Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Multi-agent Systems for Production Planning

Tomáš Staruch

Supervisor: Ing. Vojtěch Janů
January 2023

CTU
CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Staruch Tomáš** Personal ID number: **465829**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Control Engineering**
Study program: **Cybernetics and Robotics**
Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Multi-agent Systems for Production Planning

Master’s thesis title in Czech:

Multiagentní systémy pro plánování výroby

Guidelines:

The goal of this thesis is to get acquainted with the theory of multi-agent systems and to design, implement, and experimentally verify an algorithm for multiagent planning of production. The following steps should be followed:
1. Get acquainted with the theory of multi-agent systems and write an overview of concepts that are important for the task of production planning.
2. Get acquainted with the existing non-complete solution for production planning that is used in Testbed for Industry 4.0.
3. Propose sound and complete algorithm for production planning of a single product.
4. Implement and validate the proposed algorithm on the Testbed production line by executing a complete production of the product.

Bibliography / sources:

[1] Shoham, Y., & Leyton-Brown, K. (2008). Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press. doi:10.1017/CBO9780511811654
[2] FIPA Communicative Act Library Specification. (2002). Retrieved 9 January 2022, from http://www.fipa.org/specs/fipa00037/SC00037J.html

Name and workplace of master’s thesis supervisor:

Ing. Vojtěch Janů Testbed - CIIRK

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **24.01.2022** Deadline for master's thesis submission: **10.01.2023**

Assignment valid until:
by the end of winter semester 2023/2024

Ing. Vojtěch Janů Supervisor’s signature	prof. Ing. Michael Šebek, DrSc. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt	Student’s signature

Acknowledgements

I want to thank Ing. Vojtěch Janů for his outstanding support during this project. I would also like to thank the other members of the Testbed for Industry 4.0 group for their valuable advice, and I would like to thank Ing. Pavel Burget for the opportunity to work on this project at Testbed.

Declaration

I declare that the presented work was developed independently and that I have listed all source of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 10. January 2023

Abstract

The aim of this thesis is to design, implement and experimentally verify an algorithm for multiagent production scheduling. The proposed algorithm uses the principle of chronological backtracking and is an extension of the existing plan, commit, and execute algorithm, which sometimes fails to find a solution. The implementation of the proposed algorithm is done in Java and uses the currently under-development multiagent platform for agent-to-agent communication, database, and logging. The functionality of the proposed algorithm is verified and compared with the existing algorithm in a simulation environment that forms a virtual twin of the Montrac testbed production line. The results are evaluated based on data from six test scenarios.

Keywords: production planning, multiagent system

Supervisor: Ing. Vojtěch Janů

Abstrakt

Cílem této práce je navrhnout, implementovat a experimentálně ověřit algoritmus pro multiagentní plánování výroby. Navržený algoritmus využívá principu chronologického backtrackingu a je rozšířením existujícího algoritmu plan, commit, execute, který v některých případech nedokáže najít řešení. Implementace navrženého algoritmu je provedena v Javě a využívá aktuálně vyvíjenou multiagentní platformu pro komunikaci mezi agenty, databázi a logování. Funkčnost navrženého algoritmu je ověřena a porovnána s existujícím algoritmem v simulačním prostředí, které tvoří virtuální dvojče testovací linky Montrac. Výsledky jsou vyhodnoceny na základě dat ze šesti příkladů.

Klíčová slova: plánování výroby, multiagentní systém

Překlad názvu: Multiagentní systémy pro plánování výroby

Contents

Figures

Tables

Chapter 1

Introduction

This work aims to design and implement an algorithm that uses a multiagent platform to remove current agent-based solutions' shortcomings. Chapter one provides a general description of agent systems, their current uses, agent description, communication between agents, game theory, and methods for modeling, reasoning, and creating multiagent systems. Chapter two describes the Montrac line, a description of the given problem, a description of the central planning solution, and a description of the agent solution. Chapter three describes a comparison of approaches and communication design for the new algorithm. Chapter four describes the implementation of the selected algorithm. Chapter five describes the use cases and the results of the algorithms based on the use cases.

1.1 State of the art

1.1.1 Multiagent systems area of use

Multiagent systems (MAS) have gained much attention from researchers in various fields due to their ability to solve complex problems that can be broken down into smaller tasks. A subtask is assigned to an autonomous agent, which a distributed algorithm can solve. Although MAS have wide applicability, there are still several problems such as agent coordination, task assignment, and security. MAS are used in many different areas. The areas in which they are most commonly used are described below [3], [4].

MAS are mostly used in computer networks. The complexity of these networks continues to increase due to new technologies and the increasing number of devices connected to the network. Agents are used to overcoming this complexity. Given the wide range of applications of MAS in networks, we further divide them into four subcategories:

Cloud computing
: Cloud computing provides access to configurable system resources and computing services. Cloud computing often uses virtualization, meaning that a single physical machine is shared among multiple customers as multiple virtual machines. The complexity of cloud computing is primarily due to the management of cloud resources, communication, and accounting of resources and services used by individual users. These problems can be solved by using MAS [5], [6]. Other uses of MAS in cloud computing include resource monitoring, security, and resource discovery.

Social networking
: The complexity of these networks lies in their dynamic nature. It means that many users join or leave the network in a short time. MAS can be a potential solution to overcome this complexity. Gatti [7] proposed an agent-based model to predict user behavior. The agents collect a dataset of user behavior with which they create a profile. This profile is then used to predict the future behavior of the user.

Security
: The main use of MAS in network security is in the form of an agent-based Intrusion Detection System (IDS). IDS was proposed in [8]. The proposed IDS consists of five agents that work together to find suspicious behavior in the network and take appropriate mitigation actions when such behavior is detected.

Routing
: Routing is finding a path for packets between source and destination with certain metrics. Using agents for routing was one of the first applications of MAS. Agent-based routing inspired by ants behavior was described in [9].

Another area where MAS are commonly used is mobile robotics. Here, agents are used to cooperate and coordinate the robots and to plan their trajectories. For example, in this paper [10] they have used MAS for collision avoidance for mobile robots. In recent years, MAS have also been used in large cities for transport systems and traffic management or for freight distribution planning. Another use is in building management where, for example, agents manage heating.

The last thing MAS are often used for is modelling complex systems. Agents are used here for their flexibility and autonomy, which greatly facilitates the modeling of such systems. The main advantages of using agents, according to D. Helbing [11]: 1) the possibility of aggregation and combination with other modelling methods, 2) flexibility in assumptions for MAS modelling, 3) flexibility in predefined knowledge, as agents can acquire knowledge by learning from the environment, 4) the possibility to explore emergent behaviour due to the proactivity of agents.

1.1.2 Multiagent systems in production planning

Frequent changes in customer needs and requirements require reconfigurable and adaptive production systems. In the last decade, extensive research has developed on deploying multi-agent systems in several industrial environments.

Using MAS for production planning is similar to modeling a complex system. An example of such a system is the newspaper industry, where production planning using multiagent systems has been applied in Germany. This planning had better results than static centralized optimization.[12]

Currently, the use of MAS in production planning is common in supply chain management. One example is the use of agents to allocate orders efficiently according to production line capabilities [13]. Another example is the use of MAS for the collaborative forecasting and replenishment planning process between trading partners[14]. A more detailed overview of the use of MAS for supply chain management has been written in this paper[15].

1.2 Multiagent systems history

The evolution of multiagent systems in their field started simultaneously with the distributed computational environment approximately in 1980. At that time, distributed computing began to be used in LANs for expert systems, systems that are supposed to provide advice or recommend solutions to a given situation. The multiagent systems got widespread recognition in the mid of the 1990s, when local networks were connected to the Internet. The rapid growth of multi-agent systems began when massive open distributed systems became more common. The growth is due to the belief that agents are a proper approach to exploiting these systems. A multiagent system is a natural tool for understanding and designing an artificial social system, with each agent representing an object in a larger system. [16]

1.3 Description of multiagent systems

1.3.1 Agent description

Multiagent systems combine multiple computational entities, known as agents. Various authors have proposed different definitions of agents. The following definition is taken from [1, p. 4]: "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to achieve its delegated objectives." An agent is autonomous if it can decide actions for itself without the intervention of other systems or humans. An example of such behavior is depicted in Figure 1.1, an agent takes sensory input from the environment and, based on the input, produces an action that affects the environment. Usually, an agent has a set of actions that could have some preconditions. The interaction with the environment is usually an ongoing, non-terminating one. An agent usually does not have complete control over its environment. It means that an action can fail.

Figure 1.1: Behaviour of an agent in its environment, [1]

1.3.2 Description of the environment surrounding the agent

By definition Agent description, an agent is situated in some environment. Russell and Norvig, in their book [17, p. 40], classified environment properties by five attributes. These attributes are described below.

Fully observable vs. partially observable

In a fully observable environment, an agent has access to complete information about his surroundings at any given time. An environment can be partially observable because a sensors are inaccurate, or some parts of the environment state are not observed or cannot be observable at all. For example, in city traffic, one car cannot predict the actions of other cars with certainty. Most real-world environments are partially observable. The more observable environment is, the easier it is to define a capable agent.

Deterministic vs. stochastic

An environment is deterministic if any action has one guaranteed effect. There is no uncertainty about the state that will result from performing an action. The non-deterministic environment is stochastic. It is more challenging to define an agent in a stochastic environment. If the environment is partially observable, it may appear stochastic to the agent; this is especially true if the environment is complex. Therefore, it is better to consider this property of the environment from the agent's perspective.

Episodic vs. sequential

In an episodic environment, the agent is working in discrete episodes. The episodes are independent of each other, which means that agent actions in the episode depend only on the episode itself. Many classification tasks are episodic, for example, product classification on the production line or detection of defective parts on an assembly line. In some cases agent decision can affect future decisions in a sequential environment. Typical representatives are games like chess or real-life environments like city traffic. Episodic environments are easier for agent development because the agent makes decisions based only on the current episode.

Static vs dynamic

A static environment does not change over time. The static environment is simpler because an agent does not need to worry about time passing and the change in his surroundings. In a dynamic environment, other processes continuously change the environment. An agent in this environment is periodically asked what it is going to do; if it hasn't decided yet, it counts as doing nothing. An example of a dynamic environment is again city traffic. If the environment is not changing with time, but the agent utility function (a function that describes agent performance) is, then the environment is called semidynamic. An example of this environment could be chess with a clock.

Discrete vs continues

An environment is discrete if there is a finite number of percepts and actions of the agent. An example of a discrete environment is chess. An example of a continuous environment is city traffic. The speed and position of cars are continuous values.

1.3.3 Intelligent agent description

By the agent definition mentioned above Agent description, an agent can be any simple control system, and still, all aspects of the definition will be valid. An example of such a control system often used in the literature is a thermostat. A temperature sensor is situated in a room, and if the output from the sensor is below the set threshold, then the heating of the room is triggered. From our perspective, we do not view these simple systems as agents, so we add more concepts to the agent definition to define "intelligent agent". The following list of concepts was suggested in [18].

Reactivity

Reactive intelligent agents can perceive their environment and react appropriately in a timely fashion to changes that occur in it to satisfy their delegated objectives.

Proactivity
: Proactive intelligent agents can exhibit goal-directed behavior by taking the initiative to satisfy their design objectives.

Social ability
: Social able intelligent agents are capable of interacting with other agents and possibly humans to satisfy their design objectives.

Furthermore, in this thesis, we assume that "intelligent agents" are rational [19]. A rational agent behaves in such a way that he tries to maximize the expected utility value given the information at his disposal.

1.4 Communication between agents

1.4.1 Speech act theory

Interaction in the social ability concept is not only the ability to exchange bitstreams of information - an agent has to be able to interact in the human sense. An agent must be able to understand and reason about the goals of others in order to be able to negotiate to achieve its designed goal, for example. Linguistics and the philosophy of language are generally concerned with describing the communication. Of interest to us in this field is the notion of speech act, on which communication between agents is based. The speech act is an act performed by a speaker that not only carries the information but also performs an action. [20] For example, the sentence "I like tea, could you please make me some?" is a speech act because the sentence conveys not only information but also a request to make tea. Commonly included types of speech acts are apologies, complaints, compliments, responses, greetings, invitations, refusals, requests, and thanks. [21]

According to J. L. Austin [20], a speech act consists of three different components:

locutionary act
: An locutionary act is to pronounce a sentence. The meaning of this sentence is all the verbal, social, and rhetorical meanings of the sentence, which are based on its semantics, syntax, and verbal aspect.

illocutionary act

An illocutionary act is the intended meaning of a locutionary act by the speaker. For example, if the locutionary act is the sentence "Someone is talking there!" then the illocutionary act is the command "Silence!".

perlocutionary act

A perlocutionary act is the real consequence of the locutionary and illocutionary acts. In other words, it is something that someone does or becomes aware of.

Performatives are an important type of speech act for us because they are used in agent-to-agent communication. J. L. Austin first mentioned performatives in his book [20], and then the term was further elaborated by J. R. Searle in the article "how performatives work" [22]. In performative sentences, the action that the sentence describes is performed directly with the utterance of the sentence. Examples of such sentences used by Austin are "I bet you sixpence it will rain tomorrow" or "I name this ship the "Queen Elizabeth". Moreover, in explicit perfomatives, the illocution force is directly stated in that sentence, e.g., promise, order, request, etc. Searle considers performatives to be what he calls declarations. An utterance is a declaration if "the successful performance of the speech act is sufficient to bring about the fit between words and world, to make the propositional content true." [22, p. 97]

1.4.2 Multiagent communication languages

Most multiagent communication languages (ACLs) are based on the speech act theory described by Searle. One of the consequences of basing ACLs on speech act theory is the possibility of end-users to reason about agents' behavior and thus understand their behavior. In other words, the human client is able to predict the consequences of his actions, how the agent will behave. The main reason for using Searle's theory is the fact that this theory defines acts in an algorithmic way, which makes them easier to implement on computer systems. Finally, speech act theory fits well with the used models of agents that maintain their internal representation of the environment and their desires (goals).At least two standardizations have been developed based on speech act theory - KQML and FIPA-ACL [23].

The KQML (Knowledge Query and Manipulation Language) standardization was developed as part of the ARPA Knowledge Sharing Effort, which aimed to develop techniques for creating large-scale knowledge bases that are shared and reusable [24]. Then Cohen and Levesque offered a formal semantics for KQML in [25]. KQML is a communication language and is designed as a universal standard for agent-to-agent communication. KQML provides an extensible set of performatives that specify the types of communication that agents can have with each other. In addition, KQML contains some set of policies that specify a legal sequence of communication acts. KQML is predecessor of FIPA-ACL.

FIPA - ACL is a standard for agent communication developed by the Foundation for Intelligent Physical Agents (FIPA). The principles of this language are based on KQML, but it tries to define semantics and communication protocols better. Again, we have a set of messages that are fundamentally similar to performatives and are called communication acts [26]. FIPA - ACL, unlike KQML, has a closed set of acts, so new communication acts can only be created by combining basic acts. The components of the communication act model characterize both the reasons for which the act is chosen (rational effect, or RE) and the conditions that must be met for the act to be planned (feasibility preconditions, or FPs). An example of communication using the FIPA protocol is shown and described below:

Figure 1.2: Example of simple communication exchange between two rational agents

In the Figure 1.2, we assume that agents behave rationally, so they try to achieve their goal by sending messages (acts). The first act sent from agent one is a call for proposal, which starts the negotiation process. This act may include a simple query about whether the other agent can perform an action, such as whether he can put wheels on a car. Or this act may contain some query, expecting to receive a proposal, for example, at what price the other agent is able to assemble the car. To this the second agent replies with a propose act, as already mentioned this is a response to a query from the first agent. The agent also uses this act to communicate its intention to take action as soon as the preconditions are met. The first agent responds with Accept Proposal, confirming that it agrees to the proposal and may send additional information with this act, such as exactly when the action is to be performed. The second agent may additionally reply Confirm to this act, confirming to the first agent its desire to perform the action.

The following Figure 1.3 shows a state machine that represents the possible evolution of communication between two agents. This state machine is only used for a simple description of the communication, in real communication it can theoretically be infinite. One ongoing communication (message exchange) cannot give rise to another, in other words, there can be no branching, only one branch is always selected.

Figure 1.3: Example of communication state machine between two rational agents, CFP - call for proposal, NU - not understood, P - propose, AP - accept proposal, RP - reject proposal, C - Confirm, D - disconfirm

The appearance of the state machine depends on the implementation of the agents in a given system, for example, communication may terminate after an act propose. Agent one will again initiate communication by asking for a proposal. Agent two send propose or may send not understand if it cannot parse the message or, for example, expects different act content. Agent one can reject the proposal and thus end the entire communication, agree with the proposal or make a counter proposal. Depending on the agents' decisions, the communication continues. Disconfirm can be sent if for some reason the agent is no longer able to fulfill what was agreed upon.

Agents using these ACLs may only inform other agents of their actions or may perform more complex message exchanges, for example, for coordination, negotiation, or cooperation purposes. Coordination is a communication process between agents that results in rational behavior of the system as a whole with the aim of achieving an outcome in an optimal way. Negotiation is a communication process used to reach an agreement between agents. In other words, it is a negotiation of how to achieve the goals of all the agents involved. Cooperation is a communication process in which agents try to agree on a common solution to a given problem.

1.5 Modeling techniques

We have explained what an agent is, in what environment it can be situated, and have described the principle of communication between agents. We now outline various techniques for modeling, reasoning, and building multiagent systems. These techniques include knowledge of logic, probability theory, game theory, and optimization. The following information on the various techniques is taken from a book on multiagent systems by Shoham and Leyton-Brown [27].

1.5.1 Distributed Constraint Satisfaction

One of these techniques is Distributed Constraint Satisfaction. A constraint satisfaction problem (CSP) is defined by a set of variables and domains for each of the variables. Furthermore, the problem is defined by constraints on the values that the variables can take simultaneously. Agents work cooperatively to solve such a problem. The agents try to find the values of the variables that satisfy all constraints or decide that there is no solution. One well-known problem that belongs to CSP is graph coloring.

1.5.2 Distributed optimization

Distributed optimization is another technique that builds on the previous one by adding a global objective function that we want to optimize. This technique can be further divided into:

Distributed dynamic programming for path planning
Path planning aims to find the cheapest route using weighted links between the two states in the graph. For example, this approach can be used in internet networks or for car navigation where states represent the cities, links are available roads between cities, and the link weight is the time needed for crossing the road.

Distributed solutions to Markov Decision Problems
The optimal policy in Markov Decision Problems for a single agent is described by mutually recursive Bellman equations. For multiple agents, the global action is replaced by a vector of local actions of all agents. A more detailed description of Markov Decision Problems

is in the aforementioned book on multiagent systems by Shoham and Leyton-Brown [27] Appendices C.

Optimization algorithms for scheduling or assignment problems

The scheduling problem involves a set of agents and a set of time slots. Each agent has a task with a deadline and needs access to a shared resource (some time slots) to complete the task. The solution to the scheduling problem is to assign some time slots to the agents.

When solving the assignment problem, we have agents, objects, and a possible pairing of an agent and object, where each pair has a value. The algorithm finds a solution where each agent has an assigned object (optimally, all assignments maximize the sum of values).

Coordination via social laws and conventions

An excellent example of social law is the traffic rule. The rule limits the agent's options (stop at a red light), but by limiting other agents, there can be benefits. The agent can better guess the actions of other agents.

1.6 Game Theory

Game theory is one of the main tools used to design and model multi-agent systems. Game theory describes the optimal behavior of self-interested agents in a multiagent environment. Self-interested agents have their own goals, which can help other agents or harm them. This theory has two main branches, noncooperative game theory and cooperative (coalitional) game theory. In noncooperative game theory, it is not always the case that the agents' goals conflict. The division of the branches is based on whether we build the model for each agent separately (noncooperative) or the agents form a group (coalition), and we build the model for it. Other well-known categories are auction and multiagent resource allocation. [27]

1.6.1 Noncooperative game theory

Normal form representation

There are many possible formal representations of games. The most basic representation is the normal form (or matrix), as most more complex representations can be reduced to this. This representation can be described by a matrix, where for each state of the world, described by the combination of all players' actions, the player's utility is determined.

Environments where the state of the world also depends on chance (Bayesian games) can also be described in this form. Example of this representation, described by a matrix, for the game Rock Paper Scissors is shown below in Table 1.1

	R	P	S
R	(0,0)	(-1,1)	(1,-1)
P	(1,-1)	(0,0)	(-1,1)
S	(-1,1)	(1,-1)	(0,0)

Table 1.1: game of Rock Paper Scissors described using Normal Form Representation. The rows describe the actions of the first player and the columns describe the actions of the second player. The first value in parentheses always describes the first player's utility

The choices that players make in the game are called pure strategy. One strategy can consist of multiple actions. The set of strategies for each player is called a strategy profile. Not all games can be described by a pure strategy, so there is also a mixed strategy, where a player chooses a pure strategy based on a probability distribution.

Solution concepts

If we are to choose the optimal strategy in a single-agent environment, we aim to maximize the payoff of this agent. The situation is much more complex in a multi-agent environment where each agent aims to maximize his profit. The choice of one agent's best strategy depends on the other agents' strategies. This problem is solved by finding specific subsets of outcomes that are somehow significant. These subsets are called solution concepts.

Pareto optimality

An outcome a is pareto optimal outcome, if that there is no other outcome b where one player would be better off and all other players have at least the utility as in a.

Social welfare

An outcome that maximizes the sum of all players utilities.

Nash equilibrium

Nash equilibrium is a concept where the optimal outcome of a game is where there is no incentive to deviate from the initial strategy. That is, the Nash equilibrium is stable.

A pure equilibrium contains only pure strategies (in Table 1.2, the top left box and the bottom right box are pure eqilibrium - if one player changed his decision, he would lose), but it does not always exist, for example, in the game Rock Paper Scissors Table 1.1. Every game with a finite number of players and strategy profiles has at least one Nash equilibrium. This equilibrium may also contain mixed strategies.

	M	F
M	(2,1)	(0,0)
F	(0,0)	(1,2)

Table 1.2: Battle of sexes game described using Normal Form Representation

Correlated Equilibrium Each player chooses his mixed strategy independently of the others in a standard game. If we look again at the battle of the sexes game Table 1.2, this approach can lead to a state where the utility of both players is zero. Therefore, it is better for both players to choose the same strategy. The choice of this strategy is correlated with some random input, such as a coin flip. For every Nash equilibrium, there exists a corresponding correlated equilibrium.

Extensive form representations

Dynamic (sequential) games have a finite or infinite horizon. These games can also be described using a normal form representation, but this is not practical. The strategy in a dynamic game must reflect all possible situations that can be encountered in the game. Thus, we must have a predetermined action to play for every possible situation in the game. The problem with this representation is that the number of possible strategies grows exponentially. A suitable representation for finite dynamic games is the extensive (tree) form representation. This representation can be further divided into Perfect-information extensive form games and imperfect-information extensive form.

Perfect-information extensive-form games are represented by a tree (from graph theory). Each node represents a player's decision, each edge represents an action, and the leaves describe the outcome (utilities for each player). An example of a game in this representation is the Sharing game. The first player proposes a division of items and then the second player approves or rejects it. A tree describing of this game is shown in the Figure 1.4.

Figure 1.4: The Sharing game described by extensive-form game representation

Games with imperfect information in extensive form are represented in the same way as perfect games, and contain information sets (infosets) in addition. If the actions of another player are unobservable, then the states following these actions are indistinguishable and belong to the same infoset.

Every finite game can be represented as an extensive form game with imperfect information. If we did not know the action of the first player in the mentioned Sharing game, then the tree would contain one infoset. The new tree is shown in the Figure 1.5.

Figure 1.5: The Sharing game with imperfect information described by extensive-form game representation

1.6.2 Coalitional game theory

As mentioned above, coalition games are about forming coalitions between agents. A coalition is a set of agents who try to maximize the gain of the whole coalition. Finding the optimal coalition structure is an NP-hard problem, since the number of all possible coalitions grows exponentially as the number of players increases.

Each agent gains some worth from the total coalition profit independently of the actions of the other players. The solution to the game is a set of worth allocations for all players. We describe here two basic solution concepts.

Shapley value
: Shapley value describes each player's contribution to the coalition. This value is fair, meaning that players with the same contribution receive the same amount and a player who has no contribution receives zero.

Core
: Core "is a set of efficient allocations upon which no coalition can improve" [28]. Core is stable and can be considered analogous to the Nash equilibrium in non-cooperative games. The problem with the core is that it is not always unique, it can be empty, and it is also not easy to compute.

1.6.3 Multiagent resource allocation and Auctions

Multiagent resource allocation

Multiagent resource allocation is a process in which we try to distribute a certain number of objects among agents. Each agent has some own object rating, a preference.

The allocation of objects among agents is measured by social welfare. Allocation can be either efficient, where we give the object to the one who values it the most, or fair, where the goal is to make the agent who is worst off the best off.

Auctions

An auction is a protocol that allows agents to express an interest in a resource, and this interest is used to make decisions about resource allocation and agent payments [27]. Auctions can be held for one item (single-good), for multiple identical items (multi-unit) or for multiple different items (multi-item). Agents can also buy, sell, or both (exchange).

We describe here the most common example of auctions, agents buying one or more items. We start with the simple case of a single item. Auctions can be divided into five categories, depending on how they are conducted:

English
: The most famous is the English auction. The auctioneer starts the bidding at the starting price. The bidders then report their rising bids, and when no one else raises, the last bidder is the winner and gets the item for that price.

Japanese
: In the Japanese auction, the auctioneer starts at the starting price and all bidders stand. The auctioneer gradually raises the price and when the price exceeds the amount the bidder is willing to pay, he sits down. The last one standing is the winner and gets the item for that price.

Dutch
: In a Dutch auction, the price of the item starts high and gradually decreases. At some point, the bidder reports "mine" and gets the item for the actual price. There is a problem in this auction because the agent does not know the valuations of the other agents. There is a trade-off between the probability of winning and the amount the agent pays. There is no dominant strategy in this type of auction.

First price sealed bid
: In this auction, bidders write their valuation on a piece of paper. The auctioneer will then sell the item to the highest bidder at the price they wrote. This type of auction is similar to a Dutch auction in terms of agent behavior.

Second price sealed bid
: In this auction, bidders write their valuation on a piece of paper. The auctioneer will then sell the item to the highest bidder at the price written by the second highest bidder. This type of auction is similar to English and Japanese auctions in terms of agent behavior. In this auction, writing the agent's valuation as the final price is the best option, this way the agent will achieve his best result.

In sealed bid auctions, we can introduce a reserve price below which the item will not sell.

If we are going to auction for several different items, we can use the Vickrey-Clarke-Groves (VCG) auction. This auction is a sealed bid auction, so each bidder will report their valuation for the items without knowing the valuations of the others. The auctioneer then allocates the items in a socially optimal way and charges each bidder the damage he caused to the others. It is similar to a second price sealed bid auction and therefore the best option for the agent is a true valuation of the items.

Chapter 2

Problem Description

This chapter describes the Montrac production line, a description of the task and its problems, a description of the currently used MES solution and a description of the first mutliagent solution.

2.1 Description of the Montrac test line

Montrac is the name of the testing line located in the Testbed for industry 4.0 at the Czech Institute of Informatics, Robotics and Cybernetics Czech Technical University. This line is a smaller version of a real industrial production line and is used to test new technologies and principles. For this reason, the resulting algorithm was tested on this line. The Figure 2.1 shows a layout of the line.

Table 2.1: Diagram of Montrac production line

Five industrial robots of three different types from KUKA are on the line:

- The R1-R3 robots are **KUKA KR 10 Agilus-2** - High-speed industrial 6-axis robots programmed in KRL. These robots are used on the production line for quick pick and place operations. It is possible to automatically change the tools used or manually attach an electric screwdriver. The manual is available here [29].
- R10 is **KUKA LBR iiwa 14** - A modern cooperative 7-axis robot that is programmed in Java. This robot is used on the production line for pick and place operations. It can be used for more complex operations thanks to its 7 axes. At the same time, this robot can work in collaborative mode with human. The manual is available here [30].
- R20 is the **KUKA Cybertech KR 8 R1620** - A fast industrial 6-axis robot with a higher payload capacity programmed in KRL. This robot is used on the production line for pick and place operations. It has a camera mounted on it for more accurate localization of parts. This robot is the only one that can unload new parts from the stock shelf. The manual is available here [31].

The Montrac monorail transportation system connects the robots. It consists of rails called tracks, track curves, track switches (letter "C" in the figure), and positioning units (letter "S" in the figure) that ensure the exact position of shuttles in working cells. The Montrac monorail on the production line is used to transport parts or product between robots. The parts are placed in shelves that are transported by Automated Guided Vehicles (AGV). Imported parts are unloaded onto the shuttles using the R20 and distributed between the individual robots.

These robots communicate with Siemens PLC via Profinet. This communication includes safety variables, control variables, and the acquisition of diagnostic data from the robots. The PLC uses MQTT protocol to send diagnostic data and OPCUA protocol to communicate with the control system (scheduler/planner). An HMI panel is used to monitor the current status of the production line.