter-agent optimization, thereby, refers to a global evaluation of the whole negotiation process. Inter-agent optimization is insofar important because the solution obtained is evaluated according to its stability. An unstable solution can hinder sustainable success of the multi-agent system (because agents are likely to deviate from an agreement) and, thus, a negotiation should not terminate at an unstable solution. Depending on the evaluation approach, a solution stability is evaluated by using some convergence criteria. The inter-agent optimization is typically done by an environmental central agent, such as a mediator or information agent. Intra-agent optimization, on the other hand, is an optimization of proposal generation taking into account each agent's own goals, local constraints and strategy that are private information and as such unknown to all other agents.

Technically, there are two possible ways to model decentralized multi-agent optimization: in the first way, an optimization technique is applied to the problem, which solution is then re-planed by the autonomous agents. In this approach, the optimization problem is considered a *social welfare maximization problem* within a society of agents[92]. In a second approach, each agent employs optimization rules internally, followed by **agent-based modelling** ABM. ABM is about building a multiple agent structure and simulating the local interaction among agents to visualize and quantify the properties of a [complex] system that emerge from collective behaviour. There are different types of agent interactions with its neighbourhood: agents can interact freely (continuous movements) in the system, they can be connected via a grid of local neighbourhoods, or they can interact within a static or dynamic network, or move over Geographical Information Systems (see Figure 7). ABM, thus, connects the agent behaviour with the system architecture and allows the network to be tested under different uncertainty scenarios. In contrast to other modelling approaches, agent-based modelling happens through the agent's perspective.[93]

[92] Kalina P. (2014): p.21
[93] Afshari H. et al. (2014): p.652; Macal C. and North M. (2006): pp.2-24; Bandini S. et al. (2009): online

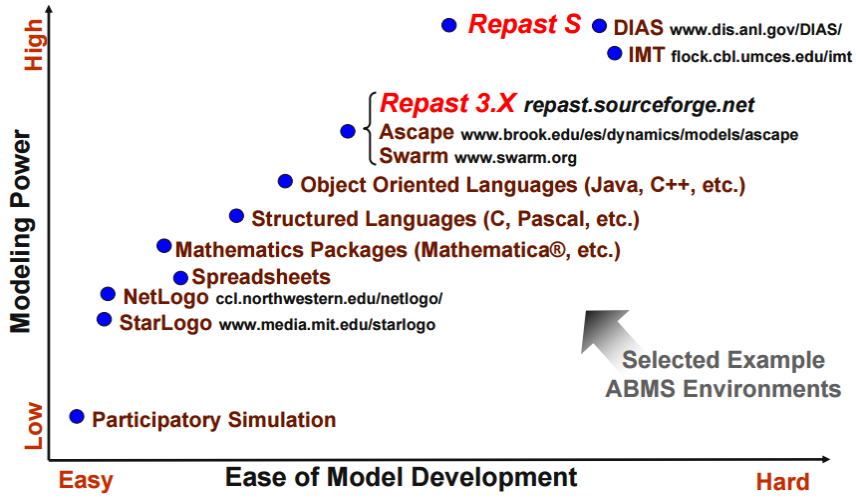
Figure 7: Agent Interaction Topologies of how agents can be connected with their neighbourhood[94]



Agent-based modelling consists of the following components[95]:

(i) a set of autonomous, heterogeneous agents (rules of behaviour)
(ii) defined agent relationships and interactions
(iii) a framework for simulation (typically provided by a toolkit like Repast (Java), NetLogo, MASON, AnyLogic (Java) or Arena Simulation, just to name a few; an abstract of specific tools used in simulation practice are shown in Figure 8 and are structured by model simplicity and scope of simulation)

Figure 8: ABMS toolkits used in practice as illustrated by Macal C. and North M. (2006)



As mentioned before, an alternative for modelling and simulating agent-based networks is to formulate a multi-agent optimization problem as a **social welfare maximization problem**. The specific optimization problem, thereby, turns into a problem of aggregate social preferences and the determination of the most optimal social states (e.g. equilibrium, Pareto optimality). The solving methods can range from methodologies based in social choice

94 Macal C. and North M. (2006): p.10
95 Macal C. and North M. (2006): p.24

theory (e.g. auction, negotiation mechanisms) to traditional optimization techniques (e.g. LP, heuristics).[96]

As already mentioned in the second chapter, the goal of multi-agent optimization is, on the one hand, to find a solution such that multiple, possibly conflicting agents can reach some overall preferred joint state, and, on the other hand, to address the dynamic structure that such systems typically employ. The most widely used formalisms in multi-agent optimization can be summarized as follows and are further described in greater detail in the subchapters below[97]:

- **STRIPS Planning**: the classical planning approach, referred to as STRIPS, is an automated planning formalism that uses predicate logics to generate and explore a state-space by sequentially applying a set of actions on a preceding condition to reach a desirable situation. Starting from an initial condition, each action involves a preceding and a result outcome that are part of a set of efficient plans aimed at reaching a set of goal states.

- **Constraint Programming**: this formal approach in multi-agent optimization is employed when a set of agents with specific constraints must cooperate on a collective problem. The search space is the specific problem being solved, formalized as a set of variables with each of the variables being owned by a single agent, and their domains and relations are expressed as constraints (feasible set). The function of these variables is the objective that describes the system performance. The goal is to find an assignment for the variables such that all constraints are satisfied and the value of the objective function is maximized. Thereby, each agent autonomously decides on the value of its own variables concerning its sub-problem (local constraints), following some protocol that combines the agents' local problem solving with communication approaches among the agents.

- **Belief-Desires-Intentions (BDI) Architecture**: this method combines logics with the mental states of the agents (the beliefs, desires and intentions), whereby this architecture allows the agent to periodically update information (local knowledge) of the other members' mental states (beliefs), its long-term goals (desires) and the temporal states that the agent commits to achieve (intentions).

- **Markov Decision Processes (MDPs)**: MDPs are used to model continuous learning in the network. Thereby, for each state and possible action, a reward is defined that is associated with the action in a specific state. Every strategy (policy) may lead to a different reward and every action performed adds up to the agent's overall utility. The optimization problem is to

[96] Kalina P. (2014): p.22
[97] Kalina P. (2014): p.25-31

select a sequence of actions that maximizes the accumulate reward. Typically, the transition states employed are described as [state-]conditional probabilities.

4.1 STRIPS Planning

STRIPS is a formal language derived from central planning that can be used in a multi-agent planning problem. There are numerous literature examples on problem statements with STRIPS implementation. In this subchapter, the formalisms of STRIPS is defined given an example of a collaborative MA-planning that allows for private information as proposed by Brafman R. I. and Domshlak C. (2013).

Every planning task in MA-STRIPS is given by a quadruple $\langle P, \{A\}_{i=1}^{k}, I, G\rangle$ where k is the number of agents, P is a finite set of atoms (propositions or facts like true or false), A_i is the set of actions that can be performed by an agent i, $I \subseteq P$ is the initial state of the system, and $G \subseteq P$ is the goal condition(s). Additionally, each action $a \epsilon A$ is given by a triplet of subsets of P, $\langle pre(a), add(a), del(a)\rangle$, which define the preconditions, add effects, and delete effects of each action.[98]

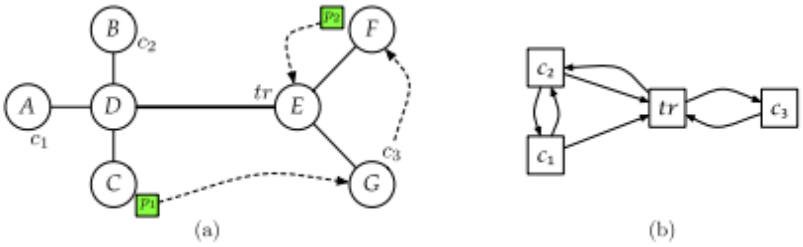
The planning semantics of STRIPS can be described as follows: action a can be applied in a state $s \subseteq P$ iff $pre(a) \subseteq s$. In doing so, the system is transformed to state $\big(s \backslash del(a)\big) \cup add(a)$, denoted as $s[\![a]\!]$. Sequential application of the respectively applicable actions $a_{i=1,..,k}$ starting at state s is denoted as $s[\![\langle a_1, \dots, a_k\rangle]\!]$. Iff $G \subseteq I[\![\langle a_1, \dots, a_k\rangle]\!]$, then the action sequence is considered a ***plan***.[99]

To give an example, let's assume a set of packages that should be moved on a roadmap from an initial location to a target location using a given fleet of vehicles. Each package location on the map and in the vehicles is associated by an *atom*, so is each vehicle location on the map. The possible actions with the respective parameters are $move(vehicle, origin, destination)$, $load(package, vehicle, at-location)$, and $unload(\dots)$. Each vehicle is represented by an agent. Figure 9 (a) illustrates the example using four agents (cars c_1, c_2, c_3 and truck tr). The truck agent can only move between the locations D and E, and the cars can only move outside the connection line D-E. The packages can be loaded and unloaded at any accessible location, except for location D in which packages can only be unloaded. Figure 9 (b) depicts the agent interaction graph (directed graph) that shows the dependencies of such a problem on the agents. For each agent $\varphi_i \epsilon \Phi$, the set of all atoms affected by and/or affecting the actions of an agent is denoted as $P_i =$

[98] Brafman R. I. and Domshlak C. (2013): p.54; Fikes R.E. and Nilsson N.J. (1971): p.197
[99] Brafman R. I. and Domshlak C. (2013): p.54

$\cup_{a\epsilon A_i} pre(a) \cup add(a) \cup del(a)$. Private information or other disjoint information that are not affected by other agents' actions are considered *internal atoms*, referring to the subsets $P_i^{int} = P_i \setminus \cup_{\varphi_i \epsilon \Phi\{\varphi_i\}} P_j$. Atoms not internal to an agent are *public atoms* (i.e. information available to every agent), denoted as $P_i^{pub} = P_i \setminus P_i^{int}$. In the graph (b), all agent locations are private atoms to the respective agents, while (see (a)) $at(p_2, F)$ and $at(p_2, G)$ are package locations only private to agent c_3. An agent's actions can be defined by its internal (here, *move*) and public (here, *load* and *unload*) actions in the partition set $A_i = A_i^{int} \cup A_i^{pub}$.[100]

Figure 9: example as depicted from Brafman R. I. and Domshlak C. (2013), based on the transport planning task from Helmert M. (2009)



A propositional representation of the MA-STRIPS transportation-planning task as shown in the interaction graph in Figure 9 is derived from Helmert M. (2009). In Figure 10 the states are represented as sets of atomic propositions, and operators are given in terms of preconditions (which proposition must be true for the operator to be applicable), true-propositions (which atoms the operator makes true; ADD effects), and false-propositions (which atoms it makes false; DEL[ETE] effects).[101]

[100] Brafman R. I. and Domshlak C. (2013): pp.55-56
[101] Helmert M. (2009): pp.504-505

Figure 10: The STRIPS representation of the transportation planning task using binary variables (atoms ∈ $\{\mathbf{0} = \boldsymbol{false}, \mathbf{1} = \boldsymbol{true}\}$) as illustrated by Helmert M. (2009) and as presented in the interaction graph in Figure 9. The notation is read as follows: $\boldsymbol{at} - \boldsymbol{p1} - \boldsymbol{a}$ states that the first package (p1) is *at* location *a*, $\boldsymbol{in} - \boldsymbol{p2} - \boldsymbol{t}$ means that the second package (p2) is currently *in*side the truck *t*

Propositions:
at-p1-a, at-p1-b, at-p1-c, at-p1-d, at-p1-e, at-p1-f, at-p1-g,
at-p2-a, at-p2-b, at-p2-c, at-p2-d, at-p2-e, at-p2-f, at-p2-g,
at-c1-a, at-c1-b, at-c1-c, at-c1-d,
at-c2-a, at-c2-b, at-c2-c, at-c2-d,
at-c3-e, at-c3-f, at-c3-g,
at-t-d, at-t-e,
in-p1-c1, in-p1-c2, in-p1-c3, in-p1-t,
in-p2-c1, in-p2-c2, in-p2-c3, in-p2-t
Init:
at-p1-c, at-p2-f, at-c1-a, at-c2-b, at-c3-g, at-t-e
Goal:
at-p1-g, at-p2-e
Operator drive-c1-a-d:
PRE: at-c1-a ADD: at-c1-d DEL: at-c1-a
Operator drive-c1-b-d:
PRE: at-c1-b ADD: at-c1-d DEL: at-c1-b
Operator drive-c1-c-d:
PRE: at-c1-c ADD: at-c1-d DEL: at-c1-c
...
Operator load-c1-p1-a:
PRE: at-c1-a, at-p1-a ADD: in-p1-c1 DEL: at-p1-a
Operator load-c1-p1-b:
PRE: at-c1-b, at-p1-b ADD: in-p1-c1 DEL: at-p1-b
Operator load-c1-p1-c:
PRE: at-c1-c, at-p1-c ADD: in-p1-c1 DEL: at-p1-c
...

So far, the STRIPS planning problem has been introduced in a single-agent framework. Brafman and Domshlak further proposed to consider the public actions as "coordination" points among the agents, in between which the agents must formulate their internal actions so that the goal outcome is reached. For instance, if the truck agent t must load package p_1 in location D, than the package should be somehow brought to D in time by the car agents c_1, c_2 and cannot be picked up from location D before t. Creating a globally consistent plan is a combination of constraint satisfaction (CSP) and planning, consisting of a **coordination constraint** (ensuring that agent performs in favour of the public actions) and an **internal planning constraint** (the agent ensures that the internal preconditions of the public actions can be achieved). The coordination constraint states, less formally, that if $p \in P_i^{pub}$ is a public precondition of a, then someone must supply p before t (i.e. $p \in add(a')$ and $t' < t$) and no one destroys p between t' and t (i.e. $p \in del(a'')$ and $t' \leq t'' \leq t$). The internal planning constraint ensures that an assignment with action commitments is solvable. According to those constraints, a possible solution plan to the transportation planning problem from above, assuming that car c_1 must perform the public actions $load(p_1, c_1, C)$ and $unload(p_1, c_1, D)$, would be

$\langle move(c_1, A, D), move(c_1, D, C), load(p_1, c_1, C)|_{int}, move(c_1, C, D), unload(p_1, c_1, D)|_{int} \rangle$.[102]

[102] Brafman R. I. and Domshlak C. (2013): pp.57-58

4.2 Constraint Programming: An Introduction

In distributed problem solving (DPS), a task is distributed among multiple agents whereby each decision variable is owned by a different agent and all agents need to ***cooperate*** to find a global variable assignment that meets the model constraints. Most typically, such agents are considered to be benevolent and they work together to solve a problem of common interest. A DPS system is a ***distributed constraint satisfaction problem*** (CSP) which aims to assign values of individual agents to a [global/central] decision variable in a way that is consistent with all the model constraints. Thereby, each variable has a domain and constraints on its values that all agents must adhere to. CSP is, however, not only used in clearly cooperative networks. Considering interest-based negotiation approaches in which a stable global solution can only be found if the agents concede to some degree to their opponents' interests, constraint programming is often implemented as part of the optimization problem. A typical example of a DPS situation are *sensor networks* which consist of multiple processing units, whereby each unit has limited processing power, limited power supply and local sensor capabilities. Sensor networks are used, for instance, to monitor environmental systems like humidity, temperature and pressure in an office, and they aim to provide global service while being limited to monitoring its local area and, thus, require cooperation among the individual processing units. The algorithm used by the individual sensors must be, therefore, designed in a way that the neighbouring sensors can effectively communicate and the centre can put together the individual sensors' inputs to achieve a global picture (output).[103]

Basic algorithms that can be implemented in a DPS situation are *domain-pruning* algorithms and *heuristic search* algorithms. In the former algorithms, agents communicate with their neighbouring agents in order to eliminate values from their domain. That can happen through a filtering algorithm in which agents communicate their domain to their neighbours and eliminate the values from their domain that are inconsistent with the values received from the neighbours until there is only one value left in each domain or no conclusive solution can be found. In the heuristic search algorithm, each agent searches the space of possible assignments that are consistent with their neighbours' domain values and informs the neighbours in case a value in the domain is consistent. Thereby, the variables x_i are ordered and the set values assigned to those ordered variables v_i are being compared recursively to a domain value v_j (the rules of a recursive, backtracking approach are shown in Figure 11). If the domain value is consistent with the set of values assigned to the variables, then the process starts for a new variable, until all variables have one solution assigned to them. Those straightforward algorithms can further be expanded to include more complex features.

[103] Shoham Y. and Leyton-Brown K. (2009): pp.1-3

Shoham Y. and Leyton-Brown K. (2009) introduce a combination and expansion of those two DPS algorithms, the asynchronous backtracking with dynamic link, which uses characteristics of both algorithms such as global total ordering on agents and an order-based message-passing protocol to distribute their local information to meet global constraints.[104]

Figure 11: recursive search through the space of possible assignments using backtracking. Backtracking can be implemented in different ways like, for instance, by removing the assignments in reverse chronological order[105]

procedure ChooseValue(x_i, $\{v_1, v_2, \ldots, v_{i-1}\}$)
$v_i \leftarrow$ value from the domain of x_i that is consistent with $\{v_1, v_2, \ldots, v_{i-1}\}$
if *no such value exists* **then**
  backtrack[1]
else if $i = n$ **then**
  stop
else
  ChooseValue(x_{i+1}, $\{v_1, v_2, \ldots, v_i\}$)

As already mentioned before, a main prerequisite and advantage of MAS is the ability to divide the problem into smaller subtasks (sub-problems). Assigning control of subtasks to different agents makes the problem easier and faster to be solved. The sub-problem can be, thereby, solved only once and stored for future interaction. The method of dividing a complex problem into simpler sub-problems to be solved, whereby each sub-problem's solution is being stored, is referred to as *dynamic programming* or dynamic optimization. Dynamic programming techniques and reinforcement learning algorithms as related area are often used in theoretical implementations of MAS in combination with CSPs. The lack of practical evidence does not allow for the analysis of dynamic programming techniques in real-life applications, but following the flexibility, learning ability and usefulness of dynamic programming techniques as well as its continuous research for MAS applications[106], it appears to be a suitable communication approach in a multi-agent setting.

4.3 BDI Architecture

A Belief-Desire-Intention (BDI) architecture has been widely used to design agents that can perform complex reasoning. One widely used BDI framework is the *Procedural Reasoning System* that consists of three steps: (1) the agent collects information about its environment (perception phase; belief formation), (2) a central interpreter helps the agent to select an appropriate action for its goals (planning phase; desire formation), and (3) the agent acts or reacts (execution phase of an intention)[107]. In recent years, an emotion-transition component

[104] Shoham Y. and Leyton-Brown K. (2009): pp.3-5
[105] Shoham Y. and Leyton-Brown K. (2009): p.9
[106] Yokoo M. and Suzuki K. (2002); Csáji B. C. and L. Monostori (2005); Valenti J. (2007); Shoham Y. and Leyton-Brown K. (2009); Wu F. et al. (2010)
[107] Caillou P. et al. (2015): p.3

was added to such agents making artificial intelligent agents capable of expressing emotions during their interaction[108].

A BDI architecture can be described using the following example of a simple task allocation problem with three types of agents, several task performing agents (TAs), a task allocation agent (AA), a resource agent (RA), and homogeneous tasks. The TA has the general *desire* to find a task. In order to do so, the TA looks for an AA that has a task to assign (it *plan*s to find a task). When it finds an AA with who the skills and capabilities match the task's requirements, the TA forms a new belief about finding the necessary resources to fulfil the task. Its new *desire* is to find an RA that offers the resources it needs at a reasonable condition (e.g. considering budgetary or timely constraints). When the TA finds an RA with available resources, it selects the new intention of requesting the resources needed. The new *plan* for the intention could be, for instance, to reallocate an existing set of tasks so that it can be assigned the resources, or the agent plans to persuade the RA by outbidding the other TAs.[109] An example of a BDI framework and simulation of a land-use model is provided by Caillou P. et al. (2015) which can be adapted to work for facility location models as well.

A BDI agent consists of a **knowledge set**, that is the ***beliefs*** (describable as a binary predicate such as *availableTask(true,(atTimeResource :: (start=x, duration=y)))*), ***desires*** (including priority values that are used to select a new intention and are formulated similarly to beliefs), ***intensions***, and a **behaviour set**, that is the set of plans that guides the behaviour to reach a specific desire (e.g. *goToPosition* or *selectPriority*).[110]

4.4 Markov Decision Processes

Markov Decision Processes (MDPs) are employed to solve sequential decision making problems under uncertainty, with the use of transition probabilities. Using multi-agent MDPs (MMDPs), however, the complexity level of state-transition descriptions can become quite problematic for which reason a sophisticated approach to multi-agent MDPs (MMDPs) is neither described here, nor further dealt with in the subsequent contents. For the sake of completeness, MMDPs are formulated in their simplest setting, that is, we assume that the probabilities of transition are known and that individual agents in the multi-agent framework have all the same payoff function. Under unknown probabilities, some reinforcement learning algorithms can be additionally employed to transition simulations.

---

[108] Puica M.-A. and Florea A.-M. (2013): pp.1ff
[109] based on the Chopper-Fire example of Caillou P. et al. (2015): p.4
[110] Caillou P. et al. (2015): p.5

In a single-agent setting, MDPs are defined as the tuple $\langle S, A, Pr, R\rangle$ where S is the finite set of states, A is the finite set of actions, $Pr: S \times A \times S \mapsto [0,1]$ denotes a transition probability function of moving from state $s \epsilon S$ to state $s' \in S$ when action $a \in A$ is performed, and $R: S \mapsto \mathbb{R}$ is the reward function.[111]

Given a policy (i.e. a probability distribution over the agent's action for any state s) $\pi(s): S \mapsto A$ and a starting state s_o, we can describe the expected value (utility; reward) function as a linear function:

$V^{\pi}(s) = R(s) + \beta * \sum_{s' \epsilon S} Pr\big(s, \pi(s), s'\big) V^{\pi}(s'), \forall s \in S$

$V^{\pi}(s_o)$ describes the utility of the starting state. The objective is to find a policy that maximizes the value function.[112]

Mathematically, $Pr\big(s, \pi(s), s'\big)$ is described as the conditional probability function that state s' is reached after taking action π in a previous state s, denoted as $P(s'|s, \pi(s))$.[113]

In a multi-agent setting, we typically have multiple agents, each with their own set of actions and tasks to be solved. We assume here that the problem is fully cooperative, hence, all agents aim to maximize the utility of a system state (joint utility function) rather than their own goals. In a MMDP, an action that is performed at any state consists of the individual action components done by the agents involved. Boutilier C. (1996) described MMDPs as an *n-person stochastic [repeated] game in which the payoff function is the same for all agents*. Because of the joint utility function, the collection of agents can be considered a single agent, which results in the above formulation of a *joint MDP.* That is, MMDP can be described as a tuple $\langle S, \alpha, \{A_i\}_{i \epsilon \alpha}, Pr, R\rangle$ where S is the finite set of states, α is the finite set of agents, A_i is the finite set of actions available to agent i, $Pr: S \times A_1 \times \ldots \times A_n \times S \mapsto [0,1]$ is a transition function that describes the probability of a transition from state s to state s' given a joint action $a \in A = \times_{i \in \alpha} A_i$, and $R(s): S \mapsto \mathbb{R}$ is the expected reward function received by the agents at state s. A stationary individual policy of agent i is denoted as $\pi_i(s): S \mapsto A_i$, the value function as described above. The joint policy is denoted as $\{\pi_i\}_{i \epsilon \alpha}$ that is the set of individual policies mapped into joint actions. In case a state s is *weakly dependent* for agent i, there is more than one individual optimal action choice for agent i to take at state s.[114]

---

111 Chadès I. and Bouteiller B. (2005): p.1595
112 Chadès I. and Bouteiller B. (2005): p.1595
113 Guestrin C. et al. (2001): p.4
114 Boutilier C. (1996): pp.4-5

5 Literature Research: MAS in Logistics

This chapter describes various literature frameworks of multi-agent automation techniques for different logistic applications. Thereby, the focus lies on the negotiation models implemented and the agent optimization used. So far, the literature implementations are mostly theoretical and a comparison between the decentralized approach implemented and a state-of-the-art centralized counterpart is, if existent, purely descriptive and based on time performance comparison, rather than on the solution quality. In some cases, a significant solution comparison might even be barely possible, as decentralized frameworks yield local optima under criteria that are difficult to be objectively replicated for a centralized system. The existing comparison points out the time complexity of multi-agent settings, which typically place the solution quality at a trade-off for a faster computation (i.e. the faster the system must be, the worse the solution becomes). The benefit of MAS lies, as altogether pointed out by the literature research, in the recognition of the environmental dynamic structures (i.e. autonomous agents react faster than a sole central agent, both in an open environment and in case of changes in closed structures). At the same time, MAS facilitates the implementation of distributed systems that typically serve a self-purpose while profiting the superordinate structure.

5.1 MAS in general logistic frameworks

A prominent general framework by Chen Y. et al. (1999) uses the Contract Net as introduced in chapter 2.3.2 by R.G. Smith (1980) for supply chain management using two types of functional agents, the *information agent* and the *role agent*. The functional agents transfer information, share knowledge and negotiate with each other using an extension of the modelling language P/T net. The predefined (fixed) number of information agents interact with the entire supply chain system, filtering relevant system information for a potential negotiation process and informing the role agents about the potential negotiation partners and their preferences. The role agent is related to a certain good and implements a special role of the supply chain. Despite the information agents, other system components are not fixed and by leaving or entering the supply chain, the functional agents have to notify an information agent. When a buyer agent looks for a cooperation partner, it communicates with the information agent about the potential sellers and then negotiate with the seller agents directly to find the most suitable seller partner concerning the respective constraints of the order. The negotiation process uses a limited number of negotiation protocols with predefined agent responses. The negotiation can take place between two functional agents (pair-wise negotiation protocol) and third party involvement (auction). Table 1 gives a summary of the possible actions a functional agent can take when a certain message from

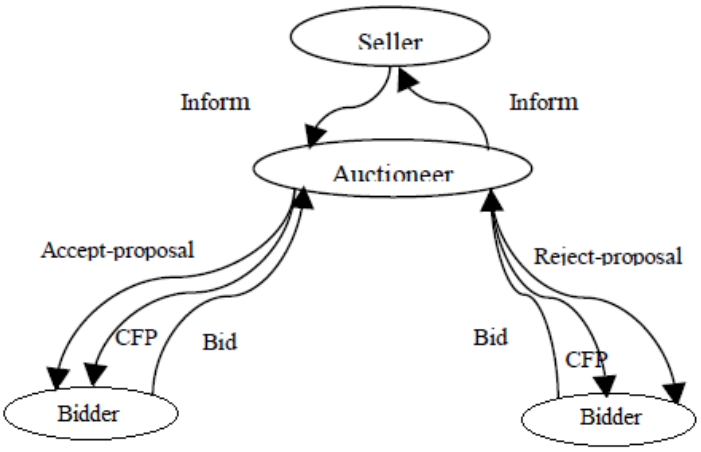
another agent comes in. In a third party involvement, the negotiation protocol follows an *auction strategy* including an additional BID-message through which INFORM-messages between the seller agent and the auctioneer are being transferred (the seller agent informs the auctioneer of what it wants to sell and the auctioneer informs the seller agent of the auction result). Figure 12 illustrates the third party negotiation scenario (auction) which is an extension of the pair-wise negotiation protocol using an additional intermediary agent (auctioneer) who carries out the negotiation with all the involved bidders (buyer agents).

There are several drawbacks to this descriptive framework (*note:* no computation was provided), such as the assumed closed environment and complete information of the negotiation process.

Table 1: pair-wise negotiation strategies and expected response as illustrated in Chen Y. et al. (1999)[115]

	Performatives followed
accept-proposal	terminate | NONE;
CFP	proposal | terminate | NONE;
proposal	accept-proposal | reject-proposal | terminate | NONE;
reject-proposal	terminate | NONE;
terminate	NONE.

Figure 12: auction strategy proposed by Chen Y. et al. (1999) using pair-wise negotiation rules in accordance to the FIPA ACL



Auctions in some form have been widely used in distributed problem solving methods in MAS literature. A practically relevant example of a decentralized task allocation method using auction-based negotiation in the logistics domain was provided by Hoogendoorn M. et al. (2007). According to their architecture, agents distribute tasks via first-price reverse

115 Chen Y. et al. (1999): p.3

combinatorial auctions with time windows and precedence constraints in a setting in which limited information are shared. Their framework considers two bidding strategies: in the first strategy, the agents only bid on tasks that are closely related to their current tasks, while in the second strategy they bid randomly. The empirical evaluation of their proposed system using the Iterative Deepening algorithm for bid evaluation showed a deviation from optimum between 1,008 (for 1 task) and 1,038 (for 10 tasks) for the first bidding strategy, whereby the solution deviation did not significantly increase for a high number of tasks to be allocated in the system. Their findings showed that distance measures in the bid evaluation is crucial to creating an auction-based MAS framework that is close to optimum.

When it comes to managing the efficiency of workflows, defect analysis becomes a crucial part of logistics. Kaplanoglu V. et al. presented, 2014, a theoretical approach to automatically dealing with breakdowns of automated guided vehicles AGV (machinery) in manufacturing systems (production logistics), a framework that could likely be applied under certain adaptations to other areas of logistics. Again, some form of auction is used in the framework proposed. The goal of the authors' research was to offer an instantaneous solution to an AGV breakdown while the system is operating. The authors defined several agents in the AGV system that all interact in case a machine breaks down: the AGV resource agent, the AGV scheduler agents and the operation agent. When a new order comes in, the manager agent assigns the order to an operation agent. The operation agent then looks for proposals from the scheduler agent to process the new order. When a machine agent is found, the order agent calls for a proposal to the scheduler agent to be assigned to the machine. The scheduler agent, then, bargains with the resource agent to allocate resources to the order. While a resource agent is operating, the machine can break down. The working status of the resource agent would then change from "in working condition" to "broken down" and send the breakdown information to the scheduler agent who informs the operation agents about the breakdown and negotiates with them a new schedule and machine relocation. In the decision-making process, the scheduler agent aims to find an operation that has the minimum earliest loading time of operation i (ELT_i) following the formula:

$$ELT_{i=1\dots n} = \begin{cases} EFT + \Delta t(NL, PCP_{i=1\dots n} & EFT > EPT_{i=1\dots n} \\ EFT + \max\{\Delta t(NL, PCP_{i=1\dots n}), (EPT_{i=1\dots n} - EFT)\} & , EFT\ otherwise \end{cases}) \quad , \quad \text{whereby}$$

EFT is the earliest free time (idle), NL is the next location of the AGV resource agent, Δt is the required time between the broken agent and the new agent, $EPT_{i=1\dots n}$ is the earliest pickup time of the operation i, and $PCP_{i=1\dots n}$ is the pickup point of operation i.

Then the scheduler agent selects the operation for which $ELT_s = min\{ELT_i\}\ i = 1 \dots n$ and submits a proposal to the operation agent(s) s. If there is a match between the current operation and the scheduler agent's proposal, the order is taken over.

Further literature concepts expanded the bidding framework by implementing a learning mechanism from rejected and accepted cooperation relationships to speed up future agreement attempts and evaluation mechanisms of the proposed cooperation partners.

Such an example of a multi-agent learning mechanism is the *adaptive agent relationship* protocol in dynamic environments by Berndt J. and Herzog O. (2011). The adaptive agent relationship aims at dynamically establishing and coordinating social structures in a MAS based on observable behaviour, thus, without prior information about agent properties and capabilities. The dynamics of the relationship start with an agent memorizing former interactions of the observed others' reactions. Each memory entry $mem_i = \langle s_i, r_i \rangle$ thereby consists of a tuple of messages s from the set S of all possible messages an agent can sent, and r as the agent's observed response to the first one from the set R of all possible responses to S. At time t, when the agent receives message r_t, it stores it in the vector $MEM = (mem_1, \dots, mem_n)$ with its own message s_{t-1}. In case memory is being exceeded, the oldest entries are being deleted. Based on the entries of the memory, an agent can estimate the outcome of possible activities based on the observed behaviour. The memory access function to estimate the probability of a specific response can be, thereby, computed as:

$$lookup(s,r) = \frac{l(s,r)}{\sum_{r' \in resp(s)} l(s,r')} \qquad [1]$$

with $l(s,r) = \frac{c}{|resp(s)|} + \sum_{i=1}^{n} \frac{n+1-i}{n} * \begin{cases} 1 & if\ \langle s,r \rangle = mem_i \\ 0 & else \end{cases}$

The mapping $resp: S \rightarrow \mathrm{P}(R)$ denotes the set of all valid responses to a given message ($resp(s) = \{r \epsilon R | val(s,r)\}$), $c \epsilon \mathbb{R}$ is a constant value that allows an agent to consider unobserved interactions while ruling out impossible message pairs, and the memory access function $lookup: SxR \rightarrow [0,1]$ is normalized to give estimated probability of an observed message pair.

By combining [1] with a utility function for each of the responses in the memory vector given an agent's objective, an agent is able to estimate the expected outcome given a certain interaction process. However, because the above formulation only considers exact matches of memory entries, it is necessary to expand the computation to include similarities between interactions, like differences in one negotiation issue of the multi-issue object. Thus, the authors defined the selection value v for a message s and its response r as a function $v: SxR \rightarrow [0,1]$ that considers the (average) neighborhood of messages $s' \in S$ which are semantically similar to s with their responses $r' \in R$ being equivalent to r:

$$v(s,r) = \sum_{s' \in nb(s)} \frac{lookup(s',r')}{|nb(s)|} \quad [2]$$

with $r \equiv r' \in resp(s')$ and $nb(s)$ denoting the set of neighbouring messages to s:

$$nb(s) = \{s\} \cup \begin{cases} \emptyset & if\ \exists r \in R: \langle s,r \rangle \in MEM \\ s': min_{s' \neq s} \Delta(s,s') & else \end{cases}$$

As before, the selection value is combined with a utility function in order to estimate the outcome of possible interactions.

In a multi-agent setting, the message selection must happen simultaneously while interacting with multiple agents. Thus, the message selection is a bundled message being evaluated under the joint utility for a given design objective. The maximum expected outcome based on a set of messages is defined in the framework as:

$$select(S) = max_{X \subseteq S}(\sum_{s \in X} \sum_{r \in resp(s)} v(s,r) * util(s,r|X)) \quad [3]$$

with any conditional utility function $util: SxRxP(S) \rightarrow \mathbb{R}$ denoting the expected value for a message pair given a set of simultaneously sent bundled messages X.

The empirical results of their adaptive agent relationship protocol demonstrated an overall significantly better performance to a predefined network configuration and a *random* message selection method based on optimal fulfilment rates in a dynamic environment. As demand changed, the quality of performance of the negotiation decreased to the level of the random selection method but could, contrary to a pre-defined negotiation pattern, significantly re-establish its utilization rate. Without environmental changes, the results even approximated optimal performance rates based on an order and delivery example. The authors concluded that implementing an adaptive agent relationship protocol could create a negotiation framework in an uncertain environment that is capable of re-establishing an efficient coordination among the negotiating agents.

5.2 MAS in Transportation Logistics and Routing Problems

In a paper from Davidsson P. et al. (2005), the existing research on agent-based approaches to transportation and traffic management was analysed through the years 1992 until year of publication, 2005. Their findings showed a relatively large research interest in agent-based systems in this domain, most likely because most transportation applications fulfil the characteristics below as defined by Parunak (1999) to be important criteria for an ideal application of agent technology:

- the entities involved contain well-defined set of state variables and are distinct from other involved entities and clearly identifiable
- the application can work decentralized
- the structure of the application is dynamic and changeable
- all information about the application is not available at design phase
- the system is strongly complex with a large number of different behaviours and dynamic interactions

According to Davidsson et al., for any application, which is strongly centralized, static and well-structured, agent technology would not offer any value added but unnecessary complexity. The paper structured the agent-based technology research based on the three parts: *domain* (transport, traffic, and terminal), *mode of transportation* (road, rail, air, water, and pipeline), and the *time horizon* considered (operational, tactical, strategic as levels of decision-making). In the *transport-domain*, a good is being moved from A to B via either air, water, road or rail. Typical transport problem areas include route planning, scheduling or fleet management. The *traffic-domain* implies a flow of the different transports within a network (e.g. train traffic) and typical problem areas include the scheduling of traffic flow, railway slot allocation and traffic management. The *terminal-domain* is a fixed place where the transport is handled. Typical application areas are resource allocation and scheduling of cranes, for instance. Figure 13 illustrates the focus of applications in the papers surveyed by the authors. The graph shows that the modes of transportation applied in MAS were dominated by air, road and intermodal (i.e. different modes of transportation used). The widest area of interest lied in the traffic domain such as allocating slots for the railway network (i.e. timetabling), allocating transport tasks to vehicles, flights in air traffic (i.e. aircrafts choose their speed and path in real-time), and traffic management to avoid congestion of roads. Figure 14 further illustrates the number of literature in which MAS frameworks were introduced in a varying degree of decentralization and automation. According to the analyses done by the authors, the main advantages for the use of MAS as mentioned by the papers surveyed is the distribution of control, the ability to cope with noisy data and to model complex, dynamic problems. However, only half of the projects surveyed used a *dynamic* MAS structure and the main application area of MAS was still in decision-support rather than a decentralized automation system. The authors also found in their selection of research papers a lack of evaluation comparison between the current or alternative approaches and the agent-based application. Two third of the approaches surveyed had a strictly theoretical implementation of the agent-based system without further evaluation comparison. Only 5 out of 56 papers included a quantitative comparison with the use of simulation.

Figure 13: the distribution of domain and transport mode, according to the extensive survey on MAS applications in transport logistics done from 1992-2005 by Davidsson P. et al. (2005)

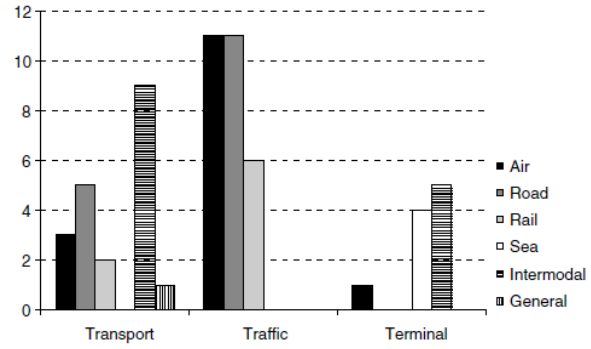
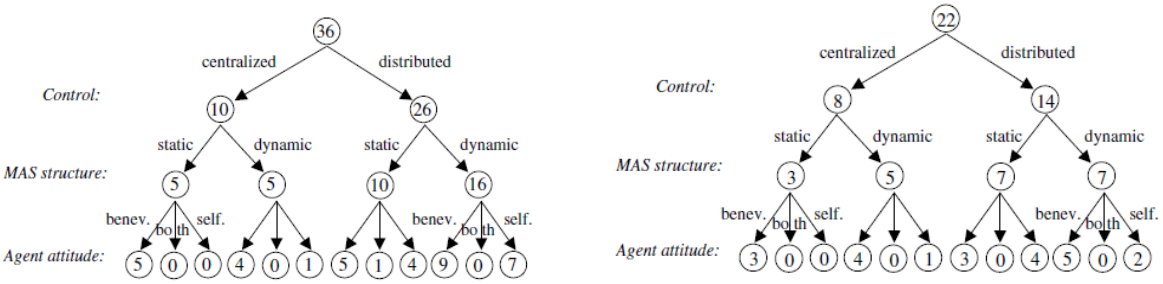


Figure 14: MAS characteristics in transport logistics' research papers according to Davidsson P. et al. (2005). Left: the number of approaches on decision-support systems; right: the number of approaches on automation systems



Half a decade later, newer MAS applications in transport logistics become more sophisticated and focused on a quantitative evaluation. One example framework was proposed by Robu V. et al. in 2011, an auction-based multi-agent platform for loads allocation in transportation logistics that allowed the comparison and evaluation of automated trading strategies and was used as a prototype for VOS Logistics. In their framework, one depot serves several destinations in one foreign country and all orders have to be delivered by a certain deadline. The platform then lists orders (loads) sequentially based on their lead-time (difference in days between the time of placing the order online and the delivery deadline) and auctions loads to a predefined number of different shipping companies. In the auction protocol, when an order appears in the platform, all bidders (carrier-companies) are informed and can make offers. If $bid_{carrier_i} > reservation\ price_{auctioneer}$ for an i, the offer is rejected and the carrier can bid again. If a bid is accepted, all carriers are informed and a new load offer is being auctioned. While the auction process is straightforward, the actual difficulty lies in how well the carriers form profitable bundles of loads. This requires all carriers to compute a cost model for each combination of loads they aim to win in the auction, containing both fixed costs and variable costs (proportional to the distance travelled). Additionally, the carrier planning must take into account several constraints such as, for example, capacity constraints for the mode of transportation used and any legal

constraints (max. working hours for the drivers). Thus, the carriers participating in the auction process at time t must each compute an optimal route (the given framework used insertion heuristic) prior to offering a bidding price for an auctioned load in order to derive a competitive selling price. This computations are repeated for the other loads being auctioned at time $t+1, t+2,$ ... considering the new bundle of loads and the new intermediary stops on the currently planned route under the set of given constraints. The bidding platform was not tested on empirical data, but transportation planners of VOS Logistics were to bid against the intelligent agents and the authors concluded with the following findings: first, the bidding support was considered to be helpful for transportation planners and could at times outperform the planners' decision; and second, using automated bidding agents has the potential to stabilize market prices and influence their convergence to realistic levels.

The work of Feng F. et al. (2015) used the concept of dynamic cooperative relationships in a large-scale automating transport plan protocol by building three modules inside the software agents used that assist dynamic negotiation behaviours. The agents are thereby defined by their task-divisions and include: the barge agent (BA) that negotiates with the terminal agent (TA) to find available time slots and cooperate with the decision-making (DMA) to assess schedule performance; the TA that receives and processes time requests from different BAs and looks for time slots; the DMA analyses the feasibility of the schedule and, if needed, initiates a new negotiation round; and the supervisor agent (SA) monitors system status and agents behaviour. The three modules for cooperative agent relationships are the *computing engine* that performs relevant task computations (e.g. task sequence), the *negotiation engine* that translates the incoming messages, and the *intelligent engine* that aims to make the agents adaptive to their environment, similar to the functionality of Robu et al.'s framework on adaptive agent relationships mentioned before. They implemented their framework on a hinterland barge transport plan, an example for a large-scale complex logistics system characterized by low communication and lack of coordination. Traditionally, a barge is scheduled in advanced to a free terminal and remains fixed throughout the plan. Thus, the traditional approach follows a strictly time-ordered sequence. The example data of the authors included six barges for containers to load and unload in eight terminals following a normal distribution. Their findings showed that the intelligent agent system found the best route with less waiting time (by 65% lower) and better turnaround time (by 30% faster) than the traditional planning approach.

5.3 MAS in Task Allocation and Scheduling Problems

Task allocation and scheduling problems deal with the division of a problem into precedence-constrained sub-problems and their allocation to other agents so that the completion time is

minimized. The amount of time invested typically involves the execution delay of a service, the job release times with respect to any precedence jobs as well as communication delays between agents. Much of the literature work presented for task allocation and scheduling problems in MAS is based on multiprocessor scheduling theory that deals with optimally allocating a set of agents (or machines) to complete a set of tasks over time. The majority of literature on such problem settings involve heuristics that are problem specific and, thus, do not work best for every problem instance.[116]

Nouyan S. et al. (2005) proposed a classification of dynamic task allocation and scheduling problems into the two subsets of *market-based approach* and *insect-based approach*. The market-based approach is typically used for coordinating asynchronous scheduling operations under imperfect information and the decision process is based on a decentralized bidding mechanism where agents (e.g. processors) bid for a task or a resource and the highest bidder wins. The bid is computed based on the agent's ability to solve a task or on the availability of a resource. The original algorithm for market-based scheduling was developed by R. Morley in 1996. In the example of the authors, several painting booths (agents) are autonomously bidding for painting a truck (tasks). If the queue of an agent is full (i.e. no capacity available), the agent does not participate in the bidding process. Any other agent participating in the bidding process bids a value given by:

$$B_k(j) = \frac{P * w_i * (1 + C * c_{i,k})}{\Delta T_k(j)^L}$$

where w_i is the priority of the task i (truck), $c_{i,k}$ is a binary value depending on a required painting setup (i.e. switching colours for the truck), ΔT is the time until task i starts to be painted in booth (agent) k, and P, C, L are parameters that weight each component's importance, w_i, $c_{i,k}$, and ΔT. Additionally, ΔT is computed using the following equation:

$$\Delta T_k(j) = q_k \cdot t_{proc} + n_k^{setup} \cdot t_{setup} + t_k^{working}$$

that is, the sum of the paint time, t_{proc}, with respect to the number of trucks q_k in a booth k's queue, total setup, t_{setup}, with regards to the total number of times the next truck in line requires a different colour than the previous truck, and the time required to finish the currently painted truck, $t_k^{working}$. The bids are compared and the respective truck goes to the queue of the highest bidding booth.

In the insect-based approach, Nouyan S. et al. categorized task allocation that focuses on a smart division of labour using a response threshold model by Theraulaz G. et al. (1998) that

116 Tompkins M. F. (2003): p. 20

helps to divide the tasks among the agents. In their model, one threshold is given to each type of task, whereby the threshold value represents the level of specialization in that task. Based on the threshold value and a stimulus to perform a task, an agent will accept a task or withdraw from it.[117] Given this response threshold model, several algorithms were developed in task allocation and scheduling. A framework that incorporates the above example with the painting booths was proposed by Campos M. et al. (2000). According to the authors, each painting booth (agent) k has a threshold value θ_{k,c_j} for each colour c_j. A truck j has a stimulus effect of s_{c_j} for each colour in the system. The probability of booth k to get task j is given by:

$$P_k\left(s_{c_j}, \theta_{k,c_j}\right) = \frac{s_{c_j}^2}{s_{c_j}^2 + \alpha \cdot \theta_{k,c_j}^2 + \Delta T_k^{2\beta}(j)}$$

where α, β are parameters that weight the relative importance of their respective terms, and ΔT_k is the same as in the market-based example. The values of P_k are then compared and the respective truck is assigned to the booth with the largest value. After a truck is assigned, the threshold value is updated for all of the booths, whereby θ_{k,c_j} decreases by an amount ε. The authors' findings showed promising results but recognized the issue that with changing processing times the relative importance for each parameter must change too which makes the problem difficult to implement.

Despite the transportation system, auction-based (market-based) frameworks are quite popular in task scheduling of cooperative agents too. An example for cooperative agents in a market-based framework was given by Zlot R. and Stentz A. (2005). In their framework proposed, they assumed complex tasks can be solved by one or more robots on the team (*distributed system*). Thereby, the task can be divided into sub-tasks (*note*: complex tasks can be represented graphically using a task tree whereby the tree nodes are sub-tasks that the complex task can be divided into) and the agents bid on the sub-task or on the entire tree. Their approach is considered to be an extension of TraderBots, in which agents trade tasks via auctions and each agent maintains a schedule of the tasks won and evaluates new tasks by computing the marginal costs of adding them to the existing schedule. Figure 15 illustrates the clearing algorithm used in auctions held by individual robots (agents) using task trees on each task under auction. In this task tree auction proposed by the authors, the goal of the auctioneer is to find a task allocation that maximizes profits (minimizing team costs). When a new complex task enters the system, the OpTrader agent divides the task into sub-tasks (using a task tree) and auctions the task nodes to the other robots. Although bidders can bid on all or multiple nodes, only one node at a time can be awarded per bidder

[117] Theraulaz G. et al. (1998): p.1

and the bid price depends on the marginal costs to perform the task. The empirical evaluation considered the example of a multi-robot simulator in which a number of robots had to overcome obstacles on a terrain map by coordination. The findings showed that the task tree allocation mechanism could outperform current state-of-the-art complex multi-robot allocation mechanisms with respect to the total distance travelled by the team and the solution cost (e.g. hitting a tree). They concluded that the distributed agent system is beneficial to be used when flexibility to re-plan and cooperate on complex tasks is required.[118]

Figure 15: Alg. 1 is used as a clearing algorithm by the individual agents (auction clearing refers to determining to which bidders to award which tasks). The auctioneer has a reserve price for the entire task tree (an entire task) in case it can perform the task itself (Alg. 2). The OpTrader decomposes new complex tasks entering the system and, in case it cannot perform the task itself, holds an auction to allocate the resulting task tree to the robots. There is no reserve price if the task can be performed by the auctioneer (Alg. 3). The auction stops when all tree nodes are auctioned

Algorithm 1: Auction clearing algorithm for restricted bids

Instance – T: a set of nodes in a tree with root R;
$p(N)$, $N \in T$: the lowest price bid on each tree node;
Initialize the solution by marking all leaf nodes;
while $N_p \neq R$ **do**
  Find N_{max}, the maximum depth node in T. Let N_p be the parent of N_{max}, and S be the set of children of N_p;
  if $Op(N_p)$ = *AND* **then**
    $p(S) := \sum_{C \in S} p(C)$;
  else if $Op(N_p)$ = *OR* **then**
    $p(S) := min_{C \in S} p(C)$;
  if $p(N_p) < p(S)$ **then**
    Mark N_p, unmark all descendants of N_p;
  else
    $p(N_p) := p(S)$
  $T := T - S$

Algorithm 2: *RoboTrader* auction clearing

while *the tree is not satisfied* **do**
  Determine and mark the cost minimizing set of remaining bids that satisfy the remaining parts of the tree, using algorithm 1;
  Sort marked nodes by profit (*i.e.* reserve minus lowest bid). Accept as many of the highest-profit bids as possible, ensuring that no bidder wins multiple nodes;
  Remove the bids of the winners from future consideration. If any of these nodes have marked descendants, unmark them and reintroduce the bids of their previous winners for future consideration;

Algorithm 3: *OpTrader* auction clearing

Determine and mark the cost minimizing set of bids that satisfy the remaining part of the tree, using algorithm 1;
For each bidder, award the node for which it has the highest margin of victory (*i.e.* first- minus second-price bids), if any. Do not include a node if an ancestor has also been awarded;

A stronger view on selfish bidding agents was presented by Liu L. and Shell D. A. (2013). In their strategic pricing algorithm used in a bidding framework, the bidders perform an optimization step aiming at computing an optimal *assignment permutation* of new tasks to an existing set of tasks formulated as a linear program. Thereby, each bidder wants to maximize their profit margin (primal program), that is the utility gained from performing a task minus the bid price (i.e. the price paid for winning the auction and obtaining the task). In the dual method, utility can be interpreted as a budget a bidder has for a task or object (or affordable price) to turn the linear program into a minimization problem. In their proposed algorithm, however, the tasks prices are not influenced by the bidders (as it is the case in a traditional auction setting) but by the auctioneer. For tasks that are preferred by multiple agents, the auctioneer raises the price by an amount that makes it unprofitable for some bidding agents who move on to bid for other tasks. This strategy clears the path for bidders who are most

[118] Dias M. B. (2004): pp.33-38

suitable for a task bid for. The findings showed that, while the traditional auction algorithm in which the bidders set the auction price, was extremely sensitive to the input data and worked best in example data with large value differences (that lead to larger bidding margins) but, overall, yielded the worst results compared to a centralized approach and the proposed strategic pricing, latter strategy significantly improved the model, particularly for a high matrix size. Nonetheless, the proposed framework yielded a solution quality that was slightly inferior to a centralized benchmark.

A joint task scheduling problem between competing but heterogeneous agents was proposed by Lang F. et al. (2016) which takes into account an agent's share of (possibly non-linear) costs on a joint schedule given private information. The efficiency of the negotiation protocol is, thereby, determined by minimizing the Euclidean distance to the closest Pareto-efficient solution (due to a potentially non-linear decision function) and minimizing social cost as the total sum of all agents' tardiness and operating costs. The negotiation protocol as proposed by the authors is based on offer modification (contract mutations) using simulation annealing, whereby deteriorating moves are accepted based on a probability distribution controlled by a virtual temperature parameter, so that worse offers are accepted in the early negotiation stage to overcome local optima. The protocol is carried out by an artificial mediator software component that generates the contract mutations randomly for the negotiation rounds. In the computational analysis, the authors evaluated their protocol based on several criteria, such as, for example, a randomly generated starting contract (offer), a pre-selection of an initial contract from a number of random contracts, adding acceptance quotas that require agents to accept a certain number of deteriorations, and partly allowing agents to execute contract mutations (agent proposal) by individually pre-selecting a best fitting contract from a set of random mutations. Their findings showed that acceptance quotas were essential to satisfactory results that were close to the Pareto frontier and the social cost minimum, whereby the outcome varied only insignificantly between random initial contracts and a pre-negotiation of a more suitable starting point. The authors additionally found, that the agent-based contract proposal led to more straightforward movements to the Pareto frontier than the basic protocol form in which contract mutations were performed randomly and without agent preferences by the mediator component only. While their protocol was found to work well with simple problem sets (of 5-10 machines), the results got considerably worse with a larger instance size. At the best outcome, the social cost solutions were approximately 1 percent worse to a central solution.

Tompkins M. F. (2003) presented a Mixed Integer-Linear Programming (MILP) approach using branch-and-bound for complex classifications of scheduling problems under different parameters. The precedence relationships and data flow, thereby, follows a control

architecture and a graph decomposition network based on the hierarchical planning of functionally independent tasks, see Figure 16. Typically, each task involves subtasks for which the processing time must be taken into consideration when computing the optimal schedule for a task or job. Thus, the framework requires two inputs: an agent network with known communication and execution delays, and a partially ordered job precedence graph. The author's framework was said to be an alternative to other heuristic based approaches that typically involve only a limited set of parameters. Additionally, the author used Multiple Objective Linear Programming (MOLP) techniques to evaluate the multi-agent setting under multiple objective functions. His computational results showed that for a small number of agents and jobs (<20), the model yielded minimal makespans for scheduling problems. According to his findings, allowing job to be solved parallel across multiple agents could further improve makespan and computational running time and, thus, allow for more agents and jobs to be modelled.

Abdallah S. and Lesser V. (2006) proposed a task allocation framework in which agents dynamically choose their strategy (i.e. negotiation behaviour) according to the benefits (reward) they reap out of the interaction with other agents. They are, thus, considered to be learners, and learning becomes the essential part in the offer generation. At the same time, the authors claimed that their algorithm introduced- a new gradient ascent-learning algorithm, which they call the *weighted policy learner algorithm*- outperformed state-of-the-art multi-agent learners in task allocation problems implemented so far, in terms of both convergence speed and sensitivity. In their framework setting, they differentiated between servers who execute tasks and mediators who receive tasks from users and allocate the tasks to servers, under the goal of finding an optimal allocation that minimizes the average turn-around-time (TAT: the time interval between a task arrival and its completion including processing and waiting times). According to the authors, any learning algorithm must fulfil the three criteria: (i) agents must be able to learn a stochastic policy instead of following a deterministic joint policy, (ii) the algorithm must converge in order for the system to create a stable solution, and (iii) the system must be adaptable as it is considered to be dynamic and open. Figure 17 shows the learning algorithm as proposed by the authors. The basic idea is that learning is increased if the strategy gradient changes its direction (if the action reward is rather small); else, it slows down (if the action reward is increasing).

Figure 16: multi-agent planning and scheduling process (left graph) and the agent network with execution and communication delays based on three example services (add, subtract, multiply) (right graph), as proposed by Tompkins M. F. (2003)

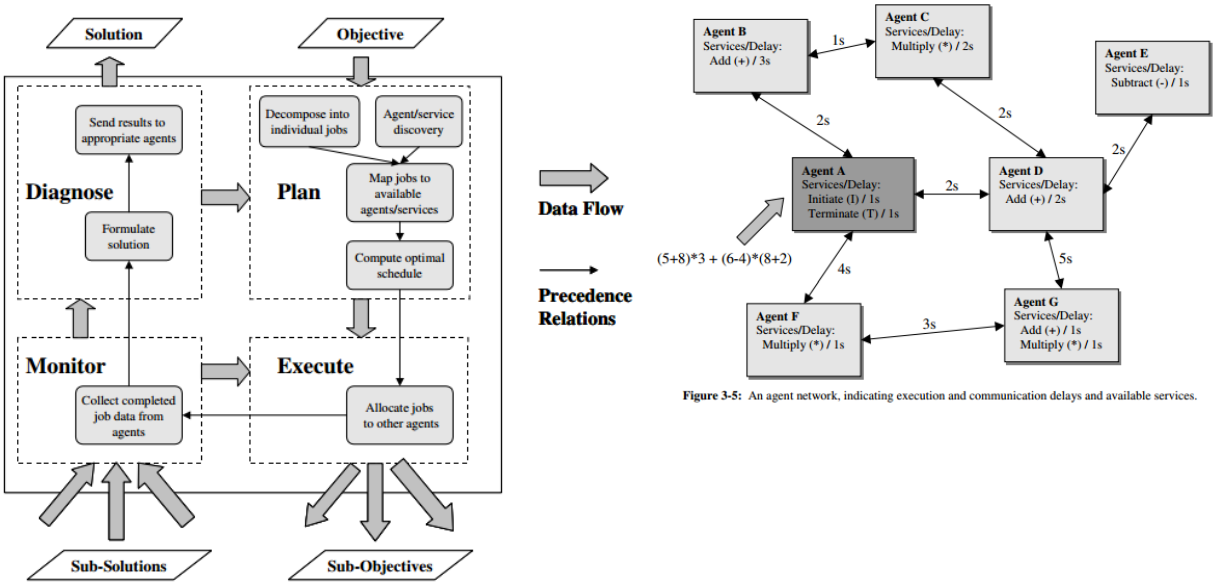


Figure 3-5: An agent network, indicating execution and communication delays and available services.

Figure 17: the learning algorithm as proposed by Abdallah A. and Lesser V. (2006) in a task allocation game, with π_i^t being the policy (strategy) of an agent i at the current running time ($t = 1$) and Δ is the strategy gradient that updates the strategy π

Algorithm 1: WPL: Weighted Policy Learner

begin
  Let $r(a)$ be the (expected) reward for executing action a
  $\widehat{r} \leftarrow$ total average reward $= \frac{\sum_{a \in A} r(a)}{|A|}$.
  foreach *action* $a \in A$ **do**
    $\Delta(a) \leftarrow r(a) - \widehat{r}$
    if $\Delta(a) > 0$ **then** $\Delta(a) \leftarrow \Delta(a)(1 - \pi(a))$
    else $\Delta(a) \leftarrow \Delta(a)(\pi(a))$
  end
  $\pi \leftarrow limit(\pi + \Delta)$
end

5.4 MAS in Facility Location Problems

The goal of any facility location problem is to establish a network that responds to customers' demand at a minimum time with the best possible quality (efficient allocation). Decentralized agent-based approaches for facility location problems are until now a poorly studied research area within logistic MAS and AI. Multi-agent facility location models could be potentially well used in land-use transport models and product-service networks within supply chains (i.e. where to locate production or service plants within an existing supply chain), particularly when it comes to facilitating the decision-process by using the advantage of the dynamic nature of multi-agent distributed systems.

One decision framework for locating facilities was presented by Afshari H. et al. (2014). They proposed an agent-based approach within product-service networks that are typically characterized as distributed problems. Agents are considered regional warehouses (RWs) and cities, and the optimization method governs the behaviour of the agents in the distribution-service network environment. The optimization rules are, thereby, derived from similar centralized optimization problems in facility locations. The idea of their framework proposed is based on the question of how many regional warehouses to install (customers belong to distinct regions) and where to locate them. For making a facility location decision, an RW agent autonomously decides based on optimization rules under multiple agent-based [constraint] elements such as, for instance, latitude and longitude location in the model, cumulative demand of covered customers, minimum distance to another RW agent, capacity and cost rates, number of facilities (RWs) to be located, the distance type method used, and the location method (objective function). The empirical evaluation of the authors considered different objective functions (min distance among all RW's, closest to central warehouse, max demand coverage, etc.) and showed that under demand uncertainty, the agent-based model led to more efficient location decisions than in a centralized approach in terms of service speed (i.e. time needed to serve customer demand) but yielded, overall, worse cost solutions than the central optimization approach. The authors concluded that the use of weighting mechanisms for both cost and time balances in the agent-based optimization model could lead to a service network that saves transportation time while, at the same time, supplies customers faster.

5.5 MAS for Lot-Sizing

As for multi-agent facility locations, decentralized lot sizing in MAS is yet not properly studied. Lot-sizing MAS covers decentralized production coordination and, thus, takes into account distributed systems of agents that follow a mutually agreeable production plan under private information. Homberger J. (2010) proposed a decentralized multi-level uncapacitated lot-sizing problem (MLULSP) using decentralized simulated annealing, consisting of a transition rule performed by a neutral mediator agent and a cooperative acceptance rule assessed by the negotiating agents. The approach of the author was to formulate the problem as a coordination problem using facility-based decomposition where the production of subassemblies and components is spread across multiple facilities. Each facility is a independent decision agent who follows the individual objective of minimizing total inventory holding and setup costs of its facility (local costs). The individual objectives are typically conflicting in the overall global setting and, in order to reach a mutually agreeable production plan, the decision agents must coordinate themselves. The typically NP-hard MLULSP is

formulated as a MILP with the objective function (to be minimized) being the sum of inventory and setup costs for all items over the entire planning horizon. The constraints include the inventory balance equation, a demand function for a production quantity of an item in an upcoming period, the appearance of setup costs in case of purchased or produced batched and non-negativity constraints on inventory and production variables as well as the binary nature of the decision variables (setup). Additionally, all self-interested agents that are assigned to a facility follow their individual objective to minimize the local costs. Thereby, the local costs cannot be minimized simultaneously (conflicting objective among agents). The negotiation model proposed by the author works as follows: in the first phase, a neutral mediator generates a random first contract c that represents a joint production plan, and each negotiating agent ag_k computes a local schedule with controls the probabilities of accepting deteriorations during the negotiation. In each negotiation round, the mediator generates a new, neighbouring contract proposal c' randomly (by transition rule that is in the framework proposed similar to a random pick from 0 to 1) and each agent votes for or against the acceptance of the proposal following a private cost function (taking its local costs into account) that influences the acceptance probability. Based on the empirical evaluation, the author concluded that the average solution quality (total costs) of the agent approach was significantly lower than the centralized method (using simulated annealing) and it decreased with an increasing number of negotiating agents as, generally, the proposals' value declined with an increasing number of agents.

6 Experimental and Real-life implementations of Autonomous Agents

Until now, real-life applications of multi-agent systems are limited and mostly implemented as a decision-making instance in which the MA-simulation proposes one or several outputs that a central agent actively chooses from, instead of transferring the decision-making processes to a number of autonomous agents. MAS proposed in literature have largely remained prototypes out of many possible reasons: on the one hand, many frameworks are limited in their practicability, particularly in a multi-agent system of possibly conflicting agents with international spread. Criteria not always well enough considered include the stability and robustness of a negotiation agreement (contract) over time, as well as the general necessity to share private information in an environment sensitive to data security, trade secrets and contractual agreements in long-term business partnerships. Literature findings also show that a MAS' solution often differs in quality based on underlying environmental conditions that can change unforeseen. Hence, MAS can be considered to yield [local] solutions that are usually worse to a state-of-the-art central solution (though, a direct comparison between a central and a decentralized system is rarely possible due to the different objective formulations). MAS is not only about creating intelligent agents capable of negotiating autonomously, but it is most likely about creating a sustainable system in which negotiation strategies can be autonomously modified dynamically by the agents with respect to the changing objectives, underlying resources (constraints) and environmental inputs (negotiation partners, legal constraints, etc.), which is difficult to be defined in practice. On the other hand, a decentralized agent system generally requires considerable investment without showing immediate and direct financial benefits over a previous central system. Additionally, the efficiency of a multi-agent system strongly depends on the nature of the business to be decentralized. This chapter introduces a number of autonomous systems that either are applied in a real-life setting or have proven to be attractive for implementation. Such projects range from agent trading systems to the autonomous movement of machines.

In the late Nineties, a particularly attractive research field has been automated trading systems in which demand is automatically matched with supply in online commerce. An example for such an intelligent automated trading system is **Kasbah**, a virtual agent marketplace on the Web, proposed by Chavez A. and Maes P. (1996), where users create autonomous agents to buy and sell goods on their behalf in bilateral and straightforward negotiation rounds. Kasbah is a simple, distributive prototype marketplace. When a user creates a new selling agent, he gives it a description of the item to sell including a desired sales deadline, desired price and lowest acceptable price. The selling agent then goes into the virtual marketplace, contacts interested parties (buying agents) and negotiates with them

to reach the best deal for the client. Similar to the selling agents, buying agents are created whenever a user aims to buy a particular good. The prototype of Kasbah does not include any machine learning techniques but allows parameters to be changed by the users at any time after the agent has been created. Additionally, the parameters include a certain dynamic structure in which, for instance, the sales price is decreased gradually the longer the selling agent negotiates in order to reach an agreement before the sales deadline. Agents can enter and leave the marketplace anytime and a notification is then send to the remaining agents.[119]

Similar to Kasbah, **AuctionBot** is an example of an academic auction platform in which autonomous agents are used to sell or buy used textbooks and individual objects at the University of Michigan that is available to the public since 1997. In AuctionBot auctions, users create new auctions to buy or sell products by choosing from a selection of auction types and parameters (e.g. clearing times, method for resolving bidding ties, closing conditions, etc.). The AuctionBot then manages the multilateral bidding processes according to the negotiation protocols and parameters of the created auctions.[120]

Automated negotiation is not only attractive in e-commerce applications. There are numerous examples of MAS-related applications in the fields of Artificial Intelligence that could be or are employed in the fields of logistics too. One example is **AIBO**, an Artificial Intelligence Robot, which describes pet robots designed by Sony who autonomously interact with their environment. Such robots take information from their surrounding using visual intake, voice recognition and to some extend also contact and motion. For industry purposes such AIBO's are implemented to a technologically more advanced extend, such as the **Kiva robots** developed by Amazon Robotics, a fully owned subsidiary of Amazon, and deployed in Amazon's fulfilment centres. The items in the warehouse are being stored in portable storage units and the Kiva database system locates the closest Kiva bots to the item ordered and directs a free robot to retrieve it. The robot navigates itself through barcode stickers on the floor and autonomously reacts to (e.g. compassing) other robots nearby. When the robot reaches the target shelf, it slides underneath it, lifts it up and carries the shelf to a specified human worker who picks the ordered items to be packed for delivery. Kiva robots can lift up to 750 pounds, their motion sensors can detect objects on their way and they can travel between 3 and 4 miles per hour.[121]

Since the active employment of Kiva robots in Amazon fulfilment centres around America, several start-up project have emerged to fill the industry gap left by Kiva: the **Fetch Robotics** from a Silicon Valley start-up, for instance, is able to independently pick orders and place

[119] Chavez A. and Maes P. (1996): pp.1-12
[120] Wurman P. R. et al. (1998): pp.1-7; Guttman R. and Maes P. (1998): pp.5-6
[121] Amazon Robotics (2015): online; CNET News (2014): online;

items into a pod on a freight robot that then autonomously navigates to a packing station. The communication thereby takes places through a real-time web-based communication and information system, which allows managers to monitor the robots' movements and degree of retrieval and delivery fulfilment. Another example is the **Harvest Automation**, a start-up project from the plant nursery segment, which employs robots being able to carry heavy workload from one location to another. Beside Amazon, many other suppliers use warehousing automation technology to assist human workers in retrieval and transportation activities, but so far only a few start-ups are offering that level of disruptive system.[122]

Since about a decade ago, MA contests have been run with agent gaming approaches aiming to stimulate research in the area of MAS development and programming by creating a competitive programming platform for testing multi-agent programming languages and tools on online scenarios that are strongly replicable to real-world applications. Thereby, the contestant teams create a MAS and test it on a given gaming field. Each team consists of different types of agents (e.g. cars, trucks, motorcycles, and drones) which differ in speed, motion ability, power (e.g. battery) and transportation capacity, and the teams collect tournament points according to the money earned at the end of the simulation. The MAS performance is tested in a series of games where the systems compete against each other. Such multi-agent gaming has led to several real-life implementations that are, up to now, held playfully, such as the **RoboCup** (robots playing football on a platform with sensors and effectors to read from and to make changes to the environment, respectively).[123]

Robotic cars refer to autonomous vehicles that are capable of sensing their environment and navigating around objects with the use of radars, Odometry and computer vision. They are able to identify appropriate navigation paths and obstacles and interact with GPS data to find the right location. Implementations of robotic cars in the most practical relevance include the **Oxford RobotCar** that uses probabilities and estimation of data from sensors like cameras, radars, lasers, aerial photos and internet queries. Additionally, the cars employ machine-learning techniques to build a picture of the world in terms of prior experience (training) and knowledge. The innovation in robotic cars like RobotCar is their ability to work outside of controlled workspaces, a limitation the Kiva Robots do not address (*note*: latter only follow barcodes on the warehouse floor). In other words, entirely autonomously functioning robotic cars must fully independently identify the ground on which they drive, the direction they must go to and the different objects in their environment they must distinguish from and react to in order to move safely. To apply those cars to a vast scale, they must also operate outside GPS areas. **Google Driverless Car** is the first Self-Driving Car project that

[122] The Robot Report (2015): online
[123] Multi-Agent Programming Contest (2016): online

evolved to actual driving application on a large scale on American streets (RobotCar is so far only being tested in Oxford). Up to now, there are still safety drivers aboard all vehicles.[124]

Nurse robots were developed to assist the healthcare system in caring for the increased elderly demographics. Since about a decade, the development of robots that assist nurses with their workloads on elderly patients has been of high interest for countries with particularly problematic demographics, like Japan. This has led to the development of robots that carry patients (**RIBA**) or medical robots that contain a medication database with respect to health issues and vital signs. The former robot types were developed about seven years ago in Japan with the goal to lift patients into or out of bed and help them stand. Latter robots, called **Terapio**, are programmed to follow a nurse in her rounds and as a nurse inputs data to the robot, it can immediately suggest certain medications, or recognize medication interdependency or allergies to look for.[125]

[124] MRG (2016): online; Google (2016): online
[125] Extreme Tech (2015): online

7 A Flow Shop Cooperative Auction Game

Various optimization problems can be formulated as *task allocation problems*, in which N objects (e.g. tasks, jobs, resources) need to be assigned to a set of M agents (e.g. processors or machines) at minimum overall costs and at a particular time in order to achieve the overall system goals. These problem types are typically NP-hard and can, thus, not be solved within polynomial time. Task allocation in a MAS is the problem of coordinating agents' interaction with each other and with the environment to determine which agent should execute which tasks to achieve the system objectives. In case of a robotic system in which robots act as agents, the problem is commonly referred to as *multi-robot task allocation* (MRTA).[126]

This paper's theoretical framework implementation focuses on a generic example of a task allocation problem, that is a multi-agent system applied to a *flow shop scheduling* problem in which new orders constantly enter the system from a job database and have to be implemented into an already existing scheduling plan. The computational example in this paper is implemented in R using a fixed number of predefined job instances of $n = \{20, 50, 100\}$ jobs in which each job with $l = 5$ operations to be done on $m = 5$ different machines must be assigned separately, and is tested against a central agent performance using the same underlying optimization method. The methodology uses an auction protocol in which *workcenter agents* (bidders) are being informed about new jobs by a *shop floor agent* (auctioneer) who assigns job candidates to the best bidder. The workcenter agents compute a bid for the job candidate based on their current job schedule (scheduling costs) and make a proposal decision based on the information they receive from the *information agent* (threshold value). At the end of the auction, the workcenter agents enter a *cooperative trading* process in which they can exchange jobs that fit their individual schedule in a way that improves the system solution. The goal of the production schedule is to ***minimize total makespan*** in a system of ***four agent types***. The end of the chapter provides a computational performance and solution quality comparison.

7.1 Multi-Agent Scheduling Framework

The scheduling objective is to organize the execution of incoming jobs to a set of machines (M) following a **flow shop problem** design (i.e. same machine order for each job) such that the total agent makespan for all jobs is minimized. The machines are grouped into work centers, whereby each work center consists of a predefined number of different machines. Each incoming job j consists of $n_j = 5$ operations (tasks) of different machine order.

[126] Korsah G. A. et al. (2013): p.1

This problem setting is build up around **four functional agent types:** two information agents (shop floor agent, information agent) and two task agent types (job agent, 3 workcenter agents). The job arrival is handled by a ***Job Agent (JA)*** who remains fixed in the system. Whenever a job needs to be scheduled, the Job Agent turns to the Shop Floor Agent for assignment. The ***Shop Floor Agent (SFA)*** is the auctioneer in the system who requests a current central solution from the Information Agent and informs the Workcenter Agents, who each hold 5 machines as resources, about any new job to be assigned, its central solution (used as a threshold value for the proposal decision) and requests them to submit their bids. The ***Workcenter Agents (WCA's)*** determine the bid value based on the scheduling costs including the new job's processing times and the individual jobs previously won. The bidding decision (*propose-or-withdraw*) is based on a threshold value that is 40%-60% of the current central solution as computed by the ***Information Agent (IA)***. Within each bid computation, the *Campbell, Dudek and Smith* Algorithm is used on the total schedule considering the new job. Thus, the lowest scheduling cost (minimum of all agents' minimum total makespan schedules) wins the current auction. Whenever a job is assigned, it leaves the JA and proceeds to the winner-WCA. Workcenter Agents can only bid on the entire job (i.e. all its operations). If an agent's scheduling cost exceeds the threshold value, the agent withdraws from the auction round and waits for the next job to bid on. In case all WCA's withdraw from the bidding round, the job is send back to the JA, entering as a last-in and its assignment is postponed to a later round.[127]

The *Campbell, Dudek and Smith* (*CDS*) Algorithm divides a n-job m-machine problem with $m > 2$ into $m - 1$ number of 2-machine n-job sub-problems. For each sub-problem, the value of C_{max} (total completion time for the n-jobs) is computed using Johnson rule, and the best job sequence of all sub-problems is selected that yields the minimum value of C_{max}.[128]

Following simplifications were used:

- no job or resource travelling time is being considered
- material needed to process an operation per machine is infinite
- no setup time is required on each machine

Figure 18 illustrates the agent interaction framework following the technical infrastructure. In practice, the MA-framework suggested could consider a virtual server Job Agent, a central server Shop Floor Agent, distinct and entirely separated private Workcenter Agent Terminals and a central, public information agent acting similar to a mainframe. The contract net

[127] Amrita and Tripathi A. (2014); Xuesong J. et al. (2016); Wu Z. (2005)
[128] Harkan I. (2010): pp.10-14; Modrák V. and Pandian R. S. (2010): p.276; Kumar Sahu L. and Sridhar K. (2015): pp.41-42

auction and cooperative trading protocol as illustrated in Figure 19 is further explained in the following procedures.

Figure 18: communication network of the JSSP example with three workcenter agents of which each workcenter has five fixed machines (resources) of different type

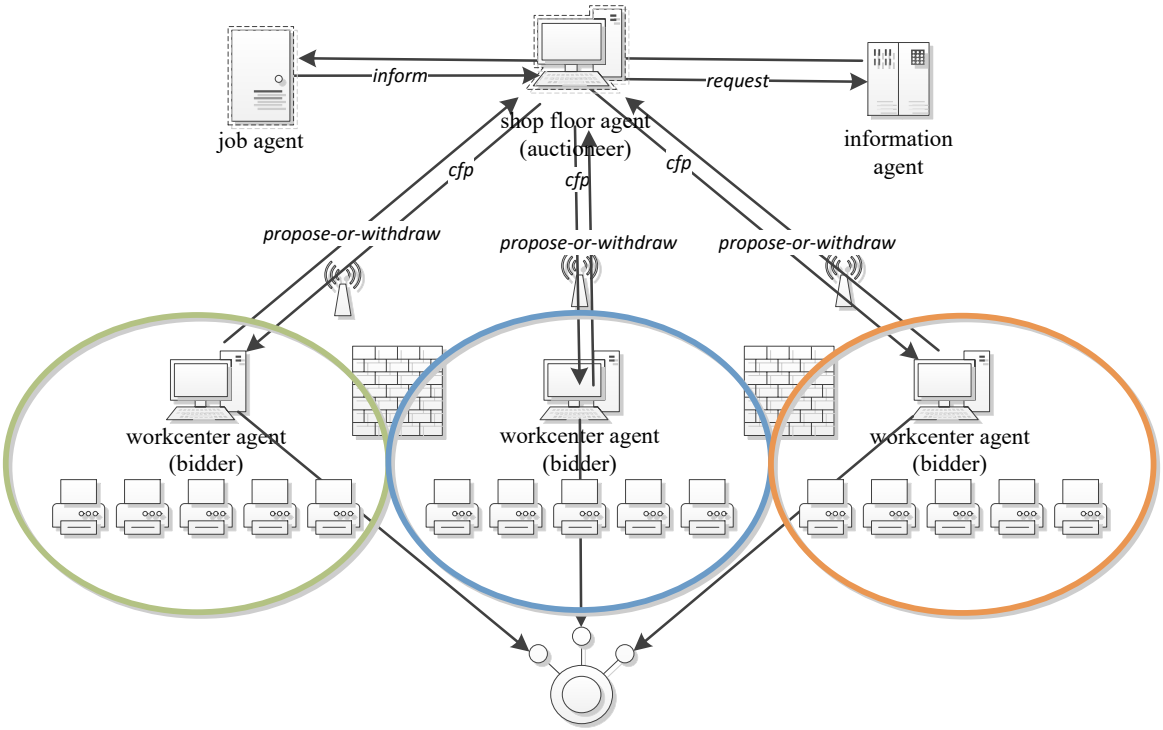
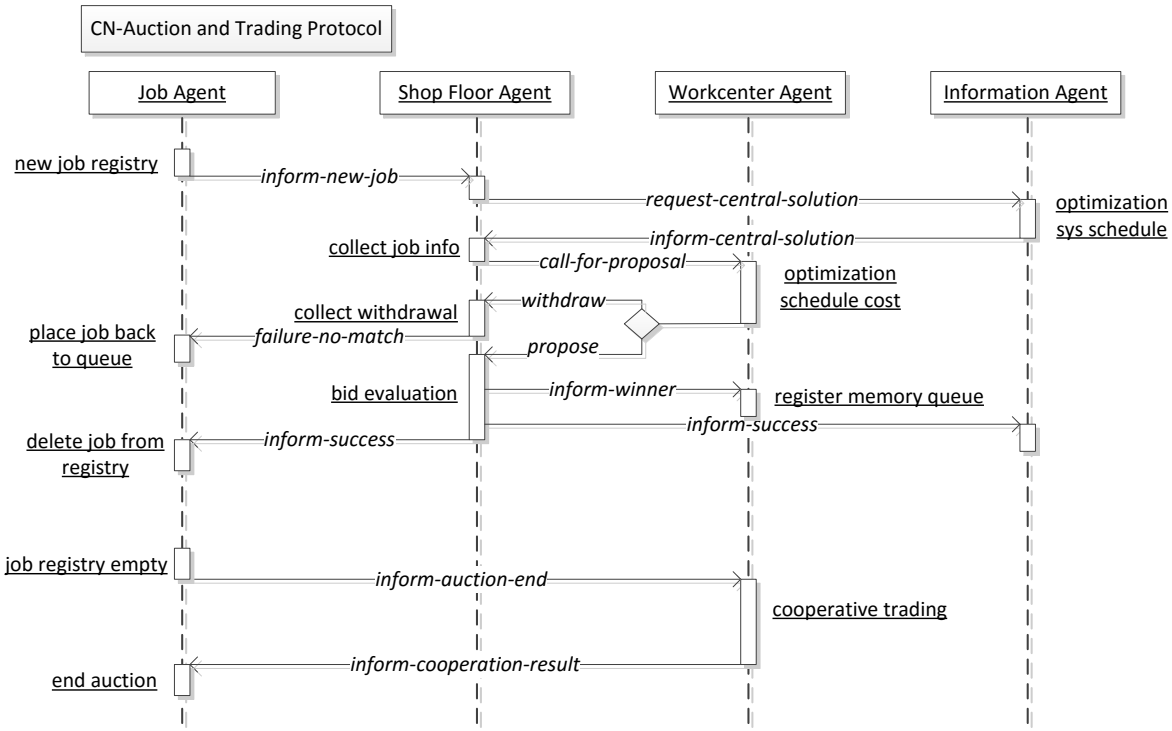


Figure 19: CNP of the MA-FSSP



Each agent consists of a knowledge unit, a functional unit and a control unit. The knowledge base contains the domain data, the functional component are the computational procedures for decision-making and the control unit consists of the protocols for the respective agents.[129]

Job Agent (JA)

The knowledge base of the job agent consists of a memory registry of all incoming jobs, the total number on uncompleted jobs as well as the job information. The functional component is for the assignment request: to select the next job from a list of queuing jobs and request an auction start from the shop floor agent. The control unit is the following communication protocol:

1) update job list (i.e. generate jobs and their information).
2) if there are several queued jobs, determine a job order for assignment (if earliest due date is given, queue order is based on that; if no priority is given, select FIFO); else go to step 3.
3) send an assignment request to the SFA on the next job; if there is no next job, go to step 5.
4) if job is assigned (i.e. all its operations), go to step 1.
5) if all jobs are assigned (queue is empty), send a cooperation-request to the WCA's, else go to step 3.

Shop Floor Agent (SFA)

The SFA acts as a job auctioneer who assigns new jobs from the Job Agent to the best bidding Workcenter Agent. As a participation decision (threshold value), the SFA requests a current central solution for the job candidate and sends it with a *call-for-proposal* to the WCA's.

The knowledge base consists of the current job information, the existing workcenter agents and their status. The functional unit is for job auctioning. Job auctioning selects a workcenter agent for the new job based on a bid comparison. The control unit can be designed as follows:

1) request central solution from Information Agent (threshold value).
2) send new job info with threshold value in a call-for-proposal (cfp) to WCAs.
3) collect bids and withdrawals from the WCAs and evaluate them.
4) if all WCA's withdraw from the bid round, inform Job Agent to place job back, else go to 5.
5) select a winner-WCA as the lowest bid (bid = scheduling costs) and inform the winner. If there is more than one winning bidder, select a winner randomly.

[129] Wu Z. (2005): pp.35-36 adopted from Sikora and Shaw (1997)

6) if all jobs are assigned (i.e. JA doesn't send an assignment request), stop; else go to 1).

Workcenter Agent (WCA)

The decentralized bidding agents have a fixed number of machines available to schedule. After receiving a *call-for-proposal* from the Shop Floor Agent, each WCA computes a bid based on the total cost of the new schedule including the job candidate represented as minimum total makespan. The idea to formulate the bid as new or current schedule costs is derived from Chien S. et al. (2000) and its main gain is that the bidder whose schedule cost is the lowest after adding the task in question to an already existing schedule wins the auction, which is closest to a desirable reality[130].

The knowledge unit of a WCA consists of the current number of jobs in each machine's queue and the processing times. The functional component is for job scheduling. Job scheduling selects the best assignment for the job candidate in an already existing schedule, whereby current jobs in the queue can be rescheduled in each auction round if total makespan is minimized. The control unit can be designed as follows (for each WCA):

1) if there is a cfp-request, collect the number of jobs currently in queue from private memory.
2) formulate a bid based on the current schedule with the candidate job using CDS-algorithm.
3) compare bid with central-threshold as computed by the information agent.
4) if job threshold is not surpassed, send bid (proposal) to SFA; else withdraw from the bid.
5) if proposal is accepted, queue new job to the current job schedule (memory) and wait for new cfp; else (proposal rejected) do nothing.
6) if there is a cooperation-request from the IA, start to trade (exchange) jobs such that the trade gain for one agent compensates for the trade loss of the other agent (i.e. overall system solution is better after the trade than it was before); stop if no solution improvement is possible.

Information Agent (IA)

Using an information agent in a multi-agent setting is quite common in the MA-literature. An information agent allows to directly connecting a decentralized system performance to a central measure for solution evaluation and, thus, improvement.

130 Zlot R.M. (2006): p.94 referring to the paper from Chien S. et al. (2000)

The knowledge base of the IA includes all the job information already assigned in the previous negotiation rounds. In a dynamic setting, any jobs that were finalized in the processing would be deleted from the global schedule memory. The functional unit of the IA is for job sequencing in a central context, that is computing a central solution of a single FSSP on 5 machines, including the processing time information of the jobs previously auctioned and the job candidate. The central solution serves as a threshold for the WCAs' decision to participate (*propose-or-withdraw*) in the auction round. The control unit of the IA looks as follows:

1) if SFA requests the threshold information, compute the central solution for the job candidate including all jobs auctioned successfully in previous rounds since auction start, send the solution information to SFA and wait for the SFA answer on the auction success; else do nothing.
2) if SFA sends an auction-success, store the job candidate in memory; else drop the job information from the system solution.
3) if there is a new request from the SFA, repeat 1) and 2); else do nothing.

7.2 Summary of Experimental Results

In the MA-example, the jobs are predefined following Taillard's data instances with 20, 50 and 100 jobs[131] and the jobs enter one-at-a-time the scheduling architecture. In this problem design, the individual job components (tasks) must be processed at the same work center (i.e. no job division between work centers is allowed) and in the same machine order of $m = 5$ machines (*flow shop scheduling problem; FSSP*). The distribution of jobs to work centers follows an auction concept in which Workcenter Agents submit bids for each incoming job and, once won, cannot withdraw from the job bid on. The job won is then inserted into the job queue of the winner agent but not further processed (*static MAS*). The bids computed by each agent is derived from the minimization of the total makespan taking into account already existing jobs in the respective agent's queue and the job candidate (i.e. total length of the current schedule including the job candidate). Within each bid computation, the CDS-Algorithm is used on the total schedule with the job candidate (the jobs can be reordered at every auction round). The agent with the minimal total makespan for the job candidate wins the current auction. The total auction process stops when all jobs in the instance database are being assigned and the *cooperative trading* (job exchange) between the Workcenter Agents starts.

[131] derived from http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html

The example data has following characteristics:

- each job follows the same machine order
- only one operation from each job can be processed at the same time
- each machine has different processing times
- job rescheduling is allowed
- no job is processed twice on the same machine
- machines cannot process more than one operation at the same time
- each work center has a fixed number of machines (5 in total) of different type
- machines can be idle within a schedule period
- jobs are scheduled in a queue until all jobs from the database are assigned (static scheduling)
- the machine processing order as given by the JA cannot be changed once auctioned

Table 2 illustrates the average computational solutions per Taillard benchmark instance from the central solution (*central CDS*), from the MA-auction game only (a*uction*) and from the cooperative trading (*coop. trade*) at the end of the auction game. The central solution was computed considering all n-jobs on $m = 5$ machines. The auction and cooperation solution is the sum of all agents' optimal schedule (min total makespan) considering the agents' respective jobs won in the auction and received in the trade afterwards, respectively. For each set of jobs $n = \{20, 50, 100\}$, four instance data sets from Taillard were taken into consideration. The column *solution* Δ *A:C* displays the percentage difference between the auction solution and the central solution. Similarly, *solution* Δ *T:C* is the percentage difference between the cooperative trade and the central solution. Column *trade:auction* shows the solution improvement yielded by the cooperative job exchange with respect to the auction's outcome.

The findings in Figure 20 show that the multi-agent setting is always worse than the central setting. Particularly, if there is a low number of jobs to fill idle times of machines, the computation solution is with appr. 30,25% after the auction and 18,43% after the trading considerably worse to a central agent structure. The higher the number of jobs to be assigned, the lower the solution difference, with about 12,01% solution difference from auctioning 50 jobs and 6,02% from auctioning 100 jobs. Following the nature of a competition, the purely competitive game unsurprisingly yields local solutions that are considerably worse than the cooperative outcome. After the coop. trading procedure, the solution difference decreases to an average of 2,63% solution difference to a central agent. While a 20-jobs' scheduling game yields considerably unfavourable agent schedules, its improvement margin is likewise the highest than for a higher number of jobs. The job

assignment in the auction results in largely uniformly distributed jobs and as the trading only allows exchanges, the cooperative schedule remains equally distributed.

The findings coincide with literature findings as follows:

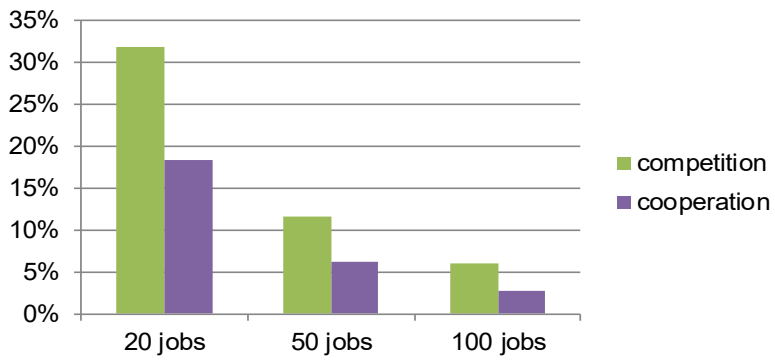
1. competitive games typically yield worse solutions than a cooperative approach, as competitive agents only focus on finding local optima that satisfy their own objective
2. MAS are computationally more complex than central systems, which significantly influences the running time (see *usertime in sec* in the Table 2)
3. MAS are most beneficial in complex (e.g. large) dynamic systems such that the solution difference to a centralized system decreases with a large problem size and is, in contrast, more robust to environmental uncertainty (e.g. machine breakdowns; latter benefit from a dynamic system was not tested)

Table 2: findings from the MA-framework compared with the centralized CDS-solution based on Taillard's test instances of nxm jobs-on-machines. Average results are based on 4 runs. Usertime results are based on an Intel(R) Core Processor with 3.20 GHz, 7.88 GB RAM

instance sets	UB	LB	central CDS *min Cmax*	auction *min Cmax*	solution Δ A:C	usertime in sec	coop. trade *min Cmax*	solution Δ T:C	usertime in sec	trade:auction
002 - 20x5	1359	1290	1424	1847	29,69%	1,27	1699	16,2%	2,84	-13,5%
003 - 20x5	1081	1073	1249	1650	32,09%	1,31	1481	15,7%	2,00	-16,4%
005 - 20x5	1236	1198	1323	1767	33,58%	1,39	1691	21,8%	2,13	-11,8%
006 - 20x5	1195	1180	1312	1735	32,22%	1,25	1642	20,1%	1,74	-12,1%
007 - 50x5	2724	2712	2866	3170	10,59%	2,75	3029	5,4%	217,91	-5,2%
008 - 50x5	2834	2808	3032	3402	12,21%	2,70	3259	7,0%	204,65	-5,2%
011 - 50x5	2863	2837	3038	3362	10,66%	3,30	3239	6,2%	190,31	-4,5%
012 - 50x5	2829	2793	3031	3415	12,67%	3,30	3238	6,4%	203,96	-6,3%
014 - 100x5	5268	5208	5563	5857	5,28%	12,33	5692	2,3%	350,75	-3,0%
015 - 100x5	5175	5130	5452	5776	5,94%	10,82	5614	2,9%	408,26	-3,1%
016 - 100x5	5014	4963	5273	5558	5,41%	10,92	5410	2,5%	654,02	-2,9%
018 - 100x5	5135	5063	5203	5603	7,68%	10,54	5354	2,8%	563,03	-4,9%

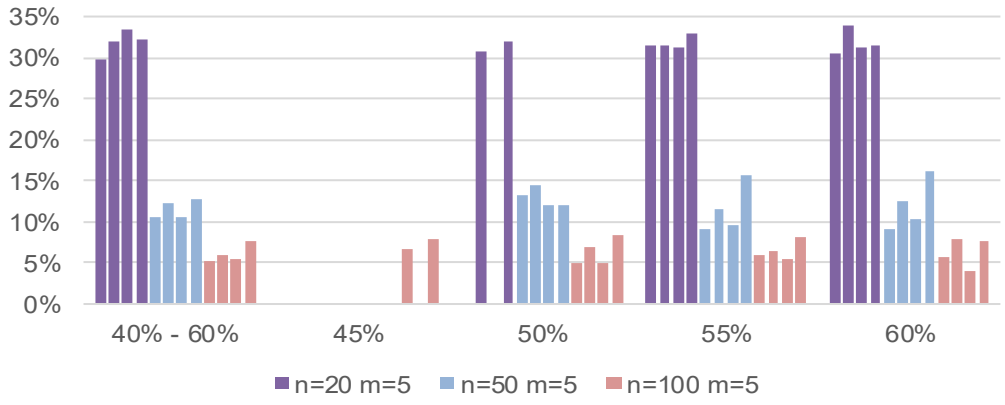
Figure 20 illustrates the computational findings in a bar chart. 0,0% refers to the central solution. It can be seen, that the solution difference to a centralized system decreases, the larger the system is (i.e. the more jobs need to be assigned).

Figure 20: comparison between central solution (global optimum) and the MA-auction as a competitive game with and without cooperative trading



In the computational example, the threshold value that influenced the *propose-or-withdraw* decision was set randomly as an integer percentage in the interval [40%,60%]. Threshold values set in the interval [35%,40%] predominantly let to job loops in which a job was continuously refused to be bid on by all the Workcenter Agents. A threshold analysis shows that, on general, the solution quality does not depend on the threshold value; thus, a lower threshold value does not necessarily lead to a higher solution quality! Selecting a threshold value randomly, however, smooths out the average solution differences and, hence, creates a more stable auction result irrespective of the data set.

Figure 21: auction solution difference to central schedule based on threshold value (x-axis) and on n jobs and m machines



7.3 Limitations and Prospects

The practical usability of the proposed example model is in several ways limited to its real-life implementation. The following list includes suggested adaptations and limitation considerations of the proposed model as well as some known risks in a MA-setting:

1) The jobs listed in the database did not include due dates. Jobs with due dates cannot simply re-enter the Job Agent and patiently wait for another auction round. Instead, the Job Agent would sort the incoming jobs according to earliest due date and, once jobs enter the auction, they must be bid on which makes the MAS less flexible.

2) The only costs involved in the example is the schedule length (processing times). Typically, a bid needs to take more costs into consideration, such as, for instance, material costs to fulfil tasks, penalty costs for late delivery and costs related to a transportation and setup time. Agents may also have limited or unlimited capacities to perform a task of a job, which needs to be taken into consideration in a real-life setting.

3) The cooperative trading includes the processing of private information, that are all the individual tasks' processing times of the agents' schedule who participate in the trading.

The exchange of private information among competing agents can be implemented using a fifth independent agent (*trader agent*) whose goal is to exchange jobs without the actual information from which agent the job is being traded from and with. As such, the trade agent acts as a central coordinator and is vital to the auction game. Such a powerful agent can realistically not be easily implemented in an auction setting between competing firms, because of its susceptibility to hacking and technical breakdowns and, most significantly, lack-of-trust in its neutrality.

4) The model's bid computation only takes queued jobs into consideration while the jobs won are not further processed until the auction ends. In a dynamic setting, in which jobs assigned could immediately enter the processing, the bid would also include any jobs currently in process, while no operation interruption is allowed, as well as the remaining time to finish a job in process.

5) The trading at the end of an auction involves exchanging two jobs between agents so that at least one agent benefits from the trade while the other agent is not worse off than the other's gain from the trade. The trading concept implemented is in several ways complicated in the practical context: first, the end of an auction must be clearly defined, while the current jobs in the agents' queue additionally influence the result of the cooperative trading's outcome. As the exchange involves swaping two jobs between agents, the trading outcome can significantly vary depending on the auction's outcome. Second, a trade in which one agent benefits at the cost of another agent requires a *future promise* or *penalty* to be negotiated successfully and remain stable throughout further negotiation rounds. A future promise can be, for instance, directly linked to a budget that an agent draws upon. Defining a penalty rate or budget cost is crucial as it directly influences the likelihood of future successful negotiation between the trading agents or the ability to continue trading with other agents.

6) In the example concept, job tasks must be done by the winner agent (i.e. operations cannot be split) and an exchange of individual operations is not possible. However, a manufacturing firm in which products need to be assigned to several identical work centers, a task transfer between machines of different work centers can significantly improve the multi-agent setting's performance by reducing idle time of machines, and makes the system more robust to machine breakdown. Allowing task transfer, however, requires to consider several additional costs in the bid formulation, such as the transportation time between machines, setup costs and fixed costs for the transportation infrastructure. Linking those costs to a task trade while neglecting to take them already into consideration in the auction process, can lead to a scheduling system in which

agents win jobs who are unfavourable but can later on be traded with a budgetary advantage, which makes the MAS suboptimal. On the other hand, considering those potential costs (probabilities) in the bid calculation makes the MAS increasingly complex, as the question arises as of the size and probability of the potential trade costs, which are likely to be unknown at the start of an auction round.

7) The biggest advantage of a MAS in comparison to a central system is its *robustness* to environmental changes (e.g. machine breakdown). The robustness advantage is clearly identifiable when considering a vehicle-routing scenario in which a truck agent breakdown occurs on the road and a breakdown signal is send to neighbouring truck agents, who then immediately recalculate their route considering the freed customer destinations and their remaining delivery points. Such clear benefits are, however, not directly viable in other problem settings, such as, for instance, in a job assignment problem in which a central agent can likewise act as a re-scheduler in case of a breakdown which can significantly benefit the complexity of such a multi-agent system, making the initially decentralized system central (but still automated) again.

8) Creating robust, dynamic multi-agent systems among geographically spread locations is largely impeded by a reference point. Thereby, the dynamics of a MAS is not only influences by the factor time, but also by wireless factors such as file transfer speed, disruptions and security. Most importantly, all connected data processors must speak the same language (protocols) which increases the thread of security lacks and hacking.

8 Conclusion

Designing a MAS is a complicated task to do with many considerations to make in the development phase, such as on the functionality and number of agents to use, the dynamics of the environment in which they operate and their negotiation nature, which are the negotiation objects, their valuation, the rules governing the agents' interaction (protocol) and their social context. The probably most difficult aspect of a MAS is the interrelation between the different negotiation components: an effective agent's strategy largely depends on the negotiation protocol. Each protocol, on the other hand, comes with certain limitations that are influences by the nature of the system's environment. A purely game-theoretic bargaining, for example, comes most likely with unstable agreements and, although it is computationally less complex to be implemented, largely yields local and, thus, poor solutions. On the other hand, an auction strategy was found to work best in systems with large differences in input data. A cooperative approach to negotiation can likewise prove to be difficult to be defined when agents must take into consideration the value interpretation of their negotiation partners while following individual goals. In order to facilitate an agreement outcome, additional reasoning steps could be included such as placing arguments on rejections or offers, whereby negotiation additions can make a negotiation model more complex and, thus, inflexible to the input data. However, expanding the agents' reasoning model to include learning mechanisms is crucial to developing a negotiation model that is both computationally beneficial and robust in the outcome.

In theory, multi-agent systems in logistics have been dominantly used in the transportation, task allocation and scheduling domains, particularly because those applications have shown to fulfil the characteristics ideal for implementing agent technology. Such characteristics include, most importantly, clearly identifiable, distinct entities with well-defined set of state variables, a high degree of structural complexity and environmental dynamics, and ill-structured information about the application at design phase. The advantages of agent-technology over central-agent systems are, thereby, found to lie in their general solution robustness of a dynamic setting in which a high degree of flexibility is required (e.g. an agent's breakdown is possible, or new agents can enter the system freely), as well as in cooperation systems in which complex tasks must be allocated among agents under some degree of information asymmetry. In distributed network in which agents have the possibility to work parallel towards achieving a goal, MAS is generally found to produce faster and less sensitive [to the type of data] results. Particularly for a global MAS, it is easier to implement new agents (or agents with new capabilities) to an already existing distributed setting than adding new capabilities to a central-agent system. Additionally, central models typically neglect self-interested agents and can, thus, not be fully employed in automated

intercompany settings. Literature also suggests, that it is particularly beneficial to use MAS for simulation purposes of social and life science problems with heuristic implementations, or in geographically distributed operations in which local agents can react faster to its location than a central agent would. Outside the proposed application fields, however, MAS showed to produce inferior results to central benchmarks both in solution quality and running times. Additionally, multi-agent mechanisms place several crucial aspects in the forefront of its development and practicability: first, distributed systems might require sharing private information or using some form of central agent as a mediator (information agent) to run effectively, which is rarely an attractive option for intercompany settings and place additional needs for the technical infrastructure in terms of data transfer security; second, in a distributed system of competing agents, it is found to be generally harder to find a global optimum that gives both a robust and stable solution at a computationally reasonable running time; third, objective functions formulated for a decentralized system in which the decision variables are heterogeneous and, most likely, self-interested agents are significantly different to a centrally formulated objective function and the model solutions can, thus, not be directly compared, which makes it difficult to objectively assess a decentralized model’s effectivity; and fourth, depending on the environmental criteria, multi-agent mechanisms can perform significantly slower as compared to a central system and, if timed, can end in a considerably lower solution quality.

In practice, additional limitations arise as of the technological abilities and security implementations that come with wireless connection. Negotiation structures between firms competing for public resources require agents to share, to some extent, private information. Equipping public agents with private information, however, creates considerable risks for hacking and manipulation. Thus, MAS is, in practice, most likely a topic of concern for internal firm's implementations in order to enhance central production processes. However, the advantages of intra-firm MAS implementations blurry out in the aspect of involving central agents who act as autonomous controllers considering the system's dynamics. For instance, embedding a central controller who automatically and immediately reacts to a machine breakdown by relocating jobs to the remaining agents can make a decentralized system considerably irrelevant.

There are several measures proposed in theory to quantify the quality of a MAS-technology, such as computational efficiency, robustness, solution stability, flexibility, responsiveness and scalability, and several MAS' projects have been proven to be useful for implementation. Such projects implemented include, for instance, TradeBots and similar auction bots used in automated trading platform that are aimed at saving the users time on buying or selling desired goods; Google cars that can, similar to Amazon Robotics, be implemented in

automated delivery systems serving locations far away; and the prototype of the loads allocation bidding platform implemented for VOS Logistics that assists transportation planners in their decision-making.

The practical implementation provided in this paper, a multi-agent flow shop problem, was aimed at using several theoretical frameworks from literature, combining a classical auction mechanism with a cooperation step at the end of the auction and used, at large, an information agent who objectively overtakes the decision of selecting a winning trade. Such hybrid methods are computationally more extensive but at the same time also more efficient in solution quality and have the potential to perform nearly equally good to a central solution, depending on the input size. The inter-agent optimization implemented is a simplification of a *social welfare maximization problem* using an auction (proposal-based CNP) with sub-optimal, unstable local solutions and finalizing the auction's outcome with an extensive cooperative trading phase among the bidder agents. The improvement steps from the cooperative trading involve finding stable solutions that benefit the global system outcome. Thus, the improvement stops, if no further trading benefits the global solution. This does not necessarily end in a locally stable solution as a job exchange can still benefit a single agent at the expense of another agent. As the global solution is a stable one, however, the agents are not equally motivated to further participate in a trade. The intra-agent optimization is a *schedule cost minimization* problem in the offer generation phase using the Campbell, Dudek and Smith heuristic on the agent's current schedule with the job candidate placed in the auction, and it is computed every time a new job must be bid on. The findings show that the cooperative approach is always better than the competitive strategy in terms of solution quality, but is computationally significantly less efficient. At the same time, the solution of the cooperative mechanism also depends on the auction outcome. On general, the solution quality of the framework implemented improves at a higher number of jobs, and ends up to be less than 3% inferior to the central benchmark. Adding threshold values to the bidding decision of the agents slightly improves the auction algorithm, benefiting the cooperation result too. The practical implementation could be further improved, if the jobs allocated are immediately being processed on idle machines while a new job is being auctioned (parallelism of jobs), and if singular tasks of a job could be allocated between the competing agents.

9 REFERENCES

1 S. Kraus (2001), “Automated Negotiation and Decision Making in Multi-Agent Environments”, book, Multi-Agent Systems and Applications, pp.150-172, Springer-Verlag New York 2001

2 Y. Chen, Y. Peng, T. Finin, Y. Labrou, S. Cost, B. Chu, J. Yao, R. Sun, B. Wilhelm (1999), “A negotiation-based Multi-agent System for Supply Chain Management”, pp.2-7, retrieved from http://www.csee.umbc.edu/~finin//papers/agents99-supply-chain.pdf, accessed on 27/03/2016

3 S. Radu (2013), "An Adaptive Negotiation Multi-Agent System for e-Commerce Applications", Ph.D. thesis for the University Politehnica of Bucharest, Faculty of Automatic Control and Computers, retrieved from http://aimas.cs.pub.ro/people/serban.radu/PhDThesisSerbanRadu.pdf, accessed on 27/03/2016

4 I. Rahwan (2004), "Interest-based Negotiation in Multi-Agent Systems", Ph.D. thesis for the University of Melbourne, Department of Information Systems, retrieved from http://web.mit.edu/~irahwan/www/docs/thesis.pdf, accessed on 27/03/2016

5 P. Stone, M. Veloso (2000), "Multiagent Systems: A Survey from a Machine Learning Perspective", article, Autonomous Robotics, Vol. 8, Issue 3, pp.345-383

6 F. Lopes, M. Wooldridge, A. Q. Novais (2009), "Negotiation among autonomous computational agents: principles, analysis and challenges", article, Artificial Intelligence Review (2008), Vol. 29, Issue 1, pp.1-44

7 A. K. Goel, S. L. Gupta, S. Srinivasan, and B. K. Jha (2011), "Integration of Supply Chain Management using Multiagent System & Negotiation Model", article, International Journal of Computer and Electrical Engineering, Vol. 3, Issue 3, pp. 375-378

8 V. Robu, H. Noot, H. L. Poutré, W. van Schijndel (2011), "A multi-agent platform for auction-based allocation of loads in transportation logistics", article, Journal Expert Systems with Applications, Vol. 38, Issue 4, pp. 3483-3491

9 P. Davidsson, L. Henesey, L. Ramstedt (2005), "An analysis of agent-based approaches to transport logistics", Transportation Research Part C: Emerging Technologies, Vol. 13, Issue 4, pp.255-271

10 V. Kaplanoglu, C. Sahin, A. Baykasoglu, R. Erol, A. Ekinci, M. Demirtas (2014), "A Multi-Agent Based Approach to Dynamic Scheduling of Machines and Automated Guided Vehicles (AGV) in Manufacturing Systems by Considering AGV Breakdowns", Proceedings of the 2014 IAJC-ISAM International Conference, pp.1-12

11 F. Feng, Y. Pang, G. Lodewijks, W. Li (2015), "Agent-based negotiation and decision-making for efficient hinterland transport plan", Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp.395-400

12 J. O. Berndt and O. Herzog (2011), "Efficient Multiagent Coordination in Dynamic Environments", Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT), Issue 2, pp.188-195

13 P. Faratin, C. Sierra, N. R. Jennings (1998), "Negotiation decision functions for autonomous agents", article, Robotics and Autonomous Systems, Issue 24, pp.159-182

14 N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge (2001), "Automated Negotiation: Prospects, Methods and Challenges", article, International Journal of Group Decision and Negotiation, Vol. 10, Issue 2, pp. 199-215

15 R. Davies (2015), “Industry 4.0 - Digitalisation for productivity and growth”, European Parliamentary Research Service, Briefing September 2015, retrieved from http://www.europarl.europa.eu/RegData/etudes/BRIE/2015/568337/EPRS_BRI(2015)568337_EN.pdf, accessed on 6th May 2016

16 J. Valenti (2007), “Approximate dynamic programming with applications in multi-agent systems”, Ph.D thesis for the Massachusetts Institute of Technology, retrieved from http://dspace.mit.edu/handle/1721.1/40330, accessed on 6th May 20164

17 F. Wu, S. Zilberstein, X. Chen (2010), “Trial-Based Dynamic Programming for Multi-Agent Planning”, Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10), Association for the Advancements of Artificial Intelligence, pp.908-914

18 B. C. Csáji, L. Monostori (2005), “Stochastic Reactive Production Scheduling by Multi-Agent Based Asynchronous Approximate Dynamic Programming”, In: Pěchouček M., Petta P., Varga L.Z. (eds) Multi-Agent Systems and Applications IV. CEEMAS 2005. Lecture Notes in Computer Science, Vol. 3690. Springer, Berlin, Heidelberg

19 M. Yokoo, K. Suzuki (2002), “Secure Multi-Agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions”, Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '02), part 1, pp.112-119

20 M. Wooldridge, S. Parsons (2000), “Languages for Negotiation”, Proceedings of the 14th European Conference on Artificial Intelligence (ECAI '00), pp.393-397

21 Y. Shoham and K. Leyton-Brown (2009), “Multiagent Systems - Algorithmis, Game-Theoretic, and Logical Foundations”, Cambridge University Press, retrieved from http://www.masfoundations.org/mas.pdf, accessed on 15th May 2016

22 Extreme Tech (2015), “This autonomous medical robot can assist nurses”, retrieved from http://www.extremetech.com/extreme/207130-terapio-autonomous-medical-robot-can-assist-nurses, accessed on 16th May 2016

23 CNET News (2014), “Meet the robots making Amazon even faster”, retrieved from https://www.youtube.com/watch?v=UtBa9yVZBJM, accessed on 17th May 2016

24 Amazon Robotics (2015), “Our Vision”, retrieved from https://www.amazonrobotics.com/#/vision, accessed on 17th May 2016

25 The Robot Report (2015), “30,000 robots now work at Amazon; competing systems emerging”, retrieved from http://www.therobotreport.com/news/amazon-has-30000-kiva-robots-at-work-alternatives-begin-to-compete, accessed on 17th May 2016

26 Multi-Agent Programming Contest (2016), “Aims and Scope”, retrieved from https://multiagentcontest.org/, accessed on 17th May 2016

27 MRG (2016), “How Robotcar works”, retrieved from http://mrg.robots.ox.ac.uk/how-robotcar-works/, accessed on 17th May 2016

28 Google (2016), “Google Self-Driving Car Project”, retrieved from https://www.google.com/selfdrivingcar/, accessed on 17th May 2016

29 A. Uhrmacher and D. Weyns (2009), “Multi-Agent Systems - Simulation and Applications”, CRC Press Taylor & Francis Group: New York, pp.1-554, retrieved from http://www.inf.ufrgs.br/~irmmenezes/home/download/Multi-Agent_Systems_-_Simulation_and_Applications_June_2009.pdf, accessed on 21st August 2016,

30 S. Kumar (2012), “Agent-Based Semantic Web Service Composition”, book, SpringerBriefs in Electrical and Computer Engineering, Springer-Verlag New York

31 A. Chavez and P. Maes (1996), “Kasbah: An Agent Marketplace for Buying and Selling Goods”, Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp.75-90

32 R. H. Guttman and P. Maes (1998), “Agent-Mediated Integrative Negotiation for Retail Electronic Commerce”, Publication, retrieved from http://pubs.media.mit.edu/pubs/papers/amet98.pdf, accessed on 4th September 2016

33 P. R. Wurman, M. P. Wellman, W. E. Walsh (1998), “The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents”, Proceedings of the Second International Conference on Autonomous Agents (Agents- 98), pp.301-308

34 C. Bartolini, C. Preist, H. Kuno (2001), “Requirements for automated negotiation”, retrieved from https://www.w3.org/2001/03/WSWS-popa/paper19, accessed on 10th September 2016

35 C. Bartolini, C. Preist, N. R. Jennings (2002), “A Generic Software Framework for Automated Negotiation”, report, Trusted E-Services Laboratory, HP Laboratories Bristol, HPL-2002-2, retrieved from http://www.hpl.hp.com/techreports/2002/HPL-2002-2.pdf, accessed on 12th September 2016

36 Z. Shi and R. Sadananda (2006), “Agent Computing and Multi-Agent Systems”, Proceedings of the 9th Pacific Rim International Workshops on Multi-Agents, Part of the Lecture Notes in Computer Science book series (LNCS), Vol. 4088, pp.1-843

37 R. Vetschera, M. Filzmoser, R. Mitterhofer (2012), “An Analytical Approach to Offer Generation in Concession-Based Negotiation Processes”, article, Group Decision and Negotiation, Vol.23, Issue 1, pp.71-99

38 G. Lai and K. Sycara (2009), “A generic framework for automated multi-attribute negotiation”, article, Group Decision and Negotiation, Vol.18, pp.169-187

39 T. Sandholm, K. Larson, M. Andersson, O. Shehory, F. Tohmé (1999), “Coalition structure generation with worst case guarantees”, article, Artificial Intelligence, Vol. 111, Issue 1–2, pp.209–238

40 S. Airiau (2010), “Cooperative Games and Multiagent Systems”, article, The Knowledge Engineering Review, Vol. 28, Issue 4, pp.381-424

41 N. Jennings (2000a), “Automated Haggling: Building Artificial Negotiators”, paper, Chapter 1 from PRICAI 2000 Topics in Artificial Intelligence, Vol.1886 of the series

Lecture Notes in Computer Science, 6th Pacific Rim International Conference on AI Melbourne, Australia

42 F. Sadri, F. Toni, P. Torroni (2002), „Dialogues for Negotiation: Agent Varieties and Dialogue Sequences", paper, Chapter 30 from Intelligent Agents VIII, Vol. 2333 of the series Lecture Notes in Computer Science, pp.405-421

43 H. Benameur, B. Chaib-draa, P. Kropf (2002), "Multi-item Auctions for Automatic Negotiation", article, Journal of Information and Software Technologies, Vol. 44, Issue 5, pp.291-301

44 Y. Lengwiler (1999), "The multiple unit auction with variable supply", article, Economic Theory, Vol. 14, Issue 2, pp.373-392

45 O. Skibski, S. Matejczyk, T. P. Michalak, M. Wooldridge, M. Yokoo (2016), "k-Coalitional Cooperative Games", Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent System (AAMAS'16), pp.177-185

46 O. Marey, J. Bentahar, E. K. Asl, M. Mbarki, R. Dssouli (2014), "Agents' Uncertainty in Argumentation-Based Negotiation: Classification and Implementation", Proceeding Computer Science of the 5th International Conference on Ambient Systems, Networks and Technologies (ANT-204), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014), Vol. 32, pp.61-68

47 A. El-Sisi and H. Mousa (2014), "Argumentation Based Negotiation in Multi-Agent System", article, International Arab Journal of e-Technology, Vol. 3, Issue 3, pp.155-162

48 I. Rahwan, L. Sonenberg, P. McBurney (2004), „Bargaining and Argument-based Negotiation: Some Preliminary Comparisons", paper, part of the Lecture Notes in Computer Science book series (LNCS), Vol. 3366, retrieved from http://www.dcs.kcl.ac.uk/staff/mcburney/downloads/pubs/2004/pm-2004-05.pdf, accessed on 14th September 2016

49 A. Kakas and P. Moraitis (2006), "Adaptive Agent Negotiation via Argumentation", Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06), pp.384-391

50 M. Rovatsos, I. Rahwan, F. Fischer, G. Weiss (2005),"Adaptive Strategies for Practical Argument-Based Negotiation", publication, retrieved from

https://www7.in.tum.de/~weissg/Docs/rovatsosetal-argmas2005.pdf, accessed on 14th September 2016

51 C. Carabelea (2002), “Adaptive Agents in Argumentation-Based Negotiation“, Springer Verlag, LNAI series, retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.8238&rep=rep1&type=pdf, accessed on 14th September 2016

52 S. Kraus, K. Sycara, A. Evenchik (1998), “Reaching agreements through argumentation: a logical model and implementation”, article, Artificial Intelligence, Vol. 104, Issue 1-2, pp.1-69

53 S. D. Ramchurn, N. R. Jennings, C. Sierra (2003), “Persuasive negotiation for autonomous agents: a rhetorical approach”, Proceedings of the IJCAI Workshop on Computational Models of Natural Argument, pp. 9–17

54 J. Levin (2006), “Choice under Uncertainty”, paper, retrieved from http://web.stanford.edu/~jdlevin/Econ%20202/Uncertainty.pdf, accessed on 15th September 2016

55 EconPort (2006), “Von Neumann-Morganstern Expected Utility Theory”, retrieved from http://www.econport.org/content/handbook/decisions-uncertainty/basic/von.html, accessed on 15th September 2016

56 S. Kraus (1997), “Negotiation and cooperation in multi-agent environments”, Economic Principles of Multi-Agent Systems, article, Artificial Intelligence, Vol. 94, Issue 1-2, pp.79-97

57 IoT Agenda (2016), “Internet of Things”, article, retrieved from http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT, accessed on 21st February 2016

58 council (2016) ,”The Internet of Things”, retrieved from http://www.theinternetofthings.eu/what-is-the-internet-of-things, accessed on 21st February 2016

59 Cerasis (2014), „E-Commerce Logistics: The Evolution of Logistics and Supply Chains from Direct to Store Models to E-Commerce“, retrieved from http://cerasis.com/2014/04/30/e-commerce-logistics/, accessed on 28th March 2016

60 M. Wooldridge (1997), “Agent-based software engineering”, publication, pp.26-37, retrieved from http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/iee-se.pdf, accessed on 28th March 2016

61 N. R. Jennings (2000b), “On agent-based software engineering”, article, Artificial Intelligence, Vol. 117, Issue 2, pp.227-296

62 J. Levin (2002), “Bargaining and Repeated Games”, paper, retrieved from http://web.stanford.edu/~jdlevin/Econ%20203/RepeatedGames.pdf, accessed on 14th September 2016

63 A. Rubinstein (1982), “Perfect equilibrium in a bargaining model”, article, Econometrica, Vol. 50, Issue 1, pp.97-109

64 P. Kalina (2014), "Agents-based Algorithms for the Vehicle Routing Problem with Time Windows", doctoral thesis, Ph.D. Programme Electrical Engineering and Information Technology, Branch of study: Artificial Intelligence and Biocybernetics, retrieved from http://cyber.felk.cvut.cz/teaching/radaUIB/disertace_Kalina_Petr.pdf, accessed on 20th September 2016

65 R. G. Smith (1980), "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", article, IEEE Transactions on Computers, Vol. C-29, Issue 12, pp.1104-1113

66 FIPA (2002), "FIPA Contract Net Interaction Protocol Specification", retrieved from http://www.fipa.org/specs/fipa00029/SC00029H.html, accessed on 2nd October 2016

67 S. Aknine, S. Pinson, M. F. Shakun (2004), "An Extended Multi-Agent Negotiation Protocol", article, Autonomous Agents and Multi-Agent Systems, Vol. 8, Issue 1, pp.5-45

68 J. Vokrinek, J. Hodik, J. Vybihal, M. Pechoucek (2007), "Competitive Contract Net Protocol", In: van Leeuwen J., Italiano G.F., van der Hoek W., Meinel C., Sack H., Plášil F. (eds) SOFSEM 2007: Theory and Practice of Computer Science. SOFSEM 2007, part of the Lecture Notes in Computer Science book series (LNCS), Vol. 4362, pp.656-668, Springer-Verlag Berlin Heidelberg

69 M. F. Tompkins (2003), "Optimization Techniques for Task Allocation and Scheduling in Distributed Multi-Agent Operations", graduate thesis for the Massachusetts Institute of Technology, retrieved from

https://dspace.mit.edu/bitstream/handle/1721.1/16974/53816027-MIT.pdf?sequence=2, accessed on 2nd October 2016

70 S. Abdallah and V. Lesser (2006), "Learning the Task Allocation Game", Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '06), pp.850-857

71 S. Nouyan, R. Ghizzioli, M. Birattari, M. Dorigo (2005), "An Insect-Based Algorithm for the Dynamic Task Allocation Problem", article, Künstliche Intelligenz, Issue 4, pp.25-31

72 L. Liu and D. A. Shell (2013), "Optimal Market-based Multi-Robot Task Allocation via Strategic Pricing", conference paper, Robotics: Science and Systems, retrieved from http://www.roboticsproceedings.org/rss09/p33.pdf, accessed on 2nd October 2016

73 R. Zlot and A. Stentz (2005), "Complex Task Allocation for Multiple Robots", Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp.1515-1522

74 M. Campos, E. Bonabeau, G. Theraulaz, J.-L. Deneubourg (2000), "Dynamic Scheduling and Division of Labor in Social Insects", article, Adaptive Behavior, Vol. 8, Issue 2, pp.83-96

75 G. Theraulaz, E. Bonabeau, and J.-L.Deneubourg, (1998), "Response Threshold Reinforcement and Division of Labour in Insect Societies", Proceedings in the Royal Society London, Vol. 265, pp.327-332

76 V. A. Cicirello and S. F. Smith (2004), "Wasp-like Agents for Distributed Factory Coordination", article, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 8, Issue 3, pp. 237-266

77 M. B. Dias (2004), "TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments", Ph.D thesis for the Carnegie Mellon University , Robotics Institute, retrieved from http://www.ri.cmu.edu/pub_files/pub4/dias_m_bernardine_2004_1/dias_m_bernardine_2004_1.pdf, accessed on 2nd October 2016

78 H. Afshari, R. D. McLeod, T. ElMekkawy, Q. Peng (2014), "Distribution-service Network Design: An Agent-based Approach", Proceedings of the 47th CIRP Conference on Manufacturing Systems, Variety Management in Manufacturing, Vol. 17, pp.651-656

79 J. Homberger (2010), "Decentralized multi-level uncapacitated lot-sizing by automated negotiation", article, 4OR quarterly journal of the Operations Research Societies, Vol. 8, Issue 2, pp. 155-180

80 C. M. Macal and M. J. North (2006), "Introduction to Agent-based Modeling and Simulation", presentation, retrieved from http://www.mcs.anl.gov/~leyffer/listn/slides-06/MacalNorth.pdf, accessed on 9th October 2016

81 S. Bandini, S. Manzoni, G. Vizzari (2009), "Agent Based Modeling and Simulation: An Informatics Perspective", article, Journal of Artificial Societies and Social Simulation, Vol. 12, Issue 4

82 R. E. Fikes and N. J. Nilsson (1971), "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", journal, Artificial Intelligence, Vol. 2, pp.189-208

83 R. I. Brafman and C. Domshlak (2013), "On the complexity of planning for agent teams and its implications for single agent planning", journal, Artificial Intelligence, Vol. 198, pp.52-71

84 M. Helmert (2009), "Concise finite-domain representations for PDDL planning tasks", journal, Artificial Intelligence: Advances in Automated Plan Generation, Vol. 173, Issue 5-6, pp.503-535

85 M.-A. Puica and A.-M. Florea (2013), "Emotional Belief-Desire-Intention Agent Model: Previous Work and Proposed Architecture", article, International Journal of Advanced Research in Artificial Intelligence, Vol. 2, Issue 2, pp.1-8

86 P. Caillou, B. Gaudou, A. Grignard, C. Q. Truong, P. Taillandier (2015), "A Simple-to-use BDI architecture for Agent-based Modeling and Simulation", Proceedings of the 11th Conference of the European Social Simulation Association (ESSA 2015), Groningen, Netherlands

87 I. Chadès and B. Bouteiller (2005), "Solving Multiagent Markov Decision Processes: A Forest Management Example", pp. 1594-1600, retrieved from http://www.mssanz.org.au/modsim05/papers/chades.pdf, accessed on 9th October 2016

88 C. Guestrin, D. Koller, R. Parr (2001), "Multiagent Planning with Factored MDPs", Proceedings in Advances in Neural Information Processing Systems NIPS-14, pp.1523-1530

89 C. Boutilier (1996), "Planning, Learning and Coordination in Multiagent Decision Processes", Proceedings of the 6th Conference on Theoretical aspects of rationality and knowledge (TARK '96), pp.195-210

90 G. A. Korsah, M. B. Dias, A. Stentz (2013), "A Comprehensive Taxonomy for Multi-Robot Task Allocation", article, The International Journal of Robotics Research, Vol. 32, Issue 12, pp.1495-1512

91 J. Xuesong, T. Sun, T. Qiaoyun and J. Wang (2016), "Flexible Workshop Scheduling Optimization Based on Multi-Agent Technology", article, International Journal of Hybrid Information Technology, Vol. 9, Issue 5, pp.303-310

92 Z. Wu (2005), "Multi-agent workload control and flexible job shop scheduling", Ph.D. thesis for the University of South Florida, retrieved from http://scholarcommons.usf.edu/etd/921, accessed on 7th January 2017

93 Amrita and A. Tripathi (2014), "Multi Agent System in Job Shop Scheduling using Contract Net Protocol", article, International Journal of Computer Applications, Vol. 94, Issue 16, pp.24-29

94 R. M. Zlot (2006), "An Auction-Based Approach to Complex Task Allocation for Multirobot Teams", doctorate thesis, retrieved from https://www.ri.cmu.edu/pub_files/pub4/zlot_robert_michael_2006_2/zlot_robert_michael_2006_2.pdf, accessed on 25th January 2017

95 L. Kumar Sahu, K. Sridhar (2015), "Job Shop Scheduling with Heuristic Algorithms", article, International Journal of Research and Innovations in Science and Technology, Vol. 2, Issue 1, pp.39-45

96 I. Harkan (2010), “Algorithms for Sequencing & Scheduling”, paper from KSU Faculty, retrieved from faculty.ksu.edu.sa/ialharkan/IE428/Chapter_4.pdf, accessed on 19.12.2016

97 V. Modrák and R. S. Pandian (2010), “Flow shop scheduling algorithm to minimize completion time for n-jobs m-machines problem”, article, Technical Gazette, Vol. 17, Issue 3, pp.273-278

98 J. Ferber, F. Michel, A. Drogoul (2001), „Multi-Agent Systems and Simulation: A Survey From the Agents Community’s Perspective”, retrieved from https://www.researchgate.net/publication/229070139_Multi-

Agent_Systems_and_Simulation_A_Survey_from_the_Agent_Community%27s_Perspective, accessed on 21st February 2016

99 P. E. Dunne, S. Kraus, E. Manisterski, M. Wooldridge (2010), "Solving coalitional resource games", article, Artificial Intelligence, Vol. 174, Issue 1, pp.20-50

100 S. Chien, A. Barrett, T. Estlin, G. Rabideau, (2000), "A comparison of coordinated planning methods for cooperating rovers", Proceedings of the 4th International Conference on Autonomous agents (AGENTS '00), pp.100-101

101 M. Hoogendoorn, M. L. Gini, C. M. Jonker (2007), "Decentralized Task Allocation using MAGNET: An Empirical Evaluation in the Logistics Domain", Proceedings of the 9th international Conference on Electronic commerce, pp.319-328

102 J. Rosenschein and G. Zlotkin (1994), “Rules of Encounter: Designing Conventions for Automated Negotiation among Computers”, book, MIT Press, Cambridge MA, USA

103 R. Sikora and M. J. Shaw (1997), “Coordination mechanisms for multi-agent manufacturing systems: applications to integrated manufacturing scheduling”, article, IEEE Transactions on Engineering Management, Vol. 44, Issue 2, pp.175-187

104 F. Lang, A. Fink, T. Brandt (2016), „Design of automated negotiation mechanisms for decentralized heterogeneous machine scheduling”, article, European Journal of Operational Research, Vol. 248, Issue 1, pp.192-203

Appendix A

Definition of **Pareto Optimality**: *a strategy s Pareto dominates another strategy s' if $\forall i \in N, u_i(s) \geq u_i(s')$, and there exists some $j \in N$ for which $u_j(s) > u_j(s')$, i.e. in a Pareto-dominated strategy some player can be made better off without making any other player worse off. A Pareto-optimal strategy (or strictly Pareto efficient strategy), there is, thus, no other strategy $s' \in S$ that dominates Pareto s.*[132]

Definition of **Nash Equilibrium**: *a strategy $s_i^* \in S_i$ is an agent i's **best response** to a strategy s_{-i} (i.e. a strategy of an agent other than i) if $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ $\forall$ strategies $s_i \in S_i$. The strategy $s^* = (s_1^*, \dots, s_n^*)$ is then in a Nash equilibrium if, $\forall$ agents i, s_i^* is a best response to s_{-i} Thereby, s_{-i} denotes a strategy s without agent i's strategy, hence , $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$. A NE is a stable strategy set (i.e. no agent can gain a higher pay-off by deviating from the equilibrium strategy, assuming every agent knows all the other agents' best response strategy) and depending on the uniqueness of the solution we differentiate between a weak and a strict NE (note: a weak strategy set yields a utility that is equally high or higher than for any other strategy, while a strict strategy always leads a strictly higher utility than any other possible strategy move).*[133]

[132] Shoham Y. and Leyton-Brown K. (2009): p.61
[133] Shoham Y. and Leyton-Brown K. (2009): p.62

Appendix B

Job_Agent: select job from database and place to auction; returning jobs place back to database into last position

input: job_database $J = \{job_1, \dots, job_N\}$; optional binary *auction_start*

while $J \neq 0$ **do**
  foreach $job_j \in J$ **do**
    $withdrawal = $ ***SFA_inform***$(job_j, auction\_start)$;
    if *withdrawal==1* **then**
      $J = J[-job_j] \cup \{job_j\}$;
    else
      $J = J \backslash \{job_j\}$;
    end-if
  end-for
end
call ***WCA_coop*;**

WCA_cfp: call-for-proposal function, each WCA computes a bid and decides if bid is being proposed

input: $i \in WCA := \{WCA_1, WCA_2, WCA_3\}$; information job_j; $centralsol_{job_j}$; $memory$

initialize $new\_schedule_i := 0$;
$agent\_Q_i =$***WCA_inform***$(job_j = NULL, i, memory_i)$;
$new\_schedule_i = agent\_Q_i \cup \{job_j\}$;
compute optimal schedule using CDS-Algorithm:
$bid_{job_j} = min\_completion\_time_{new\_schedule_i}$;
if $bid_{job_j} >= centralsol * random(seq(.4, .6), 1)$ **then**
  bid_{job_j} = "$withdrawal$";
end-if
return bid_{job_j};

WCA_inform: winning WCA receives job candidate won to queue into current optimal job schedule

input: optional job_j; $agent$; $memory$

if $job_j != NULL$ **then**
$agent\_Q_i = memory_i \cup \{job_j\}$;
else
return $agent\_Q_i = memory$;
end-if

IA: information agent who computes central solution to a job candidate and stores successfully auctioned jobs

input: information job_j; optional $request$; optional $start$; $memory$

if $start == 1$ **then**
  initialize $current\_schedule := 0$;
end-if
if $request == 1$ **then**
  initialize $new\_schedule := 0$;
  $current_{schedule} = memory$;
  $new\_schedule = memory \cup \{job_j\}$;
  compute optimal schedule using CDS-Algorithm:
  $centralsol = min\_completion\_time_{new\_schedule}$;
  return $centralsol$;
else
  $current\_schedule$ = $memory \cup \{job_j\}$;
end-if

SFA_inform: auction algorithm for cfp-requests and bid selection (winner selection); if all bidders withdraw, inform Job_Agent to place job back to auction later

input: job_j: $processing\_time\_per\_machine_{job_j}$; optional $start$

initialize winner :=NULL;
initialize bid := NULL;
$centralsol = \boldsymbol{IA}(job_j, request = 1, start = 1, memory)$;
foreach $i \in WCA := \{WCA_1, WCA_2, WCA_3\}$ **do**
  $bid_i = \boldsymbol{WCA\_cfp}(i, job_j, centralsol, memory)$;
end-for
if $length(\{bid_1, bid_2, bid_3\} == "withdrawal")) < 3$
then
  $lowest\_bidder = which(min\{bid_1, bid_2, bid_3\})$;
  if $length(lowest\_bidder) > 1$ **then**
    $winner = random\_select(lowest\_bidder)$;
  else
    $winner = lowest\_bidder$;
  end-if
    call ***WCA_inform***$(job_j, winner, memory)$;
    call ***IA***$(job_j, memory)$;
    return($withdrawal$=0);
else
  return($withdrawal$=1);
end-if

WCA_coop: after auction, workcenter agents can trade jobs; trade is successful if new solution is better than old solution

input: none

initialize $agent_i = \{1 \dots 3\}$
foreach $i \in \{1 \dots 3\}$ **do**
  $agent_i$= ***WCA_inform***$(job = NULL, agent_i, memory)$
end-for
foreach $i \in \{1 \dots 3\}$ **do**
  $l = 1$;
  while $(l \leq nrjobs(agent_i))$ **do**
    compute optimal schedule using CDS-Algorithm and
    exchange $jobs_{i=\{1 \dots nrjobs(agent_i)\}}$ with $jobs_{j=\{1 \dots nrjobs(agent_j)\}}$ $for\ i \neq j$
    $l++$;
  end
end-for