DESIGN AND SIMULATION OF A MULTI-AGENT AUTONOMOUS ROBOT SYSTEM FOR INDUSTRIAL FACILITIES

by

Sarp Baran Özkan

B.S., Industrial Engineering, Galatasaray University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering
Boğaziçi University
2006

ACKNOWLEDGEMENTS

I am grateful to my thesis advisor, Prof. H. Levent Akın, for his guiding, comments, support and endless patience in this study.

I would like to thank to Associate Prof. Ümit Bilge and Associate Prof. Eşref Eşkinat, my thesis committee members, for their kind participation.

Thanks to the AILab people, especially Kemal, Hatice, Çetin, Buluç, Serdar and Fuat, for their friendship and their valuable support to this work.

I am so grateful to my family for their endless love and support.

I have to mention my employers' support. Without their support I couldn't finish my study.

ABSTRACT

DESIGN AND SIMULATION OF A MULTI-AGENT AUTONOMOUS ROBOT SYSTEM FOR INDUSTRIAL FACILITIES

In this study we aim to observe the usage of multi-agent autonomous robots in industrial facilities and the related alternative application details of this usage.

The purpose of the utilization of multi-agent autonomous robots in manufacturing plants is to pass over the bottlenecks existing in the usage of Automated Guided Vehicle (AGV) systems, which are being used in transportation of materials in such systems. Multi-agent autonomous robots are a current research issue in the field of robotics.

Unlike AGV systems, determining their routes without using fixed transmitters, autonomous robots will be able to move liberally between loading and unloading points. Mobility in manufacturing environment very frequently causes the blockage of the route of AGV systems or gives way to the impossibility of using such environment for other activities. In case of interruption of paths by fixed obstacles or dynamic obstacles described as workers, robots will be able to continue towards their targeted destination either by maneuvers of avoidance or by determining a new route. Consequently this situation will provide a new facility for a more flexible manufacturing.

For modeling the environment in our study, we have applied Webots, a 3-D simulation program that, both with its graphical interface and its C, C++ and Java programming units, is used to create a test environment for the applications of units, various production environment layouts, robotic elements trials, robot architecture and different path planning and task assignment strategies. In our studies, robots and production environment have been modeled and programmed. Conclusions of simulations have all been evaluated to measure the proposed system's responsiveness to flexible manufacturing.

The files and instructions necessary to run the simulation program are available in the appendix CD which is attached inside the back cover of the thesis. Information related to the content and usage of the appendix CD is in the Read.Me text file.

ÖZET

ENDÜSTRİYEL TESİSLER İÇİN ÇOK-AJANLI BİR OTONOM ROBOT SİSTEMİ TASARIMI VE SİMÜLASYONU

Bu çalışma çoklu-etmenli otonom robotların fabrika tarzı üretim birimlerinde kullanımı ve alternatif uygulama detaylarını gözlemlemek amacı taşımaktadır.

Çoklu-etmenli otonom robotların üretim tesislerinde kullanımı halen bu tesislerde malzeme taşıma işlerinde kullanılan Otomatik Güdümlü Araç (OGA) sistemlerinin mevcut bazı darboğazlarını aşabilmeyi hedeflemektedir. Çoklu-etmenli otonom robotlar, robotik araştırmaları alanında güncel bir araştırma konusudur.

Çoklu-etmenli otonom robotlar, OGA sistemlerinden farklı olarak sabit bir hat, yahut sabit vericiler kullanılmadan kendi rotalarını çizerek yükleme ve boşalma noktaları arasında daha liberal bir biçimde hareket edeceklerdir. Üretim ortamındaki hareketlilik OGA sistemlerinin sık sık yolunun kesilmesine yahut bu alanların başka işler için kullanılamamasına yol açmaktadır. Sabit engellerin yahut dinamik engeller olan işçilerin rotayı kesecek biçimde yer değiştirmesi durumunda robotlar kaçınma manevraları yaparak yollarına devam edebilecekler yahut yeni bir rota saptayabileceklerdir. Bu durum daha esnek bir üretim yapılanmasına da olanak sağlayabilecektir.

Ortamı modelleyebilmek için üç boyutlu simülasyon programı Webots, hem grafik arayüzü hem de C, C++ ve Java programlamasına uygun birimleri ile, çeşitli üretim ortamı yerleşimleri, değişik robotik elemanların denenmesi ile robot mimarisi ve farklı yol bulma ve görev atama stratejilerinin uygulanmasına uygun bir test ortamı oluşturmada kullanılacaktır. Yapılan çalışmalarda, robotlar ve üretim ortamı modellenmiş ve programlanmıştır. Yapılan simülasyonların sonuçları değerlendirilerek önerilen sistemin esnek üretime tepkiselliği değerlendirilmiştir.

Simülasyon programını çalıştırmak için gerekli dosya ve talimatlar tezin arka kapağı içindeki ek CD'dedir. Ek CD'nin içeriği ve kullanımına ilişkin bilgi ise Read.Me metin dosyasındadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
BBC	Behavior Based Control
DW	Differential Wheels
FCFS	First Come – First Served
GPS	Global Positioning System
IR	Infrared
NPF	Nearest Pickup First
RRT	Rapidly-exploring Random Trees
TP	Type Priority of parts
θ	Angular error in degrees
D	Linear distance from departure to goal point
$C_{r,\,i}$	Cost of Robot r to the Node i
$\omega\,(t)$	Angular velocity of the robot at time t

1. INTRODUCTION

Robotics made its debut in factory applications as reliable machines for repetitive works where accuracy and speed in repetition is needed. Early applications like robotic manipulators, i.e. welding or painting robot arms, were only capable of executing a given task without any “intelligent” action [1]. The advance in electronics and AI gave robotics a new horizon: Now, autonomous mobile robots applications are everywhere: Many tasks like search, rescue, soccer competitions, etc. could be accomplished with success. Industry could also profit more from autonomous robot applications.

The very first one among the application fields of robotics is industry and manufacturing plants. Manufacturing plants are large workshops, where different production techniques are applied to obtain products. There are many production areas. From little workshops to great mass production manufacturing bands, workspaces need workforce. Human workers, animals, or machines may provide this workforce. Most common use of workforce are human workers since humans can think and use their arms and body for making various moves and they provide an intelligent platform for acting against materials. Animals are also a kind of common workforce. They are powerful, and are generally capable of providing more effort than humans. Scanning crop fields or turning a mill is a better-suited work for horses or cows; or insects may be used for spying. But animals need to be guided by humans to provide a useful workforce. Another possible workforce is machine: They are both powerful and useful for routine works. Standard products need precision and repetition. However, repetitive jobs are difficult for human workers. Machines could provide more production in a given time. But machines are not intelligent workforce. A machine without human intervention cannot produce anything. Human intelligence must be used every time to organize the workforce. So, only workforce and intelligence together may create production or service.

However, intervention of intelligent machines could change this situation. Intelligence combined with power and precision can solve many manufacturing problems. Fixed, semi-intelligent machines are already being deployed in manufacturing plants: Robotic arms, manipulators equipped with sensors are common examples for such

applications. Replacing humans need more features like making intelligent decisions and mobility. Consequently, autonomous mobile robots are suitable candidates: Fully mobile, decision making, individual machines with manipulators, capable of interacting with external world and other objects.

Manufacturing plants with their production lines consist of a large number of complex components. These production areas have well defined layouts, where every little space has a utility and free spaces are very precious [2]. Mobile vehicles, like forklifts, AGVs are already operational for a long time, but no fully autonomous systems are widely used in these facilities.

The use of autonomous mobile robots in industry could change many habits. Autonomous mobile robots can replace or co-operate with human workers in industry. Living and sharing the working area with robots would be easier for humans, but this is not the same case for robot builders. Making and programming robots to work out in industrial environments, where many constraints and many alternatives are possible, has some hard work to do.

1.1. Aim of the Thesis

The motivation behind this thesis is to explore the possibility of transformation of factory automation systems from AGVs to fully autonomous robots. Thus by implementing a navigation system and a task allocation algorithm for an multi-agent autonomous mobile robot team, where robust, flexible, allowing both low level reactive behaviors and high level strategies to be defined. Also, exploration of possibility to define a set of metrics that can evaluate multi-agent performance in industrial facilities for different task requirements and exploration of the important points to propose a method for autonomous robot utilization are considered.

In this study, we will first revise the related notions about autonomous mobile robotics, then introduce our research topic, discuss the theoretical issues, give and explain a proposition for multi-agent missions in industrial facilities and then make a simulation for testing, and finally discuss the results.

In the following pages, we will try to examine some of the possible autonomous robot applications in industrial workspaces. We will take an industrial place model and put the constraints, then configure our robots and simulate the environment, then measure the robots' performance for some given tasks.

1.2. Thesis Outline

A brief information about robots, mobile robots and multi-agent systems is given in the Section 2. Basics of behavior-based (or, bottom-up approach) robotics, motor schemas, brief information about multi-agent robot robotics and control strategies for multi-agent robot teams is presented also.

In Section 3, Automated Guided Vehicles (AGVs), a competitor system for industrial places, is presented in detail. Comparative information about AGVs and autonomous robot systems is given.

Section 4 discusses a possible autonomous robot design and needed hardware and software features and needed AI capabilities of the robots.

In Section 5, a simulation software and a robot model to represent and simulate the job shop model for multi-agent usage is presented.

Section 6 contains information about a job shop model of a manufacturing plant where robots are proposed to work in. Physical appearance and different aspects of the environment is given. Then, a model for job shops is presented.

Section 7 explains the proposed features of a multi-agent system to work on industrial facilities. Algorithms are explained in detail. The proposed features and robot settings are performed in experiments with a simulator in the following sections.

Section 8 presents the task assignment algorithms used by the robot team. Market-driven approach for multi-agent task assignment is explained. Also, different

configurations for task priorities are explained. Different cases are proposed to make the experiment design to test some possible task allocations due to the needs of a flexible manufacturing system.

We give in the Section 9, we discuss about the performed experiments in simulator for explained cases in given Section 8. The settings and the obtained results by test runs for different task-to-robot configurations and task assignment algorithms are observed. Comments for these test results are presented. Also, the statistical validation of the success of the robot team is checked.

Finally, in Section 10, a conclusion is given by summarizing the work done, by citing the principles obtained from previous sections and experiment results and by pointing to possible future work.

2. AUTONOMOUS ROBOTS AND MULTI-AGENT SYSTEMS

2.1. Definition of an Autonomous Mobile Robot

Autonomous robots have at least three common characteristics to be stated: They are self-sufficient in energy sources, computational resources and decisional capabilities.

An autonomous robot must need no power supply cables, must possess all the sensors, actuators, CPUs on board and must have behaviors, a descend of computational capabilities, to act without any outer intervention.

Autonomous robots can react and develop behaviors in unpredicted situations, which make them different and superior in complexity than other machines.

Autonomous Mobile Robot (AMR) is a vast application area for robotics: It comprises both mechatronics research for hardware components to resolve mobility problems and artificial intelligence research for software components to resolve autonomy problems.

In production areas, mechatronics and manipulators have already their implementations. But intelligent robotics, especially artificial intelligence applications are still laboratory applications in majority for today.

Use of purposeful intelligent mobile robots can have a great impact in industry. For dangerous or hazardous places, such machines can be used.

2.2. Definition of Collaboration

“Collaboration” is a behavior for problem solving. A collaborating group consists of elements such as robotic agents, where every member works for solving the problem by helping the other ones or just by executing its own process.

2.3. Definition of Collective Robotics

"Collective Robotics" or "Autonomous Collective Robotics" studies robot-robot and robot-environment interactions for the solutions of the tasks where the task must be solved by a group of robots. In collective robotics, typically the solutions are decentralized [4].

Autonomous collective robotics is inspired considerably much by collective intelligence (or so-called collective intelligence) demonstrated by social insects, like ants, bees, or some kind of beetle and worm colonies. These colonies are decentralized and such approaches in collective robotics seems to be a promising way to solve problems which are hard to tackle using classical control methods [7].

2.4. Robot Control Paradigms

The connection between these the widely acknowledged primitives of robotics is commonly used to describe robot control paradigms [3], [40]:

- Sense
- Plan
- Act

Sense category covers the functions which use the information that a robot's sensors provide. *Plan* category includes the functions which create a number of assignments to carry out using the information that a robot's sensors or inner knowledge provide. *Act* category covers the functions that create output instructions for a robot's actuators.

Table 2.1. Robot primitives defined in terms of inputs and outputs [3], [40].

Robot primitives	Input	Output
Sense	Sensor data	Sensed information
Plan	Information (sensed and/or cognitive)	Directives
Act	Sensed information or directives	Actuator commands

2.4.1. Reactive Paradigm

Utilizing the *sense* layer's outputs as the inputs of *act* layer, reactive architectures [3] disregard the plan phase (Figure 2.1). The robot can rapidly react to environmental changes as it links the sensory input to *actuation* layer with a very small calculative overhead. That total disregarding of the plan phase wasn't proper for general-purpose robots came to be noticed soon, even though reactive controllers' outputs are remarkable. Quick reaction, little usage of memory and taking the world for self model (no necessity for conceptual inner world model) are amongst the benefits of reactive architectures [40].

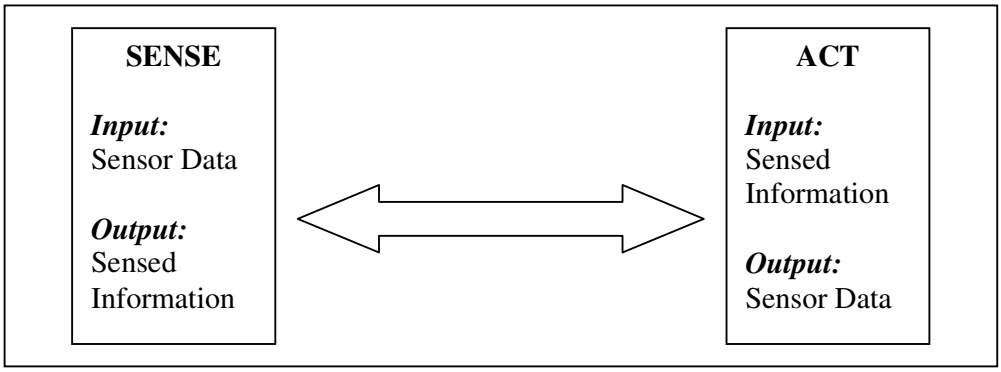

Figure 2.1. Relationships among robotic primitives in reactive paradigm [3].

2.4.2. Deliberative Paradigm

"Hierarchical Paradigm" and "Top-down Architecture" are other names for "Deliberative Architecture". *Sense* layer is used to sense the world, sensory information and knowledge base are used through *plan* layer to create inner model of the worlds and the instructions that the planning layer through *act* layer create are followed by the robot in such an architecture. Providing the robot with the ability to take strategic action following a particular aim is the main benefit of the deliberative paradigm. The high memory and calculation requirements, rare and slow reaction, and the risk of malfunction if the real world and inner model for the world are not synchronized, are the main drawbacks of this paradigm [1]. Moreover, "Closed World Assumption" (everything must match the robot's inner model for the world that supposedly includes all the necessary knowledge for the robot) affects deliberative architectures [40].

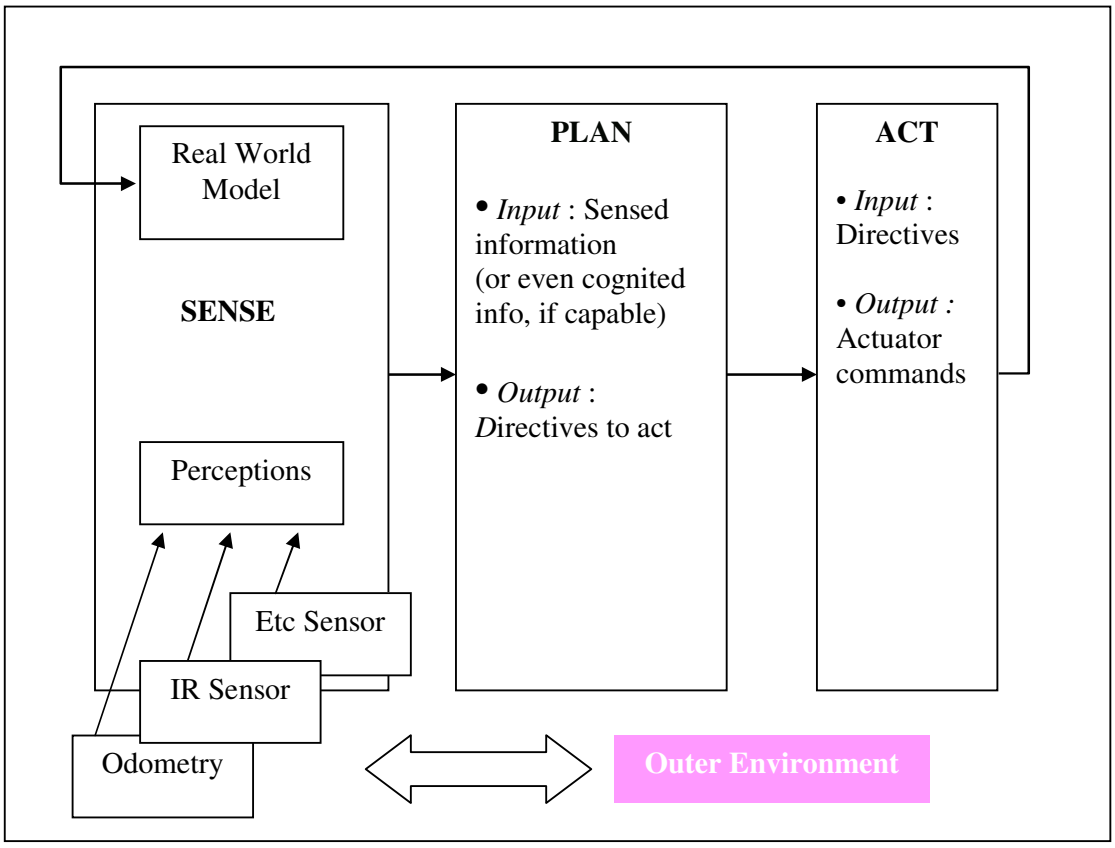

Figure 2.2. Relationships among robotic primitives in deliberative paradigm.

The Figure 2.2 shows the relationships between layers and the relations between outer world and the robot.

2.4.3. Hybrid Paradigm

The "Hybrid Paradigm" [2], [3] combines the advantages of deliberative and reactive paradigms together so that a robot can utilize planning for break the assignment down into suitable sub-assignments, and then decides on the proper behavior for their completion. Later, in accordance with the reactive paradigm, these behaviors are carried out. Because sense-act couple together follows planning like in the Figure 2.3, this paradigm is also called *Plan, Sense-Act.* A combination of deliberative and reactive paradigms creates the sense layer in the hybrid paradigm. Both the planning layer and behaviors can utilize the sensor data. The state algorithm it utilizes gives name to this third layer. Predicting ones are named d*eliberators,* and the ones whose memory includes knowledge about the past

are named *sequencers* [3]. While the hybrid approach makes up for the main drawbacks of reactive and deliberative paradigms, a proper middle layer is hard to devise.

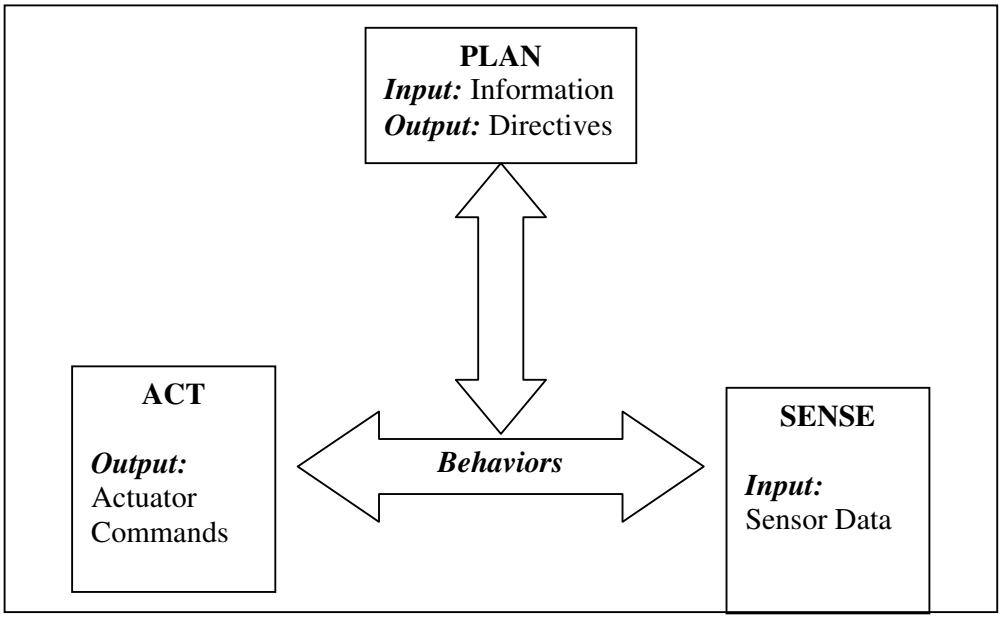

Figure 2.3. Relationships among robotic primitives in hybrid paradigm (from Murphy [3]).

2.4.4. Behavior-based Architectures

In "Behavior-based Architectures", computing processes are distributed among the behaviors (sensor-actuator couplings) [1]. Various behavior combination algorithms include cooperation and competition based algorithms and subsumption algorithm [1]. When a thorough and precise model of the real world is not available, behavior-based architectures are better suited.

2.4.4.1. <u>Subsumption Architecture</u>. In the "Subsumption Architecture", which was developed by Brooks [1], there is a hierarchical organization of behaviors in which the outcome of a behavior at a higher level can *subsume* the outcomes of the behaviors in lower hierarchical levels. Higher-level behaviors may access the lower level behaviors but lower level behaviors are not aware of the higher levels. Behaviors decide when to become active according to the sensor readings. Since the philosophy of reactive architectures is "The world is its own model", there is no internal world model or any kind of abstraction based on sensory information.

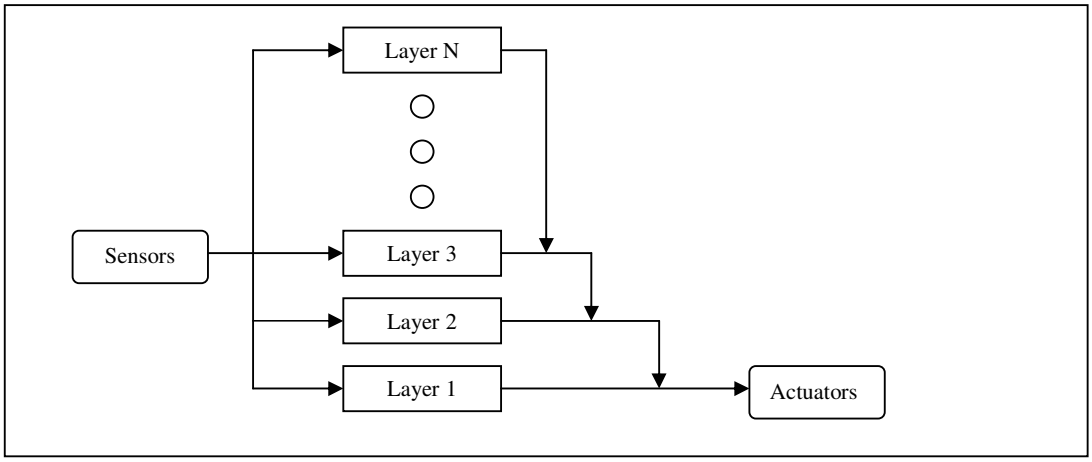

Figure 2.4. Subsumption architecture (from Meriçli [40]).

2.4.4.2. <u>Schema-based Behaviors</u>. Use of schema theory is another widely used approach in behavior-based architectures [3]. In the "Schema-based Behaviors", the behaviors consist of perceptual and motor schemas and a perceptual schema is embedded within each motor schema. The way of processing the sensor data is encoded in perceptual schemas, and in motor schemas, response is represented in a uniform format. The final action of the robot is determined by performing a weighted vector sum of individual behaviors [1].

2.5. Autonomous Robots Basics: Problems to Solve

2.5.1. Localization for Robots

"Localization" term is used for robots' process for acquisition of knowledge about their positioning, with respect to their environment. For this, they have to learn about their environment via their sensors or they can get this information via an external source such as a supervisor. If robots localize themselves by their own, they have to designate some reference points to find their actual position. By knowing the difference of distance and orientation to reference points, robots' could do localization. These differences can be measured by distance sensors (generally IR or ultra-sound) or can be calculated by odometry (actuators' number of turn).

2.5.2. Path Planning and Navigation

"Path Planning" is the key AI activity for reaching the goals. As the robots are mobile, reaching one point from another requires a plan for determining the path to follow during displacement. If, a map of the environment is available to the planner, then the planner can come up with a path from the current position to the goal in terms of goals, obstacles, etc. consisted of line segments and arcs. This type of path planning is described as "Offline" [5]. Once a path is calculated the necessary commands are sent to the low-level actuators such as electric motors, which are turning robots' wheels or legs to move them in order to displace the robot in conformity to the planned route.

2.5.3. Obstacle Avoidance

During navigation, a robot must avoid collisions with any obstacles to protect first the obstacle first and second, itself. Obstacles can be static obstacles, which cannot move, or dynamic ones, which can displace as the robots. Static obstacles are easier to avoid but avoiding dynamical ones requires quick maneuvers for the robot, because obstacle's position and speed may vary in time.

A robot can sense obstacles with their sensors. Sensor's data provide the percepts for robots. These data can be transformed to knowledge and/or a reaction. To avoid obstacles, this knowledge or reaction will command the actuators of the robot for making suitable maneuvers.

Obstacle avoidance is related to path planning: The path to be followed must be collision-free (preferred in general and mandatory in manufacturing environments), so the planned path should contain no obstacles while planning (if the obstacles positions in time are already known) or the path plan must be revised after encountering to avoid the obstacle. So, obstacle avoidance maneuvers generally triggers a new path planning for next step of the robot.

2.5.4. Computer Vision, Image Processing and Pattern Recognition

Vision in robotics can be used for providing data to external users (i.e. a sentry robot's camera can be used by its supervisors) or it can be used even for the robot's perception systems. Robotic systems use computer vision in order to recognize objects and environments [5].

Computer vision's main principle is to process the viewed data to acquire information. For this purpose, the view is divided into pixels where each pixel is defined by its color, brightness and other related values. These data must be perceived in terms of its shape and color value, but first, the data must be cleared of noise, i.e. incompatible data with the desired perception. This point is a wide research area, because the "considered" noise notion depends on the deployed principles of "what to be perceived" and how to define the objects. Image processing depends on these questions [22].

For autonomous robots, not only a single image but, sequences of images must be processed to asses dynamical change of environment since the robot is mobile. From processed images, a shape or a pattern can be recognized, and then an action plan could be devised using this information. Data acquired from sequenced images of vision can be also used like sensory data, i.e. odometry or range data could be retrieved from vision.

2.5.5. Making an Intelligent Autonomous Robot: Combination of AI Techniques

Using autonomous mobile robots depends on combined utilization of the AI techniques cited above in accordance with the requirements and goals of the robot, the suitable ones among these techniques can be chosen and applied.

For industrial environments, using suitable intelligent robots means using the proper AI techniques. Path planning of robots inside industrial spaces must consider the minimum cost path from one point to another for reaching the goal, together with obstacle avoidance [10]. The physical appearance of the robot must carry the needed devices for applying these techniques such as a different selection of sensors, actuators, locomotion devices, manipulators and the most important, the main processor.

Using computer vision is optional: Vision can be used for path planning and navigation; it can also be used as sensory and/or a communication device. It can provide valuable addition for above-mentioned AI features, but vision needs huge computational and memory resources. Data gathered by cameras depends on the chosen camera type. As the camera abilities become richer, acquired data becomes bigger and more detailed. This provides an advantage on one hand, but requires more computation and also applying different AI techniques to filtering the needed and desired information from the data. So, use of computer vision is a critical decision in designing an autonomous intelligent robot. This depends on goal and environment requirements. It can be replaced with other sensors or replaced by other means of sensors and communication devices.

2.6. Definition of Multi-agent Robotics

Groups of two or more mobile robots working together are generally called as teams or even *societies* of multiple robots, or more precisely, they are called as "Multi-agents" [3]. A multi-agent robot team can be useful for many different tasks. For example an object that a single robot cannot manage to move can be handled by many robots [30], or in tasks where an area has to be covered, a larger area can cleared in a shorter time interval more effectively. To this end, Murphy [3] in his book gives examples of robots to explore the outer space or scan a minefield. A multitude of simple low cost robots working together can achieve what a single complex structure robot cannot manage on its own. This special kind of multi-agent robots is called "Swarm" [8]. A more familiar example for these can be bees and ants. Another advantage provided by a number of robots is "redundancy". If a robot malfunctions, gets lost or fails the others can continue with the mission.

Multi-agent teams draw great attention in experimental robotic studies and in tournaments. In such competition organizations like RoboCup and FIRA, real robot teams play football against one another. There are also other games where Multi-agent robots compete. The difficulty in these competitions lies in the setting up of multi-agent robot teams. A team can be formed by placing several robots next to one another each designed and programmed as an individual robot, however existence of some capacity and talent beyond that of an individual, shall increase the chances of success of the team as a whole

[9]. When working with Multi-agents the coverage and difficulty of the study has relevance to the issues listed below:

- The difficulty in team formation: The characteristics of a problem to be solved by multi-agent robots have to be determined very well.

- Interference: With the increased number of robots in a finite space, robots may inadvertently disrupt one another's actions.

- Communication: The manner and level of communication to be established between agents is a matter of discussion. The design and mission of multi-agents determines the extents of this communication. For example communication between robots contains a lot of information on a broad band, whereas for swarm robots this is in a level more limited and does not require much computation. It still is possible for a work to be accomplished without much computation. For example let us assume a large number of small robots such as sardines employed underwater for research. Sardines propagate in mass formation without communication in sound or signal. If we are to think in terms of imitating nature, this formation must be possible to imitate with very limited computation. In other words it may not be an absolute must to employ electronic and complex devices for every task [1].

- Social dependency: Another important design problem is that how much independent action from others each individual agent shall perform in a multi-agent group. Again this can be adjusted according to the nature of the mission. It is possible to experiment with fully dependent or wholly independent models for the same mission. While working fully independent for a goal in a limited space could cause interference, working independently in open field can yield more successful results [3].

The choice of architecture in multi-agent design determines the approach to be taken in solving a problem with multi-agents. However it must be kept in mind that this design in general is always done for a single robot. For example each robot is designed compliant either with reactive or hybrid/deliberative architecture. However when more than one individual come together, their behavior may differ from that of when they are alone. This is what is called "Group Dynamics" or "Community Psychology", which can be observed

in robot groups too. If the behavior of each agent can be taken as a vector, adding of vectors more than one can result in another totally different vector. Murphy [3], comments in his book on this as, the sum of emergent behavior providing a different social emergent behavior. Apart from this, new architectures in multi-agent team design considering robots as a complete team as opposed to individual robots are also under investigation. Some researchers design a complete architecture directly for multi robots. An example is Lynne Parker's ALLIANCE architecture [39].

In conclusion, for multi-agent design, there is not yet a total consensus for team designs, architectures, etc. Researchers still published their new findings and multi-agent research grows bigger. Here, we will limit our research with the following features of multi-agents, which comprises control, cooperation and goal setting topics.

2.7. Advantages and Disadvantages of Using Multi-robot Systems

Even though there are numerous disadvantages, rich alternative solutions and advantages of multi-agent robots make working with them worthwhile. So, in working with multiple robots one has to formulate cost effective solutions and cope with the disadvantages to benefit from the advantages. Main titles are as follows:

2.7.1. Advantages

- Increased overall system performance by teamwork: If the task can be divided into subtasks, having multiple robots is an advantage because certain agents can be specialized for certain subtasks and the overall performance could be increased [4].

- Improved Task Ability: Certain tasks cannot be accomplished by a single robot. An obvious example is the case of physical helping each other between robots. For example an individual robot cannot carry a heavy object whose weight is over its capacity but multiple robots could carry the same object together.

- Improved Sensing Capability: More robots mean more sensors. Together with communication between team members, a multi-agent system can obtain more information

by sharing sensor data. This provides a better performance of the system than a singular robot, especially for exploration tasks.

- System robustness: More robots mean redundancy of the system. If one robot fails, one of its teammates can replace it. As the system's capability is distributed, one robot need not carry all the complex hardware as single robot. Carrying more hardware means more vulnerable robots. Replacing one complex robot capable of executing multiple tasks with multi-robot teams consisting of homogenous members specialized in one task could provide a more robust system.

2.7.2. Disadvantages

- Increased system uncertainty and difficulty of control: More robots mean more subsystems to be controlled. If the system is supervised, the supervisor must deal with multiple robots, which is computationally costly. If not under control, some robots could be lost or do unexpected actions.

- Interference of robots: Instead of cooperation; multiple robots could compete with each other. Especially in distributed control systems, if the communication is poor or not well designed, robots could interfere while sharing the same places.

- Cost of the whole system: Even if working with many smaller robots can be less complex in terms of hardware issues rather than working with a single complex robot, multiple robots means multiplied costs. Even for having smaller robots, smaller hardware elements cost higher than ordinary devices. Design steps are also costly, because a designer must calculate the interactions between the members of the robot team and make its design under many constraints.

2.8. Multi-robot System Parameters

The study of multi-agent systems focuses on systems in which many intelligent agents interact with each other. The agents are considered to be autonomous entities, such as software programs or robots. There are many different issues to be considered in order to characterize a team of agents. These considerations can be listed as reliability,

organization, communication, spatial distribution, congregation, and performance. Arkin [1] discusses the different properties of multi-robot teams varying from social behavior to inter-robot communication and from performance issues to ethological considerations.

- Reliability: If a system acts in a desired -correct- way in a given situation over time, then the system can be considered to be reliable.

- Social Organization: Heterogeneous societies should be developed if there is a demand for specialized skills. Some examples can be multi-level hierarchical structures, loosely structured mobs, and dominance systems.

- Communication: There are two major aspects in communication. One of them is information content, which can be described as the limit of the messages in societies. For example, for ants, there are ten to twenty different chemical signals. The other aspect of communication is mode. In different animal societies, different range of communication mechanisms including chemical, tactile, infrared, and electric communication are used [24].

- Spatial Distribution: It is very important for activities such as foraging for food. If a resource is evenly distributed, it is better for the agents to form individual, non-overlapping foraging ranges instead of roosting and foraging together [24], [3].

- Congregation: There are several strategies to keep the society remain together such as defining a colony location as a predefined meeting point, generation of a loud noise by a number of similar agents (lekking), distinctive calls, and specific assembly calls by a single agent.

- Performance: Specific metrics are required to effectively evaluate societal system performance. Speedup is one of the useful metrics that measure the performance of a team of N robots relative to N times the performance of a single robot.

In Murhpy [3] and Arkin [1], we can find taxonomies, which can categorize different multi-agent robotic systems. For characterizing a multi-agent robot team, the most underlined properties are:

- Team size (number of robots).
- Team composition (composition of agents themselves, i.e. homogeneous or heterogeneous).
- Team communication and its topology (each robot's ability to communicate directly with other team members), (the pathways by which communication can occur).
- Team configuration (flexibility regarding the structure and organization of the team).
- Team unit processing capability.
- Communication bandwidth (amount of communication available).

After multi-agents team characteristics, while organizing multi-agent teams, we have to state 3 definitive organization characteristics of these teams: Their control characteristics, their way of cooperation and their way of working for the goal [12].

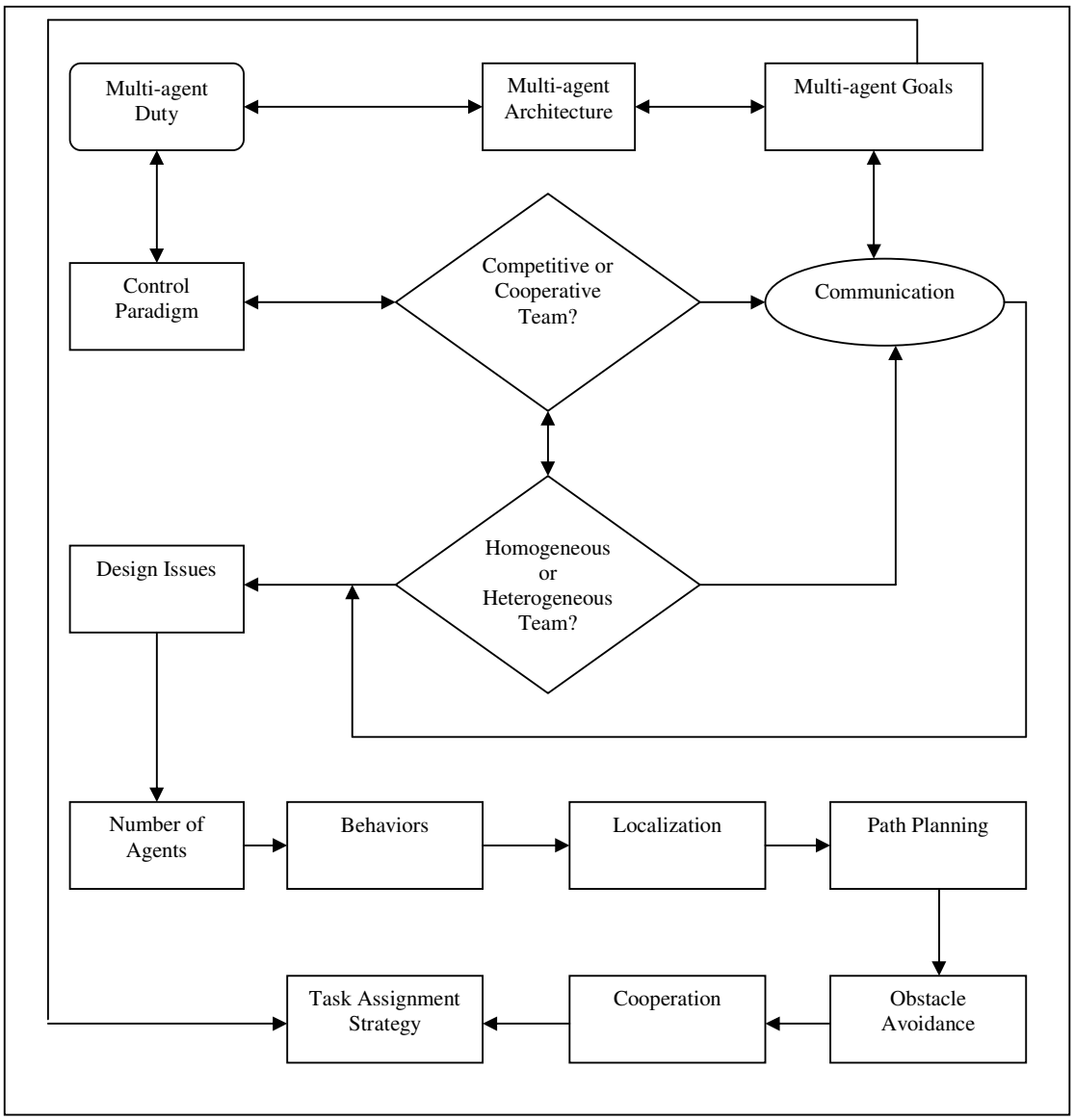

Figure 2.5. Relationship of multi-agent robot team design issues.

In the Figure 2.5 above, we can see the relations between multi-agent robot design issues. To make a goal-oriented design, all off the issues has to be considered via its relations with the other features. So, the design of such a multi-agent team depends much on task definitions and the final goal.

2.8.1. Control of Multi-agents

For multi-agent control, two distinctive control strategies are used: *centralized control and decentralized (distributed) control*. These two different strategies will be explained in detail below.

The configuration of the robots utilized determines the control strategies as well. For instance in missions where a large number of simple hardware robots are to be employed, a choice of distributed control is preferred as in insect colonies such as bees or ants. As the number of robots employed increase and their field of application widen, distributed control as opposed to centralized control shall be preferred. However in environments where the numbers and field of application is more limited, or places where robots have a wide range of communication possibilities, centralized control may provide an advantage for a full control over the whole of multi-agent group. Intelligence of the multi-agents may be a combination of these two approaches.

2.8.2. Centralized Control

"Centralized Control" is a control state-of-the-art where members of the multi-agent team are directed by supervision [7]. In centralized control, robots are communicating with a central computer, which gives directions to the agents. In general, the central computer assigns tasks to robots and it distributes goals. In the extreme case of centralized control, robots take every action direction from the central supervisor computer, and they are more like remote controlled drones than robots. This extreme case is more like a tele-operated system, where the central computer plays the role of operator. In such cases, communication between the agents is only a functionality to be used to execute the duties and the actions of the supervised process and to transmit the necessary information by the supervisor. MIROSOT robot soccer competition is a good example of centralized control systems.

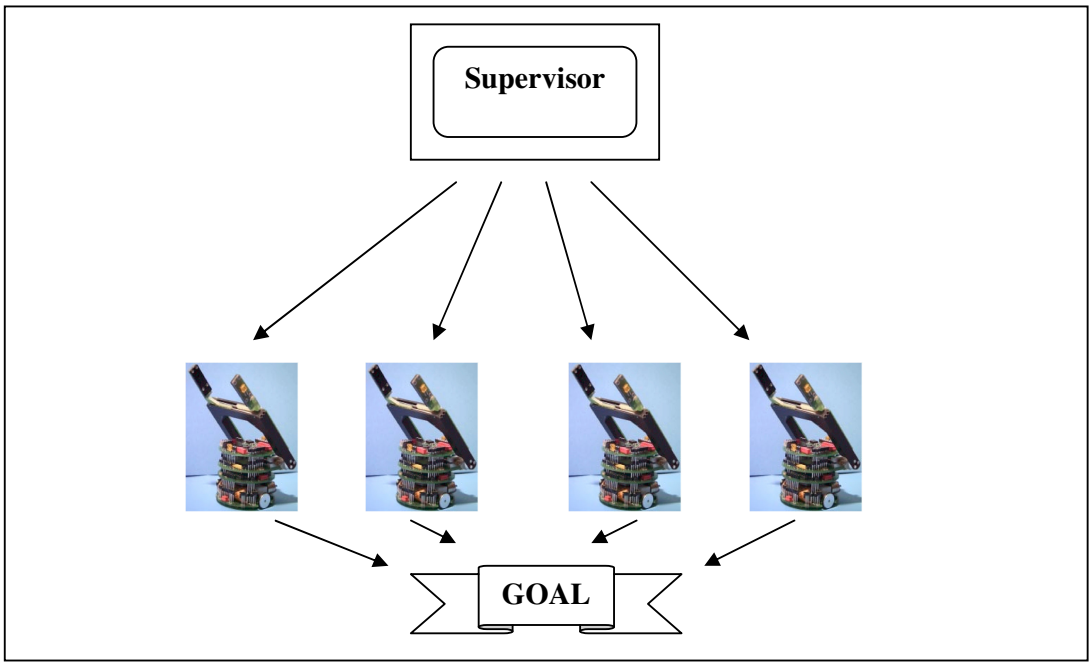

Figure 2.6. Centralized control.

2.8.3. Distributed (Decentralized) Control

In “Distributed (Decentralized) Control”, there is no supervisor, which overlooks for assignments or goals of the robots. Agents have their own control paradigm and their control code on their processor; they behave independent of each other without taking commands from a center.

This is a more liberal way of solving problems associated with a task [22]. In very vast areas where supervisor coverage is not always possible, i.e. space exploration, search and rescue, etc., distributed control has the advantage of the independence because agents can work even when no supervision exists. But the agents’ actions or task allocations are difficult to predict: Only the emergent social behavior could be foreseen but not all the agents are controllable in a desired moment if a totally distributed control is implemented on a multi-agent team.

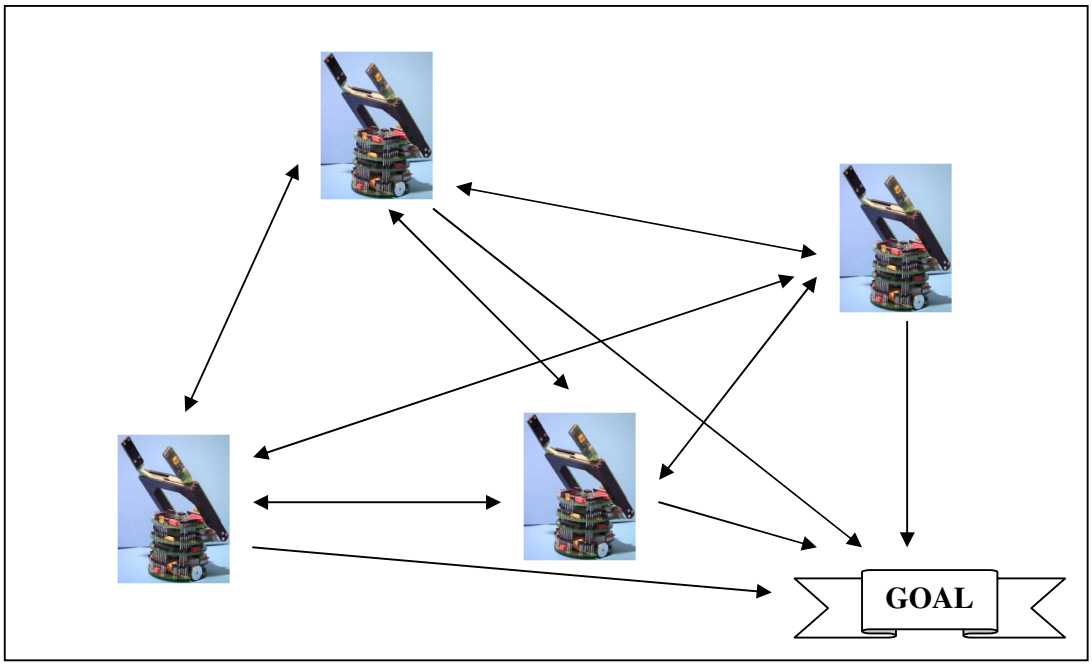

Figure 2.7. Distributive (Decentralized) control.

2.8.4. Multi-agent Control Paradigm Choice

Certainly, control regimes vary in a range from totally centralized to totally distributed ones [13]. A regime could profit from both alternative's advantages. Examples of different levels of centralized regimes can be seen in robotic soccer categories: In MIROSOT competition, small sized robots are being controlled by a central computer: Even motion commands come from the supervisor. However, in middle-sized RoboCup soccer competition, the central supervisor only gives the task assignments or general playing strategy. The agents in RoboCup have their sensors, onboard devices and programs to evaluate sensor data and execute their given tasks by the supervisor. In this case, the supervisor does not control low-level reactive tasks, but only deliberative layer's high-level strategic issues.

Another category of RoboCup challenges is four-legged robot league, where Sony Aibo robots are used. Here, a distributed control is implemented and instead of having strategies by a central command center, agents construct a group behavior, which can be measured by team performance metrics. Robots are also learning how to play by different

methods to improve their group behavior. Köse and al. [34] introduced Market-driven approaches for these issues, which will be reviewed in the following sections.

For a suitable design of multi-agent team for industrial places, the choice of control regime is crucially important.

2.8.5. Cooperation of Multi-agents

The concept of "Cooperation of Multi-agents" indicates how the interaction between robots should be while they are working together for a certain objective. According to Murphy [3] multi-agent robots can cooperate in two ways: In "Active Cooperation", each robot recognizes the other and they work together. Here the important point to emphasize is, this cooperation does not necessarily require communication. In "Passive Cooperation", the robots do not recognize each other but the work they do serve in some way for other robots to accomplish the task.

Determination of those active or passive cooperations depends on the devices and sensors capabilities found on the robot [16]. For one robot to recognize the other, sensors such as camera or wireless communication and the ability to process the information obtained from such devices are required (to form behavior) [11]. However passive cooperation does not require these and it is simpler to realize.

Another matter for thought on the subject of cooperation is "Physical Cooperation". For example, some goods that cannot be handled by one robot alone could be managed by a team of two or more robots. Especially if the multi-agent team is heterogeneous, for example in marsupial teams which consist of at least one large and one small robot, they are designed to cooperate especially in the physical sense.

As a further example, in the robot football league for the four-legged agents, Sony AIBO robot dogs can realize wireless communication with one another. Besides, robots can carry different patterns of one team blue and one team red and thus identify their teammates and opponents and this allows for active cooperation. MIROSOT, on the other hand is a different league and robots there do not know what their team mates are doing,

however a robot with his front obstructed and unable to see the goal post passes the ball to his team mates, again the other robot does not know whether he is about to receive a pass or what his team mate's action will be.

2.8.6. Goal Settings for Multi-agent Robots

Another dimension of multi-agent robot teams' characteristics is how the robots work to achieve a goal. The robots can share the same explicit single goal, or they can have different individual goals.

For the same goal, different architectures could have different settings. We should note that, for a defined goal in a given environment, the robot and team design together with the group control paradigm and architecture could be more suitable than another architecture and paradigm. Although different architectures could achieve the same goal, one of them may be more efficient or suitable or less costly for the mission.

2.8.7. Homogeneity and Heterogeneity in Robot Teams

In a group of robots, members can be all the same or they may be different. Collections of robots are called "Homogenous" if their members are all identical. On the other hand, in "Heterogeneous" teams, at least one member is different in terms of its hardware or its software capabilities [24]. Thus, even though the members of the groups are identical in hardware, the group can be heterogeneous if the individual behaviors of the members are different for a given mission.

Generally, designing multi-agent teams as homogeneous ones can reduce the design cost, the manufacturing cost and the programming cost for the entire team [3]. In case of swarm robotics, where a large number of small robots operating in conformity to the reactive paradigm and often inspired by social insect colonies like ants or bees, homogeneity is used. Heterogeneous teams often require mostly hybrid/deliberative paradigms. For an example, in case of marsupial robot teams, a big robot called *mother* robot and several smaller robots called *daughter* robots are used.

In Murphy [3], the degree of heterogeneity is given by Balch's "Social Entropy" metric, which measures the disorder. This metric is inspired by the second law of thermodynamics and expressed by the following equation (2.1):

$$\textbf{\textit{Het (R)}} = -\sum_{i=1}^{c} p_i \log_2(p_i) \qquad (2.1)$$

Where c is called caste and it defines number of robot types in a team and p_i is the decimal percent of robots that belongs to caste c_i [3]. We should note that in a homogeneous team, the value of social entropy *Het (R)* is 0 as $\log_2(1)$ equals to 0.

As we can conclude from thermodynamics, the diversity of robots increases the social entropy, that means, the instability of the multi-robot system. It is obvious that two different robots require two different control mechanisms, i.e. hardware and software issues; which increases our system's complexity and difficulty to control.

2.8.8. Collaboration in Multi-agent Robotics

"Collaboration" of robots for a given task may be in numerous forms. But in multi-agent collective robotics, accomplishing the given task is tried to be achieved by using the simplest and smallest agents where capabilities are in minimum, but always sufficient enough to complete the task. For example, if the task is carrying some objects, the agents must be capable of carrying these objects not alone but at least by carrying it by two or more agents by collaborating [30].

Here, the objective is not to limit or to reduce the capabilities of such groups but to economize in computational and physical requirements and solving problems by minimal use of robotics resources [24].

3. COMPARISON OF AGV SYSTEMS TO AMRs

In this section, we will examine AGV systems as a special example of industrial application that can make use of capabilities of fully autonomous robots.

3.1. An Overview of AGV Systems

"AGV"s (Automated Guided Vehicles) are battery powered, driverless, mobile vehicles currently deployed in flexible manufacturing environments, primarily used for material transportation of goods in warehouses and between production lines or for cleaning tasks [33]. Although AGVs are programmable for path selection and positioning issues, they are automated vehicles but not autonomous. The majority of AGV systems are wire-guided vehicles. A typical AGV follows an embedded, signal carrying magnetic wire on the floor or tracks a magnetic line [46]. In Figure 3.1, we can see an AGV; note the white lines in the floor for guidance.

AGVs are drone like vehicles, controlled by a control center station [47]. A problem in the center may cause a breakdown for the whole system, where all the vehicles would stop or be paralyzed.

Figure 3.1. An AGV executing a transport task (photo by FMC Technologies).

AGVs achieve their activity over a defined flow path layout. Besides the vehicles and the flow path layout there must be a control mechanism as the third component to manage and monitor operations of AGV fleet during material transportation. This control mechanism is called “Vehicle Control” in Bulak [43].

AGV systems can be programmed, generally from the central station, for path selection and positioning over the lines. But this path selection is always over a second or more defined path like changing tracks of a train in the railroad. A more flexible path choice can be highly profitable for the overall system [46]. The system would not have to calculate the path priorities over a fixed, one-way allowed lines but it could concentrate on the priorities of goods transportation. Today, intelligent vehicles, capable of free ranging, decision taking, interfacing with other displacing entities and avoiding unpredictable situations could solve these bottlenecks of AGV systems [33].

For the operation of an AGV fleet, some environmental modifications in the industrial facilities are required. Localization aids are the most cause of these modifications. Signal emitting active beacons, bar-code tags, magnetic or visible inductive lines and loops in the floor, rail tracks or even GPS - Global Positioning System; a system which measures the distance and positioning of the objects via triangulation of different satellite signals that can give the absolute location of the robot over the Globe with a very little error measured in meters or in millimeters under ideal conditions - are such modifications. Of course, in actual situation GPS utilization is unavailable for closed indoor environments such as factories. Design of robots that can use natural cues instead of artificial devices for navigation is an actual challenge in today’s robotics. As the GPS utilization is unavailable in closed environments such as factories, offices or hospitals, this challenge becomes particularly important. Instead GPS-like systems must be implemented for these areas.

3.2. Benefits of Using Autonomous Systems

AGV systems gained a considerable success mainly because of their ability to achieve end-point accuracy. However, this end-point accuracy is only assured by pre-determined paths by fixed magnetic lines, high-cost devices and expensively controlled

vehicles. As these AGVs have limited sensing capabilities, in presence of obstacles, their typical reaction is to stop and wait until the obstacle is removed [48]. If not controlled properly, deadlock of vehicles can occur [49]. Also, the installation of wire or magnetic guides and devices could be costly and disruptive in the working environment. Furthermore, any modification to the installed pathways is even expensive.

Another bottleneck for AGV systems is centralized control: In case of a breakdown in the central station, the whole system could stop. The extreme centralized control makes the system vulnerable.

With these features, the AGV system is similar to a railroad transportation system. For example, a faster train could pass another train only in station equipped with extra lines and rail scissors. Such a system is centrally controlled like AGVs [47]. They allow fast and accurate travels between two points, but reachable points are in a bottleneck.

As the AGV systems needs a costly computer care and high-cost setup process and they have little autonomy in problematic, an upgrade of these systems to AMR solutions would be a great progress toward a full automatic workforce instead of humans. This upgrade seems possible with introducing a few novelties for AGV systems. Modification for hardware, software and control policies to solve problems when using natural cues for functioning of these systems will be discussed in this section.

Below, we can state some forecasts for using autonomous mobile robotics inside manufacturing plants for future:

Table 3.1. AGV vs. AMR comparison.

Feature	**Automated Guided Vehicles**	**Autonomous Mobile Robots**
End Accuracy	High	Noisy
Path Planning	Pre-planned and fixed	Planned in real time
Outdoor Working Capability	N/A	Available
Sensing Capabilities	Low	High
Robustness	Low	High

As seen in the Table 3.1, autonomous robots provide a free paradigm in path planning and routing, in contrast to AGVs where path are pre-defined and their routing are always wire-guided. Autonomy carries big routing and sequencing problems to solve but once solved, it provides also the possibility of more flexible solutions [48].

In the following sections, we will try to propose solutions to solve some AGV problems with AMRs in order to obtain feasible solutions for automation of transportation of goods in factories.

Figure 3.2. AGV systems' fixed pathways.

As seen in the Figure 3.2, AGV systems operate over fixed pathways, which need high-cost setups, and they have deadlock problems over these fixed pathways.

An upgrade of these predecessor systems to successor AMR solutions would be a great progress toward a full automatic workforce capable to response to the needs.

4. IMPLEMENTATION OF AUTONOMOUS MOBILE ROBOTS IN INDUSTRY

As described in previous sections, our objective is to observe the possible techniques of using autonomous agents in industry, and especially in manufacturing workshops like factories. Already introduced by robotic manipulators, robotics has a great area of application in industry. Maybe the next step would be the use of intelligent mobile robots with (or instead) of fixed, tele-operated robots.

Mobility and autonomy of robots working in industry could come with some productive advantages: Let's think about a robotic manipulator used in welding. Instead of a fixed welding machine, suppose that the robotic welding arm is carried by a mobile platform. Displacement of the platform may allow many other positions for the robotic arm to reach the pieces. On the other hand, autonomy of that robotic arm or autonomy of the platform itself could provide choosing the positioning against pieces or working in different places, or even deciding for the welding machine where to work. Then, with mobility and autonomy, such a robot will work like a human worker.

There, the greatest dream for industry begins: Replacing humans by robots. At the first sight, it seems difficult for today. Robotic soccer games have already its place in tournaments with the aim of being capable of beating the human players in a soccer game within the next half of the century (2050). So, for economic and security reasons, this could be a similar situation for industrial places. Having different objectives than the robotic soccer, competition between humans and robots in industry would follow different criteria. Not having the obligation of being human-like, industrial robots would be different in features and in appearance than humans, in order to make processes more precise, more rapid or more economic [10].

So, working with autonomous mobile robots in industry would have two strongly correlated main research domains. These are:

- Physical appearance and design of the robot: Where robot's physical features, body, actuators, sensors and manipulators must be in accordance with the environment and the task to be executed. This part of the challenge is an interest topic for mechatronics.

- Programming and control of the robot: Where robot's control paradigm, strategy, a user-end programming would be in accordance with the physical capabilities of the robot. This part of the challenge is an interest topic for artificial intelligence.

These two domains have its influence over the other. So the question has two sides: Controlling the robots according to their mechatronic capacity or designing the robots due to their control strategy? It is not a single solution problem. Here, we will pursue the different possibilities and compare them on a basis where it is possible.

Flexibility in organization and easy moving of parts and tasks are important benefits of multi-agent robot usage in production areas. Here, the state-of-the-art still has various competing approaches. The problem to be solved with multi-agents may vary and there will be many different models that could be applied for solution. We can propose a distributed decision mechanism or a hierarchical one. Trying to model different problems and offering different possibilities of multi-agent solutions for these problems could be the profit of this research.

An important point to consider is that industry does not consist of only structured environments of manufacturing plants. Industries such as shipyards or construction sites are still industrial places but outdoor environments instead of indoor ones. So, proposed multi-agent systems should also applicable in these areas, too.

4.1. Requirements for the Team Member Robots: Properties to be Considered

For industry, robots that can work safely in industrial environments are essential. On the other hand, robots need also a safe environment for working but generally it is robots' business to care of other human beings or objects in the environments [15]. On the other hand, real-time implementation of path planning and environment learning are equally

important for a mobile robot to do something useful. Through studies carried out in experimental environments for design and implementation, autonomous robots, have acquired capabilities to cope with fast changing dynamic environments. Here, the word “to cope with” is used in the sense for the ability to suitably (without damaging environment, hindering other work or suffering damage itself) avoid unexpected obstructions by utilizing sensor-based control and displaying real time reactions.

On the other hand, subjects of real time path planning implementation and environmental learning are important for autonomous robots to achieve useful jobs. These subjects facilitate in theory missions such as lifting of load palettes, material handling, and cleaning or making rounds. However, most real time path planning and environmental learning algorithms are either insufficient or unsuitable for today’s real time applications. For this reason most autonomous mobile robot made until now are far or lacking of the full autonomous capability that is required by practical and reliable industrial applications. Production of such practical and reliable robots with full autonomous capability requires solution to difficulties mentioned below:

- Robots are complex devices with multi degree of freedom, and they can move in a rapid manner. So, the environment they are in exhibit changes as fast as the speed robot moves. This is true for robot arms as well as the autonomous robots themselves.

- For a robot to decide what move to make depends on sensory capabilities, which must be high. The difference of robots from other AGV systems is their ability to work in unpredictable and changing environments by producing required reactions. Martinoli [7] suggests that data from many sensors and not one would be crucial and it would be possible to refine data by combining data that come from many sensors containing noise, hence featuring uncertainty. However too much sensory data may reduce stability for the amount of data to be processed and increased likelihood of noise inclusion.

- Autonomous robots need to produce fast responses to be able to work in real time environments. On the other hand path planning and task allocation operations have

to be performed in parallel. These requirements call for a multi level architecture where both spontaneous reactions and planning procedures are catered for. In such situations, since optimum paths need to be modified in the process and optimum solution shall not be the theoretical closest distance calculated. Instead, a planner that re-arranges path-planning as an unexpected obstacle is encountered and provides fast commands for the low-level controllers, is desired [28], [38].

- The system to be built should be robust with respect to defects, and constitute a balanced structure that shall stop all system going out of action and should continue to work when a high or a low level component fail.

When considered from all these aspects, designing of a robot system to function in real situations is a highly complicated engineering process. Robot components such as sensors, actuators, manipulators mentioned above shall physically define the capacity of the robot. Since robot control needs to be designed in line with these physical capabilities, it is apparent that the selection of control strategy according to physical capacity or selection of physical capacity according to control strategy is directly correlated with one another. Many alternatives may arise, as there are numerous control systems and factors present. Only the designs with systems that can achieve the balance between physical components and robot control strategy can be successful.

On the other hand, robot design must be targeted for different environments and purposes as well. The system constructed must be able to cater for different purposes through modifications.

4.2. Making of a Robot Worker Team for Industrial Places

As precious individuals in factories, robots are becoming more populous than ever. To replace humans in factories, obviously, robots must work in cooperation as teams. These teams may vary in type and number of robots. To decide on these issues, one has to consider first the environment [31].

4.2.1. Industrial Workshops

Industrial production areas may vary according to their domains in macro and to their activities in micro scale. But commonly, production areas have some common physical appearances. These are:

- Indoor areas: Where workers execute their jobs closed and limited places.

- Planar surfaces: Almost all organized production area's floors are smooth planar surfaces in order to make to rolling and moving of parts, chariots or machines in an easy manner. While working on the same floor, big slopes are avoided.

- Up to down lighting: Most of the industrial places are illuminated with spotlights, which are fixed on the indoor ceiling. Some places have little windows to bring daylight inside. Lighting is important for robots' perception of the environment due to their influence over their sensors.

- Crowded places: Industrial places are commonly shared by many entities. This could be other workers, transport machines, AGVs, etc.

- Noise and strange materials: In production areas, many sounds, both in high and low frequencies are produced. Also, debris from scuttled pieces or welding fires is produced. Some vibrations from functioning machines may also be perceived by robots, which may affect their sensors.

As this list is not exhausted, a physical design of an industrial purpose autonomous mobile robot must consider these environmental constraints. Each individual of the robot team should be able to execute its duty under these circumstances [31]. Also, team control must be robust to reach its goals in order to respect the planned actions.

4.2.2. Robot Design for Manufacturing Plants

An autonomous mobile robot must be designed and controlled to work in conformity with the above-described conditions. In the first phase, we will describe possible physical appearances and required components of such a robot.

4.2.2.1. Planner. A mobile robot must plan its path within the environment to reach a destination point. For this, an offline path planner or a supervisor generates a route to follow for the robot from the start point to the destination point, containing no collisions with obstacles. In the real world, this path is followed as series of collision-free maneuvers, which are provided to the vehicle by the path planner. These maneuvers consist of line and arc segments. Then, these lines and arc segments are sent to a low-level trajectory generator and closed loop motion control to give the necessary control commands to robot's actuators. The controller assumes that it has accurate information about robot's position and gives commands to the low-level via this information. Here, robot's position estimation system must provide that information. The estimation system consists of a priori map of the environment and it has a matching algorithm, which compares the robot's estimated position with data coming from the odometry sensors to check the information's correctness and provide a better estimation of position. With these capabilities, an autonomous robot has no need for active or passive beacons or guidelines to navigate.

4.2.2.2. Sensors. Most common possible sensors would be infrared (IR) sensors for range finding and odometry. Also, ultrasound sensors can be used. A communication device must be used for sharing information with other team member robots and/or with supervisor (if exist). Also, cameras could be used for image perception. Odor sensors seem unnecessary for industrial purposes in this level.

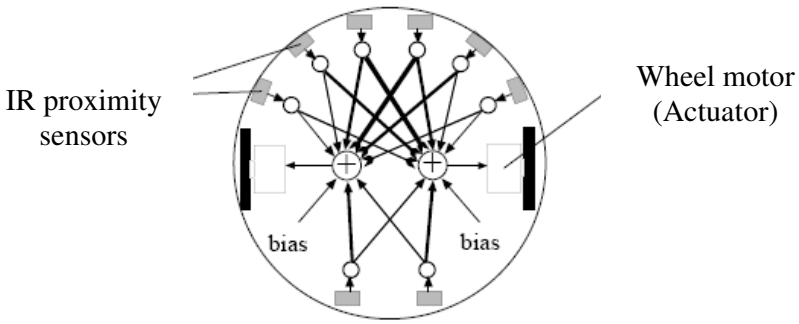

Figure 4.1. Khepera robot's sensors.

In Figure 4.1 a series of sensors covers a Khepera type robot's most vulnerable faces. Ultra-sound or IR sensor are most commonly used sensors, especially for range finding.

4.2.2.3. <u>Actuators</u>. By 4.1.1, we can predict that wheeled or tracked actuators would suit our robots. As the surface is smooth, tracks or differential wheels would provide quick displacement ability and static equilibrium [20].

Legged actuators are still costly to implement and static equilibrium difficulties make them harder to design, so for this purpose, we will not consider legged models.

Non-holonomic drivers, like automobiles can be more convenient due to their wide use. 4-wheeled robots have a good static equilibrium. But, in this study, we will concentrate on differential wheel models due to their easiness for use and program. Below, we will state some properties of differential wheel drive [20].

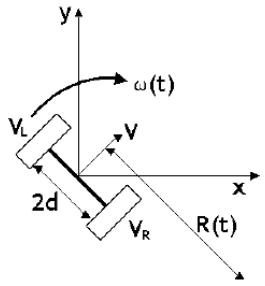

Figure 4.2. Differential wheel drive

The kinematics equations (Equations 4.1 and Equations 4.2) of differential wheels are given below. They can be used to calculate the position of the robot according to its initial position for given values of right and left wheel speed respectively. V_L represents left wheel speed; respectively V_R represents the right wheel's speed. $2d$ is the distance between two wheels. Finally, R represents the turning radius of the differential drive robot.

$$\left.\begin{aligned} \omega &= \frac{(V_L - V_R)}{2d} \\ R &= \frac{2d(V_L + V_R)}{(V_R - V_L)} \\ V &= \omega R = \frac{(V_R + V_L)}{2} \end{aligned}\right\} \quad (4.1)$$

$$\left.\begin{aligned} \theta(t) &= \int \omega(t)dt \\ x(t) &= \int V_x(t)dt = \int V(t)\cos(\theta(t))dt \\ y(t) &= \int V_y(t)dt = \int V(t)\sin(\theta(t))dt \end{aligned}\right\} \quad (4.2)$$

4.2.2.4. <u>Manipulators</u>. As obvious, an ultimate improvement for robots in manufacturing plants will be replacing human workers. As human workers use their arms and hands to use various tools, autonomous mobile robots in manufacturing plants must carry their manipulators to use, carry, pick, pull and work pieces to be processed. These manipulators must be convenient for multitasking, i.e. copy human arm and hand, with higher degrees of freedom would perfectly fit for most occasions, as the environment and pieces are commonly designed for human usage.

4.2.2.5. <u>Dimensions of the Robots</u>. Dimensions of robots depend on related products to be produced or services to be executed by autonomous robots. As the robots in the manufacturing plants would be in interaction with the production environment, they must be also in accordance to the other objects dimensions in the environment, i.e. human workers, corridors, other machines, etc. [21].

It is obvious that, if an autonomous mobile robot would work in a car factory, its dimensions would be similar to a human worker because humans for avoiding robots could ignore smaller dimensions and bigger dimensions also would cause problems for robots for avoiding humans.

4.3. Solutions for Localization Problem

As no more magnetic line or wire guidance will be used, autonomous machines have to determine their positions inside the working environment, which is an essential point.

"Localization" can be stated as the problem of determining of positions in the present environment. Positions to be determined could be objects, obstacles or robot itself. A manipulator robot has to know the location of objects to manipulate or a navigating robot has to know where it is and where the goal location is to find out its way toward a destination from a departure.

Obviously, for realizing localization robot perception is a must. For this purpose, incoming data from sensors has to be processed to resume knowledge about where things are and where the robot is. Then the perception of the environment becomes meaningful.

Localization could be done by the robot itself, or by a supervisory system. If the environment is not already known or to be explored, then a dynamical localization would be done: By sensor data, the robot percepts its environment and constructs a map. For constructing a map, Köse and Akın [23] propose the Generalized Voronoi Graph method. By known objects taken as location points, the robot generates a map of its environment. This map is re-generated or revised over time by navigating numerous times in the environment. IR distance sensors, ultrasounds, light sensors could help to perceive these objects. Also, as a verification process, wheel or track encoders could be used to verify the position knowledge from a departure point. Localization problem could be solved also, by local perception by the knowledge acquired by imaging sensors that means the cameras, which can provide features on the environment with images of the environment and models of the environment too by using the computer vision techniques. For single robots, image processing is a challenging research area.

Also, help from outside could answer the question of “Where am I?” of our robots. For highly organized indoor areas or open environments, another alternative for localization knowledge is to getting help from a device. As the robots are well acting autonomously, they can get their position via an outside system. An overlooking surveillance system could solve easily this problem, so there would be no more need for explorations. For such a system, we can give the example of GPS which, as described above, measures the distance and positioning of the objects via triangulation of different satellite signals that can give the absolute location of the robot over the Globe with a very little error measured in meters or in millimeters under ideal conditions.

The problem of GPS usage is, that the GPS cannot work indoor areas or underwater. But still working with an overlooking system is possible. The preference of working with an overlooked system comes from the end accuracy of the positioning knowledge. As the majority of the industrial places are covered, GPS cannot be used but in order to reduce computing problems, service robots working indoor areas could receive their position via an overlooking information system for positioning. Then, considering advantages of GPS, we will propose to introduce a GPS-like supervisory localization solution for indoor areas. As the robots are informed about their localization via an outer source, they will still act autonomously in their acts, like determining their navigation paths, etc…Vision perception via cameras and wheel encoders could be still used for data verification or in case of extreme conditions like communication failures or sudden environmental changes.

4.3.1. Supervisory Camera System

As GPS does not work in closed areas, a system, which can survey the covered area, must be introduced. For that, a camera system which is fixed on the ceiling of the production plant will watch the job shop from bird’s eye view. This solution provides an easy option for absolute positioning of all the robots and would provide a better coordination, because all of the positioning information could be acquired via this supervisor system (Figure 4.3). A similar approach is used by Karagöz, Bozma and Koditschek ([36], [37]) for their EDAR robot.

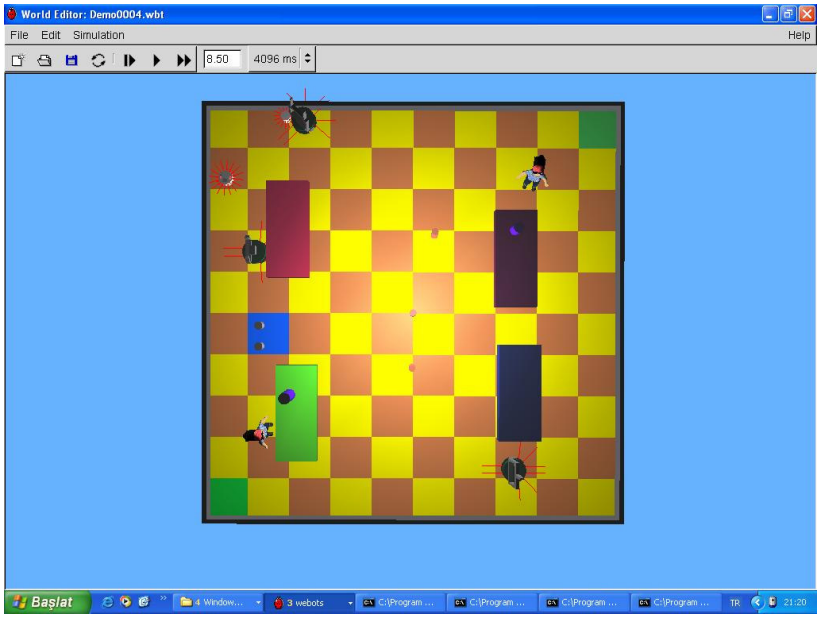

Figure 4.3. Localization via supervisor camera system.

As seen in Figure 4.3, the site's view divided in rooms first, and then divided by gridlines could be used to retrieve localization information for robots. Images taken from the above camera show the positions of the robots and other objects inside the job shop. There are two important information which could be obtained from the data provided via camera:

- Absolute positioning data: If the camera could watch all the covering area, in the image, all the robots could be watched. As the camera is fixed, if the supervisor can detect a robot in the picture, it can define its position by pixel analyzing. So, in every taken image, the positioning data is defined for robots from the supervisor.

- Odometry data: From two consecutive images, a robot's speed and direction could be retrieved. This data could be processed from the robot's own data from its odometer sensors for comparing. By that way, a supervisor could know about all the members' speed and direction in a given moment to compute their vector for path planning.

As an alternative system, every autonomous robot could have its own camera to perceive the environment and localize itself and detect other robots and objects by direct

computer vision. This system would be a robust but more costly solution for computing. This approach is better suited for a decentralized control and distributed intelligence. The previous fixed supervisory camera solution is better suited for centralized control and supervisor-supported intelligence.

4.4. Conclusions of Multi-agent Team Implementation for Industrial Facilities

Autonomous mobile robot systems find place in today's mass production industries in increasing numbers. Especially present mobile systems utilized in parts and load crates or pallet loading handling and placement may be upgraded and transformed to autonomous systems. Currently such systems consist of systems capable of following a rail or strip [40], and their perceptions are limited to stopping at an obstruction encountered on their predefined path and when it is gone to start moving again. Another frequently employed solution is systems featuring beacons. These are considered as free ranging systems obtaining their positions by retro-reflection from beacons [7]. Although these systems do not require compulsory physical guides such as magnetic strip or rail to be installed, their operating principles are the same, because the working area is required to be marked. The advantages of autonomous robots over such systems are their ability to find their way independently without required markings, plan to achieve their goals and revise these plans when unexpected situations are encountered.

An important issue here is that level of robot system autonomy should rise with falling standards of infrastructure and level of production. Robot must feature obstacle avoidance feature to go round obstacles and robust path planning, navigation and environmental learning as well as behavior adaptation against changing situations. Behavior adaptation can be implemented by increased decentralized control. Thus, without instruction from an external source robot continue functioning in line with its mission and gain "immunity" against the unexpected. On the other hand if environment is highly regulated and work area is limited then centralized control may offer advantages. For example determining task allocation from one centre, assignment of each robot with a task may provide better output for the preservation of orderliness in the factory. Besides it shall facilitate system controllability. This one presents a solution closer to the present AGV systems and its implementation can be easier in comparison. The balance between

centralized & decentralized controls for multi-agents should be adjusted according to environment and the nature of the mission.

Major application opportunities are currently opening up in outdoor multi-agent mobile robotics; such as stockyard parts acquisition and delivery, sub-sea, dockside automation, inshore navigation, civil construction and parts delivery inside nuclear power plants, or even space exploration missions. Applications that are often talked about but which seem to have very limited likelihood of being fielded over the next decade include object delivery in hospitals and offices; robot guide dogs, and sentries, e.g. in prisons [27]. Then, the same opportunity exists for industrial facilities too.

An important basic consideration is the degree of information available to the robots. For our work, robots will be informed about the arrival and the presence of the component parts to be carried; so they will make an informed search and they can concentrate on cooperation and task allocation between them. As the factories are highly ordered areas, a multi-agent robotic system designed for working on these areas would know where situated the delivery nodes or arrival nodes are. As the degree of information to be provided to robots increases, the system would focus on end point accuracies.

in the uninformed situations, not only the heuristics and the matching algorithms would change, but also the robot architecture has to be changed: The robot architecture would have to focus on the sensing of the environments and on the parts before it concentrates on task allocation or matching algorithms, because the problem of localization and object recognition is a priority for robots for making useful actions. Once these problems solved, they can continue to make their best effort to help to optimize production. We can state that the decision of implementation of centralized or decentralized control of multi-agents has a correlation with the degree of information to be provided to robots.

The ultimate autonomous robot would work on totally unknown areas, in uninformed situations, with noisy data. A robotic system's approach is filtering this noisy data to obtain the true information, to make a map of the environment, reducing the unknown factors. As an industrial place has an order of its own, like ranging of elements or materials, we can use this ordered structure and the information, to simplify our system's problems.

5. SIMULATION BASIS FOR SAMPLE PLANT MODEL AND MULTI-AGENT AUTONOMOUS ROBOT TEAM

In this phase, the principles for constructing a sample model workshop and a model of an agent and a multi-agent robot team will be determined.

Simulation is a good basis for observing models of systems. Model validation could be done via simulation and also simulation helps to carry out longer tests, which can be too costly in the real world. Also, simulation will save valuable time by fastening the simulation steps. Of course, a good robot simulator must contain many features in order to represent the robots' complexity like sensors, actuators etc. on one hand, and the environment's complexity like added noise to sensors, lightening values, adjustable physics, etc. on the other hand. This will provide a similarity between real robots and the real world and beyond this purpose, adjustable features could help to test the robots and the model performance in extreme environmental conditions. Moreover, in case of working with multiple robots, tests in real life would be too complex and costly, and then a good simulator's help becomes a must.

5.1. Software Tools

In order to simulate the explained relationships between multi-agent robot team and the environment, and then to observe the different alternatives of multi-agent usage in industry, WEBOTS simulator, version 4.0.27 of Cyberbotics Co. is chosen [6].

The files and instructions necessary to run the simulation program are available in the appendix CD which is attached inside the back cover of the thesis. Information related to the content and usage of the appendix CD is in the Read.Me text file.

In the first step of simulation, the working environment is modeled. Secondly, the creation of the workshop will be followed by design and creation of the agents: The robot features and a robot team are constructed. In the following step, control system and

behavior features of the multi-agent robots team is discussed, developed and implemented in order to answer the needs of the given goal tasks.

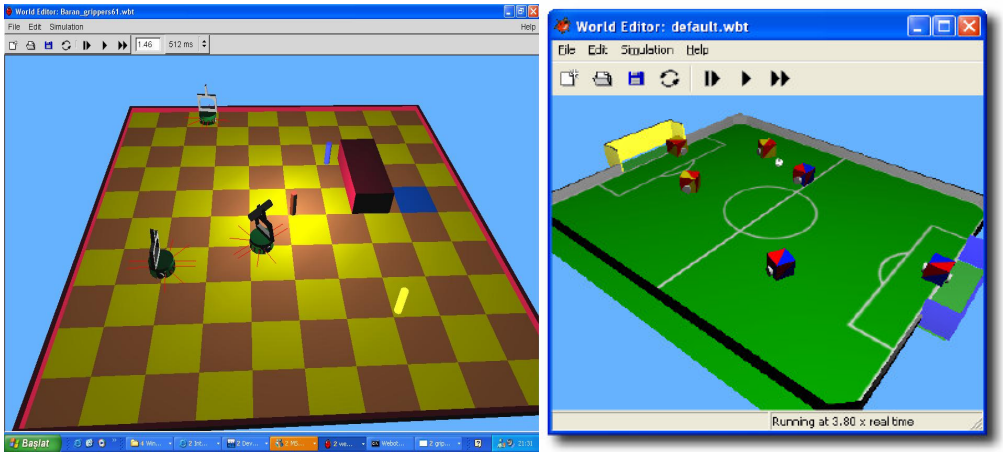

Figure 5.1. The Webots simulation platform.

The Figure 5.1 shows two scenes from Webots simulation platform. This platform is chosen to test workshop and robot controls as it provides a good basis for multi-agent simulations. As a robot simulation program, Webots has a physics library and a VRML capacity in and it can successfully provide realistic reproductions of sensor and actuator capabilities [6]. Creating your own robot; legged, wheeled or flying; and your own world is possible due to 3-D VRML capacity, physics library and well-defined sensor and actuator capabilities. With these capacities, robot-robot and robot-environment or even outer objects-environment interactions can be animated in accordance to their adjustable kinematics. All the sensors, actuators and other devices' capabilities are adjustable and their data are submitted into an adjustable noise, if wanted. The program gives also the opportunity to transfer the implemented algorithms and controllers to some real robots. Being a 3D robot simulator gives the program the authenticity of facing real atmosphere more than many 2D simulators [19]. Besides being a 3D dimension simulator, Webots has three important advantages, whose we can profit:

- A supervisor module that allows the user to monitor all internal and external variables of the robots (position, sensor and actuator data, and so on) and to interactively change them. This could be helpful to implement a centralized control.

- Extension turrets may be added to the basic module, so robot design for different purposes becomes easy [6].
- A simplified, faster sampling procedure is used for the proximity sensors. This feature allows us to simulate real-like sensor capabilities [25].

5.2. Simulated Hardware Tools

For obtaining a visible and easy to use simulation, we have chosen the Khepera robot, which is developed by Cyberbotics. This mini mobile robot's duty is to perform desktop experiments. Some features of the robots are given below:

- Diameter: 5,5 cm.
- Processor: Motorola® 68331, 32 bit processor at 16 MHz.
- Actuators: two DC motors, commanding right and left wheels respectively, and having incremental encoders to remember the turn rate.
- Sensors: 8 IR proximity and light sensors, 6 distributed in front face, 2 in backwards.

The basic Khepera robot can be extended with supplementary modules. Communications, camera and gripper modules are most common extensions of Khepera. The gripper's arm can lift up or down, and the gripper can open or close to carry an object up to 5 cm across and 20 g. of weight in maximum capacity.

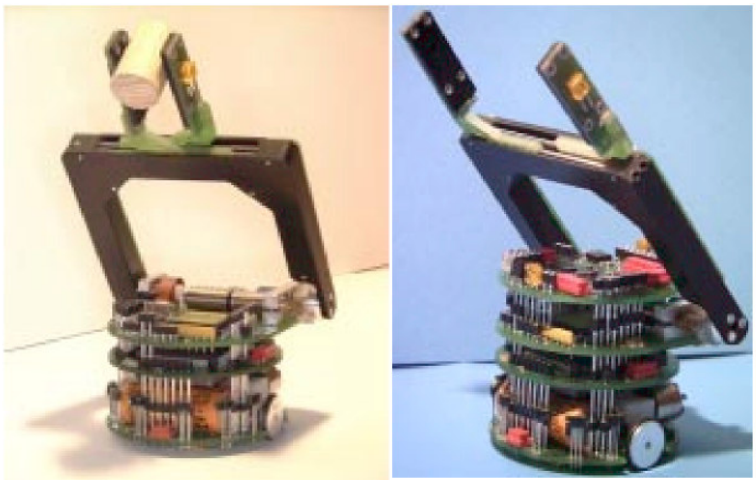

Figure 5.2. A Khepera robot equipped with a gripper module.

5.2.1. Multi-agent Modeling in Simulation

As we can create our customized robotic agent in Webots, we can also use an existing robot's model. The design process of the agent is somewhat complex but thanks to the simulation environment we can create our own prototype, equip it with sensors or actuators in needed number for free of charge. But, we can also implement our control code to a real robot by cross compilation, and then see the simulation in the real life [18]. Then, a good way to conceptualize an agent is to model an existing and already working robot. Due to this reason, we are going to choose the Khepera robot for convenience [14]. The Khepera robot is developed by Swiss Polytechnic Institute of Lausanne, where Webots software is also developed. The robot would be equipped with different extension kits such as emitters, receivers, cameras and grippers. The gripper is an important factor for our work, where in the plants, the one of the primary missions of the multi-agents could be transportation of the parts. The complete robot looks like in the Figure 5.3.

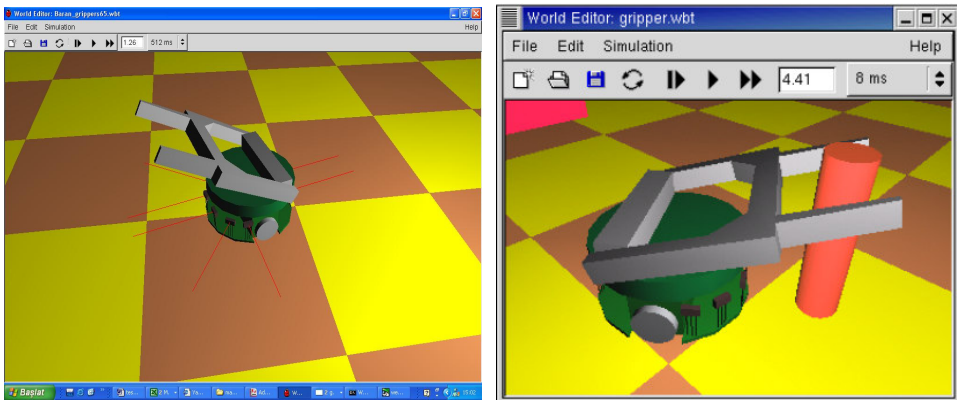

Figure 5.3. A sample agent model based on Khepera robots.

In the Figure 5.3, we can a sample agent model based on Khepera robots, equipped with sensors and a gripper. Simulated robots are representing all the features of the real robots in order to see what happens in a closest likelihood to the real world [25].

The experiments presented in the following sections were carried with a Webots simulator of Cyberbotics Ltd., provided by Boğaziçi University AI Lab.

6. A SAMPLE MODEL FOR PRODUCTION PLANT

To measure the performance of autonomous agents in indoor production areas of manufacturing plants, modeling a manufacturing plant is needed for simulation. The model must be realistic and suitable for the use of robots. For this purpose, we will look at workshop organizations in industrial places, and form a model in our simulation program. The model must have not only production notion entities, but also real life features like light arrangements, obstacles, stations and humans. Robots are working in factories for years, but here, our aim is to model a production area suitable for intelligent robot use.

As end point actions of robots, like welding, loading, picking, lifting, etc. is already simulated in many simulations for robotic manipulators, our plant model is concentrated in spatial distribution of production elements and the most important, it is concentrated to make a suitable and challenging area for multi-agent navigation.

6.1. The Workshop Subsystem: A Structured Environment

The working environment of an autonomous robot is set as a group of workcenters that performs different machining functions and processes different part types with a certain demand volume [33]. Jobs arrive at the job shop in a random manner through a receiving center and leave the system through a shipping center. Each workcenter has a pickup station and a delivery station or one dual-function transfer station. Loads are transported among the transfer stations of the work centers via robots. Robots will define a path between work centers.

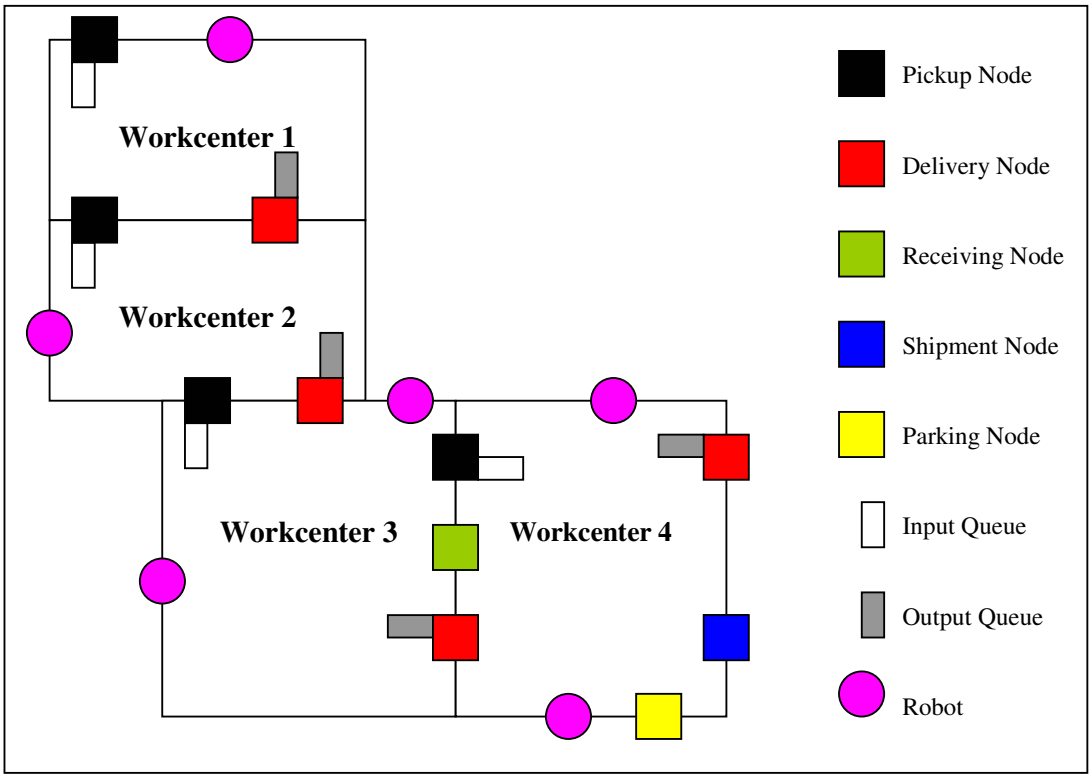

Figure 6.1. A job shop environment reference model (by Doğan [33]).

The above shown Figure 6.1 indicates a typical job shop environment where autonomous robot teams will operate. The workshop model where the robots will function could be similar or resembling to this reference layout model. The workshop model could be divided into multiple work centers; and in every work center there are multiple nodes of different purposes. In the following sections, we will try to consider the working of multi-agent robot teams inside the job shop.

A “Task” or a “Job” is an order for one unit of a given part type. There is a finite set of job types with a given job mix. Jobs arrive in the workshop through an arrival centers according to a specified inter-arrival distribution. The arrival probability of a task is determined by the defined distribution of the workshop environment. Each task can be executed with respect to a known processing rule [33].

Here, the tasks are constituted of “Unit Loads”, which will be named as “Parts” hereafter. A unit load can be defined as a unit of mass that is moved singly. In other words,

one part will represent a unit load of a warehouse, which can be transported and processed as a single entity. These parts will be transported via robots among workstations, which will be called “Nodes” hereafter. Based on the above definition, one can consider a part is a unit load, which can be considered as a single component on a conveyor, many components in a container, or several component-filled containers on a pallet [33]. Parts will be processed in workstations, which are called nodes.

The nodes or the workstations will have an entrance. Each workcenter has independently maintained input and output “Queues”. They can be also called as “Buffers”. Each buffer has specified capacities. Here, we will limit our buffer capacities with one part and one robot at each process due to concentrate more on our robot design but with the designed simulation, it will be possible to do a research with different queue capacities. On workcenters, processed parts will be loaded in robots. If there is more than one robot waiting to reach a node, the priority of the agent will be considered.

6.2. Alternative Layout Possibilities: Implementation in Webots

To watch the performance of the robots control paradigm, one can use different layout schemes to obtain different results to compare them. The layout design is very important for the performance of the multi-agent robot team. Robots’ control paradigm performance could be measured with simulating different possibilities layout possibilities.

Also, another possibility is to define cases for each layout. With different number of robots or different density of jobs, each layout design scenario could be richer to observe the results.

Below, job shop environment is simulated in Webots program. We can see the different nodes as described above, and separation walls between areas, in order to make a distinction of arrival (pickup) and delivery facilities, and moreover, to create local minima areas which makes navigating difficult for simple robots. As said before, the aim of the model is not only to see the workshop organization, but also to test multi-agent robot team performance under heavy navigating conditions. Looking-like a labyrinth, “L” or “U” shaped walls contains hard localization problems for robots. Simple behavior robots could

not navigate to their goals, because of the counter diction of sensor data in these areas. Only intelligent agents supported by successful path planners and equipped with compatible sensors could navigate in such areas, which are simulating possible real life navigation problems of robots in job shops.

6.3. Modeled Workshop in Webots Scene

The modeled workshop is presented in Figure 6.3. We will call the workshop modeled in WEBOTS as "World" as used in familiar term in WEBOTS program. The components of the "Webots World Workshop" are stated below.

- Static Obstacles: Walls, delivery and arrival (pickup) nodes.
- Dynamic Obstacles: Free moving, non-controlled obstacles (as human workers).
- Robots: Khepera robots with grippers.
- Pick-Up Nodes: Four squares near each corner. Each of them has a separate room.
- Delivery Nodes: Two squares in the inner chamber, which make them hard to access.
- Loads to be carried: Red, blue or green sticks. Their color distinct their type and priority.

In Webots, the workspace of the manufacturing plant is modeled as follows:

- As it is not possible to define all the areas in the plant, a microscoping model of a simple job-shop with workstations will be modeled.
- In the job-shop, parts to be assemble are arriving from outside to workstations.
- Incoming parts must be delivered to the assembling stations where they are delivered as finished goods. For this task, simulated parts will be transferred by autonomous robots from arrival (pickup) nodes to the delivery nodes.
  (Figures 6.2 & 6.3)
- There will be a scoring in the simulation. Number of assembled parts or number of transported parts statistics and the timings coming from the log files would be a

data for comparing the different solutions of implementation of multi-agent systems.

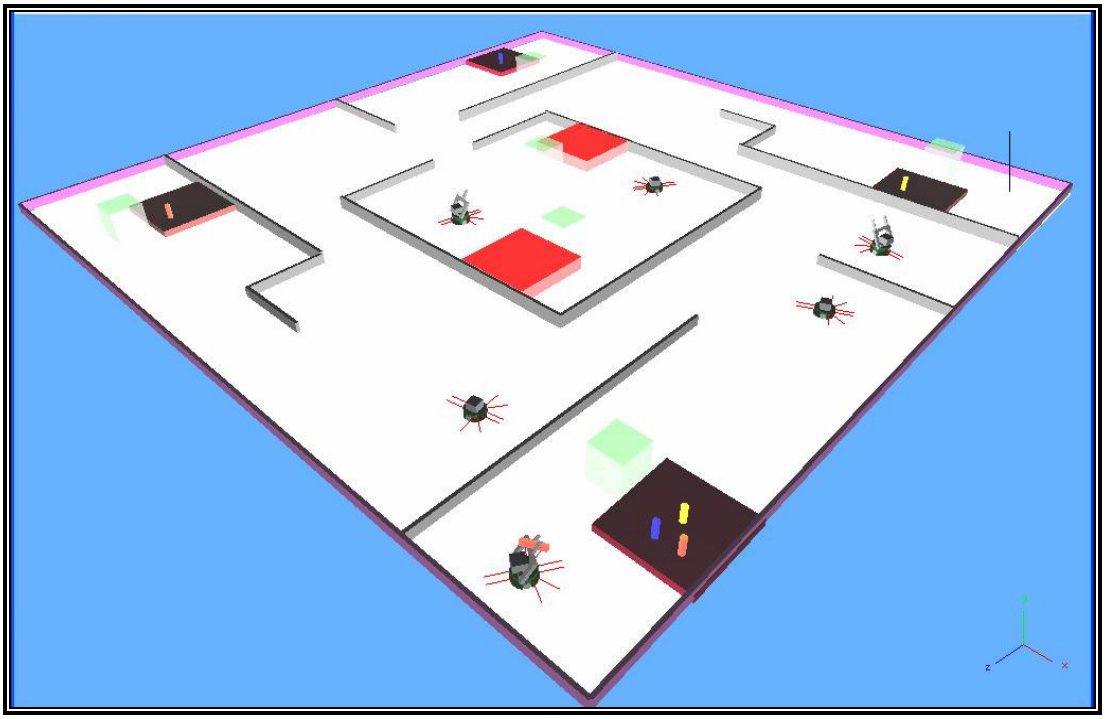

Figure 6.2. Job-shop environment in Webots simulation scene.

In the Figure 6.2, we can see four darker nodes for arrival and two red nodes for delivery. Parts to be carried are colored to show their priority. Walls around workspace chambers create local minima for robot navigation, which could be solved only by intelligent path planning techniques.

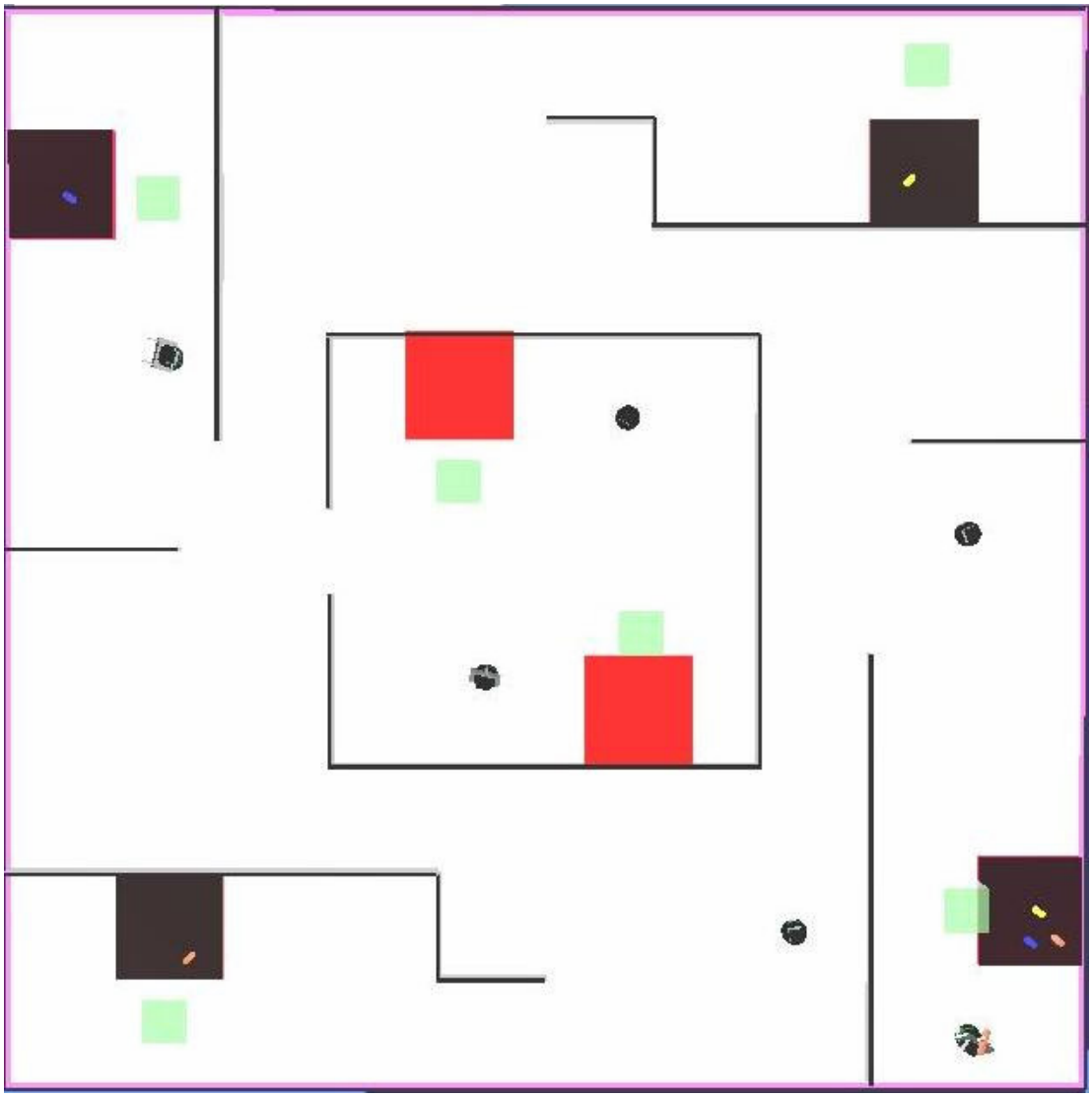

Figure 6.3. A bird's eye view of the workshop model.

In the Figure 6.3, the four darker nodes at the corners are arrival nodes; two red inside the chamber nodes are delivery nodes. Colored sticks over the node platforms are parts to be carried. Robots with grippers are our robot team members. Other moving robots are non-controlled random moving robots, introduced for creating a noisy and jammed traffic.

7. A PROPOSED SYSTEM AND ITS IMPLEMENTATION

7.1. Introduction

In this section, we will explain the features of our proposed system for a multi-agent robot team, which will execute transport and delivery duties in an industrial space. In the previous sections, we have defined a workshop and modeled it in a simulation environment, and then, we have chosen and modeled an exemplary robot to test a multi-agent team's behavior in a working environment. Here, the basic and complex behaviors of the robots to find their way inside the space, avoid collisions and organize to execute duties will be presented.

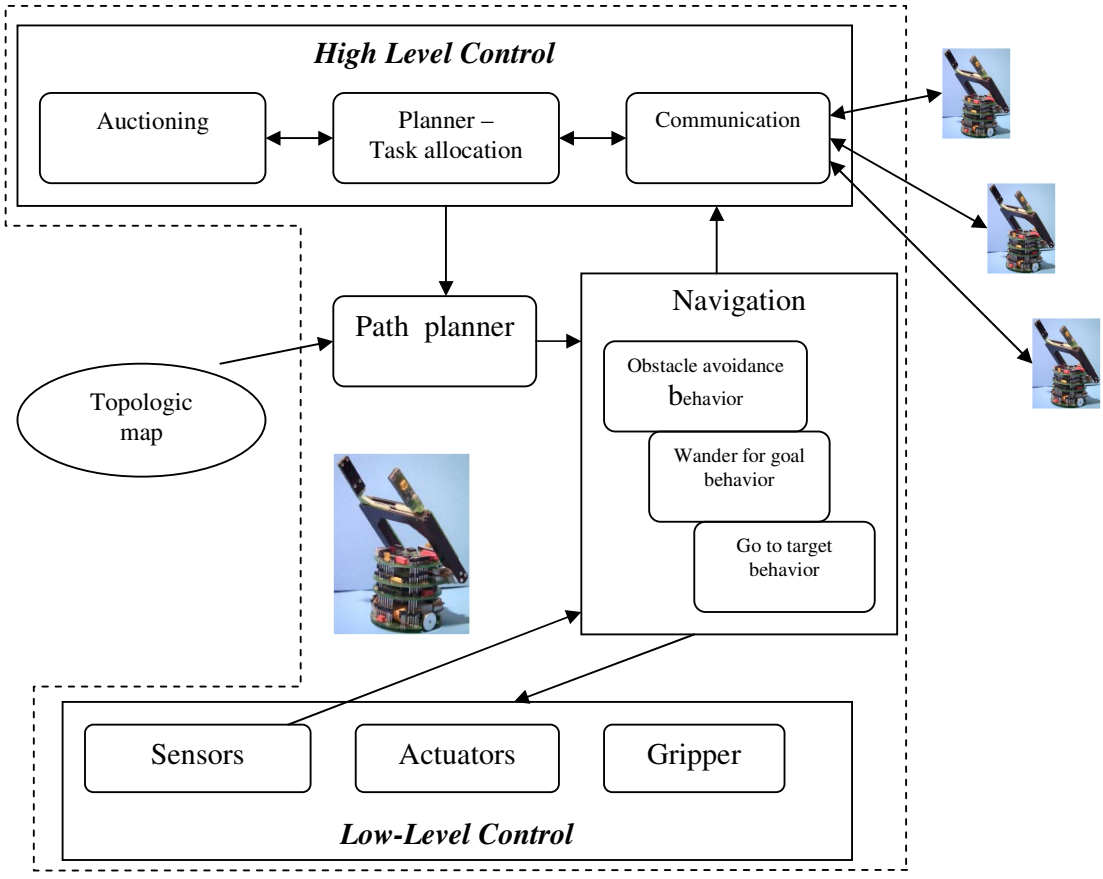

Figure 7.1. System architecture and relationships chart for proposed multi-agent configuration.

The Figure 7.1 summarizes an exemplary architecture for proposed multi-agent configuration. The team has been composed by homogenous agents. The components of this architecture will be presented in the following pages.

7.2. Basic Maneuvering and Obstacle Avoidance with Sensor-based Control

As explained in previous sections, a robot could have several layers of control, which one of them would control the mobile robot's behaviors in different levels. Among these behaviors, we will start with obstacle avoidance, which is executed by a reactive layer, commanded by sensor's direct intervention to actuators.

Sensor based control is characterized by rapid, real-time response to unexpected obstacles by the mobile robots. Let's think about our displacement processes: If we plan to move from a city to another, we can use a roadmap. But the roadmap does not show all the dangers that we can confront. In such situation, before our brain, our reflex system intervenes to protect us from the dangerous situation. While avoiding a car coming through us, we act very quickly to escape; not by a deliberative thinking process but with short and quick actions. The same process will be implemented in our robots. An industrial space is crowded by many moving entities. Among a global planning for movements, rapid reactions to unexpected situations is a requirement for our robots.

To execute these actions robots must be equipped with a series of IR distance sensors, which measures robot's distance to objects and also which commands to actuators who are moving differential wheels. In Figure 7.2, a schema of the sensors is presented. Once the sensors are triggered, differential wheel controls the wheels, so that they move in opposite directions in order to maneuver to left or right side.

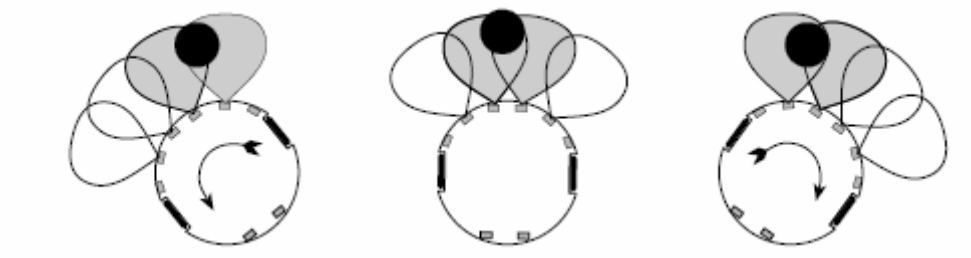

Figure 7.2. Khepera's IR distance sensor configuration.

The Figure 7.2 shows Khepera's IR sensors. The IR sensors are adjusted to command wheels' speed to avoid contacts with other objects. Possible command could be, making robots turn by wheels' speed difference or just stopping for a time

It is still possible to apply different techniques for maneuvering. Basic Braitenberg algorithm [1], or neural networks based systems could be used in maneuvering. The essential of these maneuvering is to turn the robot quick and enough to avoid collisions. Of course, after this maneuver, a robot has to continue its way to destination. So, the orientation of the robot must be quickly restored toward its destination after avoiding the obstacle. For this case, a behavior-based approach is implemented: The behavior for wandering for goal destination and the behavior for avoiding obstacles is summed to obtain an efficient, goal-oriented and collision-free displacement behavior. Over the RRT path, traveling over a defined segment of the path, that means, while traveling between two node points of the RRT path, if a moving obstacle bears or a fixed obstacle's position is changed suddenly during time to cut-off the pre-defined path, this behavior will avoid the obstacle, but also keep the robot's movement toward to destination. Figure 7.3 shows a sequence of moves for avoiding an obstacle while trying to reach the goal point.

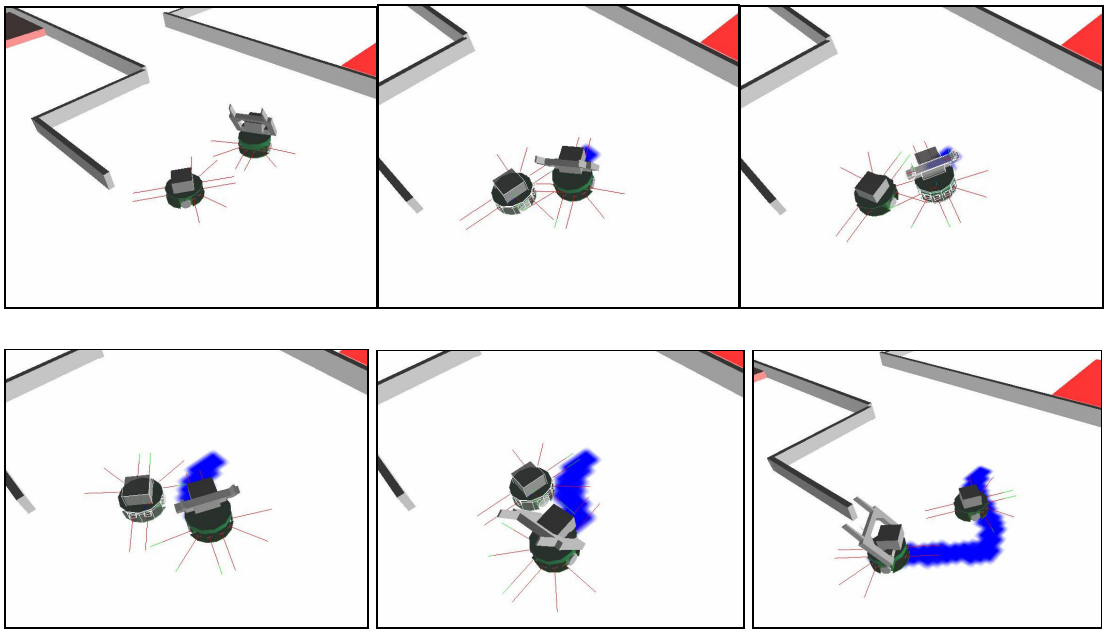

Figure 7.3. The sequence of avoiding obstacles.

In the Figure 7.3, the blue trace shows worker robot's movement. Also, sensor rays of robots are visible. The worker robot meets another moving object, sensor triggered and robot maneuvers to avoid it. After maneuver the robot continues its move toward its goal point.

7.3. Path Planning of Robots

In our proposed system, the production job-shop is observed by a fixed camera on the ceiling (Figure 7.4). All the fixed or moving objects and robots' position can be determined by this camera. So, the camera can provide "perfect information" of positions about the job shop. Although not realistic in general, this kind of model is applicable as production areas have well designed boundaries and most of the machines are fixed. Only human workers and other robots need to be considered as moving objects. Thus, a supervisor system can provide the robots, their position information. This resembles an AGV but it should be noted that robots holds still their autonomy. If desired, the robots may inform themselves by their cameras and their object recognition and map-making abilities. But this solution is more costly. An overlooking system's help could provide more accurate information. The robots could also have their own position information systems in addition to the overlooking system, in order to increase their robustness in case of failure of the

overlooking system. Figure 7.4 shows an example of such an overlooking camera. The position of fixed and moving obstacles and absolute position of robot can be retrieved from such system.

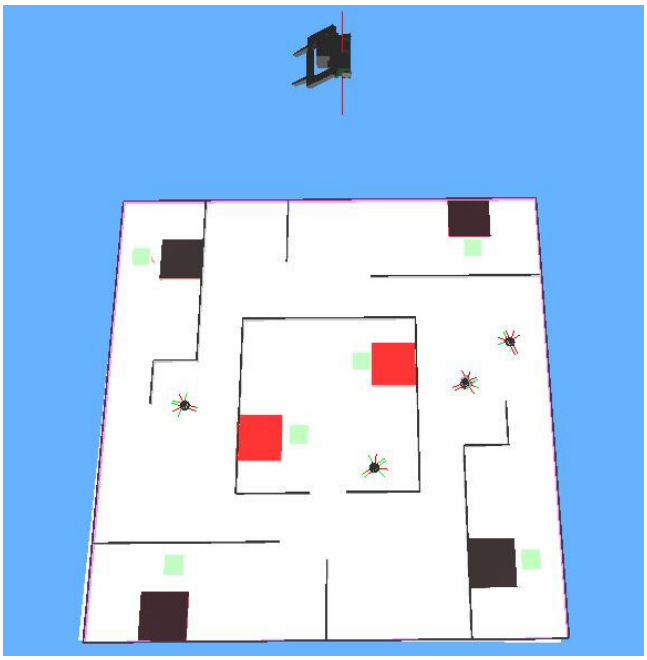

Figure 7.4. Fixed supervisor camera watches the area from bird's eye view.

For today's AGVs' applications, fixed paths are defined for displacement for vehicles. If not interrupted by trespassers, these well-defined paths are accurate and they can provide the shortest paths possible. However in real applications these kinds of defined paths have a big setup cost and they are often interrupted by other moving objects. The area on the AGV's path is not useable by the others in the job shop so very precious production area is preserved for only one purpose.

Having perfect information about objects' positions, the robots can make their own offline path planning for moving from one position to other. Given its actual position information and the destination point, a robot can plan its path to follow while displacing. This path is collision free during planning, but dynamically moving objects can interrupt this path. Of course, the path should be the shortest possible path to perform better results.

On a high level, knowing the map of the environment and position of obstacles, the planner generates a trajectory. To generate this trajectory, we have worked with the RRT algorithm [35], which will be explained in the following sections. Even, if the objects and obstacles change their positions during time, the robot could prepare a new path to follow. After defining a path and while following this path, if the robot encounters suddenly an unexpected object like a human worker or another robot, it can avoid it and follow its path. To execute this last maneuver, we have decided to use a sensor-based low-level control. By these means, the autonomous robot system gains robustness against external interruptions and they become a considerable system for production areas as well as outdoor applications. The Figure 7.5 shows the sensor configuration of simulated Khepera Robot. IR sensors on board are used to fast, low-level reactions to any moving obstacles

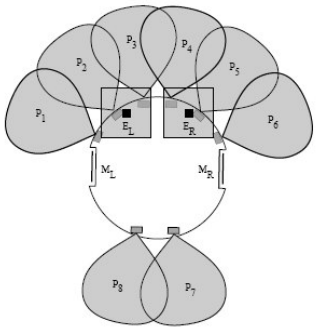

Figure 7.5. Sensor configuration of simulated Khepera Robot.

In our design, all of the robots define a path to reach their goal position taking to their actual state into consideration. This path is renewed with the changing state of the system. The new path could not be the same, but should always be a collision free one. This path is learned by the robot and improved in time if there is no obstacle over the “ideal” collision free path.

So, for reaching from one point to another, offline path planner defines intermediary points and by reaching these intermediary points, the robot could reach the final point. Over this path, there are many “between two points” displacements via a straight line or an arc. To traverse this line or arc segment, the robots will adjust the speed of their differential wheels. The information of its position and its bearing (orientation) allows the robot to

plan the path and then send the necessary commands to the low-level, which means here the actuators. Two wheels, which can turn separately, will provide necessary displacement. Hence, the steps followed by a robot to reach from one point to another are described in below:

The distance *D* between two points (when traveling in a planar 2D system) can be described with the following equation:

$$\boldsymbol{D} = \sqrt{(x_{previous} - x_{current})^2 - (y_{previous} - y_{current})^2} \tag{7.1}$$

The control system considers the distance D to be traveled as the Distance Error and tries to tend it to 0.

As discussed previously, we assume that instead the robot localize itself; a supervisor program is used to read the orientation and position of the robot explicitly. But these data are only taken for testing purpose. It is not used directly in navigation of the robot. The number of turns made by the robot is calculated from the x, y coordinates and distance as in the below equation:

$$\textbf{Angle } \theta = \cos^{-1}((y_{previous} - y_{current}) / D) \tag{7.2}$$

$$\textbf{If } \sin^{-1}((y_{previous} - y_{current}) / D) < 0 \textbf{ then Angle } \theta = \textbf{- Angle } \theta \tag{7.3}$$

In these equations, Angle θ represents how much the robot turns, at each step. If the amount of turn is above a threshold, then this turn is taken into consideration. These data would give the number of turns the controller makes. *D* is given in Equation (7.1).

The same approach could be implemented for orientation check with an addition of a feature: This time, the control system would allow an error until a given limit. If the orientation difference is higher than this threshold level, an adjustment of orientation is done before continuing to travel. A control scheme of the actuators is given in below:

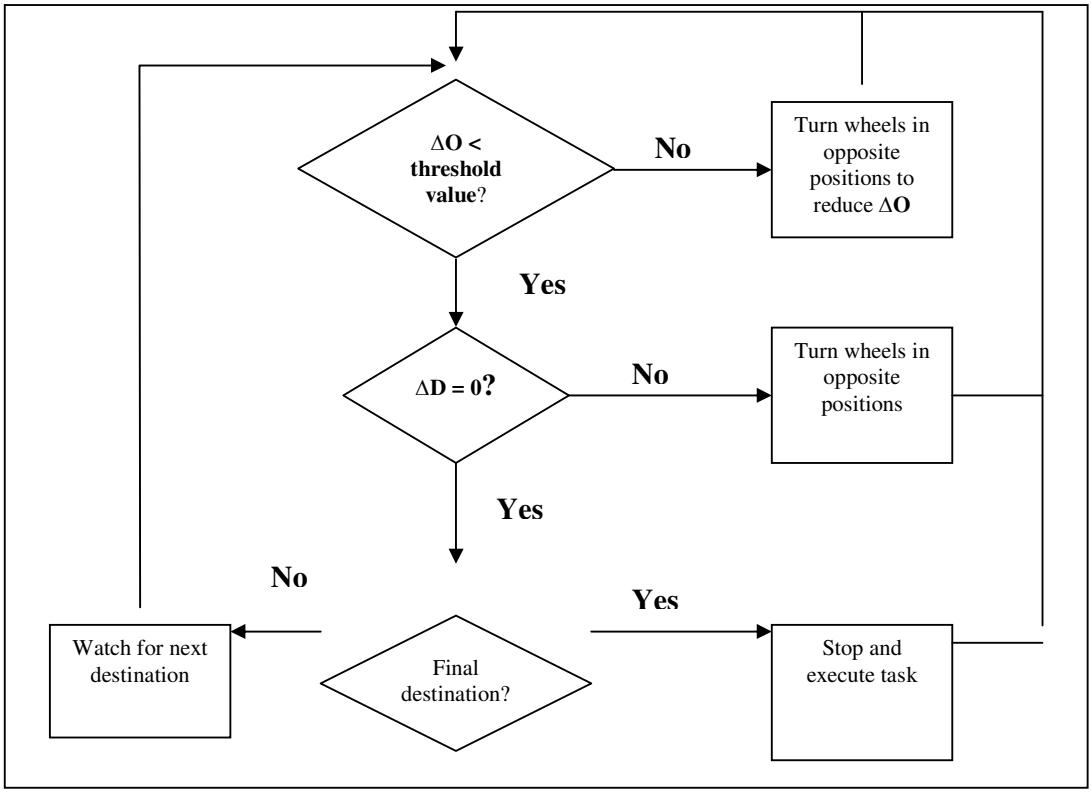

Figure 7.6. Actuator control for displacement from a departure point to a destination point.

7.4. Offline Path Planner – RRT Algorithm

A sampling-based path-planning algorithm [35], Rapidly-Expanding Random Trees (RRT) is an algorithm where a tree composed of random samples of points as leaf nodes are distributed over a 2-D space (or even 3-D) (Figure 7.6) [38]. The pseudo-code for RRT algorithm is given in below [35]. Beginning from the source node point (the origin of the tree), leaf nodes are sampled around the root node and the first expanded node point of the tree (first expanded leaf) is connected to sample nodes around it. Sampling parameters for finding leaf nodes are the distance from the root node and the sample density. In the next step, sample leaf nodes are checked whether they coincide over obstacles or not, and if true, these points cannot be used for the navigation of the robot, then, they eliminated (Figure 7.6).

In Figures 7.7 and 7.8, the algorithms are used for building and extending an RRT

Build RRT Algorithm

Input:

q_0: the configuration where the tree is rooted

n: the number of attempts to expand the tree

Output:

A tree $T = (V, E)$ that is rooted at q_0 and has < n configurations

1. $V \leftarrow [q_0]$
2. $E \leftarrow 0$
3. for $i = 1$ to n do
4. $q_{\text{rand}} \leftarrow$ a randomly chosen free configuration
5. extend RRT (T, q_{rand})
6. end for
7. return T

Figure 7.7. Build RRT algorithm (from Choset et al. [35]).

Extend RRT Algorithm

Input:

$T = (V, E)$: an RRT

q: a configuration toward which the tree T is grown

Output:

A new configuration q_{new} toward q, or NIL in case of failure

1. $V \leftarrow [q_0]$
2. $E \leftarrow 0$
3. for $i = 1$ to n do
4. $q_{\text{rand}} \leftarrow$ a randomly chosen free configuration
5. $E \leftarrow E \cup \{(q_{near}, q_{new})\}$
6. return q_{new}
7. end if
8. return NIL

Figure 7.8. Extend RRT algorithm (from Choset et al. [35]).

The following step is to connect the remaining available points between them. This process begins with connecting the point of origin to the nearest points in a given range. This is the root of our tree. Then, in every branch, the leaf point is also connected to an available point. With repeating iterations, our tree grows out. The tree grows bigger till it reaches the goal point (Figure 7.9).

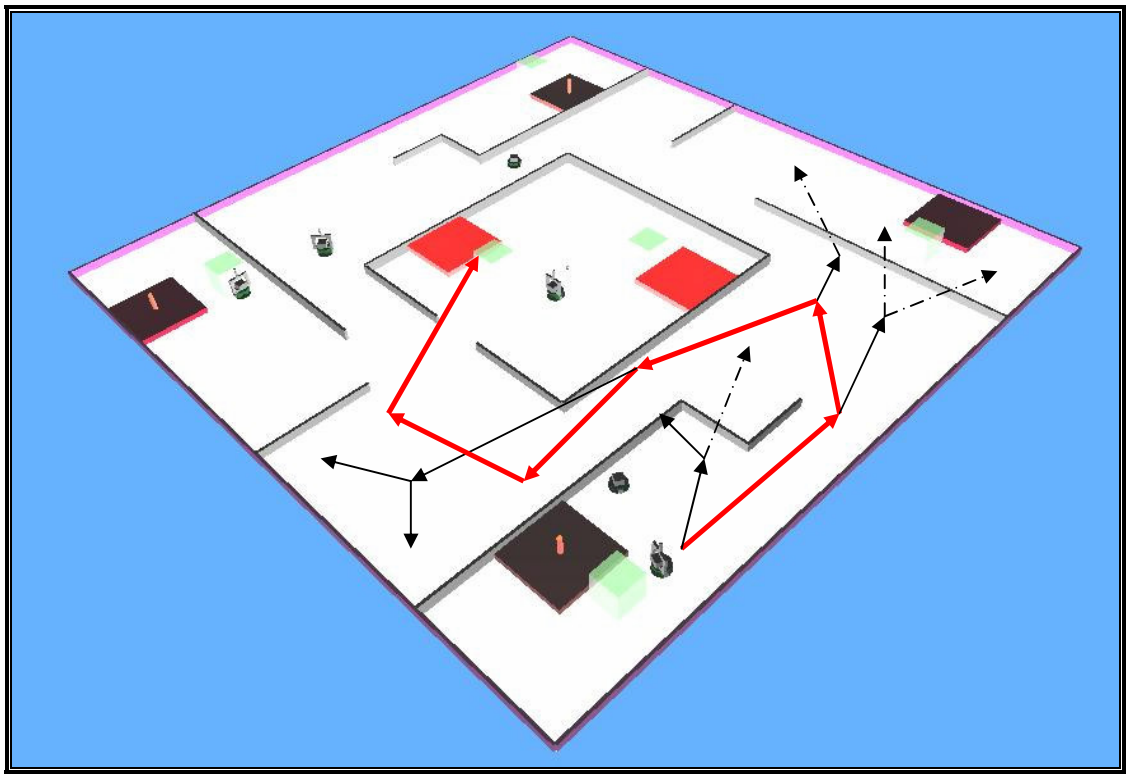

Figure 7.9. A growing RRT tree.

In the Figure 7.9, the RRT connects from departure point to goal point by reaching leaf nodes. Bold arrows show a successful tree, which gives the path for robot. The dashed arrows represent leafs coinciding with static obstacles, which are immediately eliminated.

Another version of RRT algorithm is RRT-Connect. In this version, roots are more than one. Two trees begin to grow their branches, the first from the departure point and the second from the destination point, and they are connected in the mid-way (Figure 7.11). These trees grow even faster and they tend to find a feasible route faster than the regular RRT algorithm [38].

Connect RRT Algorithm

Input:

$T = (V, E)$: an RRT

q: a configuration toward which the tree T is grown

Output:

Connected if q is connected to T; otherwise failure

1. repeat
2. q_{new} ← extend RRT (T , q)
3. until ($q_{new} = q$ or qnew = NIL)
4. if $q_{new} = q$ then
5. return connected
6. else
7. return failure
8. end if

Figure 7.10. RRT-connect algorithm (from Choset et al. [35]).

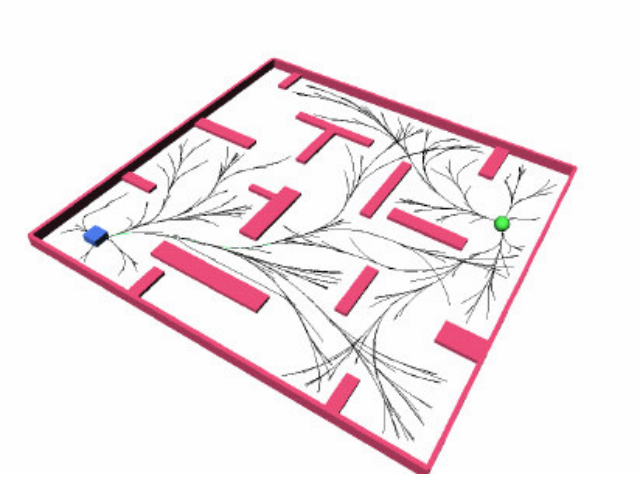

Figure 7.11. RRT-connect algorithm reaches nodes from both nodes.

In the Figure 7.11, differently from the previous Figure 7.9, one tree starts from departure and one tree starts from the arrival point and they are connected in the middle field. The search takes less time.

The final step of the path generation by this algorithm is to optimize the path by shortening the path to follow. For this, all the points are checked if between n^{th} and $(n+2)^{th}$ node, if the line segment is not coinciding with an obstacle and these points are connectable, then the path is shortened (Figure 7.11). So that, by iterations, all the nodes are checked and the feasible path is optimized whenever possible.

RRT algorithm provides a fast and effective trajectory generator, so the path planning is executed as fast as possible. The solution may not be "the optimal one" but as the environment changes during time and the dynamical obstacles are moving around, having an optimized and feasible path is more important to having the "optimal" shortest path. As during action of the robots, objects are moved around, these planned paths would be revised or changed. Then, RRT is an important tool for hybrid/deliberative algorithm, where equilibrium between reactive and deliberative paradigms of robots is established.

7.5. Optimizing the RRT Path

The RRT algorithm searches the first feasible solution and not the optimal one. If a path is found, this path could be checked to determine whether or not it provides a better solution over the given nodes.

For this purpose, the marked nodes for a feasible path are checked between them to find out if there are shortcut segments between these nodes. The following algorithm in Figure 7.12 explains the shortcut checking:

RRT Path Smoothing Algorithm

1. From node 1 to node n-2 do
2. Check node i, node i+2 feasible
3. if feasible, then eliminate i+1 and connect i, i+2
4. if not, check for i+1
5. loop until i = n-2
6. end

Figure 7.12. Path smoothing algorithm for shortcuts on RRT paths.

An example of shortcut check result can be seen in the Figure 7.11.

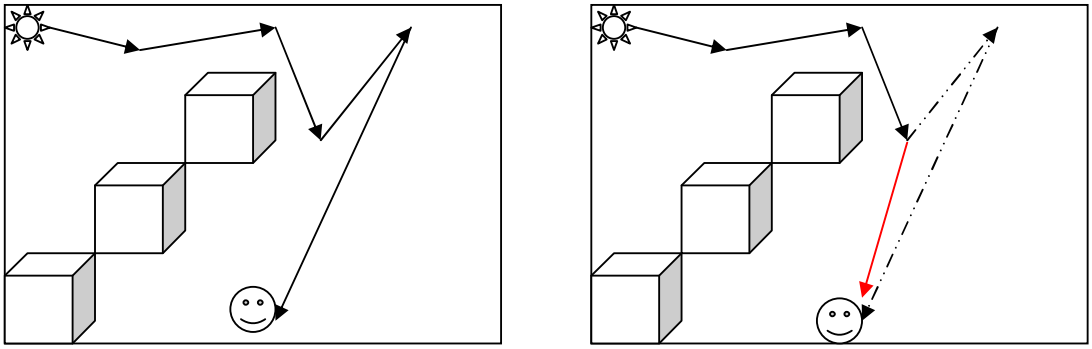

Figure 7.13. The RRT paths; before (left) and after optimization (right).

The Figure 7.13 shows us before and after situations of RRT paths: RRT path could contain excessive pathways before optimization, like in the left picture. After path optimization, RRT path is cleared off excessive leaf nodes (dashed arrows) in order to have a smooth path like in the picture on the right side.

7.6. Path Planning for Multiple Robots

Choset et al. [35] state that all sampling-based planners are adaptable to multiple robot cases and RRT algorithm can be used directly. If needed, small modifications could improve RRT path planning performance.

For multiple robots, each generated path for one robot is superposed and if there is an intersection between paths, probable arrival time to intersection point can be calculated considering the speed policy. If there is a probable collision, then the last generated path is renewed. Choset et al. [35], explains that this type of collision check is more expensive then checking robot paths simultaneously but it is more effective in the covered space.

In our simulation case, RRT algorithm will be used for the multi-agent robot team. For dynamical objects, two approaches can be considered for multi robot path generation. In the first approach, the path generator can generate a trajectory by considering dynamical objects moves or their dynamical movements can be ignored; then, if a robot-robot or a robot-dynamical obstacle (i.e. workers) close encounter occurs, robots will try to avoid these with low-level sensor based maneuvers where the sensor data directly triggers

differential wheel motor control. In the other alternative, dynamical objects trajectory could be taken into account in trajectory generation [41]. As the low-level obstacle avoidance always exists, the second alternative may need less help from the low-level avoidance maneuvers, but it is hard to foresee dynamical objects moves as they can move in a random manner. So, even if the calculation for dynamical objects is done, low-level sensor control maneuvers will be needed. This randomness in dynamical objects moves questions the profit for dynamical objects vectors' consideration. In Section 2, combination of planned paths and low-level sensor controlled maneuvers is explained in the hybrid-reactive paradigm and by superposition of behaviors as summation of wandering vectors.

Figure 7.14. A typical path followed by a worker robot with RRT path.

In the Figure 7.14, a typical path followed by a worker robot can be seen in blue traces. The RRT connection made a collision free path from a pickup node to delivery node. A free navigating non-controlled dynamic obstacle can be seen on center (the round black object in the gate).

Here, the low-level sensor based maneuvers may put the robot far away from its targeted trajectory. In such a case if the robot is still in proximity, it will try to arrive to the latest node to be reached; if it's orientation and distance difference becomes over a limit, a new path for the robot will be generated.

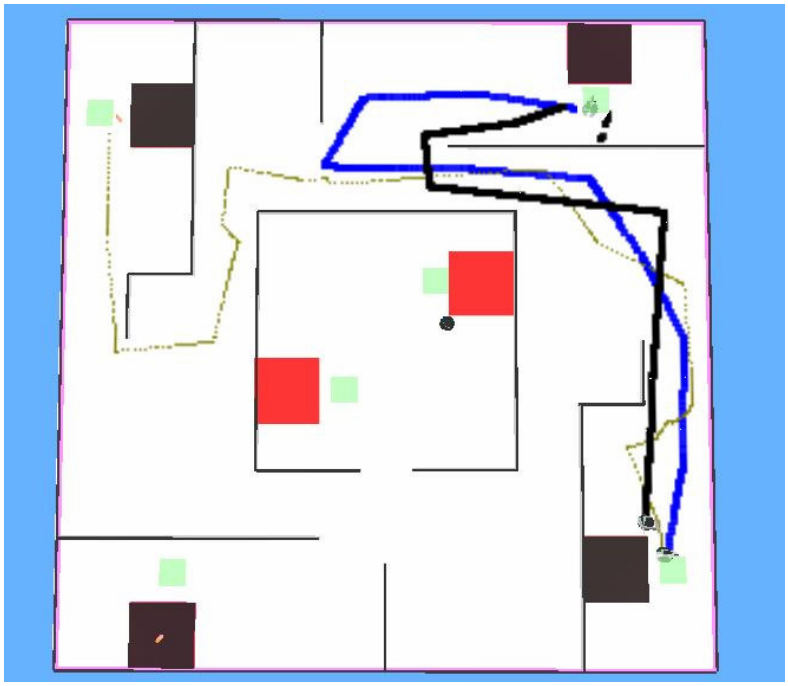

Figure 7.15. Difference of raw RRT path (dark black) and the smoothed path (light blue).

In the Figure 7.15, blue and black traces shows two robot's path. The blue one makes zig-zag maneuvers in its trajectory. However, the optimized RRT -using black trace- shows a smoothened trajectory, where excessive nodes are eliminated if these maneuvers are not necessary (i.e. corner turns, etc.).

To finalize this section, we can see the "Go to Target" algorithm given in Figure 7.14. The algorithm uses sequentially the path planning and the movement issues explained in above pages. Positions to be reached are generated by path planning algorithms.

```
Go to Target and Obstacle Avoidance algorithms

1. received:    current position
                current angle
                to Position
1.    check escaping
2.    calculate wheel speeds to reach to Position
3.    check sensors ( )
4.          distance sensors: sensed
5.          currenPosition, currentAngle, toPosition: received
6.          escaping : global
7.          escapeInfo : EscapingInfo
8.          step ( ) $
9.              if escaping
10.                 proceedEscape ( ) : velocities
11.             else
12.                 velocities = calculate according velocities to reach toPosition
13.                 checkSensors ( ) : escaping velocities
14.             end
15.             apply speed (velocities)
16.  calculate velocities ( ) : velocities
17.       vector = from currentPosition to toPosition
18.      wantedAngle = wantedAngle – currentAngle
19.     if (neededAngleChange > anglethreshold)$
20.          turn_a_bit_to_reach_angle = turn factor * angledifference
21.                                          : velocities
22.    else
23.          goforward (vectorlength * factor > vector's unit)
24.                                          : velocities
25.   result = velocities
26.  checkSensors ( ) $
27.         if (check_sensor_values_exceeding_threshold)$
28.         fill escaping info
29.         escaping = true
30.         velocities = proceedEscaping ( ) ;
31.  proceedEscape ( ) $
32.          follow escaping procedure
33.          if escaping finished
34.              escaping = false;
```

Figure 7.16. Go to target and obstacle avoidance algorithms.

8. CONTROL STRATEGIES OF TASK ASSIGNMENTS FOR MULTI-AGENT ROBOTS

8.1. Introduction

After defining purposes and design of useful multi-agent robotics for indoor production area, now we will look at the possible control processes for these agents.

There are many alternative control paradigms to be implemented for our robot team. Mobile robot control and multi-agent task allocation problems are very active research areas and many different approaches have been proposed for both single-agent of a robot and multi-agent control of a robot team. In this section, different possible deployments are examined.

8.2. Task Assignment Tests: Priorities of Transporting Duties

In a factory workshop, arrival of parts and their priority of processing for obtaining end products is a topic of industrial engineering. Due to the needs of the production control, priority in part processing management may vary. Here, our multi-agent autonomous robot team's mission is to respond to these needs of the production in order to establish the optimum output of the system.

For observing the response of a multi-agent team to different production requirements according to the priority of production, a series of tests will be executed. In this section, we will examine these tests of production priorities. The aim of the test is not only to observe the global system performance, but also to observe multi-agent team's performance while path planning, obstacle avoidance and mob psychology of the group like in the system bottlenecks, like queues in workstations, collision checks in gates or their behavior in heavy traffic jams.

As the possible prioritization schemes highly numerous, one can create limitless situations to test. In order to find a research basis for mostly confronted situations,

examples given Doğan's [33] study about AGV priorities are to be tested here with autonomous mobile robots.

Below, we will explain a series of production scenarios in order to test them with multi-agent autonomous mobile robot team.

8.3. System Performance Measures

A robot team's expected performance is to assure the manufacturing process in a suitable, intelligent, automated manner. Improving this performance must be done by improving manufacture outputs or reducing manufacture expenses of all kinds, if possible. Automation of processes, improvement in transport and delivery times or these transports in a more effective way could be the benefits of our introducing robot workers. To observe the performance of the robot team, first, we have to define the performance measures for the robot team. Possible metrics are given below:

- Waiting periods of parts: Period from arrival time to pickup time of parts, which has to be minimized.
- Delivery periods of parts: Period from pickup time to the delivery time of parts, which has to be minimized.
- Success in service period: If a part delivered less than a given time period.
- Interruptions of robots by moving obstacles (negative criteria): If robots collide with the moving obstacles and they cannot move for a while.
- Distances traveled during pickup by robots: A robot must choose the shortest possible path to reach a destination, so this is a possible metrics to be minimized.

Also, grade criteria could be introduced. For example, robots can get positive points for every collected part or for delivered parts within a given time period. Also, they can get negative points for not served parts for a long time or parts not delivered within a given time period. To attribute these points, parameters like wanted service periods or points to be earned from the results will be defined with the experiment parameters.

To see the evaluation of the system performance, an exemplary game could be like this:

Items arrive in supply posts with a given probabilistic arrival. These items are the components of an end product. Then, we need to arrange their transportation to execute a responsive manufacturing process to the production needs. Let's compose an exemplary model like this:

- 2 different products, each combined by different components: For example:

  Product 1 consists in a mix of: $Ax_1 + Bx_2 + Cx_3$

  Product 2 consists in a mix of: $Dx_1 + Ex_2 + Fx_3$
- We consider that the manufacturing process is flexible, so the product to be assembled can be changed or shifted during production.
- Component parts arrive with a probabilistic distribution.
- Multi-agent system responds to the demands to produce which product, by its organization and structure.
- Posts are arranged with a distance between them. So, when a component arrives, the closest autonomous agent goes toward it. It takes the component and carries to the assembly line.
- An agent decides to transport which component to which line by taking a decision, after communicating with others.
- The communication process is the most important feature in this design. An agent has to:
  - check the priority of the products,
  - check the present status of the line,
  - check the priority of the coming component,
  - compute the time needed to carry it to the line,
  - decide to carry or not to carry wait for another component,
  - decide to leave it or continue carrying while carrying a component.

So, the autonomous agents would aim to assure the total flexibility of the production line. The above-mentioned metrics can help to figure out their performance.

8.4. Auction-based Assignment: Market-driven Method for Robot-part Matching

A recent approach for auction-based multi-agent task allocation games is the market-driven algorithms. Market-driven approaches for multi-agent control are explained in detail in Dias [16] and Köse et al. [34]'s and market-driven controls have proven these algorithms' reliability in robotic soccer games in 4-legged robot league with an AIBO team. Here, we will first briefly look at the basics of market-driven approaches, and explain their usage in our multi-agent team for the factories.

8.4.1. Market-driven Method

This method takes its name by its similarity to sales in a market. In a market, resources are traded by individuals, where every individual looks for maximizing its personal profit [16]. In multi-agent robot platforms, resources to be traded are sensors, map info or tasks to be completed, etc. These trading resources are put on an auction and robots of the team declare their cost, or we can also say, offer their prices for the traded resource. The lowest pricing robot gains the bid. An example is that a task is not assigned to any robot and two robots are free. This task is associated to lowest cost declaring robot. An implementation of this method is applied to Sony AIBO robots for 4-legged Robot Soccer plays by Köse and al. [34].

The algorithm is suitable for both centralized and decentralized approaches. The robots make auctions to take responsibility for jobs. Then, we can call this organization as "collaboration by competition" [34]. Generally, there is an overall goal for teams, which is dividable into its components. By executing the components, multi-agent team could achieve the overall goal. For example, in an exploring mission, to see the entire map, each of team members would explore a smaller part of the area and then these maps are put together to get the complete map of the area. Another example is the robotic soccer games: To score a goal, each team member could pass the ball to another member and so on, until a player could see the opponent's goal in a suitable position to shoot. This decomposition of tasks allows dynamic auctioning during the execution of the tasks, in order to increase system performance: As in the example of soccer playing robots, once a robot gets the ball and it can see the goal of the opponent team, it can even pass the ball to another friendly

robot if this robot is in a better position for shooting, that means if it gives a better bid for the task and so, the possibility to score a goal increases, which increases the overall system performance.

In order to bid, robots must calculate a cost function. Here, we defined this cost function related to two features: The expenses of the robot to execute that task and a relation which gives the cost of not executing the task; and the inverse of the benefit to be obtained from task's execution. The net profit is calculated by subtracting estimated cost for accomplishing the task from the revenue of the task. In Eq. (8.1), an estimated cost function $C_{r,i}$ is given in terms of 1 / benefit weight points (μ_i) of the incoming load$_i$ to be transported, multiplied by rth robot's calculated path distance to the referred loading node. So, the load$_i$ will be assigned to robot$_r$ which bids the lowest min $C_{r,i}$ for that load. Equation 8.2 gives sum of the line segment distances to be traveled by robot r to the node j.

$$C_{r,i} = \mu_{part_i} . d_{node_j} \tag{8.1}$$

and

$$d_{node_j} = \sum_{j=1}^{n} \sqrt{(x_{1,j})^2 + (y_{j,1})^2} \tag{8.2}$$

These costs are calculated by robots and communicated among the team. After these auctions, task assignments are done. To define a task assignment strategy, we have to define the auctioning situations and the repetition of auctions. If the robots make auctions in every step during the runtime, it may cause the slowing of the system.

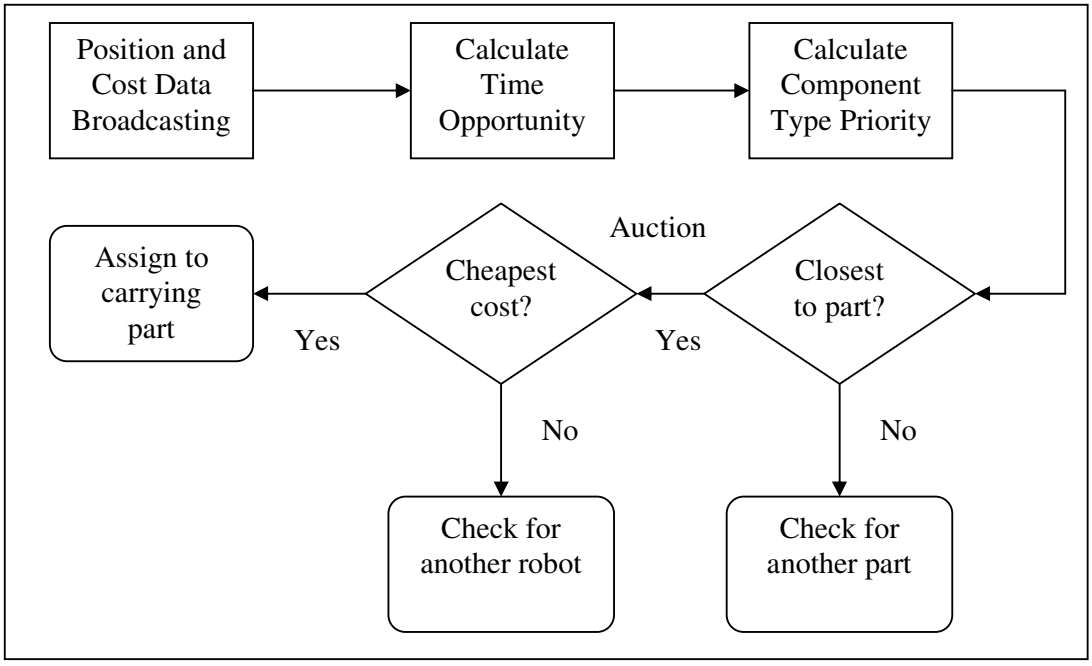

Figure 8.1. Market-driven algorithm for task assignments.

It must be noticed that a successful system should not only look for individual goals, but also check these results for individuals, and make new task assignment combinations to increase overall system performance. Equation 8.3. makes the winner bids for assignments suitable for the system's overall benefit, having *m* robots and *n* nodes to go:

$$Min.\sum_{r=1}^{m}\sum_{i=1}^{n} C_{r,i} \tag{8.3}$$

Beyond those auctioning basics, there are also optional alternatives for auction repetition [17]. The above benefit function, which minimizes the costs, could be checked over time. In one example, once a robot is assigned to a part, it is moved from occupied list and enlisted to busy list, then, until to the end of all pick-ups, transport and delivery processes, this assignment could be preserved. On the other alternative, in every status change for each one of the team member robots, a new auction can be made among the agents.

As seen in the Figure 8.2, robots make auction for parts. Part priority, multiplied by sum of the line segments which gives the total pathway to be traversed by the robot, gives the cost of the parts auction value. In every status change, free robots calculate their costs and bids for incoming parts. After calculating the best assignments for the overall system, the robot that bids the lowest cost for a part is assigned to that part. Note that, the auction's aim is to achieve the best performance not only for individual robots, but for the overall system, so that, even if a robot made the best bid for a part, it may not be assigned to that part and instead another robot could be assigned because the mentioned robot could be useful for another task, which will increase the system's overall performance.

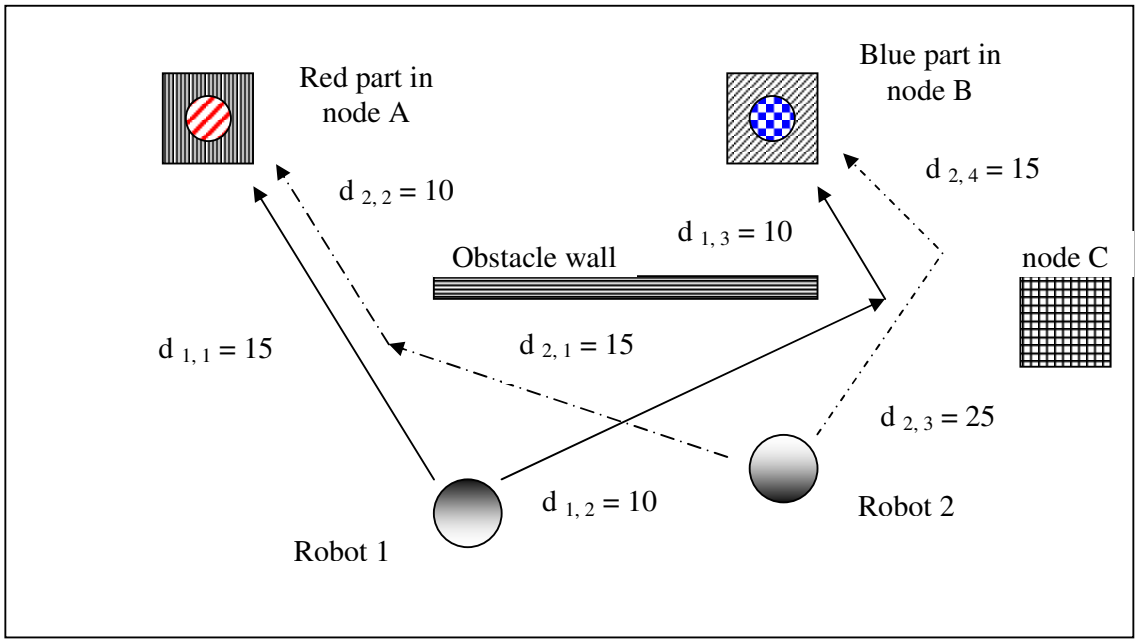

Figure 8.2. An application of market-driven auctioning for task assignments.

An exemplary scenario given in the Figure 8.2 above could help to understand Köse and al. [34] proposed process: Distance is calculated by making the sum of robot's trajectory's components, and then this cost matched by the priority of the waiting part. The system tries to maximize its overall benefit reducing the global cost. If Robot 1 chose the node A for a cost of 15 to maximize its profit, then the Robot 2 would chose node B for a cost of 40. But, if Robot 1 takes the node B for a cost of 20 instead of choosing the node A, then the Robot 2 would chose the node A for a cost of 25, which makes the overall cost reduced.

8.5. Matching and Routing of Multi-agent Robots

Generally, a transport job in a workshop consists of several elements: Decision of picking a load, arriving to loading area, loading a carry load, carrying it to its destination and then unloading and looking for the next job. Matching, dispatching, routing and scheduling terms refer to these actions. A good explication for processes is given in Doğan [33] and in Bulak [43].

Different relations between these actions, creates different production algorithms. Different algorithms have also different heuristics. One mainstream algorithm for matching and routing processes is given below. This algorithm uses sub-algorithms, which one of them will be also explained. By changing the sub-algorithms, this algorithm allows to execute different manufacturing sequences, which allows the flexibility of the system.

Mainstream pseudo-code is given in the Figure 8.3. The primary objective is to check available robots, part and node states. A robot-job matching will be done after considering the procedure to follow:

```
➢ begin
While EndTime is not reached
{
        ➢ control arrival node state
            ➢ if new part arrival, then
                inform Robots (or check RobotState for all robots),
                    ➢ Procedure MatchDecision
        choose compatible FreeRobot to match with part,
        change RobotState list
    make Auction and select robot and assign to part after bid,
    remove the last selected robot from FreeRobot list
    let robot search a path,
        ➢ control delivery node state
            ➢ if new part deployment, then
                change RobotState to available
                add the last available robot to FreeRobot list
                change PartLogs
    increase SystemPerformance points
        ➢ control PartState
            loop until DeleteTime
            ➢ if not picked for a time interval, then
                augment PartPriority
                ➢ if not picked until DeleteTime, then
                    delete Part
                    change PartLogs
                    decrease SystemPerformance points
        ➢ check RobotState list
            ➢ if RobotState list changes, then
                    ➢ Procedure MatchDecision
        choose compatible FreeRobot to match with part,
        change RobotState list
```

Figure 8.3. Mainstream pseudo-code.

8.5.1. Parameters for Multi-agent Matching and Routing Control

As main processes in a workshop are cited above heading, an improvement in a production system in terms of these common processes can be the following ones:

- Time saving for production.
- Increased output in a limited time.
- Right placements of goods.
- Prior production for preferred goods.

A flexible manufacturing system must response to above described conditions [42]. Here, our multi-agent robots will try to realize the above conditions. To make possible to respond the possible properties and priorities, an adjustable control is introduced for our proposed system: The above mentioned production priorities are associated with parameters. These parameters are presented in our simulation as follows:

Parameters:

- Agent Priorities:
  - Pickup Priority of n^{th} agent > could be more or less prior to other agents.
  - Delivery Priority of n^{th} agent > could be more or less prior to other agents.
- Node Priorities:
  - N^{th} Pickup Node's Priority > could be more or less frequented.
  - N^{th} Delivery Node's Priority > could be more or less frequented.
- Auction Properties:
  - No auction: Assignments are made in random order.
  - Path cost effect on auctions: a feasible path's distance is calculated and communicated between the groups. The job assigned to minimum costly (the highest bet) agent.
  - Time effect on auctions: Waiting time of the objects' are taken into account.

- Type priority on auctions: Some parts are prior to other parts. These parts have a greater point while scoring.
- Combination of three priorities: Time & cost, time & type, type & cost and type, time and cost effects may be calculated at same time.

• Re-auctioning:

- No re-auction: Once matched and assigned, a robot or a node participates no more to auctions. That means, once an assignment has been made, the decision is not changed until the node or the robot are liberated.
- Re-auction: They can join other auctions till reaching an assigned. That means, even if an assignments has been made, after a change in statuses as more free robots, system makes a new auction for distribution of missions.
- Re-auction time: Which status changes makes a re-auction or what time period between the re-auctions.

The re-auction property is a crucial decision on outputs. Even if the matching is made following the path costs and priorities, the conditions could change in a dynamical environment. A newly available robot could be a more convenient assignment for a task. So, when the statuses change, the system should control its prior decisions.

In a multi-agent system, this decision depends on all agents' statuses. Making the auction between all agents is an extra charge for the system and may decelerate the process. The right decision for enabling re-auctioning would be tested before introducing.

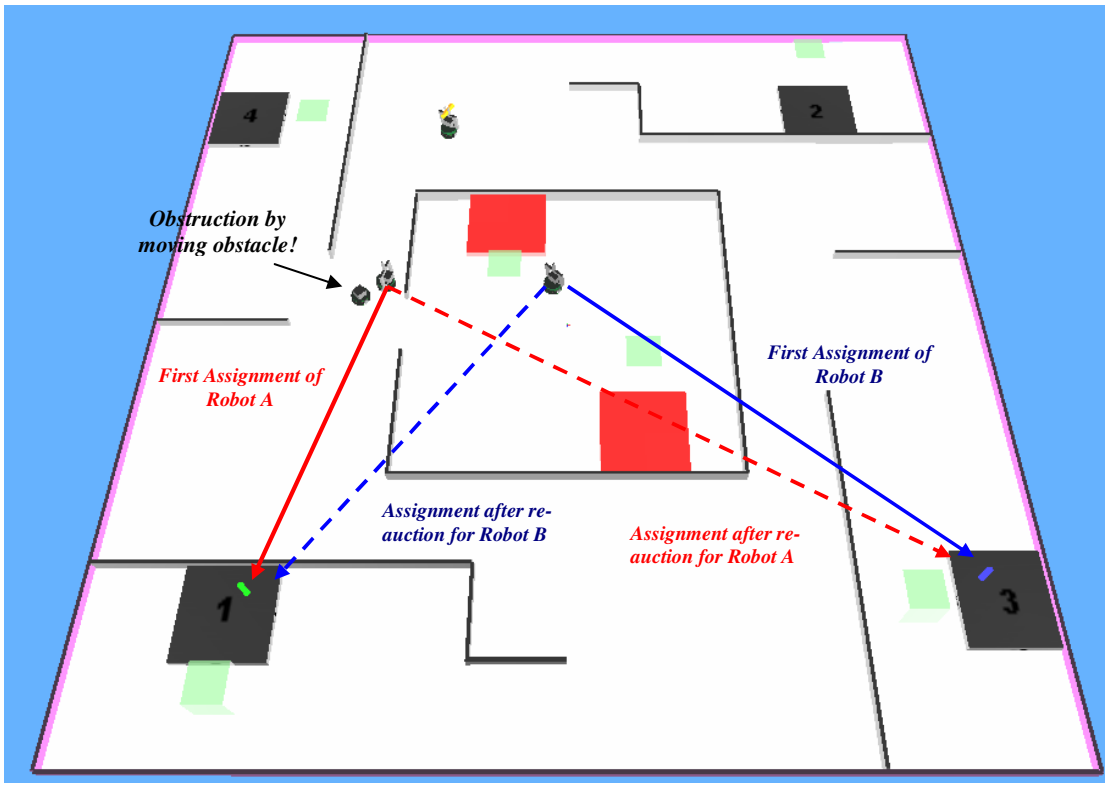

Figure 8.4. Re-auctioning.

Figure 8.4 shows an example of re-auctioning effect: Robot A is already assigned to node 1 (straight red line) as the closest robot to that node where a prior part is waiting to be served in urgency, and Robot B is assigned to node 3 where a lower priority part is waiting (straight blue line). However a moving obstacle comes in and cuts off the path of Robot A. While re-auction occurs, Robot B is assigned to node 1 (dashed blue line), while Robot A is trying to avoid the obstacle. After avoiding the obstacle, Robot A could be assigned to node 3 (dashed red line), where a lower priority part is waiting. As the environment is dynamic, it is useful to check the statuses of the agents. The re-auction checks the system elements statuses and tries to find if there is better solution opportunity, which is occurred over time.

8.6. Sub-algorithms for Matching and Routing Scenarios

Below, sub-algorithms for matching transport duties of parts to robots are presented. In manufacturing processes, different priorities for part transportations or different priorities for routing of robots are possible. These cases are given as examples to be tested, among many possible configurations. Within the overall simulation program, one is capable of testing its desired configuration after preparing the code allowing the system of multi-agent robots to act in their desired behaviors.

8.6.1. Nearest Pickup First: Assignment with Respect to Path Cost

In the scenario, parts are incoming randomly to pickup nodes to be collected by robots and then to be carried for delivery nodes. In this matching algorithm, a free robot is assigned to the closest part in its proximity. Robots do not consider parts' waiting time or transport priorities of parts. An available robot will be assigned to collect the nearest part to itself in each arrival. This situation forces robots to choose the closest part for each auction. Then, the algorithm can be referred as path cost saving algorithm.

8.6.2. First Come - First Served (FCFS): Priority due to Arrival Times

The second algorithm differs from the previous one by a transport priority. By their arrival time, the oldest part has to be delivered first. Here, the older the arrival time of the part, the prior the part to be picked. Then the first available robot is assigned to collect this part.

We should note that, in FCFS the older parts are the first assigned ones but they are not necessarily always the first arrivals. During pickup and delivery processes, the travel time of an older part could be longer than a newer part due to the route to be traveled and to traffic jams.

8.6.3. Privileges of Components: Priority due to Type of Incoming Parts

This matching algorithm is still concerned about transportation priority of incoming parts; not with their arrival time, but their type instead. Apart from its arrival time, agents, before any lower priority part, would serve a more prior part. To make a distinction of parts, red, green and blue loads are introduced. The transportation priority descending from highest order is set like blue, green and red.

8.6.4. Type & Time Priority Combined

In the latter matching algorithm, time priority of parts and priority of different type of parts is combined to obtain an intermediary solution for matching: Part type priority is multiplied by waiting time of the part, which gives the service priority.

Service priority = part type priority x waiting time of the part in the queue

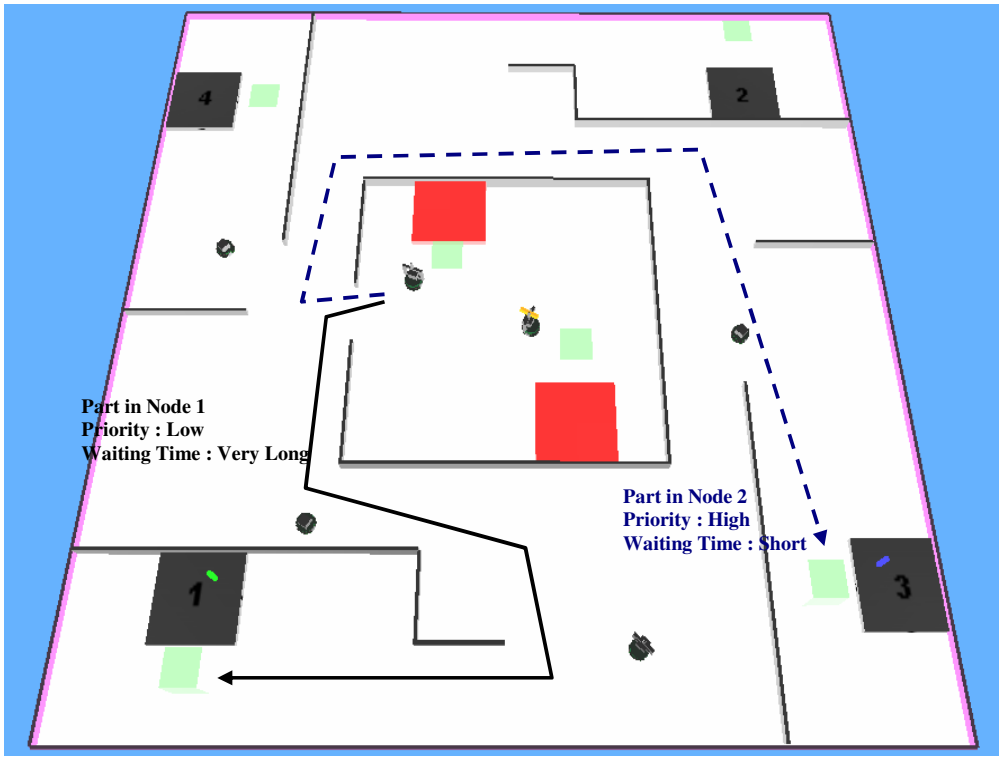

Figure 8.5. Service priorities.

In the Figure 8.5, we can see an exemplary situation: When type and time priority are combined, a longer waiting lower priority part (green part in node 1) could be picked before a newcomer, high priority part (blue part in node 3). For example, having the same arrival time, a blue part would be served first than a green colored part, but a long-waiting green colored part would be served before a new-arrived blue part.

We should note that all these scenarios could be multiplied with different arrangements. Here, the aim is to try to measure the performance of an autonomous system's responses in a flexible production by most common metrics.

In the Figure 8.6, the auctioning algorithm is given. The algorithm can be used either in calculation of the priorities of jobs and to make the assignments of jobs to best offering robots.

Pseudo-code of Auctioning Algorithm

```
1. free Robots = set of free for assignment robots;
free Jobs = set of free for assignment Jobs;
MapRobotJobCost = calculateEachPossibleCost
SortedCosts = sort MapRobotJobCost
  do while robot & job remains
select best robot job, those not registered yet
register them as assigned, to each other
{
calculate Each Possible Cost
for each job
  for each robot
    if (robot can do the job)
      register cost = calculate cost
            (robot, job);
      {
    {
  {
  result = registered cost map
{
calculate cost (robot, job)
  path cost = path (robot, job) . cost ;
  result = pathcost * pathcostmultiplier +  priority Robot * Robot priority multiplier +
priorityjob * Job priority multiplier + timejob * time multiplier + time Robot * time
robot multiplier
```

Figure 8.6. Auctioning for parts algorithm.

9. EXPERIMENTS, RESULTS AND COMMENTS

After being defined our objectives, our modeling and our tools, now we will look at the different results that we obtained from simulations and make comments on AMR use for manufacturing plants.

Simulations are done in Webots 4.0.27. The basic simulation step is set to 64 ms. Display output is set to 4096 ms for Webots 4.0.27 version.

As experimental design parameters, for each experiment, the simulation ran for 15 minutes. Each experiment is repeated for 20 times. The job-shop is designed with a number of 4 arrival nodes and 2 delivery nodes. In the experiments, the number of robots is chosen to 3 robots to better observe the trend of multi-agent team's behavior, as there are 4 arrival nodes and 2 delivery nodes. A number of non-controlled robot are introduced the human workers as dynamical obstacles. Each experiment will be repeated with an increasing number of non-controlled robots. Statistics are collected from the log of events in the simulation.

Our simulation offers to try the performance with multi-agents with different configurations. So, an option offered by our simulation is to attribute service priorities to the nodes. Also, to simulate a heterogeneous team, agents could have priorities while assigned for missions, i.e. a faster robot could be assigned to more missions than other robots. But in these experiments, we have chosen to simulate a homogenous multi-agent team; these options are not used by default configuration. Then, the nodes have equally priorities to receive robots and the robots will have the same priorities to be assigned to missions. Table 9.1 summarizes experiment design parameters:

Table 9.1. Experiment design parameters.

Pick-Up Nodes	4
Delivery Nodes	2
Simulation Time	15 minutes
Simulation Runs	20 runs for each experiment
Number of Robots	3
Dynamic Obstacles	1 to 5
Node Priorities	All nodes have equal priority to receive parts
Agent Priorities	All agents have the same priority to be assigned for pickup mission and entrance to delivery nodes

9.1. Metrics

In order to compare the performance of the multi-agent team in factory simulation, a set of metrics is proposed to measure the proposed system's performance. These metrics are:

- Average of collected parts: This metric gives us the average of collected parts after 20 runs.

- Standard deviation of collected parts: Standard deviation of collected parts after 20 runs.

- Weighted average of collected parts: The goal on the team is not only collecting parts but collect parts with respect to their priorities. Different points will be assigned for each type of different parts, to animate a priority case in the simulation. In the following experiments, weight of parts will be defined as follows:

    - Part_0 (red) : 1 point
    - Part_1 (blue) : 2 points
    - Part_2 (green) : 4 points
    - Part_3 (orange) : 8 points

- Average idle waiting duration of the parts: This metric shows the average of waiting durations of parts without having robot assignments from their arrival till their assignment. We should underline that this period will be calculated by subtraction of arrival time from the last successful assignment time of the robot that picked up the part. As the auctions are repeatedly made, this time will last longer.

- Weighted idle waiting duration of the parts: Like in the weighted average of collected parts, waiting times are multiplied by coefficients due to the type of the part, in order to have a better observation on waiting times of parts with different priority levels.

- Average idle waiting duration of the robots: Before assigned to a task, robots could wait an arrival of parts, or they can change their attributed task after auctions. All this times before the last successful auction resulted with a collection of a part is added to idle times of robots. This metric gives the average of total waiting times of robots before a successful assignment.

- Average empty trip (pickup trip) duration of the robots: Robots' average travel times which is calculated as the period between the assignment time to their assigned part and their successful pickup action time.

- Average delivery trip duration of the parts: The average of delivery times, calculated from the difference of times from pickup action until delivery action.

- Average service duration of the parts: It is the sum of the average pickup trip duration of the robots plus the average delivery trip duration of the robots and plus the average idle waiting duration of the parts.

- Weighted average of service duration of the parts: Travel durations of the parts are multiplied by coefficients to understand the importance or priorities of parts. In the following experiments, service durations of the parts will be weighted as follows:

- 0 to 500 steps services : 0 points
- 500 to 1000 steps services : -1 points
- 1000 to 1500 steps services : -2 points
- 1500 to 2000 steps services : -4 points
- over 2000 steps services : -8 points

- Average number of interruptions: The number of interruptions shows the obstructions, like total path blockings or physical collisions to robots by moving obstacles. Non-controlled moving obstacles are displacing randomly and they try to interrupt the multi-agent robots. Robots have to evade from these moving obstacles.

We should note that the number of arrived jobs in each of 20 run in a case is different, but each of the 20 runs have the same opportunity of arrival of parts due to common seed usage in random arrival of parts. We have to underline that a new part is only capable to arrive if the node is free and for each path generation, the founded path could be different in simulations.

9.2. Case One: First Come First Served Rule Based Experiments

This first case of experiments carries on three tests about FCFS rule based multi-agent organization. Under the FCFS rule, three different experiments will be arranged.

In FCFS rule, the first free agent will be assigned the oldest part in the system to minimize the waiting costs of the incoming parts.

9.2.1. Results of FCFS with Random Pickup Based Control

The first case to be tested depends on FCFS rule, which means the older part is assigned to a robot before the other parts and the experiment contains no other priorities, neither for parts, nor for robots and the team member robots are assigned to random arrival nodes. No auctioning for incoming parts has been used for this first test of case one. To provide randomness in assignments, numbers chosen over a normal distribution numbers and each quartile depends on a free robot's number to assign to a pickup node.

The parameter choices are presented below:

Table 9.2. Parameters for Experiment 9.2.1.

	Status
Auction Properties	No auction: assignments are made in random order
	Time effect on auctions: waiting time of the object's are taken into account
	Path cost effect in off: Among free robots, the job is assigned to a random one
Re-auctioning	Re-auctioning mode is off

The results obtained from the experiments are presented in following table:

Table 9.3. Results of the FCFS model with random assignment.

Moving Obstacles Inside	1	2	3	4	5
Average of Collected Parts after 20 Runs	28.25	26.65	24.25	23.85	22.95
Standard Deviation of Collected Parts after 20 Runs	2.53	1.31	0.90	1.53	1.24
Weighted Average Points of Collected Parts	105.94 pts.	99.94 pts.	90.94 pts.	89.44 pts.	86.06 pts.
Average of Idle Waiting Times of Robots	175.50	176.12	175.86	176.42	174.89
Average of Idle Waiting Times of Parts	278.92	287.25	295.30	305.80	312.14
Average of Empty Trip Times of Robots (after assignment)	461.04	482.10	492.65	517.21	526.39
Average of Delivery Trip Times of Robots (after pickup)	910.39	943.18	957.23	994.31	1015.24
Average Service Duration for Parts	1650.35	1712.53	1745.18	1817.32	1853.77
Weighted Average of Idle Waiting Parts (multiplied by time factor)	6188.81 pts.	6421.99 pts.	6544.43 pts.	6814.95 pts.	6951.64 pts.
Average Number of Interruptions of by Non-controlled Obstacles	1.15	2.25	4.05	6.45	12.10

As the only setting is to collect the older parts first, without considering path costs in assignments, starting from the oldest part in the system, robot-part matching is made randomly. This first experiment of the first case's results will be meaningful when they will be compared with other experiments.

9.2.2. Results of FCFS Based Control with Path Cost Calculation

In the second case test runs, the experiment still contains FCFS rule and no priorities of parts, but the robots are going into bid for assignments to their nearest node. The team member robots are calculating their path costs and bet for auction. Then the nearest robot to an arrival node is assigned there and it loads up the part in that node and carries it to the first available delivery node. Parameters in the simulation are set like this:

Table 9.4. Parameters for Experiment 9.2.2.

	Status
Auction Properties	Time effect on auctions: waiting time of the object's are taken into account
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet
Re-auctioning	Re-auctioning mode is off

In the following table, we have the results about this experiment:

Table 9.5. Results of the FCFS model with path cost calculation.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	33.25	32.55	31.75	29.85	28.95
Standard Deviation of Collected Parts after 20 Runs	1.51	1.31	1.12	1.17	1.49
Weighted Average of Collected Parts	124.69 pts.	122.06 pts.	119.06 pts.	111.94 pts.	108.56 pts.
Average of Idle Waiting Times of Robots	135.15	140.14	145.42	147.79	156.38
Average of Idle Waiting Times of Parts	203.68	211.59	221.94	248.37	258.43
Average of Empty Trip Times of Robots (after assignment)	357.85	366.20	382.59	405.69	425.99
Average of Delivery Trip Times of Robots (after pickup)	791.86	801.51	805.72	825.48	826.75
Average Service Duration for Parts	1353.39	1379.30	1410.25	1479.54	1511.17
Weighted Average of Idle Waiting Parts (multiplied by time factor)	5075.21 pts.	5172.38 pts.	5288.44 pts.	5548.28 pts.	5666.89 pts.
Average Number of Interruptions of by Non-controlled Obstacles	1.25	3.15	5.10	5.35	11.20

In this test, the older incoming part is assigned before the other parts. Figure 9.1 shows that except the traffic jams, the older part is likely to the delivered before a newer part. The path costs of free robots are taken into account while assignments are made. This

experiment shows better results than the previous. The reason could be that a task is assigned to a less costly, that means the nearest, free robot and this economizes durations of trips.

Note that, there is a decreasing trend in performance with the increasing number of non-controlled dynamic obstacles. But, this increases only the empty trip times and delivery times, but the idle waiting times are in a steady trend. When a robot is free, the auction and the assignment are quick. The idle waiting times are due to the reason that, there are no free robots.

A result to be underlined is that the average of collected parts number is increased but the weighted average of collected parts has not increased in same scale. On the other hand, weighted waiting times is decreased. This result could prove that the system is behaving selective and it concentrates on older parts without taken into account their type.

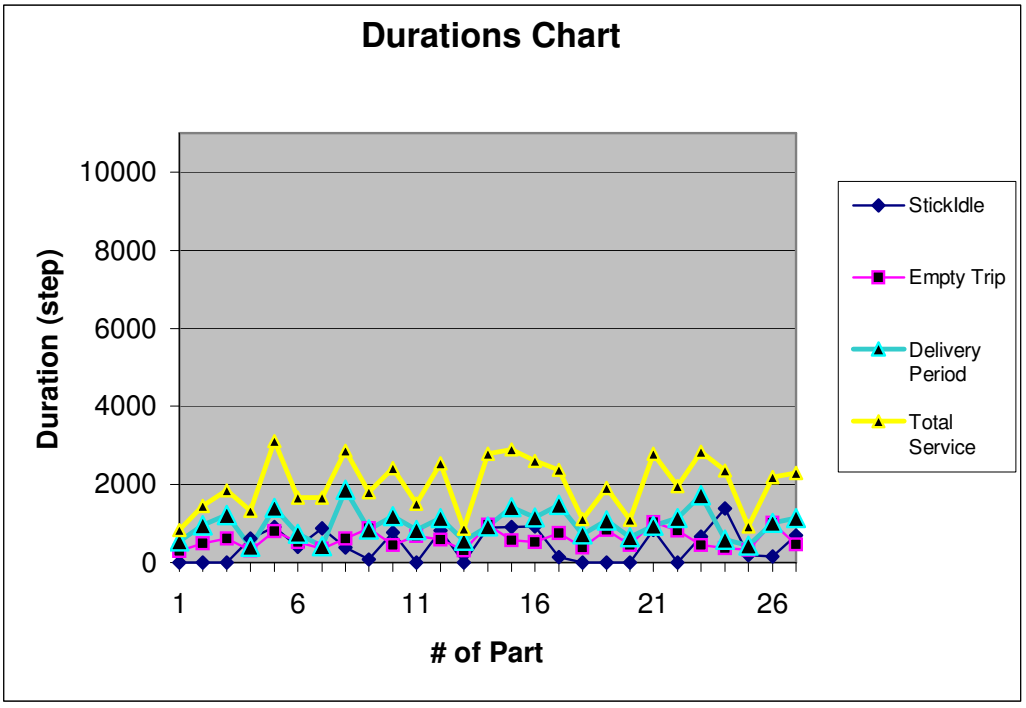

Figure 9.1. Average service durations chart from a sample of FCFS rule experiment.

The Figure 9.1 shows a sample FCFS rule experiment. Waiting time distributions show no extreme durations in total service times. As the FCFS rule aims to serve to the older parts in the system in prior, waiting time distributions are relatively steady.

9.2.3. Results of FCFS Algorithm Control with Path Cost Calculation and Re-auctioning

In addition to previous test, re-auctioning mode is activated. So, when a robot is freed from its task, there are opportunities for making a better assignment, which may execute the task with a lower cost. But the re-initialization of system assignments takes time. The set of parameters are given in the following Table 9.5.

Table 9.6. Parameters for Experiment 9.2.3.

	Status
Auction Properties	Time effect on auctions: waiting time of the object's are taken into account
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet
Re-auctioning	Re-auctioning mode is on

With these parameters, we can see the difference created by re-auctioning among the agents, compared to the previous experiment. The obtained results are presented below:

Table 9.7. Results of the FCFS model with re-auctioning.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	34.10	33.60	32.55	30.85	29.95
Standard Deviation of Collected Parts after 20 Runs	1.32	0.88	2.05	1.23	1.63
Weighted Average of Collected Parts	127.88 pts.	126.00 pts.	122.06 pts.	115.69 pts.	112.31 pts.
Average of Idle Waiting Times of Robots	141.43	144.58	148.02	148.96	150.23
Average of Idle Waiting Times of Parts	194.76	214.76	227.76	250.72	266.72
Average of Empty Trip Times of Robots (after assignment)	317.47	337.47	345.22	381.67	388.41
Average of Delivery Trip Times of Robots (after pickup)	783.70	799.70	808.87	816.87	820.50
Average Service Duration for Parts	1295.93	1350.61	1381.85	1449.26	1475.63
Weighted Average of Idle Waiting Parts (multiplied by time factor)	4859.74 pts.	5064.79 pts.	5181.94 pts.	5434.73 pts.	5533.61 pts.
Average Number of Interruptions of by Non-controlled Obstacles	2.05	2.30	5.50	9.15	11.05

Results in Table 9.7 shows that, with re-auctioning module available, the multi-agent team showed a better performance in average of collected parts, but the most important, in the weighted average waiting times. The weighted average of waiting times is tending to decrease, which is aimed by this experiment. On the other hand, that behavior created a slight decrease in weighted average of the previous case with no re-auctioning. Even the weighted average of parts is higher than the first random assignment case due to the higher average of collected parts.

The re-auctioning may cause a good influence in aimed FCFS results: In changing statuses, after re-auctioning, some task assignments can be won by another robots rather than the first assigned robots or the first assigned ones could be trapped in time. But the good influence is can be only in favor of the aimed results.

9.3. Case Two: Type Priority Based Tests

The second case is concerned about priority of incoming parts due to their identification. The aimed performance is different than FCFS model. Parts have are weight points in 1st, 2nd, 3rd and 4th grades for 1, 2, 4 and 8 points respectively. Here, highest priority parts are to be collected in prior. Collecting prior parts can score higher weighted points.

9.3.1. Results of Part Type Priority Algorithm Based Control

For the first experiment of this case, a random free robot – part assignment will be made. There is no time effect in collecting parts, so a higher priority part in the system has to be served always before the other parts. In this experiment, free robots are assigned to the prior parts first, but the choice of the assigned robot is random.

The parameter set is presented below:

Table 9.8. Used parameters for Experiment 9.3.1.

	Status
Auction Properties	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring
	Path cost effect in off: Among free robots, the job is assigned to a random one.
Re-auctioning	Re-auctioning mode is off

At the end of the runs of the experiment, we have obtained these results given in the Table 9.9:

Table 9.9. Results of Type Prioritized model with random assignments.

Moving Obstacles Inside	1	2	3	4	5
Average of Collected Parts after 20 Runs	26.60	25.45	23.35	22.95	21.45
Standard Deviation of Collected Parts after 20 Runs	0.79	0.81	1.25	1.43	1.48
Weighted Average of Collected Parts	154.28 pts.	147.61 pts.	135.43 pts.	133.11 pts.	124.41 pts.
Average of Idle Waiting Times of Robots	171.23	173.26	169.86	167.56	173.54
Average of Idle Waiting Times of Parts	279.76	297.64	316.54	324.42	334.98
Average of Empty Trip Times of Robots (after assignment)	479.18	498.30	533.73	530.29	550.49
Average of Delivery Trip Times of Robots (after pickup)	931.18	945.73	973.24	995.47	1005.36
Average Service Duration for Parts	1690.12	1741.67	1823.51	1850.18	1890.83
Weighted Average of Idle Waiting Parts (multiplied by type coefficient)	9802.70 pts.	10101.69 pts.	10576.36 pts.	10731.04 pts.	10966.81 pts.
Average Number of Interruption by Non-controlled Dynamic Obstacles	0.55	2.10	6.30	10.55	11.10

In this case, the random assignments showed no improvement in average of collected parts compared to the random FCFS case. As the averages of these two cases are similar, this could be considered normal as robots are assigned without taken into account the path costs and for the multi-agent team, there is no significance of path costs. Selectiveness for service of parts showed some results and the weighted scores are higher than the first case under the FCFS rule (experiment on 9.2.1) The weighted score points are higher because parts are pointed in 1^{st}, 2^{nd}, 3^{rd} and 4^{th} grades for 1, 2, 4 and 8 points respectively. Here, highest priority parts are to be collecting first to score higher points.

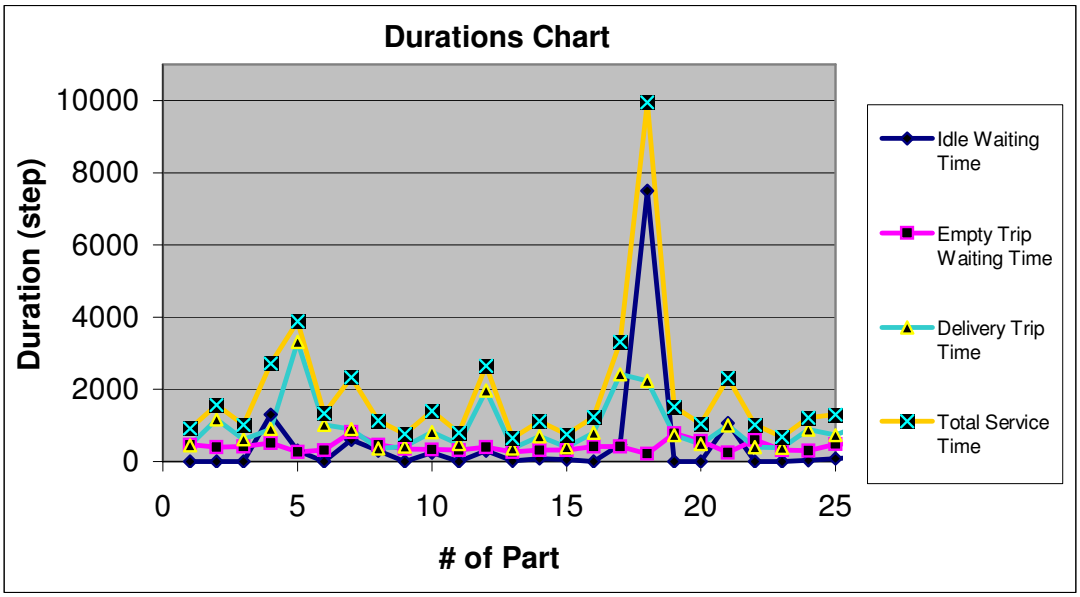

Figure 9.2. Average service durations chart from a sample of TP rule experiment.

The Figure 9.2 is taken from one of the sample runs in TP rule experiment. Total service duration for parts is the sum of the all other three series, empty trip of the robots, delivery trip of the robots and idle waiting time. The excessive high durations in delivery trips show the trapped robots by moving obstacles. The robot has tried and success to escape after a while. This picture gives a hint about the behavior of multi-agent team under this parameters set: The maxima point of the graph is due to a higher idle time of a part. As the priorities of parts are taken into account, the part #18 could be waited for service for a long time and serviced after the turn of prior parts. So the agents turned to less prior part only after terminating with the prior parts.

9.3.2. Results of Part Type Priority Algorithm Based Control with Path Cost Calculation

In this third experiment of the type priority case, the re-auctioning mode is on and robots can exchange their attributed tasks while moving to pickup. On Table 9.10, there are the parameter settings for the second experiment of the type priority case:

Table 9.10. Parameters for Experiment 9.3.2.

	Status
Auction Properties	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet.
Re-auctioning	Re-auctioning mode is off

With these parameters, the obtained results from Experiment 9.3.2 are as follows in Table 9.1:

Table 9.11. Results of Type Prioritized model with path cost calculation.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	35.65	34.90	33.55	33.05	32.20
Standard Deviation of Collected Parts after 20 Runs	1.10	0.90	1.55	1.58	1.76
Weighted Average of Collected Parts	206.77 pts.	202.42 pts.	194.59 pts.	191.69 pts.	186.76 pts.
Average of Idle Waiting Times of Robots	176.44	173.05	175.32	178.42	180.51
Average of Idle Waiting Times of Parts	288.22	318.83	332.83	344.87	354.87
Average of Empty Trip Times of Robots (after assignment)	193.61	214.23	242.21	256.21	259.86
Average of Delivery Trip Times of Robots (after pickup)	730.38	745.39	767.23	770.71	775.71
Average Service Duration for Parts	1212.21	1278.45	1342.27	1371.79	1390.44
Weighted Average of Idle Waiting Parts (multiplied by type coefficient)	7030.82 pts.	7415.01 pts.	7785.17 pts.	7956.38 pts.	8064.55 pts.
Average Number of Interruptions of by Non-controlled Obstacles	0.50	1.90	6.55	9.40	10.65

This experiment carries on priority of parts like the previous one, but differently than the previous experiment, a market-driven method is applied: When a new part comes into

the workshop, an auction is opened for assignment. By calculating their path costs, the agents bet for auction and the agent with the lowest cost agent wins the auction. This algorithm assigns a more suitable agent for reaching the pickup node. Results on Table 9.11 shows an improved average of collected parts compared to the random assignment case in 9.3.1. Even a higher weighted average of parts is obtained. Average service durations, idle times of the parts before assignments and weighted average on waiting times are decreased. These data show the good influence of market-driven auctions for the performance of the system.

9.3.3. Results of Part Type Priority Algorithm Based Control with Path Cost Calculation and Re-auctioning

This experiment is the suite of the previous one, where each part type has a different priority to be collected. In addition to market-driven algorithm, which makes auctions for task assignments, re-auctioning for assigned tasks is also applied. For this experiment, simulation parameters are set as follows in Table 9.12:

Table 9.12. Parameters for Experiment 9.3.3.

	Status
Auction Properties	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring
	Path cost effect is on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet.
Re-auctioning	Re-auctioning mode is on

After setting parameters, we have run the simulation and obtained the results presented in Table 9.13:

Table 9.13. Results of Type Prioritized model with re-auctioning.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	35.85	35.10	34.50	33.80	32.30
Standard Deviation of Collected Parts after 20 Runs	0.95	1.11	1.40	1.33	1.54
Weighted Average of Collected Parts	207.93 pts.	203.58 pts.	200.10 pts.	196.04 pts.	187.34 pts.
Average of Idle Waiting Times of Robots	180.52	182.68	184.29	183.90	191.24
Average of Idle Waiting Times of Parts	286.81	298.48	310.76	326.33	350.73
Average of Empty Trip Times of Robots (after assignment)	189.30	201.51	221.53	233.42	261.29
Average of Delivery Trip Times of Robots (after pickup)	722.43	739.57	748.36	756.05	768.05
Average Service Duration For Parts	1203.94	1239.56	1280.65	1315.8	1379.07
Weighted Average of Idle Waiting Parts (multiplied by type coefficient)	6982.85 pts.	7189.45 pts.	7427.77 pts.	7631.64 pts.	7998.61 pts.
Average Number of Interruptions of by Non-controlled Obstacles	0.65	2.85	4.75	6.90	12.05

The experiment on the section 9.3.3 carries on priority of parts like the Experiment 9.3.1 and the Experiment 9.3.2. There is a market-driven algorithm applied like in 9.3.2. In addition to the previous experiment, a re-auctioning system is implemented. Agents renew the auctions when a status changes occurs, i.e. addition a robot to the free list and also, ancient best assignments could be worsen due trapped robots while executing a task.

Here, the results on the Table 9.13 give the information that the average of collected parts and weighted average of parts are even improved than the two previous type priority experiments. Renewing market-driven assignment decisions over time scored even higher than the Experiment 9.3.2. In case of joining a free robot in the system, this free robot could be closer to a prior part and changing the assignment could make a match with higher profit. Instead of insisting to apply the first decision, making new assignment could be profitable. The overall success trend is decreasing with the increasing number of moving obstacles.

9.4. Case Three: FCFS & Type Priority Combined Experiments

In the third case to observe, the FCFS priority and part type priority is combined. The most prior part is the part, which has the highest priority point from the multiplication of its FCFS point and its type point.

9.4.1. Results of Part Type Priority & First Come First Served Algorithm Combined Control with Random Matching

In the first experiment of the third case, assignment of robots will be randomly made. Parameters are set as follows on the Table 9.14:

Table 9.14. Parameters for Experiment 9.4.1.

	Status
Auction Properties	Time effect on auctions: waiting time of the object's are taken into account
	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring.
	Path cost effect in off: Among free robots, the job is assigned to a random one.
Re-auctioning	Re-auctioning mode is off

Note that the priorities of parts are calculated from a multiplication of a coefficient for their arrival time and another coefficient from their type. Table 9.15 shows the results of the experiment after the runs:

Table 9.15. Results of FCFS and type priority model with random assignments.

Moving Obstacles Inside	1	2	3	4	5
Average of Collected Parts after 20 Runs	29.30	27.25	26.70	25.00	24.35
Standard Deviation of Collected Parts after 20 Runs	1.79	2.17	1.69	1.78	1.95
Weighted Average of Collected Parts	140.64 pts.	130.80 pts.	128.16 pts.	120.00 pts.	116.88 pts.
Average of Idle Waiting Times of Robots	172.35	186.42	185.18	179.52	183.99
Average of Idle Waiting Times of Parts	259.43	294.11	298.20	321.28	331.15
Average of Empty Trip Times of Robots (after assignment)	422.62	489.85	496.41	515.47	526.92
Average of Delivery Trip Times of Robots (after pickup)	823.19	886.01	893.46	922.53	929.09
Average Service Duration For Parts	1505.24	1669.97	1688.07	1759.28	1787.16
Weighted Average of Idle Waiting Parts (multiplied by time factor)	7225.15 pts.	8015.86 pts.	8102.74 pts.	8444.54 pts.	8578.37 pts.
Average Number of Interruptions by Non-controlled Obstacles	1.00	2.00	4.00	7.00	12.00

This first experiment of the third case shows interesting results: The average of collected number of parts is not higher than other cases, also the weighted averages of collected parts. Neither weighted averages on waiting times is the lowest score, compared to other cases. But, we can say that, equilibrium has been established between the priorities: The weighted average of collected parts is better than the FCFS random assignment case (9.2.1) and the weighted average of waiting time of the parts is lower than the type priority case's random assignment experiment (9.3.1).

9.4.2. Results of Part Type Priority & First Come First Served Algorithm Combined Control with Path Cost Calculation

In this second experiment of the third case with combined priorities, now the path cost calculations are introduced, differently than the previous experiment in 9.4.1.

Table 9.16. Parameters for Experiment 9.4.2.

	Status
Auction Properties	Time effect on auctions: waiting time of the object's are taken into account
	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet.
Re-auctioning	Re-auctioning mode is off

Table 9.17. Results of FCFS and type priority combined with path cost consideration.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	31.55	29.10	27.20	26.30	25.75
Standard Deviation of Collected Parts after 20 Runs	1.73	1.07	0.92	0.82	0.97
Weighted Average of Collected Parts	151.44 pts.	139.68 pts.	130.56 pts.	126.24 pts.	123.60 pts.
Average of Idle Waiting Times of Robots	171.43	167.25	169.39	176.09	180.20
Average of Idle Waiting Times of Parts	226.82	261.49	294.78	297.81	292.64
Average of Empty Trip Times of Robots (after assignment)	389.78	427.56	488.17	498.22	496.30
Average of Delivery Trip Times of Robots (after pickup)	812.84	826.27	886.59	898.62	930.28
Average Service Duration for Parts	1429.44	1515.32	1666.54	1694.65	1719.22
Weighted Average of Idle Waiting Parts (multiplied by time factor)	6861.31 pts.	7273.54 pts.	7999.39 pts.	8134.32 pts.	8252.26 pts.
Average Number of Interruptions of by Non-controlled Obstacles	2.25	2.35	4.10	8.15	11.20

With the given parameters in Table 9.16, we have obtained the results presented in Table 9.17. Here, we can see the results of the market-driven application on the combined case. The average of collected parts is increased than the random assignment Experiment

9.4.1 of this case. Also, the weighted average of parts is higher than the previous experiment in 9.4.1. Weighted average of idle waiting times of parts has been decreased, when compared to the previous random assignment case. Considered these points, cost calculation for bets in assignments provided better results than random assignments for this case too.

9.4.3. Results of Part Type Priority & First Come First Served Algorithm Combined Control with Path Cost Calculation and Re-auctioning

In the final test of the system, the path cost calculation is done before the auctions and also re-auctioning mode is on. Sticks are pointed in 1st, 2nd, 3rd and 4th grades for 1, 2, 4 and 8 points respectively. Also, from the FCFS rule, older parts gain priority points to be assigned to a robot by the time passed. In below Table 9.18 shows the parameters:

Table 9.18. Parameters for Experiment 9.4.3.

	Status
Auction Properties	Time effect on auctions: waiting time of the object's are taken into account
	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet.
Re-auctioning	Re-auctioning mode is on

The results of the third experiment of the third case are in the following Table 9.19:

Table 9.19. Results of the FCFS and type priority combination with re-auctioning.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	25.15	24.70	22.50	20.60	19.55
Standard Deviation of Collected Parts after 20 Runs	1.93	1.71	2.10	1.97	2.65
Weighted Average of Collected Parts	120.72 pts.	118.56 pts.	108.00 pts.	98.88 pts.	93.84 pts.
Average of Idle Waiting Times of Robots	195.35	201.22	215.37	245.84	268.96
Average of Idle Waiting Times of Parts	320.42	327.34	333.92	354.51	417.12
Average of Empty Trip Times of Robots (after assignment)	512.56	523.02	549.29	561.92	595.76
Average of Delivery Trip Times of Robots (after pickup)	921.65	923.49	985.63	1020.09	1084.66
Average Service Duration for Parts	1754.63	1773.85	1868.84	1936.52	2097.54
Weighted Average of Idle Waiting Parts (multiplied by time factor)	8422.22 pts.	8514.48 pts.	8970.43 pts.	9295.30 pts.	10068.19 pts.
Average Number of Interruptions of by Non-Controlled Obstacles	3.85	7.55	9.60	12.25	14.20

Table 9.19 presents the results of this experiment. After observing the results presented, we can say that this case has some interesting tricks: Although the path cost is gives better results than random matching in average collected parts, the re-auctioning has a slightly negative effect on results, when compared to experience on 9.4.2. While waiting service, parts are gaining time priority. So, if a newcomer prior part with a high grade is selected first, while re-auction occurs, a lower priority part could become prior with time effect. In consequence, robots change their decisions and they turn to another part. This could trigger oscillations of robots between nodes. Here, a higher average of idle waiting times of robots is visible. As the idle waiting time signs the difference of time between the end of a delivery action and the last successful assignment of the robot, higher average of idle time for robots could be due to these oscillations. As there are four nodes to be served and three robots to serve as resources, robots' idle times are not high as much. If the re-auction time is not selected properly, robots could travel between nodes and they could

reach no parts. Here, a gap between collection times occurs. So, for successful results, re-auctioning must be in accordance with type priority and time effect.

9.4.4. Results of Part Type Priority & First Come First Served Algorithm Combined Control with Path Cost Calculation and Re-auctioning – Bis

The last experiment is repeated with re-adjusted coefficients in parameters. The same experiment showed poor results with default coefficients, where the priorities are balanced and the robots are oscillated sometimes between nodes without completing action. Now, the time coefficient is set as relatively small against the type priority coefficient. Table 9.20 shows the parameters of this repeated experiment:

Table 9.20. Parameters for Experiment 9.4.4.

	Status
Auction Properties	Time effect on auctions: waiting time of the parts is taken into account. *(Note that the time coefficient is reduced in this experiment)*
	Type priority on auctions: some parts are prior to other parts. These parts have a greater point while scoring. *(Note that the coefficient of TP is increased for this repeated experiment).*
	Path cost effect in on: a feasible path's distance is calculated and communicated between groups. Among free robots, the job is assigned to the lowest cost agent (closest to part) who wins the bet.
Re-auctioning	Re-auctioning mode is on, with modified coefficients

The results of the repeated third experiment of the FCFS & type priority combined case are in the following Table 9.21:

Table 9.21. Results of the FCFS and type priority combination with re-auctioning.

Moving Obstacles Inside	**1**	**2**	**3**	**4**	**5**
Average of Collected Parts after 20 Runs	32.05	31.00	29.85	28.30	27.75
Standard Deviation of Collected Parts after 20 Runs	1.70	1.28	1.32	1.75	1.44
Weighted Average of Collected Parts	153.84	148.80	143.28	135.84	133.20
Average of Idle Waiting Times of Robots	175.50	176.12	175.86	176.42	179.89
Average of Idle Waiting Times of Parts	221.82	271.49	305.83	301.80	314.64
Average of Empty Trip Times of Robots (after assignment)	383.78	377.56	396.18	428.94	449.52
Average of Delivery Trip Times of Robots (after pickup)	813.34	826.58	844.34	895.03	928.29
Average Service Duration for Parts	1418.94	1475.63	1546.35	1625.77	1692.45
Weighted Average of Idle Waiting Parts (multiplied by time factor)	6810.91 pts.	7083.02 pts.	7422.48 pts.	7803.70 pts.	8123.76 pts.
Average Number of Interruptions of by Non-controlled Obstacles	2.25	3.10	3.95	7.85	10.10

Table 9.21 presents the results of this repeated experiment. This time, to arrange the conflict of priorities, the time priority coefficient is set relatively smaller than the type priority. Then, the obtained results in average collected parts, in weighted average of collected parts and in service times are improved compared to the NPF experience. Yet the re-auctioning gave improved results with its market-driven algorithm. Note that the weighted averages in collected parts is not higher than the type priority experiments and weighted average of service times is not lower than FCFS experiment but we can say that the multi-agent team has a success to find an equilibrated service policy, as expected from a flexible system.

9.5. The Analysis of Variance (ANOVA) Test to Check the Statistical Results

To be sure of the experiments' results, a statistical test will be applied: The Analysis of Variance (ANOVA) tests the "significance" between groups of data.

The ANOVA test is used where more than two groups of data will be checked for significance. As we have 3 + 3 + 4 = 10 groups of data, instead of checking the data pairs with the *t*-test, we could use ANOVA test. ANOVA test puts all the data into one number *(F)* and gives us one *P* value for the null hypothesis.

If we treat the collection of the 10 group means as data and find the standard deviation of those means and it is "significantly" larger than the above, we have evidence that the null hypothesis is not correct and instead location has an effect. This is to say that if some (or several) group's average leaf-size is "unusually" large or small, it is unlikely to be just by "chance".

Here are the results of ANOVA tests for 10 groups of experiments with 5 samples, which one of item represent the average of collected parts in an experiment:

Table 9.22. Averages of collected parts in order of experiments.

	1 Obstacle	2 Obstacles	3 Obstacles	4 Obstacles	5 Obstacles
FCFS - Random	28.25	26.65	24.25	23.85	22.95
FCFS - Market	33.25	32.55	31.75	29.85	28.95
FCFS - Re-auct.	34.10	33.60	32.55	30.85	29.95
Type Priority - Random	26.60	25.45	23.35	22.95	21.45
Type Priority - NPF	35.65	34.90	33.55	33.05	32.20
Type Priority - Re-auct.	35.85	35.10	34.50	33.80	32.30
FCFS & TP - Random	29.30	27.25	26.70	25.00	24.35
FCFS & TP - NPF	31.55	29.10	27.20	26.30	25.75
FCFS & TP - Re-auct.	25.15	24.70	22.50	20.60	19.55
FCFS & TP - Re-auct. Bis	32.05	31.00	29.85	28.30	27.75

Table 9.23. Confidence intervals for collected parts.

CASES	ALGORITHM	SAMPLE MEAN	%95 CONFIDENCE INTERVAL	
			Lower Limit	Upper Limit
Case One	FCFS - Random	25.20	23.43	26.95
	FCFS - Market	31.30	29.51	33.03
	FCFS - Re-auct.	32.20	30.45	33.97
Case Two	Type Priority - Random	24.00	22.20	25.72
	Type Priority - NPF	33.90	32.11	35.63
	Type Priority - Re-auct.	34.30	32.55	36.07
Case Three	FCFS & TP - Random	26.50	24.76	28.28
	FCFS & TP - NPF	28.00	26.22	29.74
	FCFS & TP - Re-auct.	22.50	20.74	24.26
	FCFS & TP - Re-auct. Bis	29.80	28.03	31.55

From Table 9.22 and Table 9.23, the obtained F value is 22.30 and p value is less than 0.0001. For rejecting null hypothesis, the p value is 0.05. Then, we can state that the difference between the experiment result groups is significant.

Then the same test is done for average service times, with the given averages in Table 9.23:

Table 9.24. Averages of service durations in order of experiments.

	1 Obstacle	2 Obstacles	3 Obstacles	4 Obstacles	5 Obstacles
FCFS - Random	1650.35	1712.53	1745.18	1817.32	1853.77
FCFS - Market	1353.39	1379.30	1410.25	1479.54	1511.17
FCFS - Re-auct.	1295.93	1350.61	1381.85	1449.26	1475.63
Type Priority - Random	1690.12	1741.67	1823.51	1850.18	1890.83
Type Priority - NPF	1212.21	1278.45	1342.27	1371.79	1390.44
Type Priority - Re-auct.	1203.94	1239.56	1280.65	1315.80	1379.07
FCFS & TP - Random	1505.24	1669.97	1688.07	1759.28	1787.16
FCFS & TP - NPF	1429.44	1515.32	1666.54	1694.65	1719.22
FCFS & TP - Re-auct.	1754.63	1773.85	1868.84	1936.52	2097.54
FCFS & TP - Re-auct. Bis	1418.94	1475.63	1546.35	1625.77	1692.45

Table 9.25. Confidence intervals for service durations.

CASES	ALGORITHM	SAMPLE MEAN	%95 CONFIDENCE INTERVAL	
			Lower Limit	Upper Limit
Case One	FCFS - Random	1755	1668	1843
	FCFS - Market	1426	1339	1514
	FCFS - Re-auct.	1390	1303	1477
Case Two	Type Priority - Random	1799	1711	1886
	Type Priority - NPF	1319	1231	1406
	Type Priority - Re-auct.	1283	1196	1371
Case Three	FCFS & TP - Random	1782	1687	1855
	FCFS & TP - NPF	1605	1517	1692
	FCFS & TP - Re-auct.	1886	1798	1973
	FCFS & TP - Re-auct. Bis	1552	1465	1639

From Table 9.23 and Table 9.24, the obtained F value is 23.91 and p value is less than 0.0001. For rejecting null hypothesis, the p value is 0.05. Then, we can state that the difference between the average service durations of experiments results is significant.

9.6. Discussion

The above experiments are only some samples among huge property combinations of this simulation. The above obtained experiment results could give us an idea about the behavior of the system tendencies.

Generally, tests are done in three main domains: Algorithms using first come first served rule, algorithms using priorities of part types and a combination of the first two domains. In all three domains, first tests are run without any assignment properties. In the second tour of tests, a market-driven algorithm making auctions is introduced. In the auctions, the agent with the lower cost for the execution of the task gives the highest bet, and than it is assigned to the most profitable task. Finally, in the third tour of tests, a re-auctioning is made for increasing the effectiveness of our system in status changes like, when a robot is pinned, trapped or become free.

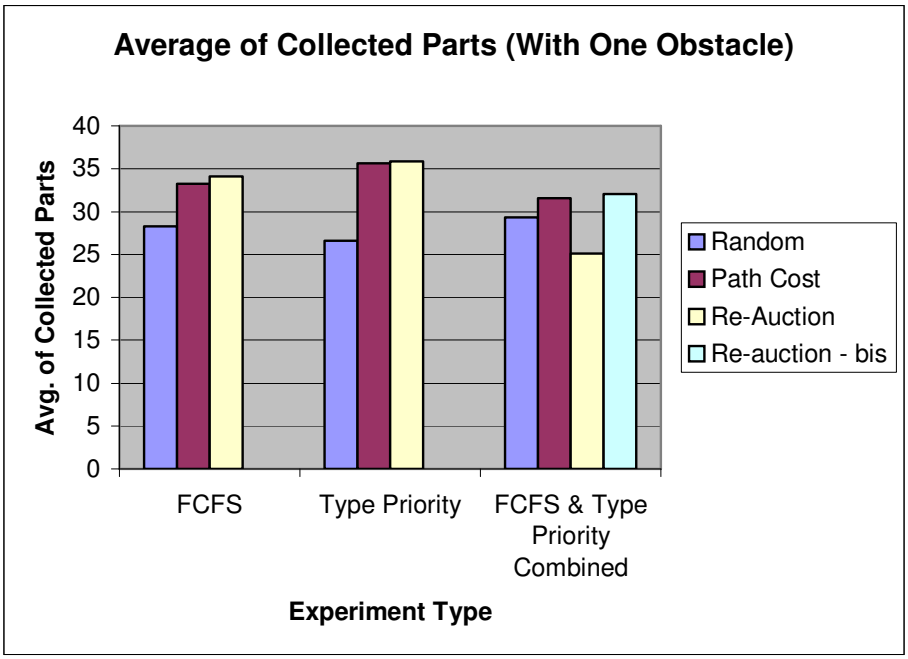

Figure 9.3. Average of collected parts in order of experiments with one obstacle.

In Figure 9.3, we can observe the average collected parts on experiments with one dynamic obstacle. Market driven methods auctioning over path costs show a good performance over random assignments. Yet, re-auctioning on market driven methods improved the results of previous market driven method, except the last category of experiments where robots are influenced by consecutive priority changes due to the balancing priorities. Re-auction-bis experiment is only repeated for FCFS and Type Priority combined experiment with different coefficients. Then the re-auction-bis experiment increased the performance on average of collected parts after an adjustment on coefficients. So, we are assured with re-auctioning influence on performances.

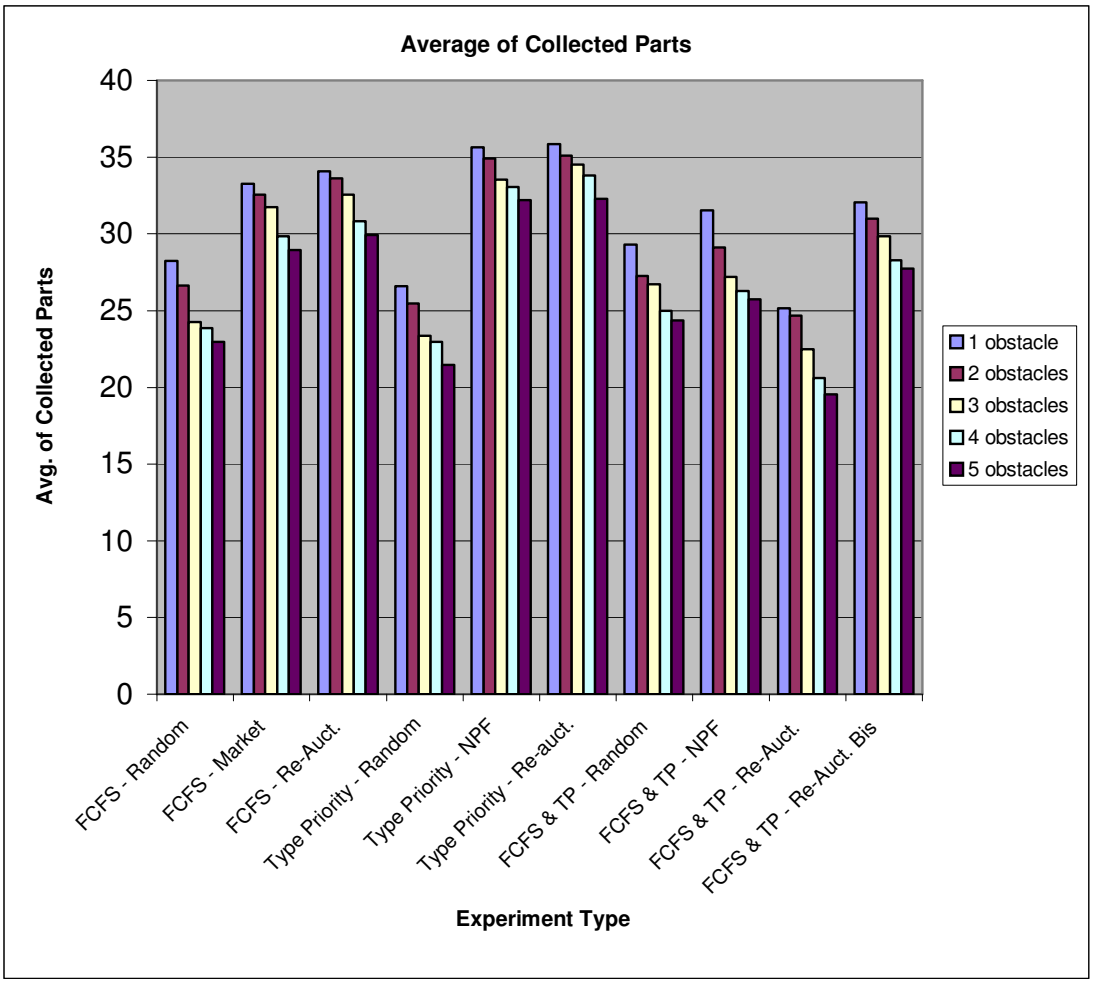

Figure 9.4. Average of collected parts in order of all experiments.

At the Figure 9.4, the average of collected parts in order of all the ten experiments is visible. Looking at the first experiment of results in each of the three cases respectively, we can predict that the random assignments' results are giving lower outputs or profits and we cannot say that they give the optimum results. The organization among the agents is not giving to system the highest profit. A more organized assignment with introducing opportunity costs and future profits could give progressed results.

The second sub-group of experiments results of all three cases results gives the performance of system with calculated path costs. This is an important introduction of a more intelligent control, which is an implementation of market-driven methods. The results of market-driven cases compared with corresponding random assignment application cases

(i.e. simulation in section 9.2.1 and in section 9.3.1) showed better results than these last ones. We can consider that the choosing the nearest available robot at the auction time reduces the pickup path to be traveled. In consequence, collected part number is increased over the same period due to the savings in empty trip times. So, the market-driven methods' profit is proved.

The third group of tests carried on re-auctioning effects: In a market-driven method, an auction for a part could be closed once an assignment has been made or with an entry of a new resource to the system (i.e. a new available robot), the auctions could be revised. Here, we renew these auctions with an expectation to find a lower cost match for the parts. For first two experiments, re-auctioning improve the results, as seen in sections 9.4.1 and 9.4.2, compared to the experiments on sections 9.3.1 and 9.3.2, only in the experiment defined on section 9.4.3 has a slightly decreased result. The reason for this decrease might be the oscillations occurred between re-auctions. Some high pickup times increase the deviation of results. This proves that the agents change their matching without picking a part. So, for successful results, re-auctioning must be in accordance with type priority and time effect. One solution could be to increase the weight of one of the priorities; the waiting time effects or type priority effects.

A general observation is that when multi-agents share information and coordinate between themselves, they can obtain better results than random assignments. Higher coordination degree of multi-agents could provide improved results but while introducing new rules, we have to take into account the after effects, like in case 9.4.3 where introduction of re-auctioning rule has decreased the overall performance in averages. The new rule must be in accordance with previous rules and configurations. After we have arranged the coefficient collision in parameters, results kept their increasing trend.

As the confidence intervals of results of Market-driven methods and Re-auctioning methods have an intersection, we can say that the improvement of re-auctioning is less significant between them, but these two methods have confidence intervals, which are strictly superior in collected parts and strictly inferior in service durations, so the improvement compared to the random methods are very distinctive.

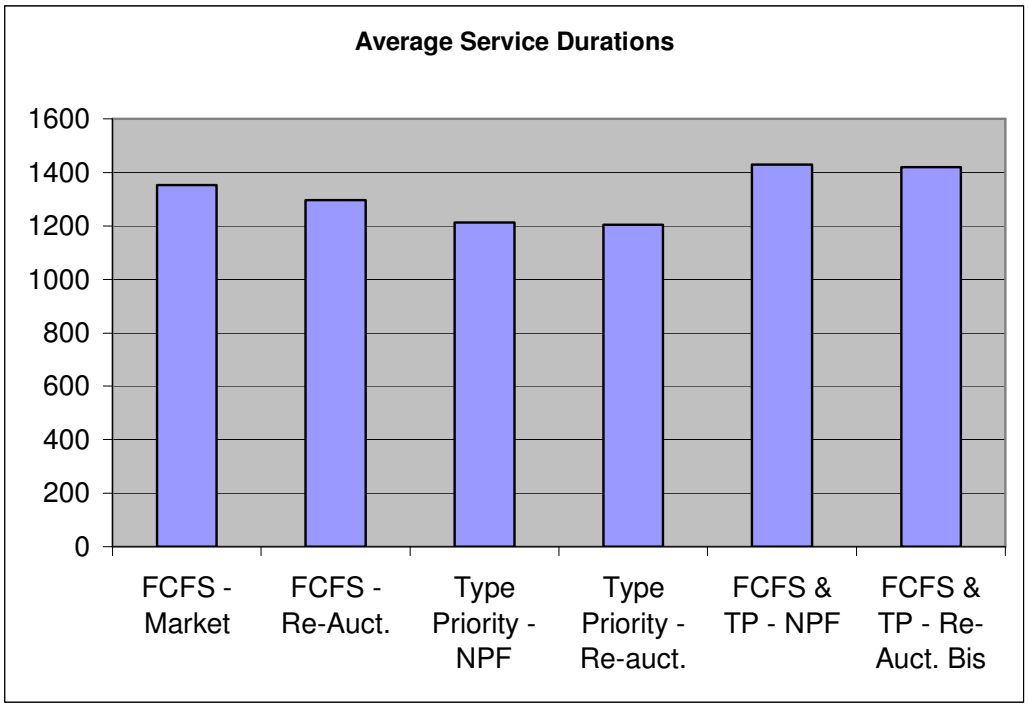

Figure 9.5. Evaluation of average service durations.

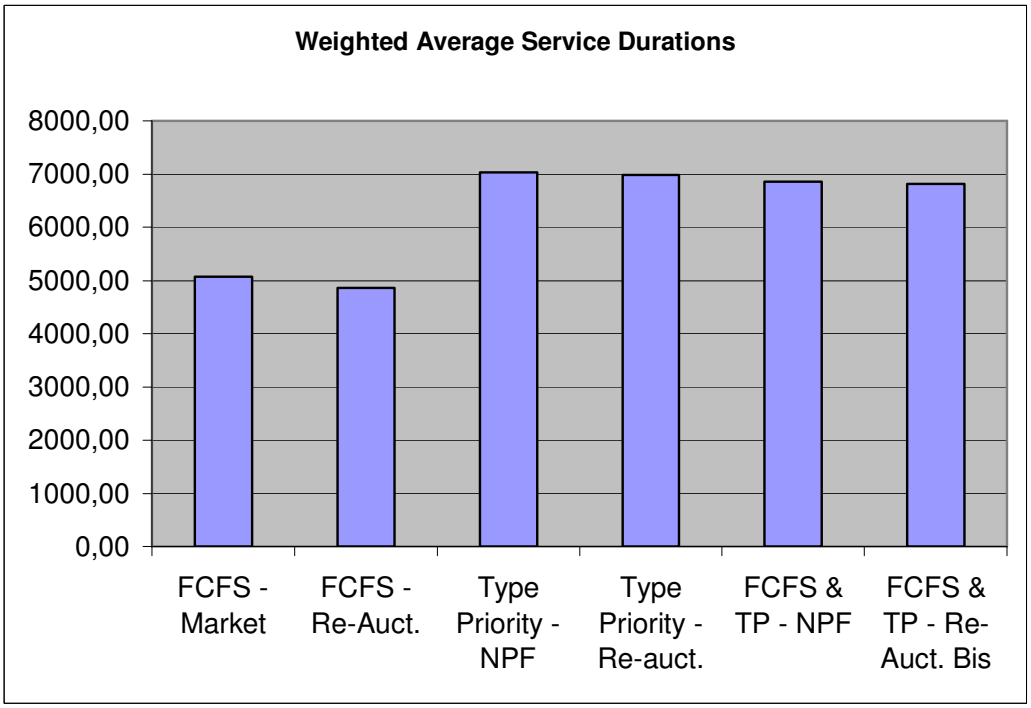

Figure 9.6. Evaluation of weighted average of service durations.

Figure 9.5 and Figure 9.6 show us an important feature of the market-driven algorithm, which is case sensitivity. The system has to maximize its profit by selecting the most prior part if they are available in time. Here, in the Figure 9.5, average of service durations in FCFS module is a bit longer than type priority case. This couldn't be a surprise: The system tries to maximize its profit with respect to weighted priorities. Than in FCFS experiments, the system tries to collect the older parts first, even though they are more far in terms of distance. This causes the average of service durations in FCFS cases to be longer than TP cases, but in the weighted average, FCFS results are lower. This is what we expect from the system: Under the FCFS rule, the aim is to serve older parts first as and the weighted average metric was for higher points for higher service times, as explained in section 9.1, the weighted average becomes lower than TP and FCFS & TP combined cases. So, under the FCFS rule, the system showed a better performance about waiting times of parts.

On the other hand, while we observe the averages of collected part and weighted averages of collected parts in the following figures, we can conclude about the success of the responsiveness of our system:

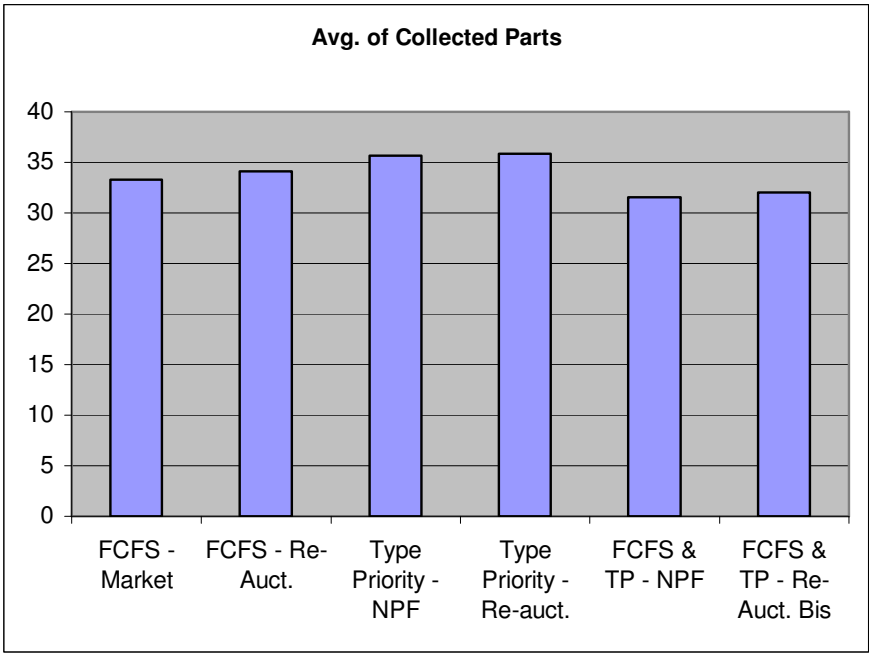

Figure 9.7. Evaluation of average of collected parts.

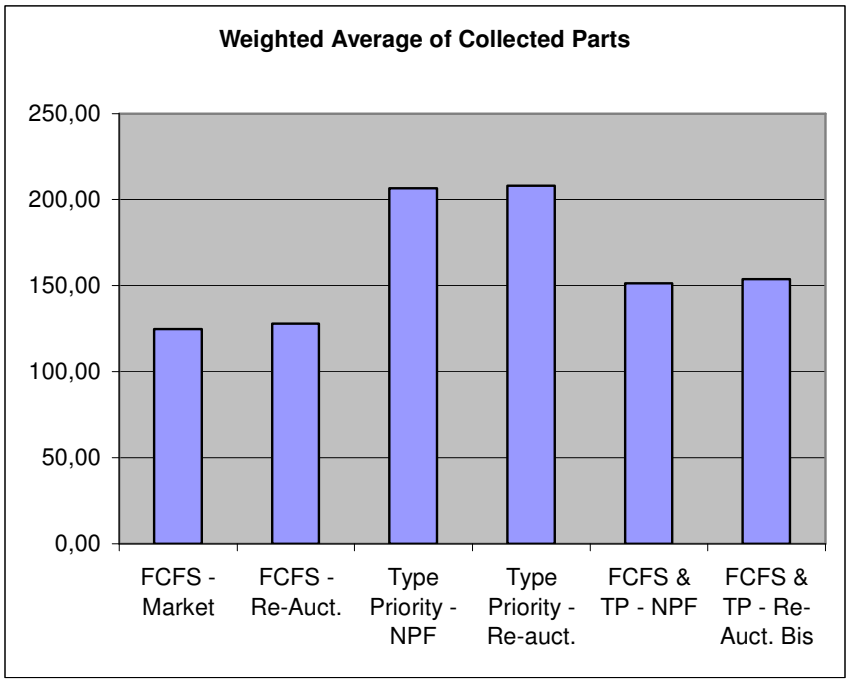

Figure 9.8. Evaluation of weighted average of collected parts.

In Figure 9.8, in the averages of collected parts, there are no big differences in number for FCFS, TP and FCFS & TP combined cases. However, when we observe the weighted averages on collected parts in Figure 9.7, our systems selectiveness is visible. In the TP case, obtained points are clearly higher. As the TP favorites the collection of parts with respect to their priorities, in TP rule, our market-driven algorithm using multi-agent team concentrated to collect the parts with higher priorities.

When we observe the FCFS & TP combined rule cases, the performance in average of collected parts, the performance is looking poor, but in weighted averages of collected parts, the system's performance on FCFS & TP combined experiment is better than FCFS case but not than the TP case. As the rules are combined, robots' behavior is a combination of two rules. Same comment can be made for weighted averages of service times for FCFS & TP case: Weighted averages of service durations for parts are higher than the FCFS case but lower than TP case on the other hand. That means, for the combined rule case, again, our system responded to the demands of a flexible production rule.

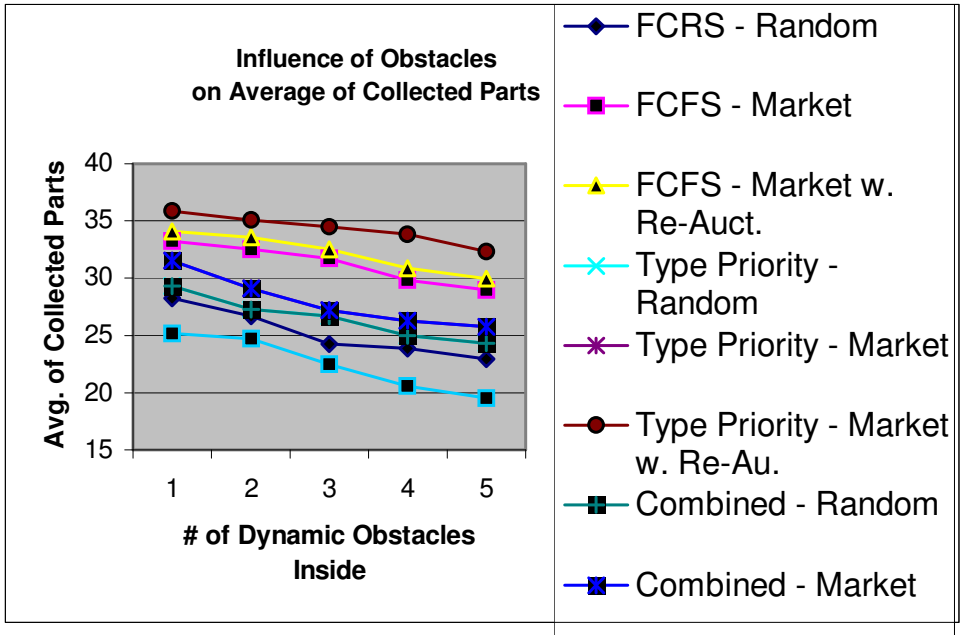

Figure 9.9. Influence of dynamic obstacles over the averages of collected parts.

The Figure 9.9 shows the influence of non-controlled dynamic obstacles in the job-shop. The increasing numbers of dynamic obstacles have a negative effect on performance on all experiments. Averages of collected parts are decreasing constantly. Even though the multi-agent robot team is designed to working in environments with human presence and they can evade from collisions, moving objects are interrupting the process and there is a decreasing trend. This trend can exponentially decrease after a saturation limit in the workspace. Even our system can work together with humans; it is advised that, if a free place could be reserved to robots, the performance of robots could be better. The same situation is visible in Figure 9.10:

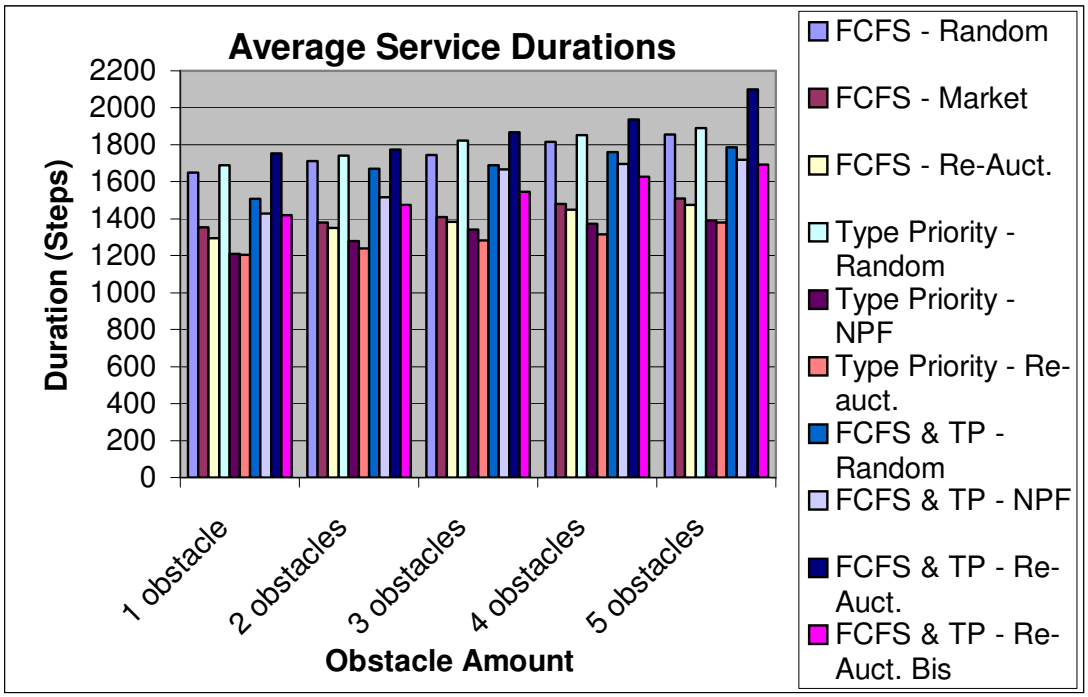

Figure 9.10. Average service durations of all experiments.

Here we can see in the Figure 9.10 the averages of service durations. These averages have a tendency with to become higher with the increasing amount of non-controlled moving obstacles inside. These results are compatible with the previous trends in Figure 9.5. We have to underline that the highest durations are obtained in the FCFS & TP combined case with re-auctioning option's first experiment. These first results in that experiment gave relatively poor solutions due to the oscillations in trips of robots. Possibility to meet a non-controlled robot is higher when the robots travel between nodes repeatedly and maneuvering from one side to another. The "FCFS & TP Combined-Bis" experiment shows a better performance after adjusting the coefficients.

10. CONCLUSIONS AND FUTURE WORK

In this thesis, the aim is to propose a fully autonomous multi-agent system to be used in industrial facilities. Such a system can replace AGV systems, which are already being used in flexible manufacturing organizations. AGV systems are already implemented in many industrial production areas and they would be considered as an ancestor of AMRs before implementation of advanced AI techniques. Most AGVs could be upgraded to AMR systems. AMR systems could benefit of the AGV infrastructure and they could provide a more robust and promising services in the production plants, outdoor production areas too, as well as indoor areas.

With this goal, first we reviewed multi-agent robots taxonomy and alternative techniques for controlling robots. We have stated the problem domains like localization, path planning, obstacle avoidance, etc.

Then, we have made a comparison between AGVs and AMRs understand better the differences capabilities. We have cited the advantages of using autonomous systems, as well as the benefiting from AGV capabilities and experiences.

Considering the multi-agent taxonomy and the problem domains, we suggested a multi-agent robot system, which can work in highly populated production areas. For this, we have tried to benefit from centralized systems control precision and distributive systems robustness. We have concentrated on localization, multi-robot path planning and obstacle avoidance issues.

Then, by observing the workshop models, we generated a simulation basis for a manufacturing job-shop environment. We have simulated our design to test the proposed system's properties and components with key technical and organizational problem domains: like localization, object-avoidance, sensing of indoor world of manufacturing shop in technical issues and disposal of the parts due to requirements and priorities in organizational parts. Finally we have found a cooperation model for better performance. The aim of analyzing our model is to define the impacts of cooperation over the mentioned

robot performance domains at the production control of a general manufacturing job-shop. The impacts of the intelligent cooperation are expressed in terms of performance results of our model. The simulation control is first built as a behavior-based control model with random task assignments, and then we tried to improve its performance by adding more AI techniques like market-driven methods.

To test the task allocation performance of the proposed system, experiments are done for three main cases with three sub-cases for each one. The results in general show that the implemented path planning algorithms are functioning effectively. Also, a good performing obstacle avoidance ability is implemented. Moreover, agents can coordinate among themselves in order to execute the tasks. Multi-agent team's output performance is increased (subject function) in expression of terms of time and successful part component deliveries (cost coefficients and key parameters) in our job-shop simulation when compared to the random assignment cases, which are representing no cooperation. Even by introducing another option in market-driven algorithm, experiment results are improved in terms of output and timesaving. The proposed system for implementation carries the advantages of autonomous robots and could profit from a centralized control issues as well as distributed control. The multi-agent team is responded to the changes in the needs of a flexible manufacturing, by their self-organization.

By this simulation basis for experiments, one can find all the key cost coefficients and key parameters of the models, in order to compare them in the functioning cost, team performance and robustness of two alternative systems. Considering a real-time manufacturing job-shop, these key factors are important to define the optimal overall system, which will provide better output performance for given conditions. The simulation basis is capable to assist to find these parameters. In this test-bed, the research could focus either on robotic features and robot control, as workshop features and production control.

We have observed that the RRT based multi-agent path planning and market-driven algorithms can be successfully implemented and work well on such multi-agent teams. Alternatives are many but these implemented solutions prove that an upgrade to fully autonomous systems is possible.

We can say that the path planning and obstacle avoidance algorithms are functioning well. The sampling based RRT-Connect algorithm ruled path planning is a good performing multi-agent path planner. Even if not gives the optimum path, it is fast and after path smoothing, it gives better results. In a highly dynamic environment, a renewable and fast path planning could be more important than finding the optimal path. Also, no one can guarantee that the optimal path would not interrupted by obstacles in time. So, as a faster and robust path-planning algorithm, RRT basis proved its performance with our multi-agent autonomous robot system together with good performing obstacle avoidance.

Also, market-driven distributed multi-agent algorithm is highly useful for this domain. Results show that the intelligent control of multi-agents can assure optimized performances, and with respect to a flexible manufacturing system's needs. Yet with re-auctioning module option, our algorithm's performance could increase but it has to be underline that re-auctioning option is parameter dependant and suitable parameters must be set for good performance.

Finally, we can state that, this work showed us the possibility of replacing AGVs with AMRs or upgrade AGVs to fully autonomous vehicles, which can profit from autonomous robots features and AI. Observations and results obtained by this system could be used to upgrade AGV systems to autonomous vehicles that will have independency and intelligent control, so global AGV system problems like blockages in the pathways are evaded. For such an improvement, the principles of control technique alternatives are many. As manufacturing plant or other industrial places have capabilities to provide the necessary information for the autonomous system, the control will be easier and the performance of the team would influenced in a good manner, but a robust system must also have its independent abilities to work alone. By getting together these two features, a multi-agent robotic system could do its best. Such capabilities are profitable for production industry, where a great flexibility opportunity is offered by autonomous systems.

As a future work, AMR systems could be implemented on outdoor production areas such as shipyards or construction sites, where localization and path planning would be more difficult due to higher outer effects of noise than the indoor areas, and job shop layout would be subject to changes during production time, differently than manufacturing

plants, where node and machine layout are almost fixed. These working environments require more uninformed search orientations: On-line path planning would be considered until a sufficient map of the environment is obtained. Or, to achieve higher level localization techniques, more vision domain usage for the robots with processing the images, filtering noises and developing a control strategy which can principally based on recognizing objects, persons, obstacles, and other robots would be in the mentioned research domain. Also, the next step for research for multi-agents in industry can be the usage of specialized agents (heterogeneous) vs. generalist agents (homogenous) and implementation of an intelligent control for multiple load transporting capacities. Another expansion could be collaboration for executing tasks where teamwork of two or more robots is required; i.e. carrying or handling packages that one robot can not handle, or towing heavier parts together where one robot's power is insufficient. As the alternative conditions are many, there are also many possibilities to try and this kind of test-beds would help us finding our way. An inter-disciplinary work and collaboration of different domains is required for future researches as well as on this one.

Still, a huge research is waiting us in this domain.

REFERENCES

1. Arkin, R. C., "*Behavior Based Robotics*", The MIT Press, 1998.

2. Bonasso, R. P., Kortenkamp, D., and Murphy, R., "*Artificial Intelligence and Mobile Robots*", AAAI Press / The MIT Press, 1998.

3. Murphy, R., "*Introduction to AI Robotics*", The MIT Press, 2000.

4. Arkin, R.C. and Balch, T., "*Cooperative Multi-agent Robotic Systems*", 1995.

5. Bischoff, R. and Graefe, V., "*Vision-guided Intelligent Robots for Automating Manufacturing, Materials Handling and Services*", Paper for WESIC'98 Workshop on European Scientific and Industrial Collaboration on Promoting Advanced Technologies in Manufacturing, Girona, June 1998.

6. Michel, O., "*Webots: A Powerful Simulator for the Khepera Robot*", Cyberbotics Editions, 2003.

7. Martinoli, A., Easton, K. and Agassounon W., "Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation", *International Journal of Robotics Review*, June 2003.

8. Martinoli, A. and Easton, K., "*Modeling Swarm Robotic Systems*", ISER 2002 Publication Paper, 2002.

9. Floreano, D., Godjevac, J., Martinoli, A., Mondada F. and Nicoud, J.-D., "*Design, Control and Applications Mobile Robots*", Swiss Federal Institute of Technology in Lausanne, 2002.

10. Bischoff, R., "HERMES - A Humanoid Mobile Manipulator for Service Tasks", *International Conference on Field and Service Robotics*, Canberra, pp. 508-515, 1997.

11. Martinoli, A., "Swarm Intelligence in Autonomous Collective Robotics: From Tools to The Analysis And Synthesis of Distributed Control Strategies", PhD Thesis, Submitted to Swiss Institute of Technology in Lausanne, Lausanne, 1999.

12. Hayes, A. T., "*How Many Robots? Group Size and Efficiency in Collective Search Tasks*", Collective Robotics Group Paper, California Institute of Technology, California, 2002.

13. Agassounon, W., "*Modeling Artificial, Mobile Swarm Systems*", PhD Thesis, Submitted to California Institute of Technology, Pasadena, California, 2003.

14. Ijspeert, A. J., Martinoli, A., Billard, A. and Gambardella, L. M., "*Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment*", Omega, Vol. 28, pp. 409-415, 2000.

15. Beetz, M., Arbuckle, T., Belker, T., Cremers, A. B., Schulz, D., Bennewitz, M., Burgard, W., Hähnel, D., Fox, D. and Grosskreutz, H., "Integrated, Plan-Based Control of Autonomous Robots in Human Environments", *IEEE,* September / October 2001.

16. Agah, A. and Bekey, G. A., "A Genetic Algorithm-Based Controller for Decentralized Multi-Agent Robotic Systems"*, Proceedings of The 1996 IEEE International Conference on Evolutionary Computation (ICEC'96),* Nagoya, Japan, 1996.

17. Dias, M. B. and Stentz, A. (T.), "*A Market Approach to Multirobot Coordination*", CMU-RI -TR-01-26, The Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, August 2001.

18. Dias, M. B. and Stentz, A. (T.), "*A Free Market Architecture for Coordinating Multiple Robots*", CMU-RI -TR-99-42, The Robotics Institute Carnegie Mellon University, Pittsburgh, Pennsylvania, December 1999.

19. Michel, O., "Cyberbotics Ltd - WebotsTM: Professional Mobile Robot Simulation", pp. 39-42, *International Journal of Advanced Robotic Systems,* Volume 1 Number 1 (2004), ISSN 1729-8806.

20. McKerrow, P.J., "*Introduction to Robotics*", Addison-Wesley Publishing Company Inc, 1991 – Reprinted in 1998.

21. Böhme, H.-J., Wilhelm, T., Groß, H.-M. and Hempel, T., "An approach to multi-modal human–machine interaction for intelligent service robots", *Robotics and Autonomous Systems, Volume 44*.

22. Russell, S. and Norvig, P., "*Artificial Intelligence: A Modern Approach*", Prantice Hall Series in Artificial Intelligence, 2003.

23. Köse, H. and Akın, H. L., "Towards a Robust Cognitive Architecture for Small Autonomous Mobile Robots" *ISCIS XV, The Fifteenth International Symposium on Computer and Information Sciences*, pp. 447-455, Istanbul, Turkey, 11-13 October 2000.

24. Cao, Y. U., Fukunaga, A. S. and Kahng, A. B., "Cooperative Mobile Robotics: Antecedents and Directions", *Proceedings of IEEE / RSJ IROS Conference*, 1995 and Autonomous Robots, no:4, pp. 1-23, 1997.

25. Olivier, M., "*Webots: A Powerful Simulator for the Khepera Robot*", Cyberbotics Ltd., 2003.

26. Rosas, D., "*Multi-Agent Supervision of Generic Robots*", MSc Thesis to Case Western University, May 2002.

27. Everett, H. R., Gilbreath, G. A., Heath-Pastore, T. A. and Laird, R. T. "*Controlling Multiple Security Robots In a Warehouse Environment*", Naval Command Control and Ocean Surveillance Center, San Diego, CA, 1994.

28. Barnes, D.P., Aylett, R. S., Coddington, A. M. and Ghanea-Hercock R. A. "*A hybrid approach to supervising multiple co-operant autonomous mobile robots*", UK Robotics Ltd., Trafford Park, Manchester, M17 1QS, UK., ICAR, 1997.

29. Lucidarme, P., "*An Evolutionary Algorithm for Multi-Robot Unsupervised Learning*", CEC, 2004.

30. Yamashita, A., Sasaki, J., Ota, J. and Arai, T. "*Cooperative Manipulation of Objects by Multiple Mobile Robots with Tools*", B002 final, August 1998.

31. Gordon-Spears, D. and Kiriakidis, K., "Reconfigurable Robot Teams: Modeling and Supervisory Control", *IEEE Transactions on Control Systems Technology*, pp. 763-769, Vol. 12, No. 5, September 2004.

32. Kaplan, K., "*Design and Implementation of Fast Controllers for Mobile Robots*", Master Thesis, Boğaziçi University, January 2003.

33. Doğan, D., "*An Object Oriented Test-bed for Automated Guided Vehicle Systems*" Master Thesis, Boğaziçi University, June 2001.

34. Hatice K., Çetin M., Kemal K. and Akın, H. L., "*All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain*", ISCIS Conference Paper, 2003.

35. Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S., "*Principles of Robot Motion: Theory, Algorithms, and Implementation*", The MIT Press, 2005.

36. Karagoz, C. S., Bozma, H. I., and Koditschek, D. E., "EDAR - mobile robot for parts moving based on a game-theoretic approach", *Electronics Letters*, Volume 38, no. 3, pp. 147-148, 31 January 2002.

37. Karagöz, C. S., Bozma, H. I. and Koditschek, D.E., "Feedback-Based Event-Driven Parts Moving", *IEEE Transactions on Robotics*, Vol. 20, no. 6, pp. 1012-1018, December 2004.

38. Kuffner, J. J. Jr., LaValle, S. M., "RRT-Connect: An Efficient Approach to Single-Query Path Planning", *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

39. Parker, L. E., "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation", *IEEE Transactions on Robotics and Automation*, Vol. 14, No: 2, pp. 220-240, April 1998.

40. Meriçli, Ç. A., "*Developing A Novel Robust Multi-Agent Task Allocation Algorithm for Four-Legged Robot Soccer Domain*", Master Thesis, Boğaziçi University, July 2005.

41. Kolushev, F. A., Bogdanov, A. A., "Multi-Agent Optimal Path Planning for Mobile Robots in Environment with Obstacles", *PSI 99, LNCS 1755*, pp 503-510, 2000.

42. Bilge, Ü., Esenduran, G., Varol, N., Öztürk, Z., Aydın, B., and Alp, A., "Multi-attribute responsive dispatching strategies for automated guided vehicles", *International Journal of Production Economics*, October 2004.

43. Bulak, C., "*Hierarchical Matching Algorithms for Automated Guided Vehicle Systems*", Master Thesis, Boğaziçi University, January 2002.

44. Nolfi, S. and Floreano, D., "*Evolutionary Robotics*", The MIT Press, 2000.

45. Holland, J., "*Designing Mobile Autonomous Robots*", Newnes, Elsevier Inc., 2004.

46. Zelinski, A. and Dowson, I., "Continuous Smooth Path Execution for an Autonomous Guided Vehicle", *IEEE Region 10 Conference. Tencon 92*, Melbourne, Australia, 11-13 November 1992.

47. Ye, R., Hsu, W.-J. and Vee, V.-Y., "Distributed Routing and Simulation of Automated Guided Vehicles", *IEEE Transactions Volume II*, pp. 315-320, 2000.

48. Wu, N. and Zhou, M. C., "AGV Routing For Conflict resolution In AGV Systems", *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, 14-19 September 2003.

49. Wu, N. and Zhou, M. C., "Modeling and Deadlock Control of Automated Guided Vehicle Systems", *IEEE/ASME Transactions on Mechatronics, Vol. 9, No. 1*, March 2004.

50. http://www.msci.memphis.edu/~franklin/AgentProg.html#agent.