RESEARCH WORK REPORT

Resource and Task Allocation in Distributed Environments. A Multi-Agent System Approach

by

Silvia Andrea Suárez Barón

PhD Program in Information Technologies
Department of Electronics, Computer Science and Automatic Control
UNIVERSITY OF GIRONA
Girona, Spain 2004

Resource and Task Allocation In Distributed Environments. A Multi-Agent Systems Approach.

SILVIA ANDREA SUÁREZ BARÓN

Research work directors:
Ph.D. BEATRIZ LÓPEZ IBAÑEZ
Ph.D. JOSEP LLUIS DE LA ROSA

Research report
Department of Electronics, Computer Science and Automatic Control
Group ARLab
Telephone:+34 972 41 84 78
Campus Montilivi, Building P4
University of Girona
Girona – Spain

Campus Montilivi Library
Girona, Spain 2004

Contents

Abstract

This research deals with coordination mechanisms for Multi-Agent Systems (MAS). In particular, it is focused on task and resource allocation mechanisms for distributed environments. The goal is to develop new methods that help to assign tasks to agents that are interacting in a distributed, dynamic environment.

In order to have a practical perspective of the research, we deal with a particular problem concerning rescue operations in a disaster environment caused by an earthquake. The rescue problem has been tackled due to it being a significant social problem, which should be tackled by the scientific community in order to look for new alternatives that allow damage in disaster and emergency situations to be minimized. The simulator provided by Robocup rescue has been used to perform some experiments.

In the Robocup rescue scenario there are agents and tasks to solve. Agents have several communication and environment constraints. Tasks arrive in a dynamic way and agents have to take decisions to accomplish their particular objectives and the group objectives. Taking into account these aspects, effective task and resource allocation is important to help agents achieve their objectives and to maximize the benefits of the system.

There are several mechanisms for tackling task and resource allocation. This research project is aimed at studying these mechanisms and their application to distributed environments. In addition, this research contributes to the development of new scheduling methods which can be applied to several areas such as: health care, tourism, and inter urban transport.

Key words: Multi-agents systems, scheduling methods, task and resource allocation mechanisms.

Chapter 1

1. INTRODUCTION

This research work concerns coordination mechanisms among cooperative agents. In cooperative situation, coordination among agents can be achieved by the use of planning techniques. The methods studied in this research focuses on a particular case of planning methods, namely, scheduling methods. In these kinds of problems, resources should be assigned according to optimisation criteria such as: maximizing the scientific results of space missions, starting rescue tasks as early as possible or maximizing manufacturing processes. From several years, several disciplines like Operational Research and Artificial Intelligence have been looking for efficient solutions to these problems.

The complexity of scheduling problems has lead to a state of the art rich in solutions to particular problems, but no general solution has been provided yet. Nowadays, advances in distributed and ubiquitous computing, networking and sensors, provide a new environment in which new scheduling challenges arise.

The ultimate goal of this research work is to contribute in the advance of the formulation of scheduling problems in this new context from the Artificial Intelligence perspective, and particularly, by using Multi-Agent Systems.

In this chapter we introduce the motivation of our work, then the problem that motivates the research from a practical point of view, and finally the specific objectives are posed.

1.1 Motivation

Advances in distributed, ubiquitous computing, networking and new sensors, have stimulated the creation of a new environment in which the capability of generating, processing and communicating data is so huge that 15 years ago it would have been impossible to imagine. Such a new situation will cause significant effects for military, commercial and scientific applications, especially when computer systems start to interact with physical systems in more diverse and different ways. [40].

In this new context, Multi-agent Systems (MAS) have emerged as a natural paradigm in which it is possible to integrate different parts of the software (called agents) developed by different designers. One of the main research issues in a MAS is coordination, that is, how to avoid extraneous activities among agents while they share resources in open environments, avoid deadlocks, and maintain applicable safety conditions.

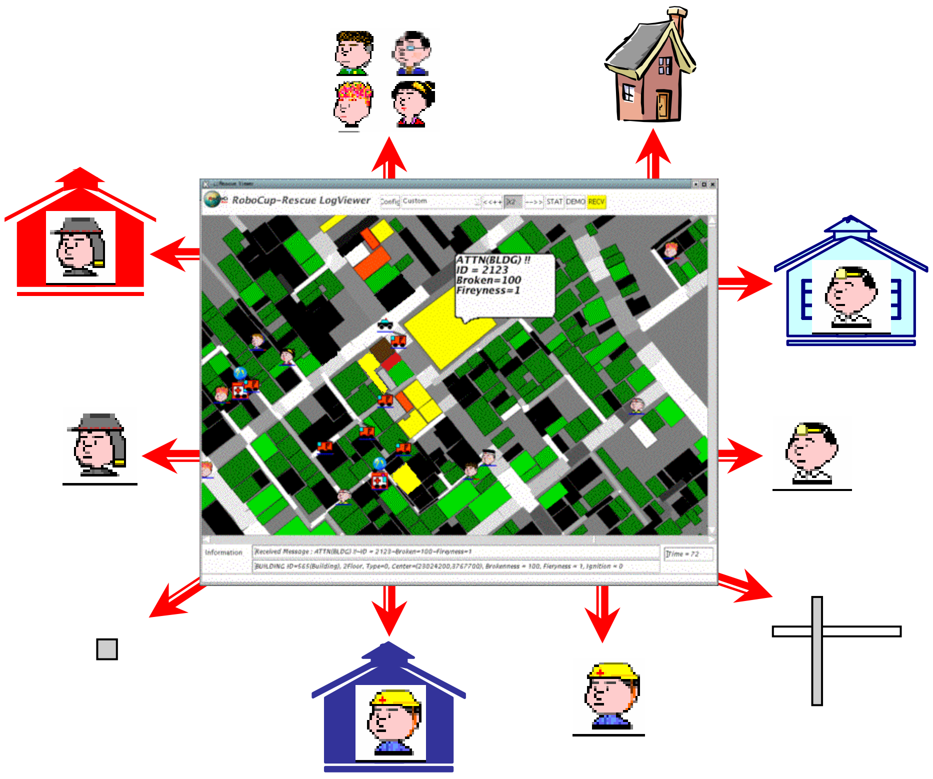
Methods and techniques should be developed to decide on which agent should perform which task in order to achieve overall performance of the system. The research work presented in this document tries to contribute to investigating this.

1.2 Rescue domain

In order to have a practical perspective of the research, we deal with a particular problem concerning rescue operations in a disaster environment caused by an earthquake. It was defined by Kitano in [25] and is the problem currently proposed as a competitive benchmark by RoboCup-Rescue (see Figure 1.1) [46]. RoboCup-Rescue is an international initiative to generate research in AI and Robotic field. Moreover, the resulting approaches can be transferred from simulation to technological reality [24].

The rescue scenario provided by RoboCup-Rescue [46] is a disaster environment caused by an earthquake. In this scenario, there are collapsed buildings, fires, and blocked highways, people in a state of panic looking for safe places, and rescue agents helping victims. Fire brigade agents, police forces and ambulance teams comprise the rescue agents, in addition to central agents made up of the fire, police and ambulance stations. In the RoboCup Rescue Simulation League, there are initially 72 civilian agents, 5 ambulance team agents, 10 brigade agents, 10 police force agents, 1 ambulance station agent, 1 fire station agent and 1 police station agent in the disaster area. All agents have the same objective - to minimize the damage and rescue victims within the earthquake scenario.

All agents have some general properties, namely id, hp, damage, position and buriedness. Id is the identification code of the agent. Hp measures the remaining life of the agents. Damage shows whether or not the agent has been hurt. Position indicates the location where the agent is in the rescue scenario. And finally, buriedness indicates whether the agent can move or is buried under a pile of objects. Other specific properties depend on the type of agent.



ATTN(BLDG) !!
ID = 2123
Broken=100
Fireyness=1

Received Message : ATTN(BLDG) !!~ID = 2123~Broken=100~Fireyness=1

BUILDING ID=565(Building), 2Floor, Type=0, Center=(23024200,3767700), Brokenness = 100, Fieryness = 1, Ignition = 0

Time = 72

Figure 1.1 Rescue scenario. Central buildings, rescue agents, houses, civilians, blocked roads (grey) and fires (from yellow to dark-red).

1.2.1 Types of agents

In the simulation environment, there are two types of agents: rescue agents and victims (civilians) [46, 47]. When an earthquake occurs, some civilians can move to nearby refuges in order to be safe. However, most of them die or are buried and hurt. The survival possibilities of the latter depend on the activity of the rescue agents.

The rescue agents are classified into moving and fixed agents (see Figure 1.2). The moving rescue agents are the fire brigades, police and ambulances. The fixed agents are the agents that cannot move, such as the fire, police and ambulance stations.

The aim of ambulance teams is to rescue civilians. Blocked roads, however, increase the difficulty of these rescue agents. Police agents should come to unblock the roads as soon as possible. In addition, ambulance agents cannot rescue victims in a fire area. Fire extinguishing is carried out by fire brigade agents. Therefore, the co-ordination activities of fixed agents (stations) in guiding the various rescue teams become a key issue.

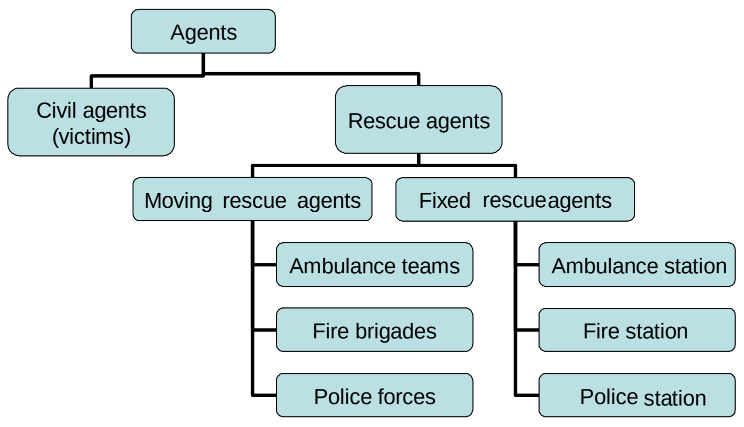


Figure 1.2. Types of agents in the RoboCup-Rescue scenario.

1.2.2 Agents capabilities

Every type of agent has certain communication and action capabilities, as shown in Table 1.1 It can be seen that ambulance teams are the only ones that are able to rescue civilians.

Type	Capabilities
Civilians	Sense, Hear, Move, Say
Ambulance team	Sense, Hear, Move, Say, Tell, Rescue, Load, Unload
Fire brigade	Sense, Hear, Move, Say, Tell, Extinguish
Police force	Sense, Hear, Move, Say, Tell, Clear
Central agents	Hear, Say, Tell

Table 1.1 Agents' capabilities

Furthermore, fire brigade agents have other properties such as water quantity, which shows how much water is in the tank, and stretched length, which shows how long the hose has been pulled [38].

Agents such as ambulances, police and fire stations can only establish communications with agents of their kind and with other stations. That is to say, the police station can receive and send messages to and from police forces, ambulance stations and the fire stations. However, it cannot send messages either to ambulances or fire brigades.

As in real situations, agents have a limited scope. Agent brigades can see visual information within a radius of 10 metres. Visual information is related to collapsing buildings, the victims' location and so on. Central agents cannot perceive visual information. Agents can exchange messages by voice (say and listen) and communication services (tell and hear). In the former, other agents located within a 10-meter radius perceive the message. In the latter case, the message is perceived by the same type of agents located within a 30-meter radius. Central agents can communicate with other central agents using communication devices.

An agent is capable of saying or listening to a maximum of 4 messages in each simulation cycle, within which a decision to perform an action should be taken. This is a hard constraint imposed by the RoboCup Rescue simulator.

This scenario poses several challenges to our research. It is a dynamic environment where situations are unpredictable. On one hand, rescue agents have to accomplish their objectives under strong communication and perception constraints in the surrounding area. On the other hand, a co-ordination mechanism should be developed to deploy rescue agents where they are needed at the right time.

To handle such scenario, two snapshots of the simulation process are proposed. The first one concerns co-ordination among ambulance teams and the second one among fire brigades. As well as satisfactory results being achieved, more complicated scenes will be tackled.

1.2.3 Snapshot 1. Ambulance operation

In this section, the snapshot of the Ambulance operation for the time of 10 seconds is presented.

Let's assume that there are six victims buried in the disaster scenario, identified by 2384, 2388, 2379, 2338, 2356, 2367 and that there are five ambulance teams, identified by 2399, 2400, 2401, 2402, 2403. The current values of the properties of the rescue agents are as follows:

Id	Availability	hp	Damage	Position	Buriedness	goal
2399	busy	10000	0	706	0	2345
2400	free	10000	0	901	0	
2401	busy	10000	0	690	0	2397
2402	free	9000	2	1850	0	
2403	busy	10000	0	76	0	2367

Table 1.2. Properties of the rescue agents

And the victims' properties are:

Id	Hp	Damage	Position	Buriedness	No. victims
2384	9200	17	23	25	1
2388	7900	21	98	60	1
2379	6000	20	1129	35	1
2338	9000	11	2098	15	2
2356	8500	16	2098	30	2
2367	7570	22	1980	16	1

Table 1.3. Properties of the victims

Observe that is possible to find more than one victim in the same position. For example, in the id 329 there could be: victim id 2384, 2388, 2379, 2338, and 2356.

The problem in this snapshot consists on coordinating all ambulance teams and ambulance centres, so that all victims are rescued on time. Taking rescue agents as resources and victims as tasks such problem can be seen as a distributed task allocation problem.

Next, snapshot 2 of the fire fighting operation is described.

1.2.4 Snapshot 2. Fire fighting operation

This is a snapshot of the rescue scenario of the fire fighting operation in the simulation time of 10 seconds:

Let's assume that the fire station knows about five fires in progress in the disaster scenario, identified by 319, 1230, 1900, 2500 and 3829; and that there are ten fire fighting teams, identified by 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423; five of these agents are free and five are busy. The current values of the properties of the rescue agents are as follows:

Id	Availability	hp	Damage	Position(id)	Buriedness	goal
2414	busy	10000	0	706	0	2345
2415	free	10000	0	901	0	
2416	busy	10000	0	690	0	2397
2417	free	9000	2	1850	0	
2418	busy	10000	0	76	0	2367
2419	busy	10000	0	2320	0	2314
2420	free	10000	0	380	0	
2421	free	10000	0	1900	0	
2422	busy	9000	2	2430	0	2312
2423	free	10000	0	15	0	

Table 1.4. Properties of the fire brigade agents

And the tasks (fires) properties are:

Id	ignition	fieryness	brokenness	Coord X	Coord Y
319	1	50	0	22975601	3786999
1230	1	65	25	23105400	3623850
1900	1	10	50	22973900	3519100
2500	1	5	100	22977999	3522750
3829	1	102	0	22996800	3760300

Table 1.5. Properties of the fires

Ignition property tells about the state of a building: 1 means that the building is in fire; 0 is not in fire. The *Brokenness* property value indicates how much the building has collapsed. 0: no damaged, 25: partly damaged, 50: half collapsed, 100: fully collapsed. *Coord X* and *Coord Y* represent the coordinates in the axis X and Y of the fire in the simulator map.

Executing the tasks represents a cost to the agent. The cost is represented by the total time to arrive to a fire position. In addition, it is necessary to take into account the physical state of the agents, which is represented by the *hp*, *damage* and *buriedness* properties, because if they are damaged their performance will be diminished.

In the two snapshots above, the problem to resolve is how to allocate these five tasks to the five free rescue agents, minimizing the associated cost in function of time and improving the total benefit of the rescue operation.

Before continuing it is interesting to know that both snapshot are the instances of the overall problem. That means, that at different time new victims and fires can be discovered and agents should adequate their commitments to the new situation.

1.3 Objectives

This research focuses on the coordination of multi-agent systems. Particularly, the work is concerning about those agents working to achieve some common goal (cooperation). The RoboCup Rescue scenario [46, 39] is used as a simulated disaster environment caused by the earthquake. In this type of environments, the agents should make their interaction in order to accomplish rescue tasks that arrive at any time.

In order to deal with this task allocation problem to agents, the following concrete objectives are proposed:

- To study the state of the art on coordination mechanisms in MAS.
- To study scheduling methods, focusing on traditional task allocation techniques and their extension to distributed approaches.
- To contribute to the development of new techniques for solving the coordination problem in the rescue problem scenario.
- To study the extension of the new methods to other problem domains.

As it will be shown in this document, part of these objectives has been achieved [ref]. However, there are still some objectives to be tackled in future research, as being detailed in the working plan proposal of chapter 4.

1.4 Outline of the document

The document is organized as follows:

First, the state of the art is presented in chapter 2. Coordination mechanisms among agents in multi-agent systems are reviewed. In addition, an overview of some techniques of traditional scheduling is given. Finally, distributed task allocation techniques and approaches are presented.

Then, the exploratory research performed until now concerning about the coordination in cooperative MAS is given in chapter 3. Studies and experimentation with several techniques to coordination among agents are presented.

In chapter 4, the doctoral thesis proposal with the thesis objective and the activities is presented. A general schedule, highlighting the main activities to be performed in the doctoral research, is also proposed.

Finally, conclusions of the preliminary research work are presented in chapter 5

Chapter 2

2 THE STATE OF THE ART

This chapter reviews relevant research that has been done in relation to our work. First we introduce Distributed Artificial Intelligence (DAI) in section 2.1 and then, Multi-Agent Systems (MAS) in section 2.2. Next, scheduling research in general is overviewed in section 2.3, and decentralized task allocation is presented in section 2.4. Finally, work on auctions and combinatorial auctions is reviewed.

2.1Distributed Artificial Intelligence

Distributed Artificial Intelligence (DAI) involves studying a broad range of issues related to the distribution and coordination of knowledge and actions in environments involving multiple entities. According to [12], there are several reasons to choose a distributed AI approach. The first one being the necessity to treat distributed knowledge in applications that are geographically dispersed, such as sensor networks, air-traffic control, or cooperation between robots. In addition, DAI can be used in large complex applications. The second reason is to attempt to extend man-machine cooperation, and the third is that DAI represents a new perspective in knowledge representation and problem solving, because it provides richer and realistic scientific applications.

The DAI field is divided into two research lines: Distributed problem solving (DPS) and research into MAS. In DPS work, the emphasis is on the problem and how to get multiple intelligent entities (programmed computers) to work together to solve it in an efficient manner. In MAS, entities are called agents and have the same properties as in real communities where the agents need to cooperate in order to achieve their goals and the goals of the communities involved [29]. Agents are autonomous and may be homogeneous or heterogeneous. Autonomy is defined as the agent’s ability to make its own decisions about what activities to do, when to do them, what type of information should be communicated and to whom, and how to assimilate the information received. Inversely to studies on DPS, in MAS, agents must reason out the coordination problem among the agents.

Taking these different facets of DAI into account, our work concerns MAS. We focus on the study of mechanisms which the agents in MAS can use to achieve their goals effectively. These mechanisms are in the coordination problem.

2.2Multi-Agent Systems (MAS)

A Multi-Agent system is composed of a number of intelligent agents, who interact with one another through communication [80, 81]. The agents are able to act in an environment; different agents have different "spheres of influence", in the sense that they have control over different parts of the environment.

Next, we review what is commonly accepted as an agent definition, then, how agents perform decision making and finally their mechanisms for cooperation.

2.2.1 Intelligent Agents.

Agents are autonomous, computational entities, which perceive their environment through sensors and act upon their environment through effectors [74]. Figure 2.1 gives a top level view of agent interaction in its environment. The agent receives input from its surrounding environment and based on this input takes the decision to perform some actions. Next, the output actions produce changes in the environment and the cycle is started.

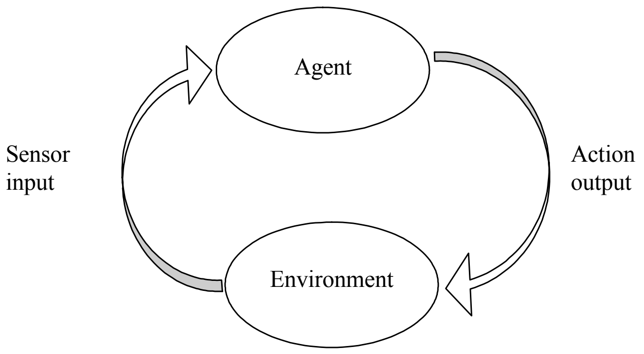


Figure 2.1. An agent in its environment. The agent takes input from the environment and generates actions that modify this environment. This is a cyclic behaviour. From [74]

According to [74] an intelligent agent must meet the following three requirements:

- *Reactivity*: Intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;
- *Pro-activeness*: Intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives;
- *Social ability*: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

In addition, intelligent agents are not independent of one another. In most cases they interact with other agents in order to reach their design objectives. This grouping of agents constitutes a multi-agent system.

2.2.2 Agent reasoning

Taking into account the tasks that an agent is able to perform, agents reason about the process required in order to achieve their goals.

According to [23] the goals may be triggered by the agent itself or by external events, for instance, receiving a message from another agent or from the user. One agent may have several goals at the same time, however, they should be compatible with each other.

Once an agent knows a goal a plan must be generated in order to achieve the goal. This is the planning phase. The planning phase consists in determining a partial order of events. The agent can use a variety of methods to generate a plan. The plan can be generated from a plan library, or the agents can create fully distributed plans. The plans must be compatible with the currently scheduled events, and this compatibility is performed in the scheduling phase.

The scheduling phase consists in determining an optimal plan. This plan must be the best in relation to minimizing the cost of the agents executing the tasks. The scheduled plan must take into account any resource constraints and other already scheduled events of the agents interacting. The scheduled plan may take into account the dynamic character of open environments and if it is necessary for a rescheduling phase to be designed.

Once the plan is scheduled, the agent is in charge of carrying it out. The execution implies calling to tasks and generating interrupts according to the scheduled events. Before the agent executes the tasks, its preconditions must be verified. Then the world state of the agent is updated according to the executed actions. In addition, the appropriate measures must be taken in case the scheduled plan fails.

2.2.3 Agent interaction: coordination mechanisms

According to [74], agent coordination is a property of a system of agents performing some activities in a shared environment. The degree of coordination is the extent to which they avoid extraneous activity by reducing resource contention, avoiding livelock and deadlock, and maintaining applicable safety conditions.

The coordination mechanism is essential if the activities that agents engage in can interact in any way [81]. For instance in the real world two agents in a company may wish to use the same resource at the same time, as for example a printer. One of them must yield their turn to the other in order to use the printer and accomplish their goals. On the other hand, dependent relationships exist in which one agent depends on the other agents in order to accomplish their objectives. In addition, in environments where the agents are not self-interested there are collaborative relations between agents. For instance, in the Internet world the recommender agents provide information to the other agents to give better recommendations to the users [37].

Figure 2.2, shows the different forms in which agents can coordinate their behaviour and activities. Two main kinds of coordination problem can be considered taking into account the interest and goals of the agents: cooperation and competition. In distributed systems where the agents pursue the same goal they need to cooperate. In order to cooperate, they must make a plan which specifies the actions that the agents must develop. However, in systems where the agents are self-interested, they must pursue their own goals and compete with each other. In these systems the agents generate their own objectives and decide whether to adopt goals from the others. The latter action depends on whether these goals are in line with the agent's personal motivation [74].

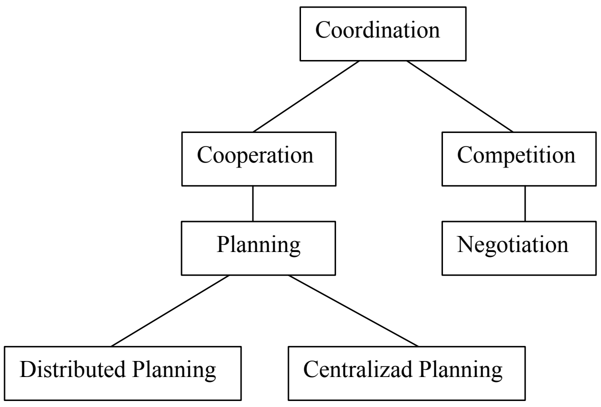


Figure 2.2. A taxonomy of some of the different ways in which agents can coordinate their behaviour and activities. From [74]

Our work is concerned with cooperative environments and in the next section we give details on planning coordinate mechanisms.

2.2.3.1 Planning

Often, getting agents to work together well to solve problems requires collective efforts. In this context there are several agents interacting, and each one is an expert in some areas. For instance, one might be an expert on the strength of structural materials, another on the space requirements for the different types of rooms, another on electrical wiring, and so on. In order to build the house they need to cooperate. In this example, tasks may be delegated from one agent to another; partial results have to be exchanged between agents; information about the state of the problem-solving process has to be given to other agents and unforeseen conflicts have to be resolved.

The process of problem-solving in this overall view of a MAS consists in the following steps:

- *Task decomposition*: the tasks are decomposed into sub-tasks which are less complex.

- *Task allocation*: in this step the tasks are assigned to the agents involved (problem solvers).

- *Problem solving*: this is the phase in which each agent develops the assigned task.

- *Solution integration*: the subproblem's results are integrated to produce an overall solution.

Task decomposition is related to planning while task allocation is related to scheduling, a special case in planning. In this section, we deal with planning in MAS, often called multi-agent planning. The next section focuses on scheduling.

Multi-agent planning (MAP) is a coordination mechanism where agents conform plans specifying all their future actions and interactions related to a particular objective: the agents commit themselves to accomplishing a plan and they behave in accordance with this. The plan benefits the coordination of the agents because they know exactly what actions to take and what actions their acquaintances will take. Since unpredictable events could happen the planning process must be configured for short-term issues of coordination [42].

According to [81], there are three possibilities for multi-agent planning:

- *Centralised planning for distributed plans*: The model consists in a centralized planner which develops a plan for a group of agents. This planner distributes the plan between the agents who are in charge of carrying out the assigned activities.

- *Distributed planning for centralized plans*: A group of agents cooperate to develop a centralized plan. The agents who develop the plan are specialists and they do not execute the plan, they are only in charge of creating the plan.

- *Distributed planning for distributed plans*: A group of agents cooperate to form individual plans of action, dynamically coordinating their activities along the way. The agents may be self-interested, and so when coordination problems arise these may need to be resolved by negotiation.

One example of the first form of multi-agent planning is presented in [10]. This approach is focused on centralized planning for distributed agents, with the Markov decision process (MDP) framework as the basis of a model of agents interacting with an environment. Two models of decision-theoretic planning for distributed agents are analysed using the tools of worst-case complexity analysis. Specifically, the MDP and partially observable Markov decision process POMDP models have been tested. The POMDP model has been extended to allow that multiple distributed agents each receive local observations and base their decisions on these observations. This extension is called the decentralized partially observable Markov decision process (DEC-POMDP).

The paper [84] presents an example of the second form of planning. This work tackles a problem where multiple agents have to find a consistent combination of actions under some constraints concerning taking actions. These problems include distributed resource allocation and are naturally described as a distributed constraint satisfaction problem. In the distributed CSP, variables and constraints are distributed among multiple agents. The agents provide a solution to the distributed CSP by finding a set of values for the distributed variables that satisfies all the distributed constraints.

In [78], one example of the third form is viewed. WALRAS, a Market-Oriented Programming Environment is presented. WALRAS is a prototype environment for specifying and simulating computational markets. To specify a computational economy it is necessary to define a set of goods and instantiate a collection of agents that produce or consume those goods. Given a market configuration, WALRAS then runs these agents to determine a balanced allocation of goods and activities. This distribution of goods and activities constitutes the market solution to the planning problem.

2.3Scheduling

Scheduling is a special case of planning where the tasks or actions are already chosen, leaving only the problem of determining a feasible order [59]. The problem consists in assigning limited resources to tasks over time to optimise one or more objectives [7]. There are three features of scheduling: firstly, scheduling problems are based on reasoning about time and resources; secondly, scheduling problems are almost always optimisation problems; and thirdly, scheduling problems also involve choices. Often this is not just confined to choices of task ordering but includes choices about which resources to use for each task.

Formally, a set R of resources (machines) $\{r_1, r_2, \ldots r_m\}$, and a set J $\{j_1, j_2, \ldots j_n\}$ of tasks (also called activities or operations) are defined. Note that usually the term job is used instead of task. Then, it means that the job is comprised of several assembly activities or operations.

The scheduling problem consists in assigning a sequence of these tasks J to the time t, using the resources R in the most appropriate way. For example, in the problem of producing two elements (10 screws and 5 nuts), two machines are available, and two jobs have to use the two machines to produce the goods. The scheduling problem consists in how to allocate jobs (manufacturing screws and nuts) to the machines in order to achieve the manufacturing goals.

There are many research studies done on scheduling problems due to their relevance to practical problems in industry. The main research areas dealing with this kind of problem are: Operational Research and Artificial Intelligence. As our background concerns the latter (particularly, Distributed Artificial Intelligence and Multi-Agent Systems), in the remainder of this section we review the different types of scheduling problems as well as the techniques used to tackle them, mainly from the AI perspective.

2.3.1 Scheduling problems

Scheduling problems can be classified taking into account different characteristics such as [15]: supply and demand of resources, the size of the problem, time required for the answer, the goal, divisibility, and distribution (see figure 2.3).

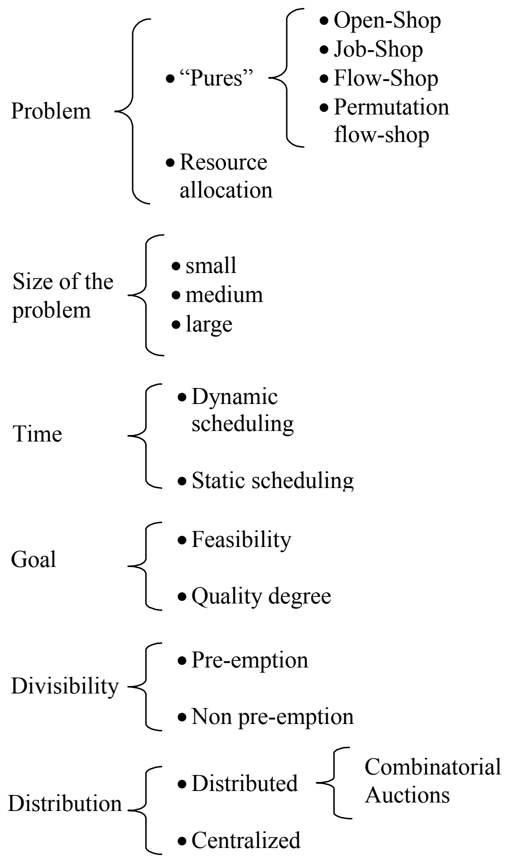


Figure 2.3. Scheduling problems.

2.3.1.1 Supply and Demand of resources

The scheduling problem can be classified in two areas according to the supply and demand of resources in a determined time: "pure" scheduling problems and resource allocation problems.

Firstly, in "pure" scheduling problems, the problem consists in covering the demand for activity resources throughout the time, without exceeding the available resource capabilities. The resources that each activity needs are known previously. This class of problem can be separated into four subgroups depending on the flow [15].

- *Open-Shop*: There are no constraints related to the ordering of the resources for the activities in each job. In an Open shop scheduling problem, the jobs don't have a pre-assigned sequence for the machines [27].

- *Job-Shop*: Each job or assembly of activities has to use the resources in a determined ordering. This ordering can be different for each one of them. Each job has its routing that defines the precedence relations between the job's operations. A feasible scheduling is a set of precedence relationships that give a complete processing order for each machine [82].

- *Flow-Shop*: Jobs use the resources in the same order. It is a particular case of a Job-Shop. There are n jobs and m machines, each job consists of m operations, and each operation requires a different machine. The n jobs have to be processed in the same sequence on m machines, that is, each job has to be processed on all the available machines in the same order 1,2,…,m and each machine can only process one job at a time [43].

- *Permutation Flow-Shop*: Jobs use the resources in the same order and, the resources process the jobs in the same order. This is a particular case of the Flow-Shop. In [72], the domain of n by m permutation flow shop sequencing problem (PFSP) is studied. Here, n jobs must be processed in the same order on m machines. Concurrency is not allowed: a machine can only process a single job at a time, and processing must be completed once initiated. Furthermore, machine j+1 cannot begin processing a job until machine j has completed the processing of the same job.

Secondly, resource and task allocation problems are defined by a set of available operations and for each operation a set of resources is available. These resources perform the same type of operation but they are not equivalent (they can require different times or process costs). The type of resource that each activity requires is given, but not the specific resource.

The problem consists then in assigning tasks to resources so that all activities are performed on time. Often, tasks and resource allocation is used without distinction: tasks can be assigned to resources. Conversely, resources can be assigned to tasks.

Regarding MAS, when agents can play different roles there is an additional terminology: role allocation. This means that agents can play different roles and the scheduling problem consists in assigning roles to the agents.

For the sake of simplicity, we will use the term task allocation throughout the rest of this document.

2.3.1.2 The size of the problem

Scheduling problems can involve a reduced number of activities or thousands of them. In this sense, there are three types of scheduling problems : small, medium or large problems. In general, scheduling problems can be sized as n x m, where n is the number of jobs or tasks and m is the number of machines or resources. This size means the variation of the duration of the manufacturing operations, which depends on the products to be manufactured and the available resources.

2.3.1.3 Time required for the answer

Scheduling problems can be classified into two groups, according to the time requested for the answer, which is conditioned by the changeable conditions of the problem: dynamic scheduling and static scheduling.

In dynamic scheduling, the scheduling is modified as time passes, due to changes in the environment or to the decision making process. The scheduling modification operation is called "*Re scheduling*" or "*on-line scheduling*", when new constraints in the environment make the last scheduling invalid. [60] present the use of incremental, constraint-based scheduling techniques as an effective basis for coordinating activity in knowledge-intensive dynamic systems (KIDS). Constraint-based scheduling models promote controlled continuous change as execution circumstances evolve, they operate under quite general representational assumptions, they integrate well with user decision making, and they directly support negotiation and conflict resolution in distributed settings.

In static scheduling, problem configuration is not changed. In paper [57] the authors present methods to combine static scheduling and online interruption handling in the real-time system for controlling the motion of vehicles. The authors state that a common representation of a static schedule is a vector, where one position in the vector represents a discrete point in time at which the execution of a task can start.

2.3.1.4 The Goal

Depending on the kind of goal the problem is related to:

- A problem with constraints which must be completely satisfied (feasibility).
- A problem with constraints which can be relaxed or partially reached (degree of quality).

However, in real scheduling problems both kinds of constraints are found, so the obtained solutions are characterized taking into account both feasibility and quality. The obtained scheduling solutions must be adaptable to the current context and these solutions must be obtained in a reasonable time period. On the other hand, to find a feasible and high quality solution can take a long time. For this reason it is important to find a balance between feasibility, solution quality and the required processing time.

2.3.1.5 Divisibility

Divisibility is concerned with the property of jobs or activities to be interrupted at a time t and then finished at a time t+d (d>0). According to this phenomenon, two kinds of scheduling problems are identified: pre-emptive and non pre-emptive scheduling problems.

Pre-emptive scheduling is characterized by the fact that each activity can be interrupted and continued later when the resources are available. Conversely, in non-pre-emptive scheduling, the execution of each activity is realized completely before leaving the assigned resource.

2.3.1.6 Distribution

In a distributed environment, resources can be spaced throughout a network and controlled by different activities. Then, scheduling problems can be tackled either from a central or distributed point of view

In centralised scheduling, the scheduling is tackled by a central unit. This model is in the same line as centralized planning. Conversely, distributed scheduling is the problem of allocating resources to alternative possible uses over time, where competing uses are represented by autonomous agents [70].

Choosing one or another distribution form depends on the communication bandwidth limitations and the self motivation of the agents involved [55].

2.3.1.7 The rescue problem

Now, we can classify our problem taking into account the different scheduling facets explained above. Rescue and disaster problems can be classified then, as follows:

a. Supply and Demand of resources

This is a problem of task allocation. In the rescue environment there are mainly three kinds of resources: ambulance teams, fire brigades and police forces, and three kinds of activities or tasks to be performed: rescuing victims, extinguishing fires and unblocking roads.

Each kind of task can be assigned to a particular resource, mainly: rescuing victims to ambulance teams, fire extinguishing to fire brigades, and unblocking roads to police forces.

Moreover, to use an ambulance team close to a victim will probably have a lower cost than using another ambulance team placed at a distant point. Then, we are clearly dealing with a resource allocation problem.

b. The size of the problem

Our problem has a dimensionality (n x m) where:

n: is not known. In the rescue environment there are 72 victims, *i* burning buildings and *j* blocked roads. So n>72.

m: In the rescue context there are 5 ambulance teams, 10 fire brigades, 10 police squads, 1 ambulance central, 1 fire station and 1 police office. So m=27

The dimension of our problem is close to 100, so we believe that we are dealing with a middle sized problem.

Note, however, that in each cycle of the simulation process, the scheduling problem is re-formulated with new information about the environment changes. So, at the beginning some of the 72 victims are known, and during the simulation new victims, fires and blocked roads are discovered. So, the characteristics of the problem evolve over time.

c. Time required for the answer

The rescue and disaster environment is a *dynamic scheduling problem* because the events and configuration in the surroundings change as time goes by.

In each simulation cycle, either a new victim can be found, a new fire discovered or an agent can find a blocked road in his way.

d. The Goal

The rescue scenario simulates a real disaster, so unfortunately there are fewer resources available than the ones required to attend to all the demands. For this reason, it is necessary to relax the constraints in order to solve the problems. See [51] in order to know more about distributed hard and soft constraints.

It is clear that constraints related to the use of resources (rescue teams) must be satisfied: one rescue agent can attend one single victim, one fire, or one blocked road at a time.

Regarding the cost, it is preferable to choose an agent close to the victim in order to minimize the displacement time. However, we can choose a distant agent if the close one is either buried or damaged. So, our problem is related to optimisation.

e. Distribution

The rescue problem is a distributed task allocation problem due to the bandwidth limitations explained in section 1.1.2. In a disaster scenario, communications are limited according to the message traffic from rescue agents to centrals, between centrals, and between rescue agents. So a distributed approach seems to be the most realistic.

2.3.2 Techniques for solving scheduling problems

Traditional techniques for solving scheduling problems can be classified into the following groups [15]: optimisation and approximation (see figure 2.4).

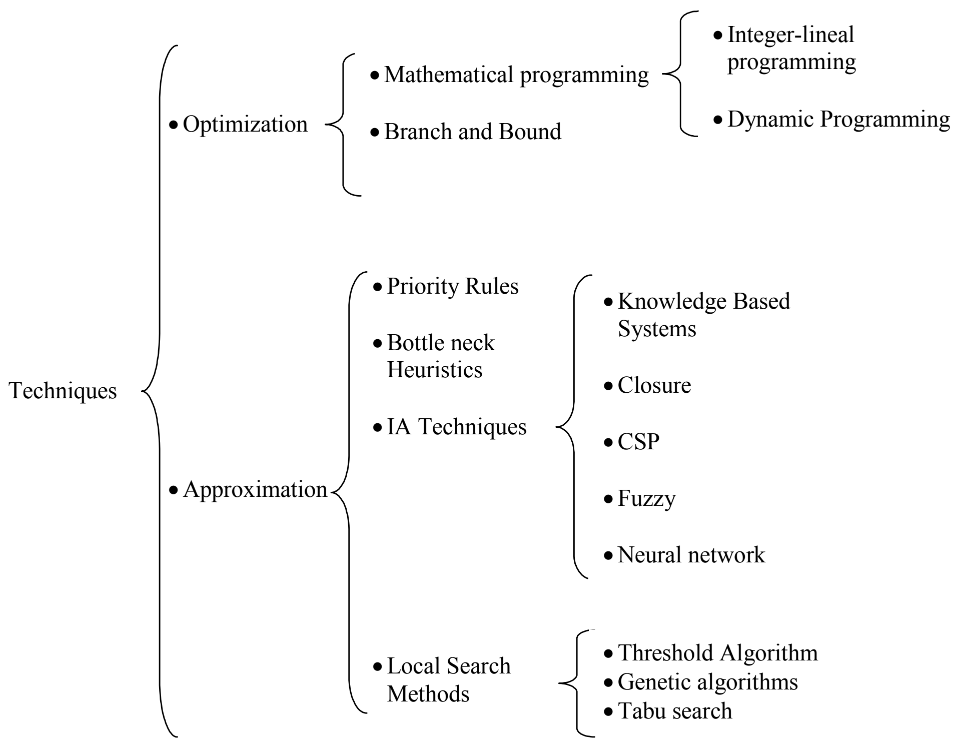


Figure 2.4. Traditional techniques for resolving scheduling problems

Optimisation techniques provide an optimal global solution, but they require a long computational time. There are two subgroups:

- Mathematical programming, and
- Branch and bound.

Approximation techniques provide a good solution in an acceptable computational time. There are four subgroups:

- Rules of priorities
- Heuristics
- Artificial intelligence techniques
- Local search methods.

Nowadays, the distributed ubiquitous computations, communications and new sensors are part of a creation process of a new environment with enormous quantities of data and processing and communicating capacities that were not possible 15 years ago [40]. In this new environment it is possible to formulate more complex scheduling problems that pose new challenges to the research of solutions. New scheduling techniques have been extended and additional techniques have been explored such as: acquaintance, contract net, market mechanisms, distributed CSP, self-organization methods, and ecosystems (see figure 2.5).

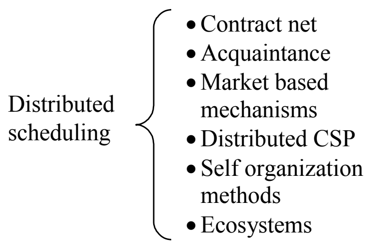


Figure 2.5. Techniques for solving distributed scheduling problems.

We have chosen to work on distributed environments. For this reason we have focused on the state of the art in these methods. A good survey of traditional methods can be found in [59].

2.4Decentralized task allocation

Decentralized resource allocation is an important key problem that requires a set of collaborative agents to effectively manage, distribute and assign their resources so that their overall performance is maximized. Examples include allocation of agents to roles, sensors to targets, equipment to jobs, goods to consumers, activities to time slots, etc. A set of agents is then faced with a global optimisation problem that must be optimised through the local allocation of their resources. This problem differs from traditional resource allocation problems in that resources are controlled or owned by different agents and single agents are not able to determine the best overall allocation individually. Instead, the agents must coordinate their local allocations to achieve good overall performance.

The agents are distributed according to the following criteria [74]:

- Avoid overloading critical resources
- Assign tasks to agents with matching capabilities
- Make an agent with a wide view assign tasks to other agents
- Assign overlapping responsibilities to agents to achieve coherence
- Assign highly interdependent tasks to agents in spatial or semantic proximity. This minimizes communication and synchronization costs
- Reassign tasks if necessary to complete urgent tasks

In the current literature there are very different representations and techniques proposed for decentralized resource allocation. We present some of them below:

- *Acquaintance mechanisms*: Task allocation taking into account the skills of acquaintance agents.
- *Contract net*: announce, bid, and award cycles
- *Market mechanisms and auctions*: tasks are matched to agents by a generalized agreement of mutual selection (analogous to pricing commodities)

- *Distributed Constraints Satisfaction Problems (CSP)*: Variables and constraints are distributed among the agents to find the solutions to the problems.
- *Self-organization methods*: Agents are based on natural behaviour in order to carry out their task and accomplish goals.
- *Ecosystems*: The dynamics of ecosystems have been used for scheduling in MAS
- *Organizational structure*: agents have fixed responsibilities for particular tasks.

In the rest of this section a detailed description of the first six mechanisms is presented. The latter, organizational structure, is not considered to be a proper task allocation mechanism because the tasks are fixed.

2.4.1 Acquaintance mechanisms

Models containing social knowledge are usually called the acquaintance models. An acquaintance model is a knowledge-based model of the agents' mutual awareness that collects the agents' knowledge about their collaborators and about suitable communication and negotiation scenarios [35].

When forming a coalition an agent usually presents other peer agents with a proposal for collaboration and they reply with a bid. The coalition forming agent analyses the responses and selects the best possible bid and then contracts the appropriate agent. If an agent had this knowledge precompiled, a substantial amount of time and resources would be saved [36], that is, the acquaintance models help Multi-Agent task allocation by means of setting a knowledge-based system with suitable information of each agent in a distributed form. An example of such a mechanism is the Tri-base Acquaintance Model (3bA) [36], which is a formal model of the agents' mutual awareness. This acquaintance model is used for supporting the process of coalition formation. The 3bA maintains the agent's social knowledge about collaborative agents that would limit the possibility of manufacturing coalitions substantially.

2.4.2 Contract Net

The contract net protocol is designed based on the contracting mechanism used by businesses in the exchange of goods and services. The contract net protocol provides a solution for finding an appropriate agent to work on a given task. The basic steps in sequential order of this protocol are [74]:

1. The manager announces a task that needs to be performed.
2. The contractor receives the task announcements
3. The contractor evaluates his capability to respond
4. The contractor responds (decline or bid)
5. The manager receives and evaluates bids from potential contractors
6. The manager awards a contract to a suitable contractor
7. The contractor performs the task if its bid is accepted
8. The contractor reports the results
9. The manager receives and synthesizes the results.

The roles of the agents are not specified. A manager can be a contractor and in a similar way a contractor can be manager. In this sense, this protocol implements the decomposition of the task because a contractor can allocate a fraction of his task to another contractor, therefore becoming a manager. In addition, if a contractor that has been selected to perform a task is not able to perform it the manager can reallocate this task to another contractor.

Each potential contractor selects one of the tasks announced depending on their capacity and availability, and the manager chooses the best bid received taking into account some criteria, for instance the bid that maximizes the benefits or minimizes the cost. Once the manager has chosen the winner it sends notification that the bid has been accepted to the contractor chosen. In the case that the bid is accepted before the deadline for task announcements, the contract net process is finished.

In [26] the contract net protocol is used for negotiations in scenarios for shipping companies where an effective strategy for resource allocation is necessary. In a shipping service it is important to plan the use of available trucks in order to use them efficiently. This approach presents a multi-agent system made up of two types of agents: the first is the shipping company agents and the second is the truck agents that are considered "resources". The task is to transport goods between different places.

In [3] an approach for assembly task allocation in holarchy manufacturing systems is developed. A mechanism for bidding is used in order to allocate tasks to the free assembly devices. The process of bidding basically consists in each of the devices returning the number of the work in their waiting queue as their bid value. Once the assembly operation holon has stored the bid set, it negotiates with the execution holons using a contract net protocol in order to choose the winners.

2.4.3 Market mechanism

The defining feature of a market-based mechanism is that the allocation is mediated through a price system [76, 77]. In a market price system, each distinct resource type is related to a good, and each good is associated with a numeric specification of its exchange terms – the price. The basic rule of a price system is that goods may be exchanged according to the relative prices of the goods involved. Market mechanisms are effective for coordinating the activities of many agents with minimal direct communication among the agents [74].

In a market the agents can measure everything related to money. The main components present in a market model are [75]:

- A collection of goods, g1,….,gn, and
- A collection of agents, which can be divided into:
    - Consumers, who simply exchange goods, and
    - Producers or sellers who can transform some goods into others.

A consumer is defined by two variables: firstly, a utility function which specifies its preferences according to certain bundles of goods, and secondly, the initial quantities of the goods [75]. A third variable could be the budget, which is the "money" that the consumer has or is ready to pay for a determined bundle of goods. Taking these aspects into account, the consumer takes the decision of how to maximize the utility according to the available budget.

The other classes of agents are the producers and sellers. The producer transforms a group of goods into others and the seller sells goods, taking into account the market characteristics. In order to do so, in the producer's case, the technology used for the transformation is taken into account. The decision which the producer must take is how to maximize the profits, which is a function of the total sales of this product and the cost of producing it.

Profit=Total price of the good sold – cost of producing the good

The general purpose in a market mechanism is to obtain a balance (equilibrium) between the consumers and producers. The consumer tries to obtain the products at the lowest price, in a way that maximizes the benefits, and the producers or sellers wish to sell the products and also obtain the maximum profit.

In [58] two mechanisms for task and resource allocation in organizations are proposed: the task allocation protocol (TAP) and the resource allocation protocol (RAP). The mechanism for task allocation TAP, takes requests and plans and allocates subtasks to agents within an organization. On the other hand, the mechanism for resource allocation RAP is based on market techniques.

The authors present a MAS scenario composed of organizations where each organization is divided into agent teams whose goal is to resolve problems. There are three types of agents: A fixed set of agents called permanent agents, (associated with each organization), a set of marketable agents who are the resources to be allocated and a set of resource manager (RM) agents.

The TAP objectives are the identification of team members in order to reach a social goal and to develop a common solution that will be accepted by the team and implied RM agents. The task allocation is developed taking into account the priority of tasks that is, the lower priority tasks are either rescheduled to accommodate a more critical task, or recommitted if deadlines make rescheduling impossible. The RAP uses the price-directed microeconomic approach to dynamically allocate marketable agents to organizations according to their computational load. The MAS is organized as a market economy where buyers and sellers interact. The goods are the marketable agents that the organizations want to purchase in order to develop a task. The organizations send fund values which determine the priority of task, the higher the priority the higher the fund that is sent.

In [5] an approach for resource allocation of two commodities is developed, namely electric and heating energy. In this scenario there are three types of agents: consumer, producer and auctioneer agents. Consumer agents control the electric and heating power consumed by the occupants of a house and they also compute the demand function. Producer agents evaluate their supply function to maximize profit. Auctioneer agents receive the demand functions of the consumer agents and the supply demand of the producer agents. Once the auctioneer agent has received this information, he determines the demand and supply of the entire electric market. The intersection between the global demand function and the global supply function determine the new market equilibrium price. This market equilibrium price is broadcast to all agents.

WALRAS [78] is a general "market-oriented programming" environment for the construction and analysis of distributed planning systems. The environment provides basic constructs for defining computational market structures, and a procedure for deriving their corresponding competitive equilibrium. In [79], a particular approach for a simplified form of distributed transportation planning is presented. In this transportation planning problem, the task is to allocate a given set of cargo movements over a given transportation network. The transportation network is a collection of locations, with links (directed edges) identifying feasible transportation operations. Associated to each link there is a specification of the cost of moving cargo along it. Market equilibrium is used to determine the amount to transport on each link in order to move all the cargo at the lowest possible cost.

2.4.4 Distributed Constraint Satisfaction

Nowadays, many real problems can be modelled as Constraint Satisfaction Problems (CSPs). CSPs involve finding values for variables subject to restrictions in which combinations of values are acceptable. A constraint graph is a representation of the CSP where the vertices are variables of the problem, and the edges are constraints between variables. Each variable has labels that are the potential values it can be assigned. CSPs are solved using search (e.g. backtrack) and inference (e.g. arc consistency) methods. [61]. The search algorithms used for resolving CSPs require an order in which variables and values should be considered. Choosing the right order of variables and values can noticeably improve the efficiency of constraint satisfaction [52].

For example, a typical example that shows how a CSP works is the colouring problem. A map colouring problem is as follows: Given a map with *N* regions bordering each other and *M* colours that can be used to colour each region, the problem is whether it is possible to assign one colour to each region such that no two neighbouring regions (regions that share at least one border) have the same colour. Figure 2.6 shows a map colouring problem.

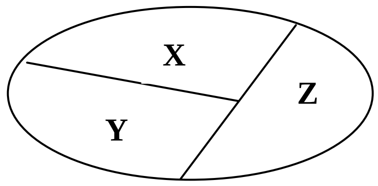


Figure 2.6. Map Colouring Problem

In this problem, variables of a CSP are the regions (**X**, **Y** and **Z**), the values are the different colours (*red*, *blue* and *green*), and the constraints are that no neighbouring regions can have the same colour (no two variables connected by a *'not equal'* edge can be assigned the same value).

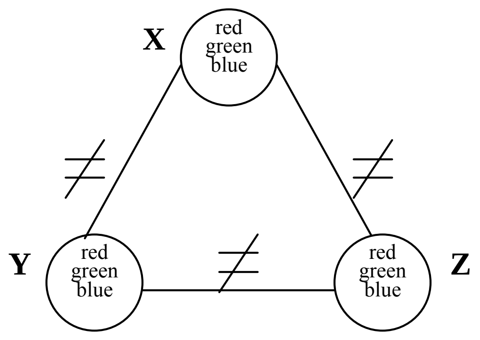


Figure 2.7. Constraint Graph

Figure 2.7 shows the constraint graph of this map colouring problem. The nodes represent the different regions, the labels for each node are the different colours a region can have, and the existence of an edge between two nodes indicates that the two corresponding regions cannot have the same colour (there is a 'not equal' constraint between these two nodes). One solution of this problem is to assign *red* to **X**, *blue* to **Y** and *green* to **Z** [61].

A distributed CSP is a constraint satisfaction problem where variables and constraints are distributed among multiple agents. A solution to a distributed CSP is a set of values for the distributed variables that satisfies all the distributed constraints.

For example [33] presents a new algorithm for solving frequency assignment problems in cellular mobile systems using constraint satisfaction techniques. Experimental evaluations using standard benchmark problems show that this algorithm can find optimal or semi-optimal solutions for these kinds of problems.

See [34], for a good review of this approach.

2.4.5 Self organization methods

In [13], a system named Ant Colony Control (AC^2), coordinates a population of simple agents, namely the ants, via indirect communication accomplished by altering and reacting to the commonly shared distributed environment of the factory. The overall behaviour that emerges from this completely autonomous and decentralized system performs in a similar way to the optimal routing strategy for simple problems.

Analogies to the way in which ants and other agents solve problems in nature have been used to solve some difficult real–world problems. Approaches for coordinating multiple robots in the performance of tasks such as object sorting and object clustering [8] have mimicked the ways in which ants sort their brood or collect their dead in cemeteries. For a more complete overview of the field, see [11].

2.4.6 Ecosystems

An ecosystem analysis begins with an examination of the rules of the game. These rules are the constraints within which organisms of that ecosystem must operate. An example of a physical constraint in biological ecosystems is the conservation of mass and energy. Plants can obtain these resources by photosynthesis; access to sunlight and land area is the critical limiting resource. Biomass and free energy are needed to sustain activity and can be transferred by predation [20].

The dynamics of ecosystems have been used to develop MASs, which demonstrate emergent cooperation as a result of acting with selfish interests. For instance, cooperation in higher animals, such as wolf packs, has generated advances in cooperative control [4].

Ecosystems seem mostly to be limited by resource availability, for this reason their complexity is staggeringly high. This approach follows a distributed control with individually controlled agents. These systems have robustness and adaptability, and scale well partly because they are self organizing around local concerns [28].

2.4.7 Others mechanisms

In [68] an approach for dynamic task allocation for multiples robots is proposed based on conflict resolution techniques. First, the mechanism proposes a division of tasks into subtasks or modules. For instance, if a task is: " carry object A to the goal", this task is subdivided into the following subtasks: "push object A", "avoid obstacle", "approach an obstacle", "push an obstacle" and so on. Once the task division is developed the robots choose the module or subtask to develop taking into account a priority function and a strategy for module conflict resolution.

2.5 Auctions

Market mechanisms have been applied in prior works to scheduling (see [76, 83, 78, and 79]). As stated in the previous section, market mechanisms follow an equilibrium theory in order to establish prices. It has been demonstrated that such methods sometime do not work, particularly when problems involve indivisible resource units [70].

The research community is now focusing on new marked-based paradigms such as auctions. An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants [70].

An auction consists of an auctioneer (seller) and potential bidders (buyers). Auctions are often used in situations where the auctioneer wants to sell an item and get the highest possible payment for it while the bidders want to acquire the item at the lowest possible price.

An auction includes at least the following component message types [76]:

- *Bids*: represent an agent's willingness to engage in some exchange or set of exchanges.

- *Price quotes*: Information about the state of an auction before an allocation.

- *Notifications*: Informs the agents about the result of an auction, that is, the final allocation and price.

Auctions can be characterized by several parameters, mainly [70]:

- Price determination algorithm (protocol)
- Bidding rules: that regulate the process in which bidders can submit bids
- Bidding policies [55] which reflect the strategy for the decision making process performed by agents in order to decide on bids. It is possible to distinguish among:
    - Private value auctions: The value of the good depends only on the agent's own value.
    - Common value auctions: an agent's value of an item depends entirely on other agent's values of it.
    - Correlated value auctions: an agent's value depends partly on its own preferences and partly on the other agents' values.

The auction mechanisms are distributed in the sense that each agent performs a local computation regarding their own bid strategy. Regarding the final allocation of the goods, such processes are distributed or not depending on the price determination algorithm. Following there is an overview of auction mechanisms regarding protocols together with other classification issues. At the end of this section, we focus on a special kind of auction called a combinatorial auction and its application to distributed scheduling problems.

2.5.1 Auctions types

Auctions can be classified taking into account several issues, such as: the protocols, the number of units of items, the order in which the items are auctioned, the number of buyers, the role of participants and the trading phases (see figure 2.8).

2.5.1.1 Protocols

There are several protocols to deploy an auction-based mechanism: English, First-price sealed-bid, Dutch, Vickrey, and all-pay auctions are the most popular [55].

In English (ascending) auctions, each bidder is free to raise his bid. When no bidder wishes to raise his bid, the auction ends, and the highest bidder wins the item at the price of his bid. In an English auction the more common strategy is always to bid a little more than the current highest bid and then stop when the private value is reached.

In the first-price sealed-bid auction, each bidder submits one bid without knowing the other bids. The highest bidder wins the item and pays the amount of his bid. The agent's strategy is to bid an amount that is a function of its private value and past information of other agents' valuations.

In the Dutch (descending) auction, the item price is continuously diminished by the seller until one of the bidders takes the item at the current price. Strategically, the Dutch auction is similar to the first-price sealed-bid auction because no information is known during the auction process by the bidders.

In the Vickrey auction, each bidder submits his bid without knowing about other bids. The highest bidder wins but the winner has to pay the second highest bid value.

2.5.1.2 Number of units of items

This issue is related to the number of items to be auctioned. There are two main kinds of auction:

- *Single unit auctions*: An auction in which only one item is available for sale.

- *Multiunit auctions*: An auction in which several (more than one) items are available for sale. Mechanisms for allocating multiple units include discriminatory and uniform price auctions [30].

2.5.1.3 Order in which the items are auctioned

There are three main orders in which items can be auctioned:

- *Sequential auctions*: An auction in which the items are auctioned one at time [14].

- *Simultaneous or parallel auctions*: An auction in where the items are open for auction simultaneously [83].

- *Combinatorial Auctions*: A multiunit auction in which each bidder offers a price for a collection of goods (of the bidder's choosing) rather than placing a bid on each item separately. The auctioneer selects a set of these combinatorial bids which raises the most revenue without assigning any of the objects to more than one bidder [54, 32, and 48].

2.5.1.4 Number of buyers

Regarding the number of buyers involved, it is possible to distinguish two main types of auctions:

- *Reverse*: (n sellers – 1 buyer) While traditional auctions involve a single seller and many buyers, a reverse auction generally involves many sellers and one buyer. For example, procurement auctions are used to obtain competitive bids to provide goods or services. In [56] it is explained that the buyer can hold a reverse auction to try to obtain the goods. Each seller submits “asks” that say how much the seller asks for each bundle of goods they can provide.

- *Direct*: (1 seller – n buyers) Auctions involve a single seller and many buyers.

2.5.1.5 Roles of the participants

Regarding the roles of the participants, that is, becoming an auctioneer (seller) or a bidder (buyer) two classes of auctions are found:

- *Double Auctions*. Standard auction theory assumes a single seller controls the trading mechanism, while many buyers submit bids. In a double auction, buyers and sellers are treated symmetrically with buyers submitting bids and sellers submitting asks. Both buyers and sellers submit bids [21]. A single agent may even submit both, offering to buy or sell depending on the price.

- *Asymmetric Auctions*: In a scenario that is specific to auctions, the buyers and sellers cannot change their roles.

2.5.1.6 Trading phases

Regarding the different phases involved in the auction process, two main auction design are defined:

- *Continuous double auction (CDA):* The main characteristic of CDAs is that they run in continuous mode i.e., there are no "trading phases" like other auctions may have. In a CDA, after a transaction occurs, the outstanding bid and ask are removed, and a new round of CDA starts, in which the auction process is repeated [71].

- *Discrete double auctions:* In these auctions there are “trading phases”, i.e. they run in discrete mode.

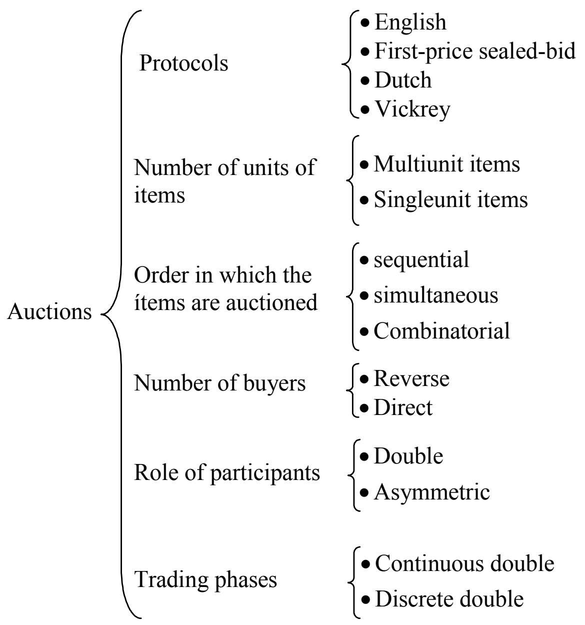


Figure 2.8. Types of Auctions

2.5.2 Combinatorial Auctions

In an auction process, resource availability cannot be guaranteed until contracts are finished. Trade situations can result in the development of a task depending on two resources, this could be unfeasible due to the fact that only one of the resources has been achieved. This is known as the complementarities issue. The value of an asset to a bidder depends strongly on which other assets the bidder wins. In this sense, a synergy or superadditivity of values, that is the assignment of two (or more) goods to a single agent, has a superior surplus than allocating the two goods separately to two different agents. In order to solve this problem, researchers have grouped the negotiation of several goods in a single auction, in what is called combinatorial auctions.

In an auction, the seller wants to sell certain items and get the highest possible payment for them, while each bidder wants to acquire the items at the lowest possible price. Combinatorial auctions are the auctions in which the bidders are allowed to submit bids on both individual items as well as on a combination of items. In a combinatorial auction, there is one seller (or several sellers acting in unison) and multiple bidders who may place bids on combinations of items [53]. The final objective is to obtain the maximum benefit for the seller by determining the appropriate set of winning bids (i.e., the winners).

In a combinatorial auction, each bidder sends the seller their offers for various groups of items, and the seller's objective is to allocate the items in a way that maximizes his revenue. The winner determination problem means choosing the subset of bids that maximizes the seller's revenue, subject to two constraints; that each item can be allocated only once, and that the seller wants to sell all the items.

Formally, let's say M = $\{t_1, t_2, \ldots.. t_m\}$ the set of items to be auctioned and let $B=\{b_1,\ldots,b_n\}$ be a set of bids. A bid is a pair $(p(b_i),s(b_i))$ where $p(b_i) \in \Re^+$ is the price offered by the bid bi and $s(b_i) \subset M$ is the set of items requested by bid b_i. For each bid b_i, an indicator variable x_i is defined. This indicator is a binary variable representing whether bid b_i is selected or not.

The Combinatorial Auction Winner Determination problem can be proposed as the following integer programming problem [2, 17]:

Minimize: $\sum_i x_i \, p(b_i)$ (2.1)

Subject to: $\sum_i x_i \leq 1 \quad \forall t \in M$ (2.2)

$x_i \in \{0,1\} \quad \forall i$

This problem is known to be NP-complete. However, [50] argues that the winner determination algorithm does not deal with all the possible combinations depending on the number of items in M, but with only those bids that are submitted. The bid space is often sparsely populated and under such restricted circumstances it is possible to solve the winner determination problem in a reasonable time (see also [55]).

Moreover, he argues that the complexity is not due to the size of B but rather to its structure. Particularly, [50] has proved the manageability of the problem for the following structures: tree-structure, cardinality-based structures, and geometry-based structures. For such structures of B, the optimal outcome can be determined in $o(n^3)$ or less. A family of sets B form a tree structure if for all b,b' $\in$B, either they are disjointed or one is a subset of the other. Regarding cardinality, while the number of goods submitted in each bid is less or equal to 2, the problem is manageable. Finally, regarding the geometry, if bids can be linear or cyclically ordered, the problem has a good shape. Unfortunately, for other dimensions (such as geographic positions on a map) the complexity of the problem remains NP-complete.

There are several algorithms proposed in the literature to solve the winner determination problem, some come from the Operational Research (OR) field, and others from the Artificial Intelligence discipline. All of them impose constraints on the bids in order to make the problem manageable. In this work, we explain two: Sandholm's algorithm and the Combinatorial Auction Structured Search (CASS) algorithm. These algorithms have been shown to outperform previous works. Other interesting equivalent approaches coming from OR can be found in [2]

2.5.2.1 Saldhom' algorithm

According to [53], the optimal winner determination problem can be solved by using a search algorithm. The search space is defined as follows: nodes keep information on bids and paths provide combinations of bids.

The list of bids is denoted by $\{B_1, ..., B_n\}$. Each bid B_j is a tuple $<S_j, \bar{b}_j>$ composed of the set S_j of items in the bid and the price $\bar{b}_j$ of the bid.

Each path is a sequence of disjoint bids, so that no items are shared. That is, for any path $p_k=\{B_{k1}, B_{k2,} ... B_{km}\}$, it holds that $S_{k1} \cap S_{k2} \cap ... \cap S_{km} = \emptyset$. One possible solution is a path in which

$$S_{k1} \cup S_{k2} \cup ... \cup S_{km} \leq M.$$

In order to generate a solution, nodes are generated in lexicographic order of the set of items of the bids. For example, given the following bids:

$$B_1=<(319, 1230),10>$$
$$B_2=<(2500,3829),21>$$

bid B_1 will be selected first, since the first item in the bid combination of B_1 is 319, and the first item of B_2 is 2500, i.e., $319 < 2500$.

The cost g_k of the path p_k is defined as:

$$g_k = \sum_j \bar{b}_{kj} \quad (2.3)$$

The paths that interest the auctioneer are the ones that lead to a maximum g value.

Sandholm proposes an A* admissible heuristic method to search this auction space in order to find the winning combination. The heuristics of the search algorithm are defined as f=g + h, where g is the cost of the path up to the current node, according to (2.3), and h is the cost of the bid to be selected, that is, $\bar{b}_i$

The novelty of the Sandholm approach compared with its predecessors is that this algorithm works well with significantly large amounts of items and bids based on the fact that, in practice, the bid space is sparsely populated [53]. However, the algorithm uses several pre-processing steps to achieve this functionality, which make the application not so straightforward.

In [32] a simple version of this algorithm (without pre-processing) has been applied to the task allocation in the rescue scenario. A discussion about this experiment can be found in section 3.6.

2.5.2.2 The Combinatorial Auction Structured Search (CASS) Algorithm

Another approach to winner determination in combinatorial auction is CASS [17]. This algorithm creates one list for each of the items that are being auctioned (each of these lists are called bins in [17]), and each bid is stored in exactly one list. Taking into account the ordering of the items in the bid, each bid is stored in the list which corresponds to the first item that appears in the bid.

The algorithm to search for the optimal solution follows a branch and bound strategy. The basic steps are the following (from [17]):

1. For each pair of bids (b_i,b_j) where $s(b_i) \subseteq s(b_j)$ and $p(b_i) \geq p(b_j)$, b_j is removed from the lists of bids to be considered during the search as b_j is never preferable to b_i.
2. Organize the bids in lists according to the good and bid ordering heuristics, which are explained later.
3. Choose the first set from the first list.
4. Add (to the current solution) the first disjoint set from the first list, corresponding to an item not included in the current solution and containing such a disjoint set, if any.
5. Repeat step 4 until
6. Nothing more can be added: Check if this solution is the best so far.
7. The value of the current solution cannot exceed the value of the best combination found so far: this branch of the search can be pruned.
8. Backtrack: Replace the last chosen set with the next valid set in its list and go to Step 4.

In order to organize the bids in the search in [17] two heuristics have been developed: good ordering heuristics and bid ordering heuristics.

On one hand, the goal of the good ordering heuristic is to determine the order of the goods, and therefore the list order.

For each good i a $score_i$ is computed according to the following expression:

$$score_i = \frac{numbids_i}{avggoods_i} \qquad (2.4)$$

Where:

- $numbids_i$ is the number of bids that request good i
- $avggoods_i$ is the average number of goods requested by these bids.

The list corresponding to the bid with the lowest score is selected and labelled as D_1. Then, the scores for the remaining goods are recalculated and the process is repeated until a total order of the list is achieved, from the lowest score to the highest: D_1, …, D_m.

On the other hand, the goal of the bid ordering heuristic is to determine the order of the bids within lists.

Given a partial allocation π, bids in the lists are sorted into descending order according to their score. The score of bid j is computed according to the following equation:

$$score(b_j) = \frac{p(b_j)}{|S(b_j)|} + h(\pi \cup b_j) \qquad (2.5)$$

Where $h(\pi \cup b_j)$ is an upper bound that estimates how promising the units not yet allocated are. Such a boundary is computed as follows:

$$h(\pi) = \sum_j v(j) \quad (2.6)$$

Where v(j) is defined as follows:

For each bid b_i, let $a(b_i)$ be the average price per good of bid b_i:

$$a(b_i) = \frac{p(b_i)}{m} \quad (2.7)$$

Let $L_\sigma(j)$ be the list of bids that refer to good j and that can be found in the remainder of the search. Elements in $L_\sigma(j)$ are sorted into decreasing order according to their $a(b_i)$. Let b^{ok}_i be the first bid in $L_\sigma(j)$ that does not conflict with π. Then $v(j)=a(b^{ok}_i)$.

The CASS algorithm is currently the winner determination algorithm that performs best [2].

2.5.3 Distributed task allocation based on combinatorial auctions

Combinatorial auctions have emerged as an important model in economics and show promise for being a useful tool for tackling distributed task allocation problems in AI. Some of the advantages, explained above, are the superadditivity issue and their capability of handling complementarities. However, another important fact is that as a coordination mechanism they combine local and overall computation. On one hand, local computations are performed on the agent level. Each agent estimates its potential contribution to the group activity and its costs by examining their individual contexts and commitments. On the other hand, overall computation needed to solve the winner determination algorithm takes into account the potential contribution of all agents looking for an overall benefit.

In the following table, we summarize some of the approaches for task allocation and combinatorial auctions:

Reference	Problem/domain	Comments	Techniques
[44]	Airport Time Slot Allocation		Sealed-bid combinatorial auctions.
[69]	Supply Chain Formation.		• Monte Carlo simulations • Bayes Nash equilibrium
[18]	MUCA	Finds an optimal solution and reduces computational time	Branch and Bound Heuristics
[19]	Solving Combinatorial	Finds solutions close to being	Stochastic Local search

	Auctions.	optimal.	
Ref [32, 64, 66]	Rescue Task Allocation.		• Integer-Lineal Programming • Branch and bound • Heuristics

Table 2.1. Distribute approaches about task allocation and combinatorial auctions

In [44] a sealed-bid combinatorial auction is developed for the allocation of airport time slots by competing airlines. This auction procedure permits airlines to submit various contingency bids for flight-compatible combinations of individual airport landing or take-off slots.

In [69], a particular combinatorial protocol consisting of a one-shot auction and a strategic bidding policy is developed. The authors present the problem of task allocation as a task dependency network, which is a directed, acyclic graph where consumers and producers are the agents who are represented by the nodes (rectangles and octagons). Goods are also nodes that are represented by circles. Relations between consumers, producers and goods are represented by the edges. Edges connect agents with goods they can use or provide. An allocation is feasible if all agents are feasible and all goods are materially balanced, that is the number of edges in a good equals the number of edges out of a good. A solution is a feasible allocation such that one or more consumers acquire a desired good. Bayes-Nash equilibrium is used in order to define a plausible strategic bidding policy. The Monte Carlo simulation is employed in order to develop experiments.

In [18], the authors use the branch and bound technique in order to deal with Multi Unit Combinatorial Auctions (MUCA). They explore and study the three components of this technique which are: a good upper-bound for the value of the optimal solution of the MUCA concerning the items still unallocated, the order in which the bids are entered and the good heuristics. Firstly, if the sum of the upper bound and the values of the bids of the partial solution are not larger than the value of the best solution found so far, it is possible to backtrack immediately. Secondly, it is essential to find a good assignation early so as to choose the order in which the bids are entered into partial solutions and try the most promising bids first. And thirdly, a good heuristic is needed to pick the most promising bid in a set of bids. This heuristic is used to find the most promising bid in the MUCA concerning the items still unallocated iteratively in order to find the most promising of those bids that do not conflict with the partial solution already found. The authors have tested three methods in order to find the bounding: The linear programming technique, the optimal solution to the knap-sack problem and average price consideration. They present some functions in order to choose the most promising bid and finally, they present some experimental results to test the upper bound techniques and heuristics functions proposed using the same distribution as in [30]. The authors conclude that the linear programming technique for finding the upper–bound, provides the best result but they also found that this technique is extremely costly (in computational time) to use. For this reason this is unfeasible for large combinatorial auctions. The second best option in order to find the upper-bound is the average price technique, which is easier to implement.

In [19], stochastic local search (SLS) techniques are used to solve the winner determination problem in combinatorial auctions. The basis of SLS techniques consists in an intuitive approach: start with a good allocation and then try to improve it iteratively. The authors show that SLS can be applied with great success to the winner determination problem, finding high quality solutions much quicker than systematic techniques and often finding optimal solutions. In addition, these techniques can tackle problem instances of considerably larger sizes than existing systematic methods.

In [32] the combinatorial auction techniques are used in a disaster scenario. In particular Sandholm's algorithm, is applied to rescue systems from the Robocup rescue simulation league.

2.6 Conclusions

In this paper we have presented some previous work related to our research work. We have reviewed coordination in multi-agent systems, from which we have outlined the cooperative framework in which planning and scheduling mechanisms have been revisited. Particularly, we have identified our rescue problem as a distributed task allocation problem and then we have reviewed techniques in this line. Special emphasis has been placed on the study of new techniques that have arisen in the last years, such as auctions and combinatorial auctions. They seem to be a promising mechanism for tackling distributed scheduling problems that some authors have already started to explore.

We believe that combinatorial auctions will be a suitable technique for solving our rescue problem. The advantages of local and overall computation, together with the ability to deal with indivisible goods (resources) and complementarities, make combinatorial auctions an adequate mechanism for handling our problem. Moreover, regarding local computation, this is an important factor when agents may be unable to share complete information, as in a critical scenario like a rescue situation.

In the next section we detail our exploratory work regarding the application of combinatorial auctions to our problem, together with other previous, failed attempts to use other techniques successfully.

Chapter 3

3 RESEARCH WORK

In this chapter the exploratory research work in the field of coordination and cooperation in multi-agent systems is presented. We restrict ourselves to distributed task allocation problems such as the rescue scenario domain explained in the motivation of our work. First, the general problem formalization is given. Then, we focus on the formulation of examples of the general problem in the particular case of the rescue problem. We continue by providing the general MAS architecture where our coordination scheduling mechanisms are deployed. Next, the proposed mechanisms are explained, namely: case based reasoning, fuzzy filters, multiple criteria decision making and combinatorial auctions.

3.1 General Problem formalization

In order to formulate the kind of problems we are dealing with, in this section we provide some notation and a formal definition.

3.1.1 Notation

Regarding the tasks, let $T=\{T_1,\ldots,T_n\}$ be the set of the tasks to be developed by the agents. Each T_i is constituted by several parameters as follows:

$$T_i=<d_i, K_i, C(T_i), U(T_i), S_i, Pr_i, t_i, Ag(T_i), Av(T_i)>$$

Where,

- d_i is the deadline (due date) in which the task T_i has to be finished.

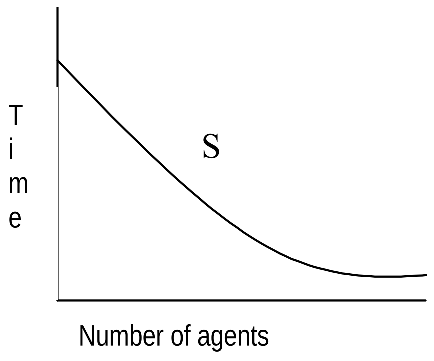


Figure 3.1. Graphical representation of time slots.

- K_i is the type of activity related to the task.
- $C(T_i)= \{c_{i_1},...,c_{i_k}\}$ represents the abilities required to perform task T_i.
- $U(T_i)=\{u_1,\ldots,u_k\}$ is the set of numbers of the units required from each ability in order to perform the task. There is a unit number for each ability expressed in $C(T_i)$, that is,

$$\forall c_j \in C(T_i), \exists u_j \in U(T_i)$$

- S_i is the time slot required to execute task T_i. Figure 3.1 shows how S is diminished proportionally to the number of agents (noted as $Ag(T_i)$) assigned to the task, that is,

$$S_i=f(Ag(T_i)).$$

- Pr_i represents the priority of Task T_i. 0 indicates the lowest priority and 10 indicates the highest priority.
- t_i indicates the initial execution time of the task (solution of the problem).
- $Ag(T_i)= \{A_{i_1},...,A_{i_p}\}$, are the agents to which the task has been assigned. A single task can be assigned to more than one agent. For example in the problem of moving a table, two agents are required. In fact, agents should cover all the abilities required for the task, that is,

$$\forall c_j \in C(T_i), \exists A_h \in Ag(T_i) \mid c_j \in C(A_h).$$

Where $C(A_h)$ are the capabilities C of the agents (see below).

Note: than a given task requires $\bar{u} = \sum_{i \in U(T_i)} u_i$ abilities and can be assigned to p agents, with $p \leq \bar{u}$, since one agent can have more than one ability that is needed for the task.

- $Av(T_i)= \{A_{i_1},...,A_{i_q}\}$ indicates agents available to perform a given task. It holds that, $Av(T_i) \supset Ag(T_i)$. It is desirable that $\forall c_j \in C(T_i), \exists A_h \in Av(T_i) \mid c_j \in C(A_h)$. However it could be the case that $\exists\ c_j \in C(T_i)$, such that $\nexists\ A_h \in Av(T_i) \mid c_j \in C(A_h)$. In this case, task T_i cannot be executed until an agent with the necessary abilities becomes available or a new agent enters the system.

The t_i, $Ag(T_i)$ parameters, are the solution to the problem. In addition, it is possible to configure a matrix P of precedence relationships among the different tasks, nxn, where: $P_{ij}=1$ if task T_i precedes task T_j

Regarding resources, let $A = \{A_1,...,A_l\}$ be the set formed by the agents (solvers of the tasks). They are modelled according to the following parameters:

$$A_i=<C(A_i), T(A_i)>$$

Where:

- C(A$_i$)= $\{c_{i_1},\ldots,c_{i_k}\}$ are the agents capabilities.
- T(A$_i$) indicates the tasks assigned to the agent.

In addition, let B(K$_i$) be the time distance required by an agent to perform two consecutive tasks of the same kind K$_i$.

Finally, let C={c$_1$, …, c$_m$} be the set of capabilities that can be defined in any agent (C(A$_i$) $\subset$ C); conversely, the set of abilities that any task could require (C(T$_i$) $\subset$ C).

3.1.2 Problem formulation

Given a set of tasks T={T$_1$, …, T$_n$}, each one of which is described as a tuple T$_i$ = < d$_i$, K$_i$, C(T$_i$), U(T$_i$), S$_i$, Pr$_i$, t$_i$, Ag(T$_i$), Av(T$_i$)>, assign to each task their corresponding initial time t$_i$ and resources Ag(T$_i$) in order that the following constraints are satisfactorily executed:

1. The execution of a task should be performed before its due date

$$\forall i (t_i + S_i) \le d_i \qquad (3.1)$$

2. If a task T$_i$ precedes another task T$_j$, T$_i$ should be finished before T$_j$ is initiated

$$\forall i, j (P_{ij} = 1) \Rightarrow (t_i + S_i - 1) < t_j \qquad (3.2)$$

3. Tasks assigned to an agent should not be overlapped

$$\forall i, j, k \;\; T_j, T_k \in T(A_i), j \ne k \Rightarrow \left[(t_j + S_j - 1) < t_k\right] \vee \left[t_j > (t_k + S_k - 1)\right] \quad (3.3)$$

4. Two tasks of the same kind performed by the same agent should be executed in a time “distance” determined by B.

$$\forall i, j, k \;\; T_j, T_k \in T(A_i), j \ne k, (K_j = K_k) \Rightarrow |(t_j + S_j) - t_k| > B(K_j) \qquad (3.4)$$

5. If a given task T$_i$ has a higher priority than task T$_j$, then T$_i$ should be executed before T$_j$ (at least not later, or in parallel)

$$\forall i, j \middle| \Pr_i > \Pr_j \Rightarrow t_i \le t_j \qquad (3.5)$$

This formulation follows a CSP approach, which is one of the common representations used to describe scheduling problems in the literature [16]. Constraints 1, 2, 3 and 4 should be completely satisfied. However, 5 can be partially reached. The reason is that in a critical environment task priority is not directly related to the resources available for accomplishing this task. That is to say, if there is a victim with high priority in a blocked road, and it is known in advance that there isn't enough time to unblock the road (the victim has not enough life time), then another task (victim) can be assigned to one agent.

3.2 Examples of the general problems in the rescue scenario

In this section snapshot 1 and snapshot 2 described in chapter 1 are formulated according to the notation formulated in the previous section.

3.2.1 Snapshot 1: Ambulance operation

In snapshot 1, the capabilities required to handle rescue victims in the RobocupRescue scenario are the following:

C={rescue, load, unload}

Rescue means unbury one victim; load is the action of carrying the victim to the ambulance team, and unload is the action of leaving the victim in a safe place (refuge).

Regarding the tasks, in snapshot 1, there are:

$$T=\{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}\}$$

Where,

T_1= rescue_victim_2384,
T_2= rescue_victim_2388,
T_3= rescue_victim_2379,
T_4= rescue_victim_2338,
T_5= rescue_victim_2356,
T_6= rescue_victim_2367,
T_7= load_victim_2384,
T_8= load_victim_2388,
T_9= load_victim_2379,
T_{10}= load_victim_2338,
T_{11}= load_victim_2356,
T_{12}= load_victim_2367,
T_{13}= unload_victim_2384,
T_{14}= unload_victim_2388,
T_{15}= unload_victim_2379,
T_{16}= unload_victim_2338,
T_{17}= unload_victim_2356,
T_{18}= unload_victim_2367,

All tasks refer to victim identifiers.

Due date (d_l) is related to the *buriedness* property of the task. This property has an upper bound of 60 and its value is increased by one point in each simulation cycle. So, if a victim arrives at time t_i, with a *buriedness* degree b_i, then in time t_i+1 the victim has a *buriedness* degree b_i+1. Then, the maximum additional time that the victim can remain alive without help is:

$$60-b_i$$

So, the deadline d_i for this task is:

$$d_i=t_i+60-b_i.$$

Then, in snapshot 1, the following deadlines are defined:

$d_1= 45$	$d_7= d_1 + 1 = 46$	$d_{13}= d_7 + 1 = 47$
$d_2=$ due date is over	$d_8=$ due date is over	$d_{14}=$ due date is over
$d_3= 35$	$d_9= d_3 + 1 = 36$	$d_{15}= d_9 + 1 = 37$
$d_4= 55$	$d_{10}= d_4 + 1 = 56$	$d_{16}= d_{10} + 1 = 57$
$d_5= 40$	$d_{11}= d_5 + 1 = 41$	$d_{17}= d_{11} + 1 = 42$
$d_6= 54$	$d_{12}= d_6 + 1 = 55$	$d_{18}= d_{12} + 1 = 56$

Table 3.1. Deadlines for tasks in snapshot 1

First six tasks consist in rescuing victims, the next tasks concern loading victims to the ambulance team and finally, the last tasks consist in unloading victims:

$$K_1= K_2= K_3= K_4= K_5= K_6= \text{rescue\_victim}.$$
$$K_7= K_8= K_9= K_{10}= K_{11}= K_{12}= \text{load\_victim}.$$
$$K_{13}= K_{14}= K_{15}= K_{16}= K_{17}= K_{18}= \text{unload\_victim}.$$

In order to rescue a victim, three abilities are required:

$$C(T_1)= C(T_2)= C(T_3)= C(T_4)= C(T_5)= C(T_6)= \{\text{rescue}\}$$
$$C(T_7)= C(T_8)= C(T_9)= C(T_{10})= C(T_{11})= C(T_{12})= \{\text{load}\}$$
$$C(T_{13})= C(T_{14})= C(T_{15})= C(T_{16})= C(T_{17})= C(T_{18})= \{\text{unload}\}$$

The number of units of each ability depends on the ability. First, the rescue ability depends on the degree of the victim's buriedness. If the buriedness degree is n, then n units are required to rescue the victim. Regarding the first victim, 2384, its degree of buriedness is 25 (see table 1.3 of chapter 1). Then, the number of units for the rescue ability is 25. Second, loading requires one unit (it is a single operation). And third, unloading also requires one unit.

Then, we get:

$U(T_1)=\{25\}$	$U(T_7)=\{1\}$	$U(T_{13})=\{1\}$
$U(T_2)=\{60\}$	$U(T_8)=\{1\}$	$U(T_{14})=\{1\}$
$U(T_3)=\{35\}$	$U(T_9)=\{1\}$	$U(T_{15})=\{1\}$
$U(T_4)=\{15\}$	$U(T_{10})=\{1\}$	$U(T_{16})=\{1\}$
$U(T_5)=\{30\}$	$U(T_{11})=\{1\}$	$U(T_{17})=\{1\}$
$U(T_6)=\{16\}$	$U(T_{12})=\{1\}$	$U(T_{18})=\{1\}$

Table 3.2. U values to snapshot 1

Regarding time slots S_i depends on the solution. For instance, task T_1 requires 25 units, so if 25 agents are assigned to this task then it can be achieved in a single time slot. Of course, there are not 25 ambulance teams available so other solutions will be required. Therefore, this value is assigned dynamically as the solution proceeds.

Priorities between tasks are established depending on two main factors:

- The number of victims placed in the same area, $nv_i \in [1, n]$, n being the total number of victims (tasks).
- The buriedness degree of the victim, $b_i \in [0,60]$

Then,

$$\Pr_i = (0.7\frac{b_i}{60} + 0.3\frac{nv_i}{n}) * 10 \qquad (3.6)$$

According to the example in snapshot 1 in which there are two victims, 2338 and 2356, in position 2098 and in the remaining positions there is only one victim, then, the corresponding priorities are:

$Pr_1= 2.9$	$Pr_7= Pr_1= 2.9$	$Pr_{13}= Pr_7= Pr_1= 2.9$
$Pr_2=$ is not calculated because the due date is up	$Pr_8=$ is not calculated because the due date is up	$Pr_{14}=$ is not calculated because the due date is up
$Pr_3= 4.1$	$Pr_9= Pr_3= 4.1$	$Pr_{15}= Pr_9= Pr_3= 4.1$
$Pr_4= 1.8$	$Pr_{10}= Pr_4= 1.8$	$Pr_{16}= Pr_{10}= Pr_4= 1.8$
$Pr_5= 3.6$	$Pr_{11}= Pr_5= 3.6$	$Pr_{17}= Pr_{11}= Pr_5= 3.6$
$Pr_6= 1.9$	$Pr_{12}= Pr_6= 1.9$	$Pr_{18}= Pr_{12}= Pr_6= 1.9$

Table 3.3. Priorities of tasks to snapshot 1

Concerning precedence, it is clear that one ambulance team cannot load a victim until he/she has been unburied. Then, in snapshot 1 we get an 18x18 preference matrix, with the following contents (see figure 3.2):

$$P=\left(\begin{array}{ccccccccccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

Figure 3.2. Precedence matrix.

Concerning agents, all agents in snapshot 1 have the same capabilities, since all of them are ambulance teams. Thus, the available free agents are, according to table 2 of chapter 1, the following:

$$Av(T_1)= Av(T_2)= Av(T_3)= Av(T_4)= Av(T_5)= Av(T_6)= \{2400, 2402\}.$$

Then, the set of agents A is composed by A= {2400, 2402}. The capabilities of each agent are:

$$C(2400)=C(2402)=\{rescue, load, unload\}.$$

The task to be performed by each agent, $A(T_i)$ should be determined (the solution of the problem).

Regarding time distances B, there are some restrictions concerning loading the victim. That is, when one agent has carried a victim, it needs a time slot to unload the victim in order to proceed with another activity. Then,

$$B(rescue) = 0,$$
$$B(load)=1,$$
$$B(unload)=0.$$

Finally, regarding the solution of the problem, we need to determine the starting times $t_1, t_2, \ldots t_6$ and the assigned agents $Ag(T_1), Ag(T_2), \ldots Ag(T_6)$.

3.2.2 Snapshot 2: Fire fighting operation

Now, we give examples for snapshot 2. The capabilities required to extinguish fires are the following:

C={extinguish}

Regarding the tasks, there are five fires that have been discovered:

$$T=\{319, 1230, 1900, 2500, 3829\}$$

Being $T_1=319$, $T_2=1230$, $T_3=1900$, $T_4=2500$, $T_5=3829$. All tasks refer to fire extinguishing.

The due date (d_1) is related to the *fieriness* property of the task. This property has an upper bound of 100 and its value is increased by one point in each simulation cycle. So, if a fire is discovered at time t_i, with a *fieriness* degree f_i, then in time t_i+1 it has a *fieriness* degree of f_i+1. Thus, the maximum additional time that the fire can remain without the building being burnt down is:

$$100-f_i$$

So, the deadline d_i for this task is:

$$d_i=t_i+100-f_i.$$

Then, in snapshot 2, the following deadlines are defined:

$d_1= 60$
$d_2= 45$
$d_3= 100$
$d_4= 105$
$d_5=$ the due date is over.

The kind of task involved in this scenario is ”extinguish”, so:

$$K_1= K_2= K_3= K_4= K_5= K_6=\{extinguish\}$$

The abilities required of agents in each task are the following:

$$C(T_1)= C(T_2)= C(T_3)= C(T_4)= C(T_5)= C(T_6)= \{extinguish\}$$

The number of units of the extinguishing ability depends on the fieriness degree. If the fieriness degree is n, then n units are required to extinguish the fire. Then, in the snapshot 1 scenario we have:

$U(T_1)=\{50\}$
$U(T_2)=\{65\}$
$U(T_3)=\{10\}$
$U(T_4)=\{5\}$
$U(T_5)=\{102\}$

Regarding time slots S_i, depends on the solution, as has previously stated. For instance, task T_1 requires 50 units, so if 50 fire brigade agents are assigned to this task, then it can be achieved in a single time slot. Of course, this is an infeasible solution. This value is assigned dynamically.

Priorities among tasks in this snapshot depend on two main factors:

- The number of victims located in the same area, $nv_i \in [1, n]$, n being the total number of victims (tasks). A fire in which civilians are involved has a higher priority than another fire in which there are no civilians.

- The fieriness degree of the victim, $f_i \in [0,100]$

Then,

1. There are some fires in which victims are involved:

$$\Pr_i = \frac{nv_i}{n} * 10 \qquad (3.7)$$

2. There are no fires in which victims are involved

$$\Pr_i = \frac{f_i}{100} * 10 \qquad (3.8)$$

Let's assume that there are no victims involved in any of the fires of snapshot 2 (no information about them are provided). Then, the following priorities are computed:

Pr_1= 5
Pr_2= 6.5
Pr_3= 1
Pr_4= 0.5
Pr_5= is not calculated because the due date is up.

There are no precedence relations in P to be defined.

Concerning agents, all agents in snapshot 2 have the same capabilities, since all of them are fire brigades. Thus, the available agents are, according to table 4 of chapter 1, the following:

$Av(T_1)= Av(T_2)= Av(T_3)= Av(T_4)= Av(T_5)=$ {2415, 2417, 2420, 2421, 2423}.

The set of agents is then formed by:

A= {2415, 2417, 2420, 2421, 2423}

The agents' capabilities are the following:

C(2415) = C(2417) = C(2420) = C(2421) = C(2423) = extinguish[1].

Regarding time distances, B(Ki), how much water is in the fire brigades' tank should be taken into account. However, this issue is implemented in version 0.41 of the Robocup

[1] Some capabilities are related to the *stretchedLenght* which is the property that shows how long the hose is pulled to the nearest, however, in the current version of the Robocup rescue, this particularity is not yet taken into account.

Rescue, and we are still dealing with version 0.40. So we deal with this constraint in further work. Therefore, with the current configuration, we get:

$$B(\text{extinguish}) = 0.$$

Finally, regarding the solution of the problem, we have to determine the starting times $t_1, t_2, \ldots t_5$ and the assigned agents $Ag(T_1), Ag(T_2), \ldots Ag(T_5)$.

The tasks to be assigned to each agent, $A(T_i)$, are considered to be the solution of the problem too.

3.3 Multi-Agent Architecture.

In order to test a coordination mechanism for the rescue simulator scenario, we have developed a MAS architecture which we have implemented the Girona Eagles rescue system [65]. In this section we first explain the organisational structure of the agents, and we proceed with the details of the architecture defined for the moving agents, and the fixed agents.

3.3.1 Organisational structure

In order to coordinate the activities among the different agents involved in the rescue scenario, we propose the construction of a MAS such as the one shown in figure 3.3.

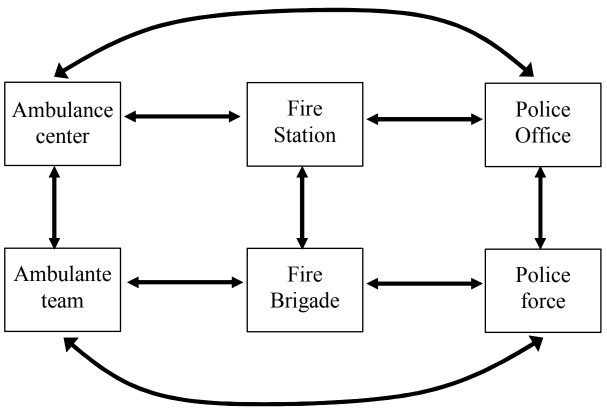


Figure 3.3. MAS organization

The organisation corresponds to the information flow concerning agents in the rescue scenario. The role of the moving agents is to gather information about tasks (position), and the role of the fixed agents is to pass on this information between them.

For instance, the ambulance teams keep the ambulance station informed about their condition: hp, damage, position, buriedness, availability and goal. The first four types of data have already been described in section 1.2. *Availability* means the current activity being carried out by the agent: it is "busy", if the ambulance team is trying to rescue a civilian; "free" if the ambulance team is looking for civilians; and "blocked" if the ambulance team cannot perform the task it has been assigned because of blocked roads. Finally, the *goal* descriptor indicates the current target of the ambulance team, i.e. the identification of the civilian that it is trying to rescue. On the other hand, the police force and fire brigade agents gather information about buried or damaged civilians. This information is passed on to the ambulance centre that is in charged of taking decisions about which victim to rescue. Finally, the decision is communicated to the ambulance teams (see figure 3.4).

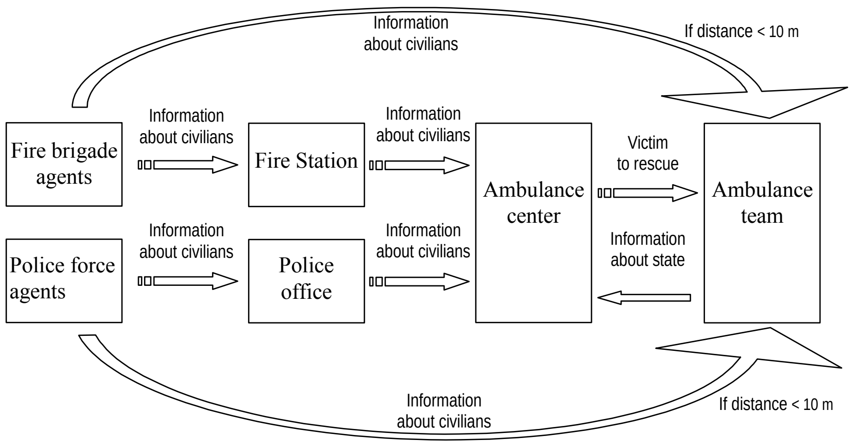


Figure 3.4. Strategy for the communication flow through the ambulance centre.

In the same way, the fire brigades and police forces deal with the fire station and the police office, respectively.

This organisation underlies the importance of cooperation between heterogeneous agents in order to find tasks. Since there is no agent with a global view of the scenario and no agent knows about civilians, fires and blocked roads in advance, if there is no cooperation rescue agents will move without goals around the simulation scenario.

To check the importance of this communication flow, we performed three experiments:

- No coordination: that is, there was not communication at all between agents. The results showed that in this case ambulance teams get lost in the rescue scenario and cannot find victims that need to be rescued.
- Coordination between homogeneous agents: that is, communication between agents of the same kind (between ambulance teams and the ambulance station, between fire brigades and the fire station, and between police forces and the police station). The results improved and two civilians were rescued. One ambulance close to a group of victims was able to receive help from another ambulance and rescue civilian agents.
- Coordination between heterogeneous agents, according to the strategy presented above. Results improved even more, since many more victim positions were known, and they could be rescued. Given the evaluation equation provided by Robocup Rescue Organization [46], and using the MCDM technique explained

in section 3.5, our score was 38. The best score obtained in RoboCup Rescue 2002 was 90 by Arian; the second 87 by YowAI2002; the third one was 46 by NITRescue; and fourth was 34 by Kures2002; out of 10 teams that participated in the 2002 tournament [46].

This study shows the importance of ambulance team coordination, although the remarkable impact of cooperation between the heterogeneous agents was also made clear by the simulation process results.

3.3.2 Moving agents architecture

The architecture designed for the Girona Eagles' moving agents is the one shown in Figure 3.5. In the architecture three main blocks can be distinguished: perception, decision and action.

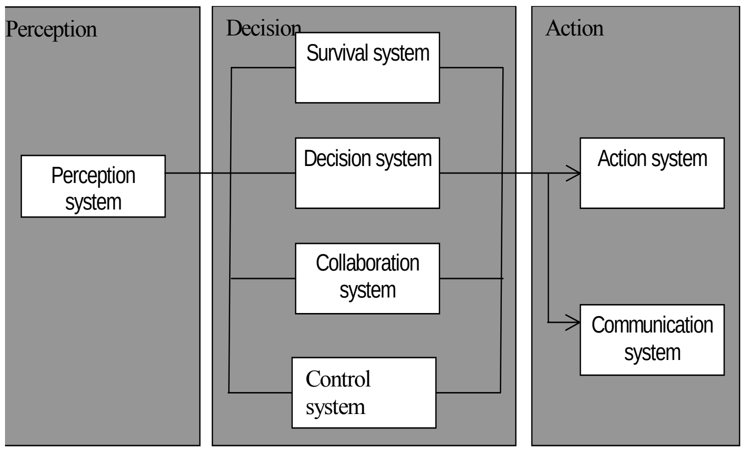


Figure 3.5. The moving agents' structure.

The *perception system* is in the perception block, which is in charge of modelling the surrounding world, from the local information perceived within a 10 m radius and the information received from other agents.

The decision block consists of four modules: the survival system, the decision system, the collaboration system and the control system. The *survival system* tries to guarantee that rescue agents will not die when performing their assigned tasks. For this reason it is important to take into account the degree of damage to the rescue agent and control this aspect. If any rescue agent is severely injured, they should be moved to a refuge. The *decision system* provides the moving rescue agents with decisions considering the surrounding context and its constraints. The hp (measure of agents vitality), damage and water availability (fire brigade agents) are examples of constraints that should be taken into account, due to their direct influence on rescue actions. The aim of the *collaboration system* is to achieve the rescue goals, i.e. all moving agents behaving in a co-operative way. In particular, the police force and fire brigade agents should cooperate providing the ambulance station with information about victims found on the road. Using this information, ambulance teams can rescue civilians. Finally, the *control system* gives priority to the task to be performed. For instance, the survival decision becomes a priority over other decisions made in the remaining modules of the decision block.

The action block is made up of two modules: the action system and the communication system. The *action system* translates the agent's decision into commands that can be interpreted by the simulator. The *communication system* is important considering the constraints imposed by the simulator.

3.3.3 Fixed agents architecture

The architecture for fixed agents in the Girona Eagles team is shown in Figure 3.6. Again, we distinguish three main blocks: perception, decision and action. The main difference from the moving agent architecture is the absence of a survival module in the decision block.

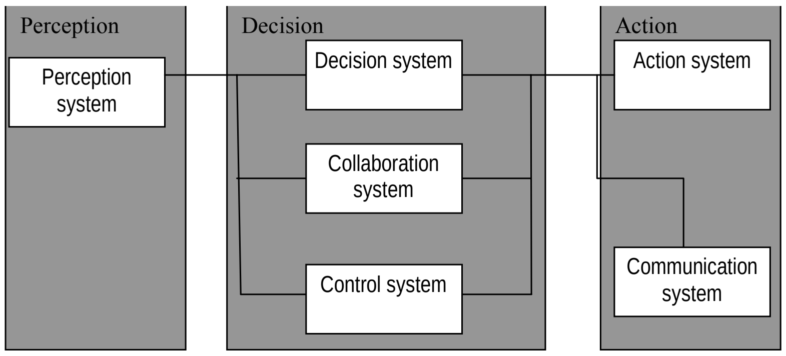


Figure 3.6. Central agent architecture

In our approach fixed agents were used as coordination centres for every agent team. For instance, the ambulance station takes decisions about resource allocation to rescue civilians based on two different types of information - firstly, information about its own team agents (damage, hp of ambulance teams), and secondly, information regarding injured or buried civilians.

3.4 The case-based reasoning and fuzzy filter approach

Our work starts from a national project in which CBR and Fuzzy filters were proposed to deal with agents' decision making processes in the Robocup rescue scenario. In this section we explain our attempts to develop such mechanisms.

3.4.1 Case-based local reasoning

Case-based reasoning (CBR) is a problem solving methodology in AI coupled to learning [74]. The learning facet of the methodology makes it especially suitable for dynamic environments in which agent behaviour should be adapted to environmental changes. The task of CBR can be divided into four phases: Retrieve, Reuse, Revise and Retain. When a new problem arrives, similar cases are retrieved from the memory.

Then, the information and knowledge of retrieved cases are used to solve the problem. Finally, the proposed solution is revised and retained, as it is likely that it will be useful for future problem solving [73].

In our approach cases store information about each agent decision. Components of the case are, consistently:

- Goal of the agent, for example, to extinguish the fire of building 23.
- Unsolved petitions: messages from other agents asking for help that are unattended, together with the trust value of the emitter agent.
- The agent's physical capabilities, that vary through time:
    - *vision*: is a variable which determines the capability to perceive visual information from the surroundings.
    - *hp*: is a indication of life. The initial value is 10000. The value is decreased by damage received in every step. A value of ´0´ means death.
    - *damage*: The damage value is set when the a civilian suffers from fire or building collapses. It goes from 0, when the victim is safe in a refuge, to 200 in the case of building collapses.
- Context: information related to the agent's scope, degree of buildings on fire, distance from the support teams, closeness to a victim, life expectancy of the victim, etc. Most of this information is gathered by the visual capability of the agent and also by the information provided by other agents.
- Solution: action the agent has decided on (including, leaving the current goal and attending other agent petitions).
- Result: evaluation of the action performed (leading to success or failure).

Each team agent sends back the decision made on a simulator cycle (case) to its central; central agents gather team agent decisions (cases) and make decisions regarding agent distribution around the rescue scenario.

Retrieval is based on the goal, unsolved petitions, agent capabilities and context components of the case base. Reuse consists in transferring the solution. In the next simulation step, the result of past experience (expected result) is compared with the current result. If the result differs from the expected one, the new case is retained.

3.4.2 Fuzzy logic Systems and Fuzzy Filters

Due to the problem presented above, agents should know the reliability of all messages sent by their partners. For example, one agent can send help messages to their colleagues but their damaged situation does not guarantee that the help will arrive on time.

One way to get a trust value for each agent is by means of a fuzzy filter according to the work proposed in [62] (See figure 3.7). A fuzzy filter is defined as a special case of the Mandami fuzzy inference systems in which fuzzy rules have the following form:

$$if \quad A_1 \text{ is } S_1 \text{ and} \ldots\text{and } A_n \text{ is } S_n \text{ and } V \text{ is } L_1 \text{ then } W \text{ is } L_2$$

Where:

- S1, S2, ...Sn are linguistic values, defined by fuzzy sets in universes of discourse X1, X2... Xn respectively.
- A1, A2... An are fuzzy variables taking values over the fuzzy power sets of X1, X2... Xn.
- L1, and L2 are linguistic values, defined by fuzzy sets in the same universe of discourse U.
- V and W are fuzzy variables taking values over the fuzzy power set of U.

Thus, an example of basic fuzzy rule implementation in order to determine the message trust value sent by a fireman agent, taking into account its physical capabilities is:

If *vision* is *diminished* and *hp* is > *5000* and *damage* is < *80* then *trust* is *high*

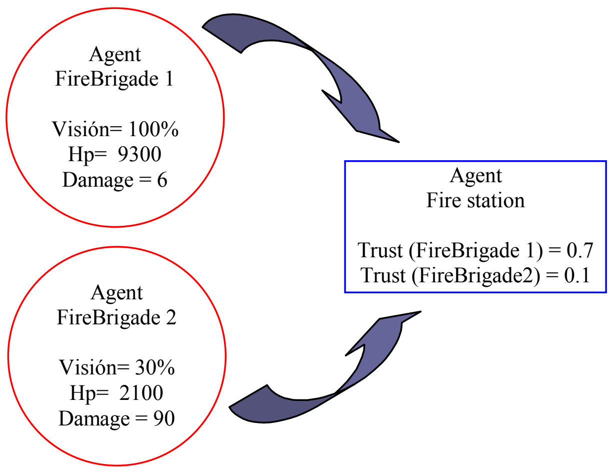


Figure 3.7. Trust filtering according to the agent's capabilities

Then, each agent has fuzzy rules to compute the trust value of the agents in its neighbourhood. This trust value is a feature on which CBR retrieval relies in order to recover past experiences to decide on the next action (see section 3.4.1).

3.4.3 Discussion

In order to test the feasibility of our approach we have performed a communication study to find out the amount of information gathered by each agent and the decision making process that the agent should do. From this study we realised that our approach presents serious difficulties regarding learning by the agents [62]. In the first instants of the simulation there is only a small number of cases gathered by the central, for this reason the agents don't have enough information of past cases in order to perform the reuse phase of CBR. On the other hand, in the rescue scenario, the cases gathered by the central are sparsely, that is, are quite different among them. For this reason, it has no sense to use a CBR approach, in which the core is on similarity measures.

3.5 Application of Multiple criteria decision making methods (MCDM)

Our next attempt to deal with the rescue coordinating problem was by means of the use multiple criteria decision making methods because the rescue scenario in our approach refers to making decisions in the presence of multiple, usually conflicting criteria. For instance, the rescue agents present several characteristics (objectives, attributes, or goals) such as *hp*, *damage*, *buriedness, localization,* and others. Then, when the agent executes a task, their performance is influenced by these characteristics and for this reason these factors should be taken into account, both intrinsically and extrinsically, in the decision making process.

A Multiple Criteria Decision Making (MCD) method is characterized by the different alternatives which a decision should take into account, and the set of criteria used for the decision. For our purposes we need to apply the MCD method twice (see Figure 3.8). First, to know the priority of the different activities to be performed, namely, the order in which civilian agents should be rescued. Second, to select the resource to be assigned to the most urgent activities.

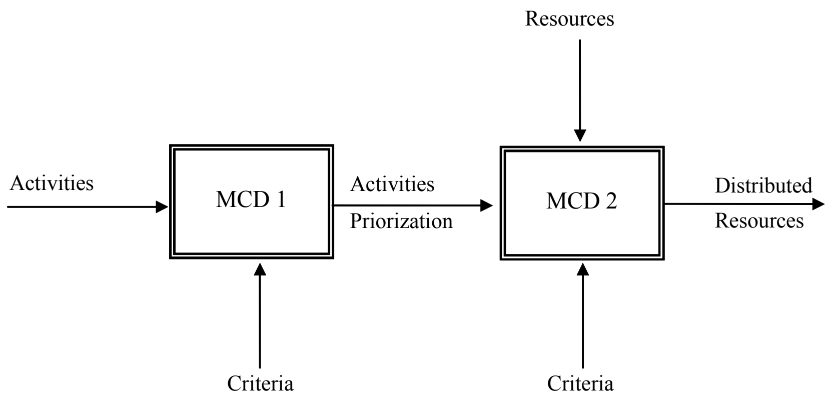


Figure 3.8. MCD method application.

3.5.1 Activity prioritisation

At this stage, the different rescue activities are the alternatives of the MCD method. Activities are featured by their hp, damage, position, buriedness and the number of victims at the same position. The criteria involved in the decision are the different features that constrain the activities, namely, the hp, damage, buriedness and number of victims, all of them related to the life expectancy of the civil agents to be rescued.

The MCDM procedure is based on two main steps:

1. Rating of the different alternatives according to the different decision criteria.

For each alternative A_i, and each criteria C_j, a value V_{ij} is assigned. This V_{ij} is normalized in [0,1]

2. Rating of the different alternatives according to the importance of each decision criteria. One possible way of rating the alternative is by using aggregation operators, such as the WA [85].

   So, for each alternative Ai, the corresponding rating Ri is computed as:

$$R_i = \sum_{j=1}^{m} V_{ij} W_j \qquad (3.9)$$

Where m is the amount of criteria, and W_j the weight associated to the criteria C_j.

In the rescue problem, the importance of the different criteria has been established as follows:

Damage: W=0.9
Hp: W=0.7
Number of victim in current position: W= 0.6
Buriedness: W = 0.5

These weights are used at the rating stage of the MCDM procedure. The weight assignation is based on the importance of each of the criteria. For instance, if a civil agent has a low hp, it means that it is urgent to rescue them. Therefore, they should be the first civilian to be rescued.

3.5.2 Resource distribution

Now, the different alternatives are the resources available (rescue agents), characterized by their identification, hp, damage, position, buriedness, availability and goal. The criteria from which the MCD method makes the decision is based on these features, namely: hp, damage, and distance to the closest prioritised activity.

Regarding the criteria, we have used the following:

Hp: W = 0.9
Damage: W = 0.7
Distance: W = 0.6

It seems clear that the rescue agent should first have enough life expectancy (hp) in order to perform rescue activities; second, it shouldn't be damaged; and third, the closest activity is preferred.

The rating is now applied in inverse order. That is, the alternative with the minimum value is ranked first.

3.5.3 Example

To illustrate the MCDM process with an example, let us suppose that the current information about victims at the ambulance station is what is shown in the Table below. At the rating stage we thus obtain the following normalized values for each alternative:

Alternatives (Victims Id)	Criteria			
	hp	Damage	Buriedness	No. victims
2384	0,08	0,17	0,21	0,5
2388	0,21	0,21	0,5	0,5
2379	0,4	0,26	0,29	0,5
2338	0,1	0,11	0,13	1
2356	0,15	0,16	0,25	1
2367	0,24	0,22	0,13	0,5

Table 3.4. Information about victims at the ambulance stations

And the ordered results (ranking) are the following:

Alternatives (Victims Id)	Value criteria * weight				Result
	Hp * weight	Damage * weight	Buriedness * weight	No. Victims * weight	
2356	0,15 * 0,7	0,16 * 0,9	0,25 * 0,5	1 * 0,6	0,974
2379	0,4 * 0,7	0,26 * 0,9	0,29 * 0,5	0,5 * 0,6	0,959
2388	0,21 * 0,7	0,21 * 0,9	0,5 * 0,5	0,5 * 0,6	0,886
2338	0,1 * 0,7	0,11 * 0,9	0,13 * 0,5	1 * 0,6	0,834
2367	0,24 * 0,7	0,22 * 0,9	0,13 * 0,5	0,5 * 0,6	0,731
2384	0,08 * 0,7	0,17 * 0,9	0,21 * 0,5	0,5 * 0,6	0,614

Table 3.5. Ranking of the alternatives (victims)

The table shows an ordered list of civilians to be rescued. The highest ranked victim is the one whose identification number (id) is 2356, so this one will be rescued first.

The ambulance central decides which ambulance team will perform the rescue activity. Suppose that the available information on ambulance teams is the following:

Id	Availability	hp	Damage	Distance
2399	free	0	0	0,1392
2400	free	0	0	0,1197
2402	free	0,1	0.02	0,248

Table 3.6. Information about ambulance teams

The ordered results (ranking) are the following:

Alternatives (Victims Id)	Value criteria * weight			Result
	Hp * weight	Damage * weight	Distance * weight	
2400	0	0	0,1197 * 0,6	0,07182
2399	0	0	0,1392 * 0,6	0,08352
2402	0,1 * 0,9	0,02 * 0,7	0,248 * 0,6	0,2528

Table 3.7. Ranking of alternatives (ambulance teams)

The table shows an ordered list of available ambulance teams. The highest ranked ambulance team is the one whose identification number (id) is 2400, so it is requested to rescue the victim 2356.

The resource distribution step is applied as many times as needed, in order to assign a prioritised activity to each free resource.

3.5.4 Discussion

We have implemented this approach for the ambulance team and central [Ref 69, 70]. In particular, the MCDM was applied in the decision making block of the ambulance centre (see figure 3.3) and [63].

Given the evaluation equation provided by the Robocup Rescue Organization [46], our score is 38. The best score obtained in RoboCup Rescue 2002 was 90 by Arian; the second 87 by YowAI2002; the third one was 46 by NITRescue; and fourth was 34 by Kures2002; from among the 10 teams that participated in the 2002 tournament [46].

Multiple criteria decision making has several advantages, such as: the possibility of incorporating differences into interest views. MCDM also allows a reduction of the available information, which allows the reduction of the number of alternatives to be considered.

However, it also has several disadvantages:

- The WA operator of the aggregation applied is a simple method. For this reason, it is necessary to explore another methods that allow the decision making process to improve for the agents in the rescue scenario.

- In this approach we need to apply the MCDM method twice. Firstly, to find out the priority of the different activities to be performed, namely, the order in which civilian agents should be rescued. Secondly, to select the resource to be assigned to the most urgent activities. For this duality, the computational cost of using the method proposed is incremented.

- Taking into account that the responsibilities of moving rescue agents in this approach are based on passing information about themselves and information about victims to the centrals, the decision making process is completely performed by the central agents. For this reason this is a centralized approach, and probably a distributed coordination approach can give better results.

3.6 Task allocation using Combinatorial Auction

At this point of the research, we realised that classical approaches such as CBR and MCDM were hard to follow in distributed problems like the one we want to solve. Then looking at the MAS literature, we saw the importance of market mechanisms for coordination issues in MAS. We were particularly interested in combinatorial auctions as an alternative method for agent coordination in distributed environments.

Combinatorial auctions [70]:

- Computes high-quality allocation of bundles of goods,
- Agents do not receive undesirable partial bundles.

Although the general problem of winner determination is NP-complete, it has been experimentally shown that it can solve many large problems quickly [2] and converge to a solution [54, 70]. In particular, when the bid space is sparse, the problem can be solved with a polynomial time [50], as explained in chapter 2. This is precisely the kind of space we are dealing with in the RobocupRescue scenario. We can even have more than one hundred tasks, experiments show that few task are auctioned at a time, due to the lack of information regarding the rest of the tasks (localisation of victims, fires, etc.). It is along the simulation evolution when tasks are found and then auctioned.

3.6.1 Rescue formulation as a combinatorial auction

In this approach the central agents (fire station, ambulance centre and office police) are the auctioneers and the rescue teams (fire brigade, ambulance team and police force) are the bidders. The bid items are the tasks and the final objective is to obtain the maximum benefit for the whole system.

At the beginning there are no rescue operations to be performed, since the agents are just exploring the situation in their surrounding area. As they discover victims, fires and blocked roads, the central agent receives the information about these tasks. This communication strategy is explained in section 3.3.

When agent stations have information on new tasks, they start the combinatorial auction process. Rescue agents send their bids, which correspond to combinations of tasks to be performed in sequential order, to the central agents. The rescue agents select each activity, taking into account the distance of their location and the place where these are required. Only the activities available in the agents' perception area are taken into account in the bids. In this approach, the rescue agents initially send combinations of the nearest places where it is necessary to perform any task.

The cost of performing these activities is also included in the bid. This is computed following equation 3.10, below:

$$\text{Cost estimate} = \sum_i f(Dis, CPT) \qquad (3.10)$$

$$f(Dis, CPT) = Dis * CPT \qquad (3.11)$$

where:

- i = Number of tasks to be performed sequentially.
- *Dis*= Distance from the agent location to the place where the task will be performed.
- *CPT*= Cost of Performing the Task.

For instance, the CPT of the police agent is the cost required for clearing the road that is provided by the Robocup Rescue simulator (property *repairCost* of the blocked road); the CPT of fire brigades is the degree of fieriness (spelt fieryness in the simulator) that specifies how much a building is burning. If the estimated cost of equation 3.10 is greater than the current agent's property (hp, damage) and capabilities (i.e., water quantity) the bid cost is set to ∞.

The *f(Dis,CPT)=Dis*CPT* function was defined because both the distance and the cost of performing the task are crucial factors in deciding the overall cost.

All bids received by the central station are processed using the *winner determination algorithm* explained in the following section. Tasks are selected by ensuring that sets conform to the maximum number of tasks and the minimum cost. That is, winner determination consists in finding the solution that minimizes the following:

$$\min_{W \in A} \sum_{S \in W} \bar{b}(S) \qquad (3.12)$$

where

$$\bar{b}(S) = \min_{i \in bidders} b_i(S) \qquad (3.13)$$

Among the different algorithms of the state of the art that solve the winner determination problem, we have applied the Sandholm algorithm and CASS, due to their properties explained in chapter 2.

Such algorithms can be extended to deal with the rescue problem, and in section 3.6.3, 3.6.4 and 3.6.5 we introduce some new preliminary alternatives.

3.6.2 Application of Sandholms' Algorithm

In this section, we explain our approach using the winner determination algorithm proposed by Sandholm [53] in the rescue scenario which has been explained in section 2.5.2.1.

Next, an example of the application of the algorithm is presented.

3.6.2.1 Example

Let's assume that the fire station knows about four fires in progress in the disaster scenario, identified by 319, 1230, 2500 and 3829; and that there are three fire fighting teams bidding for them (see Figure 3.9). The set of items to be auctioned is therefore, M={319, 1230, 2500, 3829}. These are the tasks to be performed by the fire brigades.

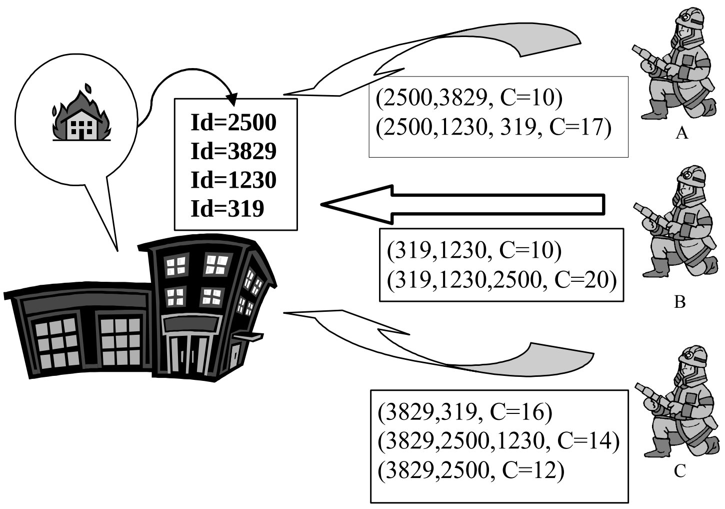


Figure 3.9. Fire fighter agents send bids to the fire station

After announcing the tasks, the bids gathered by the fire station are the following:

From agent A:

B_1=<S_1, $b_A(S_1)$)>

S_1 = {2500, 3829} $b_A(S_1)$ = 10

B_2=<S_2, $b_A(S_2)$)>

S_2 = {2500, 1230,319} $b_A(S_2)$ = 17

From agent B:

B_3=<S_3, $b_B(S_3)$)>

S_3 = {319, 1230} $b_B(S_3)$ = 10

B_4=<S_4, $b_B(S_4)$)>

S_4 = {2500} $b_B(S_4)$ = 20

From agent C:

B_5=<S_5, $b_C(S_5)$)>

S_5 = {3829,319} {$b_C(S_5)$ = 16

B_6=<S_6, $b_C(S_6)$)>

S_6 = {3829,2500,1230} $b_C(S_6)$ = 14

B_7=<S_7, $b_C(S_7)$)>

S_7 = {3829,2500} $b_C(S_7)$ = 12

First of all, we re-order the different items in each combinatorial auction S, resulting in the new set of bids shown in the following table.

Bid	S	b
B_1	{2500,3829}	10
B_2	{319,1230,2500}	17
B_3	{319,1230}	10
B_4	{2500}	20
B_5	{319,3829}	16
B_6	{1230,2500,3829}	14
B_7	{2500,3829}	12

Table 3.8. The set of bids

We can see that two combinations of items are identical, S_1 and S_7, so the more expensive one S_7 is removed in line with equation (3.13). The set of bids that make up the winner determination problem is now as follows:

Bid	S	$\bar{b}$
B_1	{2500,3829}	10
B_2	{319,1230,2500}	17
B_3	{319,1230}	10
B_4	{2500}	20
B_5	{319, 3829}	16
B_6	{1230,2500,3829}	14

Table 3.9. The set of bids for winner determination problem

This is the set of bids that is submitted to the winner determination algorithm, being $W = \{S_1, S_2, S_3, S_4, S_5, S_6\}$.

The search space corresponding to the current data is as follows (Figure 3.10):

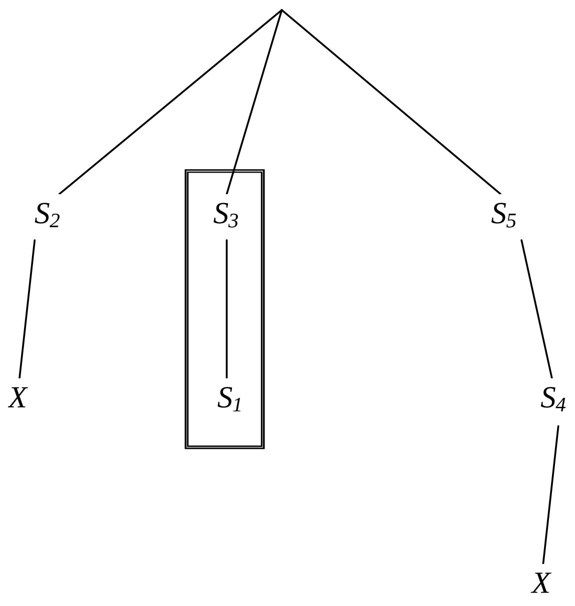


Figure 3.10. Search space example

Note that the solution is the path $S_3 \cup S_1$. When applying a heuristic search, S3 is selected as the first node to be expanded since it is the bid with the lowest price. The search tree that is finally generated is shown below (Figure 3.11), obtaining the solution $S_3 \cup S_1$:

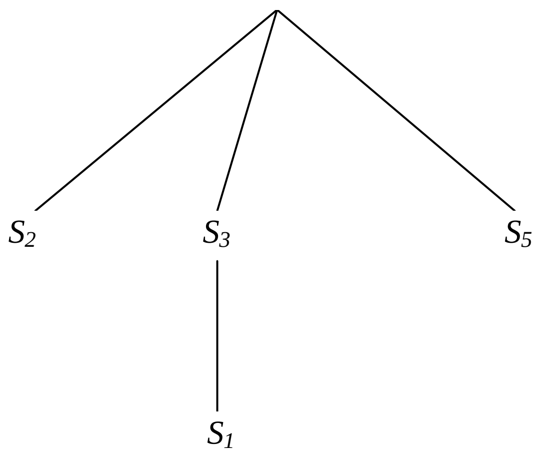


Figure 3.11. Search tree example

3.6.2.2 Discussion

We have studied the Sandholm algorithm used for decision making in the centrals. The complexity of the approach, however, has lead us to experiment only with particular examples.

The main problems presented in this approach are:

1. *A Continuous double auctions philosophy should to be used:* Due to the dynamic characteristics of the scenario of our study, the rescue agents send the bids for tasks in at any time, and the combinatorial auction process needs to be repeated until all the tasks are allocated and no new tasks arrive. Sandholm presents some variations to the algorithm proposed here, in order to deal with such an incremental auction

2. *Precedence relations are convenient:* In the problem formulation (section 3.1.1), the precedence constraints are presented as follows:

   $$\forall i, j(P_{ij} = 1) \Rightarrow (t_i + S_i - 1) < t_j$$

   That is, precedence among rescue tasks must be taken into account. For instance, police force agents must clear blocked roads to allow the ambulance teams to pass on the roads leading to the victims. Such constraints are not taken into account in this approach.

3. *Pre-emption methods should be explored:* Once the tasks are assigned to the agents, they can suffer damage or even die. For this reason, pre-emption methods should be used to assign substitute agents.

3.6.3 Combinatorial auction structured search (CASS)

The best algorithm known to solve the winner determination problem is currently the CASS algorithm, explained in section 2.5.2.2. We have used it in our domain [66]. Next we explain the application of CASS to the rescue operation.

3.6.3.1 Example

In this example, we refer to snapshot 2 of the scenario in section 1.1.4. After announcing the tasks, the bids gathered by the fire station are the following:

From agent A:
$b_1=(s(b_1), p(b_1))$
$s(b_1) = \{319, 1230\}$ $p(b_1) = 10$
$b_2=(s(b_2), p(b_2))$
$s(b_2) = \{319, 1230, 1900\}$ $p(b_2) = 20$
$b_3=(s(b_3), p(b_3))$
$s(b_3) = \{319\}$ $p(b_3) = 6$

From agent B:
$b_4=(s(b_4), p(b_4))$
$s(b_4) = \{2500, 3829\}$ $p(b_4) = 12$
$b_5=(s(b_5), p(b_5))$
$s(b_5) = \{2500\}$ $p(b_5) = 7$

From agent C:
$b_6=(s(b_6), p(b_6))$
$s(b_6) = \{1230, 1900, 2500\}$ $p(b_6) = 17$
$b_7=(s(b_7), p(b_7))$
$s(b_7) = \{1230, 1900\}$ $p(b_7) = 12$
$b_8=(s(b_8), p(b_8))$
$s(b_8) = \{1230\}$ $p(b_8) = 6$

From agent D:
$b_9=(s(b_9), p(b_9))$
$s(b_9) = \{1900, 2500\}$ $p(b_9) = 15$
$b_{10}=(s(b_{10}), p(b_{10}))$
$s(b_{10}) = \{1900\}$ $p(b_{10}) = 9$

From agent E:
$b_{11}=(s(b_{11}), p(b_{11}))$
$s(b_{11}) = \{2500, 3829\}$ $p(b_{11}) = 10$
$b_{12}=(s(b_{12}), p(b_{12}))$
$s(b_{12}) = \{3829\}$ $p(b_{12}) = 12$

We can see that two combinations of items are identical, $s(b_4)$ and $s(b_{11})$, therefore the more expensive one $s(b_4)$ is removed by step 1 of the algorithm. The set of bids that make up the winner determination problem is presented in table 3.10:

Bid b_i	$s(b_i)$	$p(b_i)$
b_1	{319,1230}	10
b_2	{319,1230,1900}	20
b_3	{319}	6
b_5	{2500}	7
b_6	{1230,1900,2500}	17
b_7	{1230,1900}	12
b_8	{1230}	6
b_9	{1900,2500}	15
b_{10}	{1900}	9
b_{11}	{2500,3829}	10
b_{12}	{3829}	12

Table 3.10: Set of bids submitted to step 2 of the algorithm.

This is the set of bids that is submitted to step 2 of the winner determination algorithm. Following and using the algorithm, the order of the goods is determined by using the heuristic 1:

Good 319 is requested by bids b1, b2, and b3. Then $numgoods_{319}$=3. Bid b1 requests 2 goods, while b2 requests 3 and b3, 1. Then, the average number of goods requested by these bids is:

$$avggoods_{319} = \frac{1+2+3}{3} = 2$$

According to equation (2.4) in section 2.5.2.2, the corresponding score is:

$$score_{319} = \frac{numgoods_{319}}{avggoods_{319}} = \frac{3}{2} = 1.5$$

In the same way, we computed the score for the remaining goods:

good	requested by	Numgoods	avggoods	score
319	b1,b2,b3	3	2	1.5
1230	b1,b2,b6,b7,b8	5	2.2	2.27
1900	b2,b6,b7,b9,b10	5	2.2	2.27
2500	b5,b6,b9,b11	4	2	2
3829	B11,b12	2	1.5	1.3

The highest scores are for goods 1230 and 1900. We choose 1230 following a lexicographic order. The remaining lists are for goods 319, 2500, 1900 and 3829.

Now, we recalculate the scores for the remaining lists:

good	requested by	numgoods	avggoods	score
319	b1,b2,b3	3	2	1.5
1900	b2, b9,b10	3	2	1.5
2500	b5,b9,b11	3	1,6	1.8
3829	b11,b12	2	1.5	1.3

The highest score is for good 2500. The remaining lists are for goods 319, 1900 and 3829. We recalculate its corresponding scores:

good	requested by	numgoods	avggoods	score
319	b1,b2,b3	3	2	1.5
1900	b2, b9,b10	3	2	1.5
3829	b12	1	1	1

The highest scores are for goods 319 and 1900. We choose 319 following an alphabetical criterion. The remaining lists are for goods 1900 and 3829. We recalculate its corresponding scores:

good	requested by	numgoods	avggoods	score
1900	b9,b10	2	1.5	1.3
3829	b12	1	1	1

The highest score is for good 1900. The final decreasing order of the lists according to heuristics 1 is the following:

D1	D2	D3	D4	D5
b6, s(b6)=(1230,1900,2500) b7, s(b7)=(1230,1900) b8, s(b8)=(1230)	b5,s(b5)=(2500) b11, s(b11)=(2500,3829)	b1,s(b1)=(319,1230) b2,s(b2)=(319,1230,1900) b3,s(b3)=(319)	b9,s(b9)=(1900,2500) b10,s(b10)=(1900)	b12,s(b12)=(3829)

Table 3.11. Final decreasing order of the lists according to heuristic 1

Now, the order of bids within the lists (bins) is determined by applying heuristic 2, explained in section 2.5.2.2: The first list being sorted is D_1, and to do so the average prices for the goods are computed. According to equation (2.7), and taking into account that the number if items to be auctioned is m=5, the average price per good $a(b_i)$ of each bid is the following:

Bi	s(bi)	p(bi)	a(bi)
b1	319,1230	10	2
b2	319,1230,1900	20	4
b3	319	6	1.2
b5	2500	7	1.4
b6	1230,1900,2500	17	3.4
b7	1230,1900	12	2.4
b8	1230	6	1.2
b9	1900,2500	15	3
b10	1900	9	1.8
b11	2500,3829	10	2
b12	3829	12	2.4

Table 3.12. Average price per good.

Then, the score for each bid in D1 is computed according to equation (2.5). Given that the current partial allocation is empty ($\pi=\emptyset$). Let's start by computing the score of bid b6, then the estimate to be computed is $h(\pi\cup b6)$. This means, that a list $L\sigma(j)$ is being computed for the goods that do not belong to b6, i.e. for 319 and 3829. Thus, $L\sigma(319)=\{b3\}$, since bid b3 is the only one that contains the good 319 and does not conflict with $\pi\cup b6$. Similarly, $L\sigma(3829)=\{b12\}$. The average price of the first (and only) element of $L\sigma(319)$ is 1.2 while the average price of the first element of $L\sigma(3829)$ is 2.4 (see table 3.12). Then,

v(319)=a(b3)=1.2
v(3829)=a(b12)=2.4.

According to equation (2.6), we get:

$$h(\pi \cup b_6) = v(319) + v(3829) = 1.2 + 2.4 = 3.6$$

Finally, the score of b6 is the following (according to equation (2.5)):

$$score(b_6) = \frac{p(b_6)}{|s(b_6)|} + h(\pi \cup b_6) = \frac{17}{3} + 3.6 = 9.3$$

Now the score for bid b7 is computed, and so the estimate to be computed is h(π∪b7). Then, the following lists are constituted:

Lσ(319)={b3}.
Lσ(2500)={b5,b11}.
Lσ(3829)={b11,b12}.

Note that elements are ordered according to increasing cost. The corresponding v(j) are the following:

v(319)=a(b3)=1.2
v(2500)=a(b5)=1.4
v(3829)=a(b12)=2

So we get: $h(\pi \cup b_7) = v(319) + v(2500) + v(3829) = 1.2 + 1.4 + 2 = 4.6$

Finally, the score of b7 is the following

$$score(b_7) = \frac{p(b_7)}{|s(b_7)|} + h(\pi + b_7) = \frac{12}{2} + 4.6 = 10.6$$

Similarly, we get score(b8)=15.6. By sorting the bids in D1 according to the increasing value of their scores, we get:

D1
b6, s(b6)=(1230,1900,2500), score(b6)=9.3
b7, s(b7)=(1230,1900),score(b7)=10.6
b8, s(b8)=(1230),score(b8)=12.4

Now, following step 4 of the algorithm to search for the optimal solution, the first element of D1 is allocated, setting π=b6, and we proceed to order the bids in D2. Since both bids conflict with b6, they are removed and we proceed to the next list, D3.

Bids b1 and b2 of D3 are removed since they are not compatible with the current allocation π=b6. Bid b3 is the only one in D3, no reordering in D3 is required and b3 is selected.

Step 4 is repeated in D4. Since all bids in D4 conflict with the current allocation π=b6 ∪ b3, no option in D4 is chosen.

Step 4 is then applied in D5. The only bid selected in this list is b12.

D1	D2	D3	D4	D5
$\boxed{\text{b6, s(b6)=(1230,1900,2500)}}$	~~b11, s(b11)=(2500,3829)~~	$\boxed{\text{b3,s(b3)=(319)}}$	~~b9,s(b9)=(1900,2500)~~	$\boxed{\text{b12,s(b12)=(3829)}}$
b7, s(b7)=(1230,1900)	~~b5,s(b5)=(2500)~~	~~b1,s(b1)=(319,1230)~~	~~b10,s(b10)=(1900)~~	
b8, s(b8)=(1230)		~~b2,s(b2)=(319,1230,1900)~~		

Table 3.13. Search tree for the first solution.

The current allocation π=b6 ∪ b3 ∪ b12 is a solution to the problem (see table 3.13). The cost of this solution is p(b6)+p(b3)+p(b12)=35.

According to step 6 the process backtracks to look for better alternative solutions. Next backtracking point is in D1, by choosing b7 instead of b6.

We proceed to order the bids in D2. The scores for bids b5 and b11 for heuristics 2 are computed according to the partial allocation π=b7. The scores are 10.6 and 6.2 respectively, and bids are ordered consistently. Then, the first bid of D2 is chosen (see table 3.14). The cost of the current allocation is p(b7)+p(b11)=12+10=22, which does not go over the cost of the current solution (35).

The next step consists in choosing a new bid from D3. Bids b1 and b2 of D3 are removed since they are not compatible with the current allocation π=b7∪b11. Bid b3 is the only one in D3, no reordering in D3 is required and b3 is selected.

D1	D2	D3
b6, s(b6)=(1230,1900,2500)	$\boxed{\text{b11, s(b11)=(2500,3829)}}$	$\boxed{\text{b3,s(b3)=(319)}}$
$\boxed{\text{b7, s(b7)=(1230,1900)}}$	b5,s(b5)=(2500)	~~b1,s(b1)=(319,1230)~~
b8, s(b8)=(1230)		~~b2,s(b2)=(319,1230,1900)~~

Table 3.14. Search tree for the second solution.

The cost of the current solution is p(b7)+p(b11)+p(b3)=28 <35. So this is the best solution computed so far.

Next backtracking choice is in D2 (b5 instead of b11). The cost of the current allocation is p(b7)+p(b5)=19 < 28.

Bids b1 and b2 of D3 are removed since they are not compatible with the current allocation π=b7∪b5. Bid b3 is the only one in D3, no reordering in D3 is required and b3 is selected. The current cost of this partial allocation is p(b7)+p(b5)+p(b3)=25 < 28

Step 4 is repeated in D4. Since all bids in D4 conflict with the current allocation π=b7 ∪ b5 ∪ b3 no option in D4 is chosen. Again, step 4 is then applied in D5. The only bid selected in this list is b12 (see table 3.15).

D1	D2	D3	D5
b6, s(b6)=(1230,1900,2500)	b11, s(b11)=(2500,3829)	$\boxed{\text{b3,s(b3)=(319)}}$	$\boxed{\text{b12,s(b12)=(3829)}}$
$\boxed{\text{b7, s(b7)=(1230,1900)}}$	$\boxed{\text{B5,s(b5)=(2500)}}$	b1, s(b1)=(319,1230)	
b8, s(b8)=(1230)		b2, s(b2)=(319,1230,1900)	

Table 3.15. Search tree for solution b7 ∪ b5 ∪ b3 ∪ b12 .

Current cost of this partial allocation is p(b7)+p(b5)+p(b3)+p(b12)=37 > 28, so this solution is rejected.

Next backtracking choice is in D1 (b8 instead of b7). The corresponding scores for the bids of D2 are:

score(b11)= 8
score(b5)= 12.4

So, the ordering of bids in D2 is maintained. The first bid of D2, b11, is chosen. The current cost of the partial allocation is p(b8)+p(b11)=6+10=16.

Regarding D3, again only b3 does not conflict with the current allocation. Thus, b3 is selected and the cost of the current allocation is p(b8)+p(b11)+p(b3)=22 <28. Regarding D4, bid b9 conflicts with the current allocation, so it is removed. The bid chosen is then b10 (see table 3.16).

D1	D2	D3	D4
b6, s(b6)=(1230,1900,2500) b7, s(b7)=(1230,1900) $\boxed{\text{b8, s(b8)=(1230)}}$	$\boxed{\text{b11, s(b11)=(2500,3829)}}$ b5,s(b5)=(2500)	$\boxed{\text{b3,s(b3)=(319)}}$ ~~b1,s(b1)=(319,1230)~~ ~~b2,s(b2)=(319,1230,1900)~~	~~b9,s(b9)=(1900,2500)~~ $\boxed{\text{b10,s(b10)=(1900)}}$

Table 3.16. Search path for allocation b8∪b11∪b3∪b10.

The cost of the current solution is p(b8)+p(b11)+p(b3)+p(b10)=31, which is greater than the best solution computed so far (28), so this branch of the search tree is pruned.

Next backtracking choice is in D2 (b5 instead of b11). The cost of the partial allocation is p(b8)+p(b5)=13.

Regarding D3, again only b3 does not conflict with the current allocation. Then we choose b3. The cost of the current allocation is p(b8)+p(b5)+p(b3)=19 <28.

Regarding D4, bid b9 conflicts with the current allocation, so it is removed. The bid chosen is then b10 (see table 3.17).

D1	D2	D3	D4
b6, s(b6)=(1230,1900,2500) b7, s(b7)=(1230,1900) $\boxed{\text{b8, s(b8)=(1230)}}$	b11, s(b11)=(2500,3829) $\boxed{\text{b5,s(b5)=(2500)}}$	$\boxed{\text{b3,s(b3)=(319)}}$ ~~b1,s(b1)=(319,1230)~~ ~~b2,s(b2)=(319,1230,1900)~~	~~b9,s(b9)=(1900,2500)~~ $\boxed{\text{b10,s(b10)=(1900)}}$

Table 3.17. Selection of bid in D4 to current allocation π=b8∪b5∪b3

The cost of the current allocation is p(b8)+p(b5)+p(b3)+p(b10)=28. So this branch of the search tree is pruned.

No more solutions can be found.

So the best solutions found are b7, b11, and b3. As a result the best combination found is: (1230,1900)(2500,3829)(319) because it has the lowest cost (28).

Then, the fire station sends a message to the fire brigades (bidders) in order to inform them who are the winners. In this example the winners are: Agent A extinguishes the fire id 319, Agent C extinguishes the fires ids 1230 and 1900, and Agent E extinguishes the fires ids 2500 and 3829.

In this way, our tasks in the rescue scenario are allocated, minimizing the cost of developing these tasks.

3.6.3.2 Conclusions

Determining the outcome of combinatorial auctions is an optimisation problem that is NP-complete in the general case. The CASS algorithm is an approach that tries to overcome this intractability by structuring the search space and improving pruning of the bids received. The CASS algorithm presents several beneficial properties, such as the possibility to be used as an anytime algorithm, as it tends to find good allocations quickly. However, it presents the same disadvantages regarding precedence and pre-emption requirement for our rescue problem.

3.6.4 Structured search with duplicates and pre-emption

The results obtained with the last approach show that the application of winner determination algorithms in a disaster environment is not so straightforward. This is due to the fact that in a disaster environment pre-emption and continuous auction mechanisms should be taken into account due to the dynamics of the environment.

The proposed algorithm [64] takes pre-emption into account, and is more appropriate for task allocation in dynamic and unpredictable environments than its predecessors.

Let $A=\{A_1,..., A_n\}$ be the set of bidders (agents that request at most one set of items), and agent(b_i) that predicts the returns of the agent which has requested bid b_i.

The algorithm we propose is based on CASS and Sandholm's winner determination algorithms with the following considerations. From the CASS (Combinatorial Auction Structured Search) we get the idea of structuring the search space. However, we do not use a Branch and Bound mechanism to search for the solution. Instead of such a mechanism, we adopt a search mechanism based on the A* algorithm (more similar to Sandholm's work).

The main steps of the algorithm are the following:

1. Assign to M' the set of items requested by the agents in such a way that if there is any item in M that has not been requested in any bid, then it is not included in M'.
   $M' \subseteq M, M'=\{t_i \mid \exists\, b_j , t_i \in s(b_j)\}$
2. For each pair of bids (b_1, b_2) where $s(b_1) \subseteq s(b_2)$ and $p(b_1) \geq p(b_2)$, remove b_2 from the lists of bids (b_2 is never preferable to b_1).
3. Partition the search space in lists. There is at most one list per good, and each bid is stored in the list corresponding to its first item.
4. Organize the lists according to the good ordering heuristics.
5. The current solution is empty, $\pi=\emptyset$.
6. For each list D_i (corresponding to item t_i):

```
    a. Organize the bids in the lists according to the bid ordering heuristics
    b. Pick the first bid $b_i^1$ of the list.
    c. If $\exists b_j \in \pi$, such that agent($b_i^1$)=agent($b_j$),
        i. If $s(b_i^1) \subseteq s(b_j)$ then continue.
        ii. If $s(b_j) \subseteq s(b_i^1)$, then $\pi=\pi — b_j \cup b_i^1$
        iii. If $\exists b_k \in \pi$, such that $t_i \in b_k$ then continue.
        iv. If $\pi — b_j$ contains all the items $t_1, \ldots, t_{i-1}$, corresponding to the previous
            partitions, then $\pi=\pi — b_j \cup b_i^1$
        v. Otherwise, remove $b_i^1$ from $D_i$ and go to 6b.
7. Check if $\pi$ is a solution. Otherwise, backtrack.
```

Steps 1 and 2 are pre-processing steps of the algorithm, in which unachievable items and bids are removed from the auctioning process. Regarding step 6.b, several circumstances are considered in the case that the bid to be added to the solution corresponds to an agent that has a bid already assigned in the partial solution computed so far. Due to this treatment, it could be the case that at the last partition D_n, we get a partial solution π that does not contain the complete list of items. Thus, a backtracking step allows the recovery of a previous branch of the search tree where a solution can be found.

3.6.4.1 Good ordering heuristics:

The goal of this heuristic is to determine the ordering of goods.
For each good i, the score is computed according to the following expression:

$$score_i = \frac{numbids_i}{avggoods_i} \qquad (3.13)$$

Where:

- $numbids_i$ is the number of bids that request good i
- $avggoods_i$ is the average number of goods requested by these bids.

The list corresponding to the bid with the highest score is selected and labelled as D_1. In the case of a tie, select the lowest good according to a lexicographical order. Then, the score for the remaining goods is recalculated and the process is repeated until a total order of the list is achieved, from the lowest score to the highest: $D_1, \ldots, D_n$.

This heuristic is the same as the one used in CASS [17], with a slight difference, in that we invert the order of applications.

3.6.4.2 Bid ordering heuristics:

The goal of this heuristic is to determine the ordering of bids within lists.
We have tested two heuristics. The first one aims at getting solutions with a low cost, while the second one is biased towards replication of items in the solution.

- **Low cost preference.** Given a partial allocation π, bids in the lists are sorted in ascending order of their score. The score of bid j is computed according to the following equation:

$$score(b_j) = p(b_j) + h(\pi \cup b_j) \qquad (3.14)$$

Where $h(\pi \cup b_j)$ is an upper bound that estimates how promising the units not yet allocated are. This bound is computed as follows:

$$h(\pi \cup b_i) = \sum_j v(j) \qquad (3.15)$$

Where v(j) is defined as follows.

Let $L_\sigma(j)$ be the list of bids referring to good j that can be encountered in the remainder of the search and which is not already included in $\pi \cup b_j$. Elements in $L_\sigma(j)$ are ordered in increasing order by their $p(b_i)$. And let b^1_i be the first bid in $L_\sigma(j)$. Then $v(j)=p(b^1_i)$.

This heuristic is similar to the one proposed by CASS. However, in CASS the elements in $L_\sigma(j)$ are ordered according to the average price of the bid per good, and the elements that conflict with the partial solution computed so far are removed from $L_\sigma(j)$ (so, no item duplication is allowed).

- **Replication Preference.** The score of bid j is computed according to the following equation:

$$score(b_j) = \frac{p(b_j)}{|S(b_j)|} + h(\pi \cup b_j) \qquad (3.16)$$

Where $h(\pi \cup b_j)$ is computed as (3.16).

The main difference between this heuristic and the previous one is that in (3.16), we take into account the length of the bids. Thus, we are favouring the bids with a low cost per item and as a consequence, more duplicate items can be included in the final solution.

Note that the solution computed is the optimal one according to the heuristics. So in the case of using a low-cost heuristics, π represents the cheapest solution. By backtracking, the next solution is the second cheapest one, and so on. Thus, the alternative solutions can be easily generated and used in a pre-emption algorithm if required.

3.6.4.3 Pre-emption algorithm

Regarding pre-emption, any agent A_i that has been the winner of a rescue task by means of the allocation mechanism can become injured or even die before the accomplishing the task. Then, the following policies have been defined in order to assign a substitutive agent.

Let's say that π is the current allocation and $b_i \in \pi$ the bid assigned to the rescue agent A_i, with the corresponding items $t_{ij} \in s(b_i)$. Then, for every item t_{ij}, the following re-assignation process is started:

1. If t_{ij} is only requested by A_i, then t_{ij} will not be allocated until a new agent appears in the scenario with the capabilities of performing t_{ij}.
2. If there is a bid $b_j \in \pi$ (and $b_i \neq b_j$), so that $t_{ij} \in s(b_j)$, then no reallocation is performed since the item t_{ij} is replicated in the solution, that is, it is already assigned to another agent.
3. If there is a bid $b_j \notin \pi$ (and $b_i \neq b_j$) so that $t_{ij} \in s(b_j)$ and agent(b_i) $\neq$ agent(b_j), then
   a. Check if agent(b_j) participates in the current allocation π. If so, localise the bid b_k of agent(b_j) that belongs to π. Then,
      i. If $s(b_k) \subseteq s(b_j)$, reassign the bid b_j to agent(b_j) instead of b_k.
      j. Otherwise compute the items in b_j not covered by b_k, that is: uncovered(b_k)= $s(b_k)$-$s(b_j)$. If for every item $t_{un} \in$ uncovered(b_k) there is a bid $b_l \in \pi$ that requests it ($t_{un} \in s(b_l)$), then reassign b_j by b_k to agent(b_j).
   b. Look for the next best solution of the allocation algorithm, π_{next}. For every bid in π_{next}, request the corresponding agent for the current cost of the bid. If it is maintained, then reassign the new solution π_{next}.
   c. Otherwise, no reallocation is performed. No rescue operation for the items assigned to agent(b_i) will be performed until new rescue agents appear in the scene and a new auction process is started.

We need to stress that the reassignment process is not simple. Since most of the agents have started to deal with their winning bids, then a reassignment process requires a merging process that maintains the precedence relations between items according to the current situation of agents.

3.6.4.4 Example

In this example, we refer to snapshot 2, which concerns fire brigade operation. After announcing the tasks, the bids gathered by the fire station are the following:

Agent	Bid b_i	$s(b_i)$	$p(b_i)$
A	b_1	{319,1230}	10
	b_2	{1900.1230.319}	20
	b_3	{319}	6
B	b_4	{2500.3829}	12
	b_5	{2500}	7
C	b_6	{2500,1230,1900}	17
	b_7	{1230,1900}	12
	b_8	{1230}	6
D	b_9	{1900,2500}	15
	b_{10}	{1900}	9
E	b_{11}	{3829,2500}	10
	b_{12}	{3829}	12

Table 3.18. Set of bids gathered

After applying step 2 of the algorithm, bid b_4 is removed from M' since it dominates b_{11}. The set of remaining bids is shown in table 3.19. All items are requested by at least one agent, so all items are included in the auction process (M'=M in step 1).

Bid b_i	$s(b_i)$	$p(b_i)$
b_1	{319,1230}	10
b_2	{1900.1230.319}	20
b_3	{319}	6
b_5	{2500}	7
b_6	{2500,1230,1900}	17
b_7	{1230,1900}	12
b_8	{1230}	6
b_9	{1900,2500}	15
b_{10}	{1900}	9
b_{11}	{3829,2500}	10
b_{12}	{3829}	12

Table 3.19. Bids to be auctioned.

Then, the search space is divided into the following sets (step 3):

b_1,s(b_1) b_3,s(b_3)	b_7,s(b_7) b_8,s(b_8)	b_2,s(b_2) b_9,s(b_9) b_{10},s(b_{10})	b_5,s(b_5) b_6,s(b_6)	b_{11},s(b_{11}) b_{12},s(b_{12})

Following, in step 4 the ordering of the lists is determined by using the good ordering heuristics.

Good 319 is requested by bids b_1, b_2, and b_3. Then numgoods$_{319}$=3. Bid b_1 requests 2 goods, while b_2 requests 3 and b_3, 1. Then, the average number of goods requested by these bids is:

$$avggoods_{319} = \frac{1+2+3}{3} = 2$$

According to equation (3.13), the corresponding score is:

$$score_{319} = \frac{numgoods_{319}}{avggoods_{319}} = \frac{3}{2} = 1.5$$

Similarly, we compute the score for the remaining goods:

good	requested by	Num goods	avggoods	Score
319	b_1,b_2,b_3	3	2	1.5
1230	b_1,b_2,b_6,b_7,b_8	5	2.2	2.27
1900	b_2,b_6,b_7,b_9,b_{10}	5	2.2	2.27
2500	b_5,b_6,b_9,b_{11}	4	2	2
3829	b_{11},b_{12}	2	1.5	1.3

Goods 1230 and 1900 have the highest scores. We choose 1230 following the lexicographic order. So the list corresponding to good 1230 is labelled as D_1. The remaining lists are for goods 319, 1900, 2500 and 3829.

Now, we recalculate the scores for the remaining lists:

good	requested by	Num goods	avggoods	Score
319	b_1,b_2,b_3	3	2	1.5
1900	b_2, b_9,b_{10}	3	2	1.8
2500	b_5,b_9,b_{11}	3	1,6	2
3829	b_{11},b_{12}	2	1.5	1.3

Good 2500 has the highest score. We repeat the process, until we get the following labels:

D_1	D_2	D_3	D_4	D_5
b_7,s(b_7) b_8,s(b_8)	b_5,s(b_5) b_6,s(b_6)	b_1,s(b_1) b_3,s(b_3)	b_2,s(b_2) b_9,s(b_9) b_{10},s(b_{10})	b_{11},s(b_{11}) b_{12},s(b_{12})

Table 3.20. Final decreasing order of the lists according to the good ordering heuristics

Now the process continues with step 6. First, the order of bids in D_1 is set according to the bid ordering heuristics. Given that the current allocation is empty ($\pi=\varnothing$ in step 5), the following lists can be formed:

$$L_\sigma(319)=\{b_3,b_1,b_2\}.$$
$$L_\sigma(1900)=\{b_{10},b_9,b_6,b_2\}.$$
$$L_\sigma(2500)=\{b_5,b_6,b_9,b_{11}\}.$$
$$L_\sigma(3829)=\{b_{11},b_{12}\}.$$

Then, if we select b_7 in D_1, items 319, 2500 and 3829 will be required, but 1900 will not be mandatory. Consequently, the estimate for b_7 is the following:

$$h(\pi \cup b_7) = v(319) + v(2500) + v(3829) = 6 + 7 + 10 = 23$$

The score for b_7 according to equation (3.14) is the following:

$$score(b_7) = p(b_7) + h(\pi + b_7) = 12 + 23 = 35$$

We proceed in the same way with bid b_8, getting the score(b_8)= 38. So the order of bids in D_1 is first b_7, and second b_8 (see table 27). Therefore, current allocation is $\pi=b_7$.
Next the iteration of step 5 in the algorithm is applied to list D_2. The bid order in D_2 is computed according to the current allocation. That is, the current lists are the following:

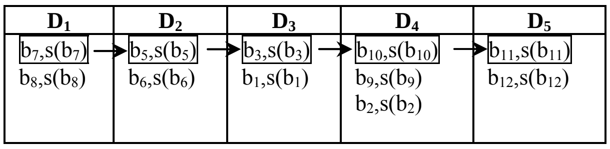
$$L_\sigma(319)=\{b_3,b_1,b_2\}.$$
$$L_\sigma(1900)=\{b_{10},b_9,b_2\}.$$
$$L_\sigma(3829)=\{b_{11},b_{12}\}.$$

Then, if we select b_5, the only remaining mandatory items to be assigned are 319 and 3829. So the estimate for b_5 and the corresponding score are the following:

$$\boldsymbol{h(\pi \cup b_5) = v(319) + v(3829) = 6 + 10 = 16}$$

$$\boldsymbol{score(b_5) = p(b_5) + h(\pi + b_5) = 7 + 16 = 23}$$

The score for b_6 is 33. So b_5 is selected from D_2 to form the current allocation. And the execution of the algorithm is continued until the final solution is achieved, as shown in table 3.21.

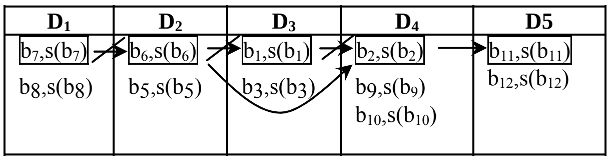


D_1	D_2	D_3	D_4	D_5
$\boxed{b_7,s(b_7)}$	$\boxed{b_5,s(b_5)}$	$\boxed{b_3,s(b_3)}$	$\boxed{b_{10},s(b_{10})}$	$\boxed{b_{11},s(b_{11})}$
$b_8,s(b_8)$	$b_6,s(b_6)$	$b_1,s(b_1)$	$b_9,s(b_9)$	$b_{12},s(b_{12})$
			$b_2,s(b_2)$	

Table 3.21. Solution for the rescue example by using the low cost heuristics

The best solution found is $\pi=\{b_7, b_5, b_3, b_{10}, b_{11}\}$, requested by agents C, B, A, D and E respectively. The cost of the solution is 44, and contains two items that are allocated twice (1900 and 2500). Then, the fire station sends a message to the fire brigades (bidders) in order to inform them who the winners are.

The bid ordering heuristics favours duplicates instead of low cost solutions, and therefore we get the following result: $\pi=\{b_6,b_2,b_{11}\}$ (see table 3.22).



D_1	D_2	D_3	D_4	D5
$\boxed{b_7,s(b_7)}$	$\boxed{b_6,s(b_6)}$	$\boxed{b_1,s(b_1)}$	$\boxed{b_2,s(b_2)}$	$\boxed{b_{11},s(b_{11})}$
$b_8,s(b_8)$	$b_5,s(b_5)$	$b_3,s(b_3)$	$b_9,s(b_9)$	$b_{12},s(b_{12})$
			$b_{10},s(b_{10})$	

Table 3.22. Solution for the rescue example by using the high duplicate heuristics.

The crossed out arrows correspond to allocated bids that have been removed because they were requested by agents with bids that had already been assigned (Step 6c of the algorithm).

3.6.4.5 Pre-emption example

Let us suppose that the heuristic used in step 6a is the bid ordering – low cost heuristics, that led to the solution $\pi=\{b_7, b_5, b_3, b_{10}, b_{11}\}$, requested by agents C, B, A, D and E correspondingly.

If A fails, condition 1 of the re-assignment algorithm holds, that is, no other agent has requested the items in b_3. Thus, no alternative agent can be assigned to cover the tasks contained in bid b_3. The same happens if E fails.

In the case that B fails, whose allocated bid is b_5, it holds that $s(b_5) \subseteq s(b_{11})$, the latter assigned to agent E. So condition 2 of the pre-emption algorithm holds and no reassignment is performed.

In the case that C fails, the items to be re-assigned are the ones for b_7: 1230 and 1900. The latter is already assigned to D in the current solution ($1900 \in s(b_{10})$), so no reassignment is required. Regarding item 1230, we proceed with step 3 of the pre-emption mechanism. According to this, there are two bids, b_1 and b_2, requested by a different agent, A, and that contain item 1230. Agent A has already been assigned to perform task b_3. Moreover, it holds that $s(b_3) \subseteq s(b_1)$ and $s(b_3) \subseteq s(b_2)$. Since both bids satisfy the condition of step 3.1.a of the algorithm, the bid with the lowest cost, b_1, is chosen. Then, agent A is reallocated, changing its current bid b_3 by b_1.

3.6.4.6 Conclusions

We have compared our results with the ones obtained by applying the CASS [17] and the Sandholm's algorithm [53]. Table 3.23 summarises the different results obtained.

Method	Solution	Number of agents	Duplicated items	Cost
Low-cost	$b_7,b_5,b_3,$ b_{10},b_{11}	5	2	44
High-duplicates	b_6,b_2,b_{11}	3	3	39
CASS	b_7,b_1,b_3	3	0	28
Sandholm's	b_7,b_1,b_3	3	0	28

Table 3.23. Summary table of results

As shown, our algorithm is the one that uses the resources (agents) widely: all agents have a bid assigned to them. This is the case when the bid ordering heuristic selected focuses on low-cost. However, the solution obtained with the low-cost heuristics is the most expensive one, with a significant difference regarding the CASS and Sandholm's algorithms. In a rescue environment, however, the appropriate use of all the available results is a more important issue than the overall cost. Moreover, we are getting duplicated items, which is an important issue when replacing damaged agents.

Nevertheless, we set such a result at a high cost. So a new alternative algorithm will be explored.

3.6.5 Structured search with balanced duplicates

The next alternative that we will explore concerns getting duplicates like the previous one, but with a proportional ratio.

The algorithm to search for the optimal solution is as follows (from [17]):

1. For each pair of bids (b_1,b_2) where $s(b_1) \subseteq s(b_2)$ and $p(b_1) \geq p(b_2)$, b_2 is removed from the lists of bids to be considered during the search because b_2 is never preferable to b_1.
2. Dividing the search space into lists. There is at most one list per good, and each bid is stored in the list corresponding to its first item.
3. Organize the lists according to the good ordering heuristics (see below).
4. Current solution is empty, $\pi=\varnothing$.
5. Current list is $D_i = D_1$
6. If the item corresponding to D_i is already in π, then put the list in the set of redundant agents R, and proceed with the next list
7. Organize the bids in the lists D_i according to the bid ordering heuristics (see 3.2)
8. Pick the first bid b_i^1 of the list and add it to π
9. If the current list is D_m then proceed with the redundant agents R. Otherwise, select the next list and go to 6.
10. For each list in R, select the best bid according to the bid ordering heuristics

Note that the algorithm has two main parts: one concerning the allocation of at least one of the items (from 1 to 9), and the second part concerning allocation replication with free agents that have not been assigned to any rescue task.

3.6.5.1 Good ordering heuristics

The goal of this heuristic is to determine the order of goods, and therefore the list order. For each good i a $score_i$ is computed according to the following expression:

$$score_i = \frac{numbids_i}{avggoods_i} \qquad (3.17)$$

Where:

- $numbids_i$ is the number of bids that request good i
- $avggoods_i$ is the average number of goods requested by these bids.

The list corresponding to the bid with the lowest score is selected and labelled as D1. Then, the scores for the remaining goods are recalculated and the process is repeated until a total order of the list is achieved, from the lowest score to the highest: D1, …, Dm.

3.6.5.2 Bid ordering heuristics: balancing duplicates

The goal of this heuristic is to determine the order of bids within the lists.

Given a partial allocation π, bids in the lists are sorted in descending order of their score. The score of bid j is computed according to the following equation:

$$score(b_j) = \sum_j v(j) \qquad (3.18)$$

Where v(j) is defined as follows.

Let $L_\sigma(j)$ be the list of bids that refer to good j and that can be found in the remainder of the search and that are not in π. For every element b_k in $L_\sigma(j)$, the standard deviation value regarding the population of items in $\pi \cup b_k$ is computed.

$$\sqrt{\frac{n\sum x_i^2 - \left(\sum x_i\right)^2}{n^2}} \qquad (3.19)$$

Where: *x* is the number occurs of item t_i en π and
n is the total number of item in π.

Elements in $L_\sigma(j)$ are sorted according to an increasing order of such value. In case of getting the same value, the elements are ordered according to their cost (less expensive first). Let b^{ok}_i be the first bid in $L_\sigma(j)$. Then $v(j)=a(b^{ok}_i)$.

3.6.5.3 Example

In this example, we refer to the snapshot 2 that is the fire fighting operation in section 3.2.2. After announcing the tasks, the bids gathered by the fire station are the following:

Agent	Bid b_i	$s(b_i)$	$p(b_i)$
A	b_1	{319,1230}	10
	b_2	{1900.1230.319}	20
	b_3	{319}	6
B	b_4	{2500.3829}	12
	b_5	{2500}	7
C	b_6	{2500,1230,1900}	17
	b_7	{1230,1900}	12
	b_8	{1230}	6
D	b_9	{1900,2500}	15
	b_{10}	{1900}	9
E	b_{11}	{3829,2500}	10
	b_{12}	{3829}	12

Table 3.24. set of bids gathered

Bid b_4 is removed from the list since it dominates b_{11} (step 1 of the algorithm). Then, the set of remaining bids is the one shown in table 3.25. All items are requested by at least one agent, so all items are included in the auction process.

Bid b_i	$s(b_i)$	$p(b_i)$
b_1	{319,1230}	10
b_2	{1900.1230.319}	20
b_3	{319}	6
b_5	{2500}	7
b_6	{2500,1230,1900}	17
b_7	{1230,1900}	12
b_8	{1230}	6
b_9	{1900,2500}	15
b_{10}	{1900}	9
b_{11}	{3829,2500}	10
b_{12}	{3829}	12

Table 3.25. Bids to be auctioned.

Then, the search space is divided into the following sets (step 2):

b_1,s(b_1) b_3,s(b_3)	b_7,s(b_7) b_8,s(b_8)	b_2,s(b_2) b_9,s(b_9) b_{10},s(b_{10})	b_5,s(b_5) b_6,s(b_6)	b_{11},s(b_{11}) b_{12},s(b_{12})

Following, in step 3 the list orders are determined by using the good ordering heuristics. Good 319 is requested by bids b_1, b_2, and b_3. Then numgoods$_{319}$=3. Bid b_1 requests 2 goods, while b_2 requests 3 and b_3, 1. Then, the average number of goods requested by these bids is:

$$avggoods_{319} = \frac{1+2+3}{3} = 2$$

According to equation (3.17), the corresponding score is:

$$score_{319} = \frac{numgoods_{319}}{avggoods_{319}} = \frac{3}{2} = 1.5$$

Similarly, we compute the score for the remaining goods:

Good	requested by	Num goods	avggoods	Score
319	b1,b2,b3	3	2	1.5
1230	b1,b2,b6,b7,b8	5	2.2	2.27
1900	b2,b6,b7,b9,b10	5	2.2	2.27
2500	b5,b6,b9,b11	4	2	2
3829	b11,b12	2	1.5	1.3

Goods 1230 and 1900 have the highest score. We choose 1230 following the lexicographic order. So the list corresponding to good 1230 is labelled as D_1. The remaining lists are for goods 319, 1900, 2500 and 3829.

Now, we recalculate the scores for the remaining lists:

Good	requested by	Num goods	avggoods	Score
319	b_1,b_2,b_3	3	2	1.5
1900	b2, b9,b10	3	2	1.8
2500	b5,b9,b11	3	1,6	2
3829	b11,b12	2	1.5	1.3

Good 2500 has the highest score. We repeat the process, until we get the following labels:

D_1	D_2	D_3	D_4	D_5
b_7,s(b_7) b_8,s(b_8)	b_5,s(b_5) b_6,s(b_6)	b_1,s(b_1) b_3,s(b_3)	b_2,s(b_2) b_9,s(b_9) b_{10},s(b_{10})	b_{11},s(b_{11}) b_{12},s(b_{12})

Table 3.26. Final decreasing order of the lists according to the good ordering heuristics

Now the process continues with step 4. First, the order of bids in D_1 is set according to the bid ordering heuristics. Given that the current allocation is empty ($\pi=\varnothing$ in step 4), if we select b_7 in D_1, items 319, 2500 and 3829 will be required, but 1900 will not be mandatory. Then, the following lists can be formed:

$$L_\sigma(319)=\{b_3,b_1,b_2\}.$$
$$L_\sigma(2500)=\{b_5, b_{11}, b_6, b_9\}.$$
$$L_\sigma(3829)=\{b_{11},b_{12}\}.$$

The order of bids within the list has been determined according to the bid ordering heuristics. For example, for good 319, according to (3.19) the standard deviation including b1 is 0,47: the corresponding population of items is 1230 (twice, one from b1 and another from b7), 1900 and 319. For b2 and b3 the standard deviation is 0.47 and 0 respectively. Then b3 is the first one on list $L_\sigma(319)$, and b1 the second (it has a lower price than b2), and finally b2. The rest of the lists are obtained in the same way.

Consequently, the score for b_7 is the following:

$$\boldsymbol{score(b_7) = v(319) + v(2500) + v(3829) = 6 + 7 + 10 = 23}$$

We proceed in the same way with bid b_8, getting the score(b_8)= 32. So the order of bids in D_1 is first b_7, and second b_8 (see table 3.27). So, current allocation is π=b_7.
Next iteration of step 5 of the algorithm is applied to list D_2. The bid ordering in D_2 is computed according to the current allocation.
Then, if we select b_5, the only remaining mandatory items to be assigned are 319 and 3829. The lists that we constituted are the following:

$$L_\sigma(319)=\{b_3,b_1,b_2\}.$$
$$L_\sigma(3829)=\{b_{12},b_{11}\}.$$

So the estimate for b_5 is the following:

$$\boldsymbol{score(b_5) = v(319) + v(3829) = 6 + 12 = 18}$$

Similarly, the score for b_6 is 18. Arbitrarily, b_5 is selected from D_2 to form the current allocation. The execution of the algorithm is continued until the final solution is achieved, as shown in table bellow.

Note, that when choosing and selecting from D_4, the corresponding good already belongs to π, so bids in D_4 are considered as redundant and are not assigned until the end of the algorithm (Step 10).

D_1	D_2	D_3	D_5	D_4
$\boxed{b_7,s(b_7)}$	$\rightarrow \boxed{b_5,s(b_5)}$	$\rightarrow \boxed{b_1,s(b_1)}$	$\rightarrow \boxed{b_{12},s(b_{12})}$	$\rightarrow \boxed{b_{10},s(b_{10})}$
$b_8,s(b_8)$	$b_6,s(b_6)$	$b_3,s(b_3)$	$b_{11},s(b_{11})$	$b_9,s(b_9)$
				$b_2,s(b_2)$

Table 3.27. Final decreasing order of the lists according to the good ordering heuristics

The best solution found is π={b_7, b_5, b_1, b_{12}, b_{10}}, requested by agents C, B, A, E and D respectively. All agents are involved in the allocation, although D is redundant.
The cost of the solution is 37, and contains two items allocated twice (1900 and 2500). Then, the fire station sends a message to the fire brigades (bidders) in order to inform them of who the winners are.

3.6.5.4 Discussion

We have compared our results with the ones obtained by applying the CASS [17], and the Sandholm's algorithm [53], and the algorithms of the previous sections. Table 3.28 summarizes the different results obtained.

Method	Solution	Number of agents	Duplicated items	Cost
Balancing duplicates	B7,b5,b1, b2,b10	5	2	37
Low-cost	B7,b5,b3, b10,b11	5	2	44
High-duplicates	b6,b2,b11	3	3	39
CASS	b7,b1,b3	3	0	28
Sandholm's	b7,b1,b3	3	0	28

Table 3.28. Summary table of results

Our solution improves the cost of the previous versions of the algorithm, although the cost is still over the results obtained with the CASS and Sandholm's algorithms. In a rescue environment, however, the appropriate use of all the available results is more important than the overall cost. Moreover, we are getting duplicated items, which is an important issue when replacing damaged agents.

Note, however, that the proposed algorithm is applied simultaneously by each central in order to allocate their resources. This means that the ambulance central distributes ambulance teams; the fire station allocates fire brigades and the police station police forces. As a continuation of our work, we are thinking of extending the algorithm in such a way as to take into account action coordination between centrals, which would involve precedence relations between rescue tasks. For example, police forces usually need to clear roads before ambulance teams can act. Other aspects to be taken into account are pre-emption and continuous auctioning due to the dynamics of the environment. Regarding the former, rescue teams can die or be injured before they accomplish the tasks they have won, so a pre-emption mechanism should be developed (as the one of 3.6.4.3). Concerning the latter, new victims can be found at any moment, so items can be auctioned at any time. Work such as that which is started in [6] and [41] could be a starting point.

3.7 Conclusions

In this section, we have presented exploratory research work on multi-agent system coordination. Several AI techniques have been used to facilitate cooperation and task allocation among agents in a distributed environment like the Robocup rescue scenario.

Case-based reasoning techniques have been tested in order to facilitate decision making based on experience and fuzzy filter techniques. They have also been used to listen to message selection, taking into account the trust of emitter agents. MCDM techniques have been applied to a rescue scenario in order to provide the agents with decisions about resources (rescue agents) and tasks (victims, fires and blocked roads). Based on the abilities of the agents to perform the tasks, the central agents allocate rescue agents to the victims with the highest priority, that is, the victims that are most damaged. Finally, we have applied combinatorial auction techniques to task allocation in the rescue scenario. This technique has shown to be effective and provide optimal allocations. In addition, we have developed a new algorithm for the rescue scenario based on CASS and Sandholm algorithms [11, 53].

The results obtained so far are quite satisfactory; we are dealing with the rescue problem partially. That is, we are only concerned with part of the constraints formulated in 3.1.2: i.e. due dates and overlapping. Other constraints, such as precedence, distances and priorities are still to be studied. In the next chapter we present a proposal to tackle the complete problem.

Chapter 4

4. THESIS PROPOSAL

This doctoral thesis proposal is in the field of multi-agent systems coordination. Specifically, the thesis proposal is addressed to improve task and resource allocation in distributed environments such as MAS. According to the concrete objectives of chapter 1, a preliminary state of the art has been performed and experimentations with AI techniques have been done. Concretely, some of these techniques have been applied to the RoboCup Rescue scenario. Based on this experimentation, we have concluded that combinatorial auctions seem to be an effective technique to tackle the distributed task allocation problem.

In what follows, a working plan is proposed to achieve the general objective of this thesis: to make its contribution to the development of new coordination mechanisms in MAS, specifically, to study the adequate techniques and mechanisms to task and resource allocation in distributed environments.

In addition, there are also scheduled two pre-doctoral internships in international research groups. The first internship is planed to be 3 months of duration in a specialized research group on combinatorial auctions in order to obtain a complete understanding of this technique. The second pre-doctoral internships will be performed at the end of the working plan, in order to compare the new methods developed with the ones of other research groups.

4.1Working plan proposal

To accomplish the proposed general objective in this section, the following tasks are proposed:

- **Deepening the state of the art on combinatorial auctions**

This task comprises the study of complexity measures, conditions of coverage and computational time of the winner determination algorithm.

- **Performing extensions to the proposed winner determination algorithm**

The algorithm proposed in chapter 3 solves part of the problem formulated. For example, no overlapping tasks are assigned to the same agent. Taking into account that other constraints should be incorporated into the algorithm, the following subtasks are proposed:

- Study of precedent relations that are related to our approach. In [6] temporal and precedence constraints are tackled to maximizing agents preferences over schedules of multiples tasks. This approach can be a starting point for the study.

- The priority of tasks should be taken into account in the proposed algorithm.

- **Extensions to the dynamical problem**

The approaches presented in the chapter 3 refer to snapshot from a static point of view. However, this kinds of problems that we are trying to solve are dynamical, that is, tasks and resources coming in and going out of the system. For this reason, it should study the mechanisms to tackle with such dynamicity. In this sense, it is planning to develop the following subtasks:

- To study continuous double auctions as a way to deal with tasks and bids at any time. In [71, 67] approaches about CDA's auction have been developed. However, there are a few researches in this field.

- To extend the preliminary pre-emption mechanisms, that is, to look for alternative mechanisms to deal with the damaged resources. In [31, 41], an approach concerning about the speculative auctions is presented, which could be a starting point for this subtask.

- **Integration of strategies for tasks allocation in a rescue scenario.**

The approaches presented so far deal with the task allocation among homogeneous agents. For example, the ambulance centre decides upon ambulance teams and victims; the fire station assigns fire brigades to fire buildings; and the police office decides which police force unblocked determined road, etc. Then, the algorithm should be extended to deal with all the resources and tasks from a global point of view. In this line, the use of electronic institutions [22] and scenarios for trading agents [49] can be of great help in modeling such scenario.

- **Implementation and experimental prototype**

So far, we have experimented with the RoboCup Rescue simulator. However this environment of simulation presents some drawbacks as its slow execution and complexity of the code source of the simulator to make modifications in the performance of experiments. For this reason, we will develop a prototype using new tools for auctions application as REPAST [45] and agent platforms as JADE.

- **Extensions to other domains**

The new methods developed will be applied to other domains. So far, we have focused on the rescue problem. However, there are some benchmarks [9] in which we can apply the new methods and compare the obtained results.

Moreover, there are other real applications as health care, tourism and interurban transporting in which the Agents Research Lab is dealing with. New methods will be applied to such frameworks in order to verify their viability and benefices in real problems.

- **Elaboration of final thesis documentation**

This task concerned about reporting the results and main contributions in international forums, such as conferences and journals related to this research and application area. And finally, the doctoral thesis will be written.

4.2Schedule of activities

The proposed schedule is summarized in the following flow-chat:

Activity Description	Time Line											
	Year 1						Year 2					
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
1. Deepening state of the art on combinatorial auctions	▨	▨										
2. Performing extensions to the proposed winner determination algorithm		▨	▨	▨								
3. Extensions to the dynamical problem			▨	▨	▨							
4. Integration of strategies of tasks allocation in a rescue scenario				▨	▨	▨						
5. Implementation and experimental prototype			▨	▨	▨	▨	▨					
6. Extensions to other domains							▨	▨	▨	▨		
7. Pre-doctoral internship in international research group	▨	▨							▨	▨		
8. Elaboration of final thesis documentation										▨	▨	▨

Table 4.1. Proposed schedule for doctoral research work (B= Two month period)

Chapter 5

5. CONCLUSIONS

Resources are limited and should be used in a rational way in order to solve problems. In the particular case of cooperative multi-agent systems, agents are the resources available to perform the task required in order to solve a distributed problem. Coordination mechanisms should be provided in order to distribute task allocation among the different available agents.

There are several mechanisms that deal with task and resource allocation in MAS, which we have reviewed in the chapter 2 of this document. However, it is important to design new methods for task allocation in distributed environments.

Among them, combinatorial auctions are an effective technique for tackling task allocation problems in MAS. In particular, this work proposes the study and application of combinatorial auctions to the distribution of tasks among agents in emergency response management scenarios. The preliminary results obtained so far have encouraged us to use combinatorial auction techniques to provide optimal solutions for this problem.

However other extensions of the current algorithms in the literature are necessary. First of all, a detailed computation cost analysis for the specific application we are dealing with must be studied. Second, it is necessary to tackle the task priorities and precedence relationships among tasks. And third, the dynamic feature of open environments in which the problem description changes at every moment, that is, new tasks arrive at any time, agents come in and out of the system, means that re-scheduling of tasks is necessary. The new method resulting from the proposed research should provide an effective solution to the distributed task allocation problem.

5.1List of publications

The work reported in this memory has been published in the following workshops and congresses:

1. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis, Del Acebo Esteve. "Integration Of Fuzzy Filtering, Case-Based Reasoning and Multiple Criteria Decision Techniques in Rescue Operations". In Proceedings Workshop of physical Agents (WAF) 2003. Alicante, Spain April 3, 4 and 5 of 2003.

2. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. "Co-operation strategies for strengthening civil agents' lives in the RoboCup-Rescue simulator scenario". In Proceedings: First International Workshop on Synthetic Simulation and Robotics to

Mitigate Earthquake Disaster. Associated to RoboCup 2003. Padova, Italy July 5 of 2003.

3. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “Girona-Eagles Rescue Team”. In Proceedings of the Internacional Symposium. Rescue Team Description Papers RoboCup 2003 (formato CD), Padua, Italia, 2003.

4. López, B., Suárez, S., De la Rosa, J.L. “Task Allocation in rescue operations using combinatorial auctions”. En: I. Aguilo, L. Valverde, M.T. Escrig (eds), Artificial Intelligence Research and Development, páginas 233-243, IOS Press, Frontiers in Artificial Intelligence and Applications 100, Netherlands 2003. ISBN 1-58603-378-6.

5. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “MCD Method for resource distribution in a large-scale disaster”. En: X Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA), volumen II, pág. 261-264, San Sebastián (España), 2003. ISBN 84-8373-564-4).

6. Suárez Silvia, López Beatriz, Meléndez Joaquim. "Towards holonic multiagent systems: Ontology for supervision tool boxes". In “Workshop de Agentes Inteligentes en el tercer milenio (CAEPIA 2003)”. November 11 of 2003. San Sebastián Spain.

7. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “The use of combinatorial auctions for resource and task allocation in disaster situations”. Report IIiA 2004.

8. Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “A Combinatorial Auction Algorithm for Dynamic and Unpredictable Environments. Application to a Disaster Scenario”. Report IIiA 2004.

9. López Beatriz, Suárez Silvia, De la Rosa Josep Lluís. Improving heuristics in a combinatorial auction mechanism for task allocation in disaster environments. Report IIiA 2004.

Chapter 6

6. REFERENCES

[1] Aamodt A., Plaza E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications 1994. IOS Press, Vol. 7: 1, pp. 39-59.

[2] Anderson Arne, Tenhunen Mattias, and Ygee Fredrik. Integer programming for combinatorial auction winner determination. In ICMAS 2000, pages 39-46.

[3] Arai Tamio, Izawa Hidemitsu, Maeda Yusuke, Kikuchi Haruka, Ogawa Hiroki and Sugi Masao. Real-Time Task Decomposition and Allocation for a Multi-Agent Robotic Assembly Cell. Assembly and Task Planning, 2003. Proceedings of the IEEE International Symposium on , 10-11 July 2003. Pages:42 – 47.

[4] Arai Tamio, Pagello Enrico and Parker Lynne. Editorial: Advances in Multi-Robot Systems. IEEE Transactions on Robotics and Automation, Vol. 18, No.5, October 2002: 655-661.

[5] Arnheiter Tim. Modeling and Simulation of an Agent-based Decentralized Two-Commodity Power Market. MultiAgent Systems, 2000. Proceedings. Fourth International Conference on , 10-12 July 2000. Pages:361 – 362.

[6] Babanov Alexander, Collins John and Gini Maria. Scheduling tasks with precedence constraints to solicit desirable bid combinations. In proceedings AAMAS'03, July 14-18, 2003, Melbourne, Australia.

[7] Baker Kenneth R., Elements of Sequencing and Scheduling. ISBN: 0-9639746-1-0 / 0963974610. Revised edition (January 1995).

[8] Beckers R., Holland O. E., and Deneubourg J.L.. From local actions to global tasks. Stigmergy and collective robotics. In R. A. Brooks and P. Maes, editors, Artificial Life IV: Proceedings of the Fourth International Workshop on the Systhesis and simulation of Living Systems, pag 181 – 189, 1994.

[9] Benchmarks research: http://mscmga.ms.ic.ac.uk/info.html and http://www.neosoft.com/~benchmrx/

[10] Bernstein Daniel S., Zilberstein Shlomo, and Immerman Neil. The complexity of Decentralized Control of Markov Decision Processes. In Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence, 2000. Pag. 32-37.

[11] Bonabeaun E., Dorigo M., and Theraulaz G. Swarm Intelligence. From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Pres, 1999.

[12] Chaib-draa, B., & Moulin, P., “Architecture for Distributed Artificial Intelligent Systems,” IEEE Proceedings, Montreal, pp. 64-69, 1987.

[13] Cicirello Vincent A. and Stephen Smith. Ant Colony Control for Autonomous Decentralized Shop Floor Routing. Fifth International Symposium on Autonomous Decentralized Systems. March 26 - 28, 2001 Dallas, Texas

[14] Craig Boutilier, Moisés Goldszmidt and Bikash Sabata. Sequential Auctions for the allocation of Resources with complementarities. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Pages: 527 – 523. Year of publication: 1999. ISBN:1-55860-613-0

[15] Escolano, F.R., Cazorla, M.A., Alfonso, M.I., Colomina, O., Lozano, M.A. Inteligencia artificial. Modelos, Técnicas y Áreas de Aplicación. Thomson, 2003.

[16] Fox Mark S. and Norman Sadeh. Why is Scheduling Difficult? A CSP Perspective. ECAI 1990.

[17] Fujishima Y., Leyton-Brown K., and Shoham Y. Taming the computational complexity of combinatorial auctions: Optimal and approximate approach. In Proceeding of IJCAI-99, Stockholm, 1999.

[18] Gonen Rica and Lehmann Daniel. Optimal Solutions for Multi-Unit Combinatorial Auctions: Branch and Bound Heuristics. EC’00, October 17-20, 2000, Minneapolis, Minnesota. 2000.

[19] Hoos Holger. Solving Combinatorial Auctions using Stochastic Local Search. American Association for Artificial Intelligence (www.aaai.org). 2000.

[20] http://www.agorics.com/Library/agoricpapers/ce/ce4.html.

[21] Hu Junling and Wellman Michael. Online Learning about Other Agents in a Dynamice Multiagent System. In proceedings of the Second International Conference on Autonomous Agents (Agents-98), Minneapolis, MN, USA, May 1998.

[22] Institutions research: http://www.iiia.csic.es/~jsabater/SLIE-web/EI.html.

[23] Jennings Nicholas and Wooldridge Michael. Agent Technology. Foundations, Applications, and Markets. ISBN 3-540-63591-2 Springer-Verlag Berlin Heidelberg New York.

[24] Jonker P.P., Caarls J., Bokhove W.J., Altewischer W., and Young I.T. RoboSoccer: autonomous robots in a complex environment, in: Wei Sui (eds.),

Proc. 2nd Int. Conf. Image and Graphics (Hefei, China, Aug.16-18), Proc. SPIE, vol. 4875, SPIE, Bellingham, WA, USA, 2002, 47-54.

[25] Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, R., Shinjou, A., Shimada, S. Robocup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research. Proc. 1999 IEEE Int. Conf. on Systems, Man and Cybernetics, Vol. VI, pp. 739-743, October, Tokyo, 1999 (SMC 99).

[26] Kuhn N., Muller H.J. and Muller J.P. Task decomposition in Dynamic Agent Societies. Autonomous Decentralized Systems, 1993. Proceedings. ISADS 93., International Symposium on ,30 March-1 April 1993. Pages:165 – 171

[27] Kyung-Mi Lee and Takeshi Yamakawa. A Genetic Algorithm for General Machine Scheduling Problems. Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on , Volume: 2, 21-23 April 1998. Pages:60 - 66 vol.2

[28] Laddaga Robert. Tolerant Software. A White paper for the Workshop on New Visions for Software Design and Productivity.

[29] Lesser Victor R. Cooperative Multiagent Systems: A personal view of the state of the art. IEEE transactions on knowledge and data engineering, vol. 11. No. 1, January/February 1999.

[30] Leyton-Brown Kevin, Shoam Yoav and Tennenholt Moshe z. An algorithm for Multi-Unit Combinatorial Auctions. American Association for Artificial Intelligence (www.aaai.org). 2000.

[31] Li Li, Smith Stephen F. SpeculationAgents for Dynamic Multi-period Contiuous Double Auctions in B2B Exchanges. Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 7. January 05 - 08, 2004 Big Island, Hawaii.

[32] López, B., Suárez, S., De la Rosa, J.L. "Task Allocation in rescue operations using combinatorial auctions". En: I. Aguilo, L. Valverde, M.T. Escrig (eds), Artificial Intelligence Research and Development, páginas 233-243, IOS Press, Frontiers in Artificial Intelligence and Applications 100, Netherlands 2003. ISBN 1-58603-378-6.

[33] Makoto Yokoo and Katsutoshi Hirayama. Frequency Assignment for Cellular Mobile Systems Using Constraint Satisfaction Techniques. Proceedings of the IEEE Annual Vehicular Technology Conference (VTC2000-Spring), 2000.

[34] Makoto Yokoo, Durfee Edmund, Ishida Toru, and Kuwabara Kazuhiro. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. IEEE Trans. On Knowledge and DATA Engineering, vol. 10, No.5 September 1998.

[35] Marik Vladimír, Stepankova Olga and Pechoucek Michal. Acquaintance Models for Integration-oriented Multi-agent Systems. Proceedings. XXI International Conference of the Chilean , 7-9 Nov. 2001. Pages: 186 – 192.

[36] Michal Pechoucek, Marik Vladimir and Stepankova Olga. Coalition Formation in Manufacturing Multi-Agent Systems. Proceedings. 11th International Workshop on, 4-8 Sept. 2000. Pages:241 – 246.

[37] Montaner M. López B. De la Rosa Joseph LL. Improving Case Representation and Case-Based Maintenance in Recommender Agents. ECCBR'02

[38] Morimoto Takeshi, Kenji Kono, and Ikuo Takeuchi, YabAI. The first Rescue Simulation League Champion. In proceedings of Rococup 2001, pp. 49-59.

[39] Morimoto Takeshi, Manual: How to develop a RoboCup Rescue Agent, for RoboCupRescue Simulation System version 0, 1st edition.

[40] Murray, R.M., Aström, K.J., Boyd, S.P., Brockett, R.W., Stein, G. Future Directions in Control in a Information-Rich World. A summary of the report of the Panel on Future Directions in Control, Dynamics, and Systems. IEEE Control Systems Magazine, April 2003, pp. 20-33.

[41] Oliver G. and Guerrero J. Multi-Robot Task Allocation Strategies using Auction-like Mechanisms. En: I. Aguilo, L. Valverde, M.T. Escrig (eds), Artificial Intelligence Research and Development, páginas 111-121, IOS Press, Frontiers in Artificial Intelligence and Applications 100, Netherlands 2003. ISBN 1-58603-378-6.

[42] Ossowski Sascha, Co-ordination in Artificial Intelligence Agent Societies. Social Structures and Its Implications for Autonomous Problem-Solving Agents. Lecture notes in computer science; 1535: Lecture notes in artificial intelligence. ISBN 3-540-65495-X. Springer, 1999.

[43] Peng Jin, Kaoping Song. Fuzzy Flow-Shop Scheduling Models Based on Credibility Measure. Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE International Conference on , Volume: 2 , 25-28 May 2003. Pages:1423 - 1427 vol.2 pp. 3-55, 1996.

[44] Rassenti A.J., Smith V.L., and Bulfin R.L. A Combinatorial Auction Mechanism for Airport Time Slot Allocation. In Proceedings of Autumn 1982. Pp. 402-417.

[45] REPAST home: http://repast.sourceforge.net/.

[46] RoboCup-Rescue Official Web Page. http://www.r.cs.kobe-u.ac.jp/robocup-rescue/

[47] RoboCup-Rescue Simulator Manual-Versión 0 rev. 4

[48] Rodríguez-Aguilar Juan A. Giovanucci Andrea, Reyes-Moro Antonio, Noria Francesc X., Cerquides Jesús. Agent-based decision support for actual-world

procurement scenarios. IEEE/WIC International Conference on Intelligent Agent Technology October 13 - 17, 2003. Halifax, Canada.

[49] Rodríguez-Aguilar Juan A., Martin Francisco J., Noriega Pablo, Garcia Pere, Sierra Carles. Competitive Scenarios for Heterogeneous Trading Agents. Agents'98. Minneapolis May 10-13(1998).

[50] Rothkopf Michael, Pekec Alesandar. Computationally Manageable Combinatorial Auctions. DIMACS Technical Report 95-09 April 1995.

[51] Salido Miguel A. and Barber Federico. Distributed Hard and Soft Non-binary constraints: An Any-Time Proposal. X Conference of the Spanish Association for Artificial Intelligence (CAEPIA), ISBN: 84-8373-564-4, Vol(1), pp:449-458 San Sebastian, 2003.

[52] Salido Miguel A. Giret Adriana, and Barber Federico. Distributing Constraints by Sampling in Non-Binary CSPs. In Proceeding of IJCAI International Workshop on Distributed Constraint Reasoning. Acapulco, México, 2003.

[53] Sandholm Tuomas, Algorithm for optimal winner determination in combinatorial auctions. Artificial Intelligence, 135, 1-54. (Early version appeared as an invited talk at ICE-98, as a tech report 1/99, and as a paper in the Proceedings of IJCAI-99

[54] Sandholm Tuomas. Improved algorithm for optimal winner determination. Generalizations. Artificial Intelligence, 145, 33-58. Early version: Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations. In Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 90-97, Austin, TX, July 31-August 2, 2000.

[55] Sandholm Tuomas. Negotiation self-interested computationally limited agents. PhD Dissertation. Septembre 1996.

[56] Sandholm Tuomas. Winner determination in Combinatorial Auctions Generalizations. AAMAS'02 july 15-19, 2002, Bologna, Italia.

[57] Sandström Kristian, Eriksson Christer, and Fohler Gerhard. Handling Interrupts with Static Scheduling in an Automotive Vehicle Control System. Real-Time Computing Systems and Applications, 1998. Proceedings. Fifth International conference on ,27-29 Oct. 1998.

[58] Shaheen Fatima, Uma G, Perraju Siva. TRACE – An adaptive Organizational Policy for Multi Agent Systems. MultiAgent Systems, 2000. Proceedings. Fourth International Conference on , 10-12 July 2000. Pages:383 – 384.

[59] Smith David E., Frank Jeremy, Jónsson Ari K. Bridging the Gap Between Planning and Scheduling. Knowledge Engineering Review, 15(1), 2000.

[60] Smith Stephen. Coordinating Activity in Knowledge-Intensive Dynamic Systems. KIMAS 2003, October 1-3, 2003, Boston, MA, USA

[61] Sqalli Mohammed H., Purvis Lisa, Freuder Eugene C. Survey of Applications Integrating Constraint Satisfaction and Case-Based Reasoning. 1999

[62] Suárez Silvia, López Beatriz, De la Rosa Josep Lluis, Del Acebo Esteve. "Integration Of Fuzzy Filtering, Case-Based Reasoning and Multiple Criteria Decision Techniques in Rescue Operations". In Proceedings Workshop of physical Agents (WAF) 2003. Alicante, Spain April 3, 4 and 5 of 2003.

[63] Suárez Silvia, López Beatriz, Meléndez Joaquim. "Towards holonic multiagent systems: Ontology for supervision tool boxes". In “Workshop de Agentes Inteligentes en el tercer milenio (CAEPIA 2003)”. November 11 of 2003. San Sebastián Spain.

[64] Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “A Combinatorial Auction Algorithm for Dynamic and Unpredictable Environments. Application to a Disaster Scenario”. Report IIiA 2004.

[65] Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “Girona-Eagles Rescue Team”. In Proceedings of the Internacional Symposium. Rescue Team Description Papers RoboCup 2003 (formato CD), Padua, Italia, 2003.

[66] Suárez Silvia, López Beatriz, De la Rosa Josep Lluis. “The use of combinatorial auctions for resource and task allocation in disaster situations”. Report IIiA 2004.

[67] Tesauro Gerald, Das Rajarshi. HighPerformance Bidding Agents for the Continuous Double Auction. EC’01, October 1417, 2001, Tampa, Florida, USA.

[68] Uchibe Eiji, Kato Tatsunori, Asada Minoru, Hosoda Koh. Dynamic Task assignment in a Multiagent/Multitask Environment based on Module Conflict Resolution. Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on , Volume: 4 , 2001. Pages:3987 - 3992 vol.4.

[69] Walsh William E., Wellman Michael P., and Ygge Fredrik. Combinatorial Auctions for Supply Chain Formation. EC’00, October 17-20, 2000, Minneapolis, Minnesota.

[70] Walsh William, Wellman Michael, Wurman Peter and MacKie-Mason Jeffrey. Some Economics of Market-Based Distributed Scheduling. In proceedings of the Eighteenth International Conference on Distributed Computing Systems (ICDCS-98), Amsterdam, The Netherlands, May 1998.

[71] Wang Changjie and Leung Ho-fung. Anonymity and Security in Continuous Double Auctions for Internet Retails Market. Proceedings of the 37th Hawaii international Conference on System Sciences – 2004.

[72] Watson Jean-Paul, Barbulescu Laura, Howe Adele E., and Whitley L. Darrell. Algorithm Performance and Problem Structure for Flow-shop Scheduling. American Association for Arti_cial Intelligence. 1998.

[73] Wei-Chou Chen; Shian-Shyong Tseng; Jin-Huei Chen; Mon-Fong Jiang. A framework of features selection for the case-based reasoning Systems, Man, and Cybernetics, 2000 IEEE International Conference on, Volume: 1 , 2000. Page(s): 1 -5 vol.1.

[74] Weiss Gerhard. Multiagent Systems: a modern approach to distributed artificial intelligence, MIT Press 1999.

[75] Wellman Michael P. A computational Market Model for Distributed Configuration Design. In proceedings of EDAM 1995.

[76] Wellman Michael P., and Wurman Peter R. Market-Aware Agents for a Multiagent World. Based on an invited talk at MAAMAW-97.

[77] Wellman Michael P., Walsh William E., Wurman Peter R. and MacKie-Mason Jeffrey K. Auction Protocols for Descentralized Scheduling. Games and Economic Behaviour. Revised and extended version of “Some economics of market-based distributed scheduling”, presented at the Eighteenth International Conference on Distributed Computing Systems, Amsterdam, May 1998.

[78] Wellman Michael, A market-oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems. Journal of Artificial Intelligence Research 1 (1993) 1-23.

[79] Wellman Michael. A General-Equilibrium Approach to Distributed Transportation Planning. in Proceedings AAAI-92, San Jose, CA, 1992, pp. 282—290

[80] Wooldridge Michael, Ciancarini Paolo. Agent-oriented software engineering. Handbook of Software Engineering and Knowledge Engineering Vol. 0, No. 0 (1999) 000-000. World Scientific Publishing Company.

[81] Wooldridge Michael. An Introduction to MultiAgent Systems. Published in February 2002 by John Wiley & Sons (Chichester, England). ISBN 0 47149691X.

[82] Wu Zhiming and Zhao Chunwei. A Genetic Algorithm for Job Shop Scheduling in Real Time. American Control Conference, 1997. Proceedings of the 1997, Volume: 1 , 4-6 June 1997. Pages:162 - 163 vol.1.

[83] Wurman Peter R., Wellman Michael P., and Walsh William. The Michigan Internet AuctionBot: A configurable Auction Server for Human and Software Agents. In proceedings of the second international Conference on Autonomous Agents (Agents-98), Minneapolis, MN, USA, May 1998.

[84] Wkatsutoshi Hirayama, Makoto Yokoo. An Approach to Over-constrained Distributed Constraint Satisfaction Problems: Distributed Hierarchical Constraint Satisfaction. In proceedings of ICMAS 2000.

[85] Yager R.R.. On Ordered Weighed Averaging Aggregation Operators in Multi-criteria Decision Making. IEEE Transactions on SMC, volume 18, 1988.