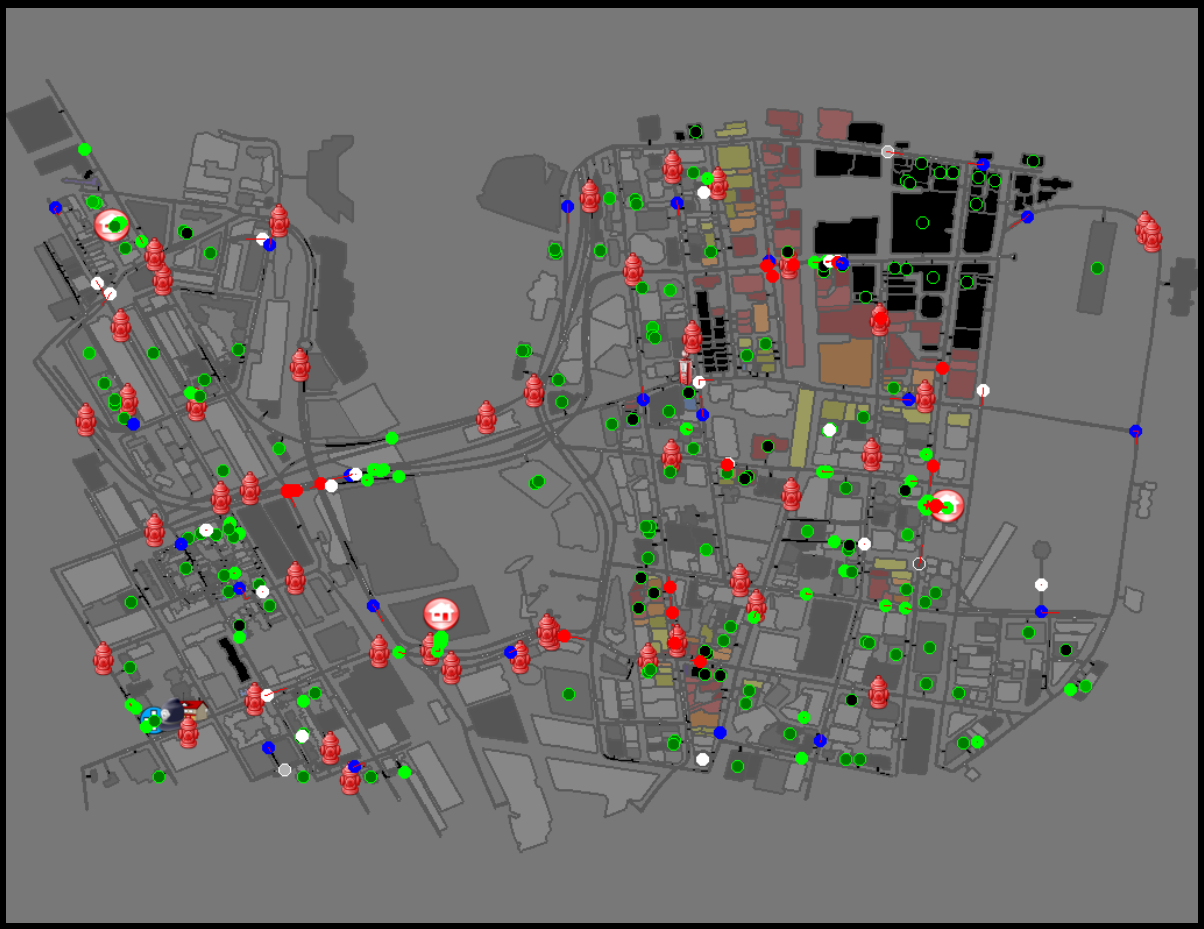
RoboCup Rescue Agent Simulation: Max-Sum as a Decentralized Solution to the Distributed Constraint Optimisation Problem with the use of the AIT-extension



Jesper van Duuren

Layout: typeset by the author using LaTeX.
Cover illustration: Snapshot RoboRescue Agent Simulation Sydney

RoboCup Rescue Agent Simulation: Max-Sum as a Decentralized Solution to the Distributed Constraint Optimisation Problem with the use of the AIT-extension

Jesper van Duuren
10780793

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Jan 31st, 2020

Contents

Abstract

In the Robocup Rescue Agent Simulation (RRS) the objective is to save victims and buildings in a simulated disaster to achieve the highest score. Many challenges are faced simultaneously in this multi-agent system. One of the major challenges is the assignment of tasks to agents in the field, also knows as target allocation. The target allocation of agents can be modelled as a Distributed Constraint Optimization Problem (DCOP). The standard RRS is however not suitable for algorithms like the Max-Sum algorithm for solving the DCOP. This is due to the restrictions in communication. Therefore, the AIT-extension is used to bypass this problem and effectively apply the Max-Sum algorithm in a decentralized manner. This thesis finds that the decentralized Max-Sum algorithm has equal performance to the centralized Max-Sum algorithm that is dependent on performance of the central-agent. Therefore, it is concluded that the existence of the central-agent is not always necessary to converge to a near optimal solution.

1 Introduction

In the early morning of January 15, 1995, Kobe was hit by the Great Hanshin-Awaji earthquake. 6,308 people were killed and 35,000 were injured [1]. About 400,000 buildings were damaged by the earthquake, among others medical facilities [1]. In his research, Takashi Ukai found that the disaster damaged many roads, highways, bridges and railways, making it difficult to move around the city of Kobe after the earthquake [1]. Moreover, communication via telephone was lost due to disconnections and overloading of the network. Traffic congestion and the loss of telephone communication made it difficult to move severely injured people out of the disaster area to medical facilities in safe zones. All in all, these problems led to the inefficient use of emergency services, enabling fires to spread and resulting in a lack of immediate medical care for injured people.

In 2001, in the aftermath of the Great Hanshin-Awaji earthquake, RoboCup Rescue Agent Simulation League was initiated. The main purposes of the competition are to make structures of the disaster relief problem visible, resolve potentially hidden problems, develop new algorithms and to contribute to practical problems [2].

1.1 Robo Rescue Agent Simulation (RRS)

RoboCup Rescue Agent Simulation (RRS) simulates disastrous events such as an earthquake [2]. During and right after such events, roads can be blocked due to collapsed buildings, fires can break out and victims tend to be scattered around the city. In the RRS-event, agents are divided by the task they can perform. Fire brigade agents can control fires around the city, ambulance agents can rescue victims and police force agents can clear obstructed roads.

Cities in the RRS are based on either real or fictional cities. The cities are typically formed by buildings and roads. Buildings are divided into two categories. First, there are refuges that provide a safe harbor for citizens and where fire brigades can resupply their water. Second, there are gas stations that form a potential hazard to the surrounding if they catch fire.

The RRS is equipped with a special fire simulator, that simulates fire break-outs depending on several variables such as building height, construction material and map layout. A building can either be not involved in a fire, heating, burning or inferno. The RRS rewards points for every building saved and keeps track of the rate at which fires are spreading. The fire brigades are able to carry a limited amount of water to extinguish the fires. Their water supply can be refilled at refuges or pumping stations. The rate at which fires are extinguished will improve when fire brigade agents work together. Such teamwork is particularly helpful in case of a severe, fast-spreading fire where a single fire brigade agent won't suffice.

The RRS also contains a collapse simulator that simulates the collapse of buildings. This simulator takes into account the intensity of the earthquake combined with building characteristics and fire-state of the building. The collapse of a building will result in a blocked road. This blocked road can only be cleared by police forces. The speed of clearing a single part of the road will not increase when police forces work together.

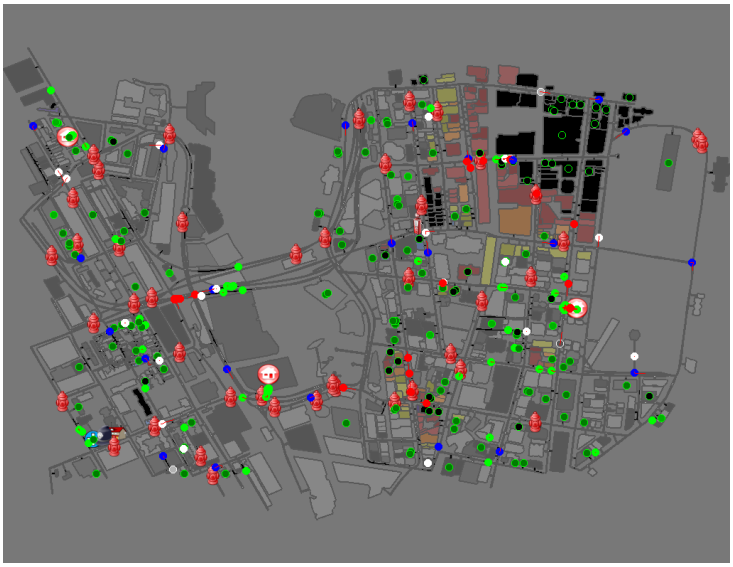


Figure 1: A simulation in RRS

Civilians can be covered by rumble and/or get injured as a result of the earthquake simulated in the RRS-event. Civilians who are not covered by rumble and remain uninjured, will move to a refuge on their own. Civilians always move at a slower pace than emergency agents in the field.

If a civilian is covered by rumble and/or injured, an ambulance agent is needed to perform a rescue. Ambulance agents are able to work together to speed up the process of rescuing a victim from the scene. Depending on the injury, the victims' hit-points deplete while waiting for help from an ambulance agent. When the number of hit-points equals zero, the victim died. The RRS keeps track of the number of civilians alive and subtracts points for each civilian losing his life.

Agents are able to communicate with other agents that are nearby. In addition, radio communication with the central agent is possible. The central agent is an immobile agent with the ability to process information received from the kernel of the RRS and communicate with agents in the field. However, depending on the particular scenario the agent is in, the number of radio channels is limited. Moreover, potential noise in communication strands can lead to unreceived or unintended messages.

1.2 Decentralized Coordination

In the event of a disaster, emergency services needs to resolve many challenges at the same time. In the decision making process, multiple factors need to be taken into account. First, it is vital to keep in mind that emergency services are probably outnumbered by the number of tasks. Second, there are different problems that need to be solved by determined agents. Third, the environment in the RSS is highly dynamic, meaning that some tasks will disappear, while new tasks will be added or requirements for existing tasks will change [3].

To overcome the challenges described above, it should be noted that coalitions of different agents need to be formed. To prevent single point failure of the system, such coalitions should be formed in a

decentralized way. In addition, long range communication should be minimized since resources for such type of communication is most likely damaged and or limited as a result of the given disaster [3].

The RRS simulates emergency agents in a multi-agent system (MAS) [4]. One of the main models to control the agents behaviour in a MAS is Distributed Constraint Optimization Problem (DCOP) [5]. To solve such a DCOP, the Max-Sum algorithm is one of the many suitable solutions [6]. It is a popular algorithm that can be modified in several ways. Thus, many variations on the Max-Sum algorithm exist: Binary-Max-Sum [7] [8], Lazy-Max-Sum [9], F-Max-Sum [3], Bounded-Max-Sum [10] [11] and many more [5]. The Max-Sum algorithm provides a decentralized message-passing solution. This means that it effectively adapts to a dynamic environment with limited communication [3]. In section 2.6, the precise mechanism of the algorithm is explained.

1.3 Aim

The current RRS makes it difficult to apply a DCOP algorithm for task allocation [12]. Therefore, Miyamoto et al. [12] proposed an extension to the current RRS which makes it possible to communicate multiple times within each time step. This addition of communication makes it possible to effectively apply a DCOP algorithm [12]. This extension is referred to as the AIT-extension.

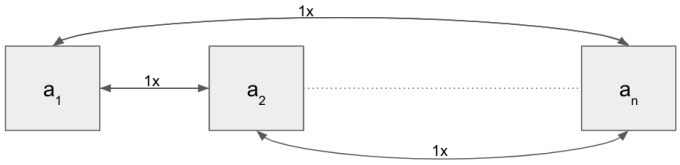


Figure 2: Communication abilities in the normal RRS

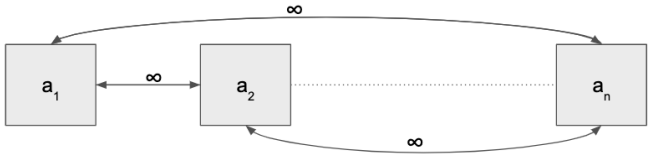


Figure 3: Communication abilities when AIT-extension is enabled

AIT [12] has used Max-Sum in a centralized manner, but not in a decentralized way. The question can be raised if the Max-Sum can perform as good in a decentralized way as in a centralized manner with the use of the AIT-extension. Therefore, this thesis will provide an answer to the following research question: *How does the Max-Sum algorithm perform when used in a decentralized manner in the RoboCup Rescue Agent Simulation with the AIT-extension enabled?*

This question can be divided into two sub-questions which gives insight into the mechanisms of the algorithm. First, how does the Max-Sum algorithm perform in a centralized manner? Second, how does the Max-Sum algorithm perform in a decentralized manner?

2 Theory

This chapter elaborates on the algorithms and theory used throughout this thesis. First, the general concepts of multi-agent systems and constraint programming are described. Second, the theory and definition of a distributed constraint optimization problem are discussed. Finally, the mechanism of the Max-Sum algorithm is explained and how factor graphs are useful for this algorithm.

2.1 Multi-Agent Systems (MAS)

In the RRS, multiple autonomous agents interact with each other to rescue victims and save buildings. The autonomous behavior of agents and interaction with other agents are the two main characteristics of a multi-agent system (MAS) [13]. An example of interacting mechanisms of agents is the clearing of roads by police agents so that ambulance agents can reach their destination faster.

2.2 Constraint Programming

Within the field of constraint programming, the objective is to solve decision-making problems by programming constraints [14]. The programming of constraints prevents the search for variables that have inconsistent constraints with respect to the objective: σ^*, an optimal assignment.

2.3 Constraint Satisfaction problem (CSP)

In a constraint satisfaction problem (CSP), the objective is to assign variables to values with the use of the notion of constraints [5]. These constraints provide insight into the more optimal relation between variables and values.

A CSP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where [5] [15] [16]:

- $\mathcal{X} = \{x_1, \ldots, x_n\}$, a set of variables.
- $\mathcal{D} = \{D_1, \ldots, D_n\}$, a set of domains, where D_i corresponds to the set of values that x_i can take.
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ denotes m sets of constraints where C_i is a subset of k variables so that $\{x_{i_1}, \ldots, x_{i_k}\}$.

The goal is to find a complete set of assignments σ^* that include and solve all problem constraints.

2.4 Constraint Optimization Problem (COP)

In some cases it might not be necessary to calculate a complete assignment in the CSP. A good solution for this is a violation degree, so the rules of a complete assignment can be violated in a systematic manner [5]. This feature comes with the weighted constraint satisfaction problem (WCSP), also known as the constraint optimization problem (COP) [5].

A COP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{F} \rangle$, and like a CSP, $\mathcal{X}$ denotes a set of variables and $\mathcal{D}$ denotes a set of domains associated with $\mathcal{X}$. $\mathcal{F}$ however, is a set of cost functions, where $f_i(x^i) \in \mathcal{F}$ denotes the cost of the connection of variables x^i, that is a subset of $\mathcal{X}$ with k variables, $x^i = \{x_{i_1}, \ldots, x_{i_k}\}$.

The objective is to calculate an optimal solution to solve the COP. The optimal solution is a solution with the lowest sum of cost of each assignment. The cost of an assignment σ is calculated by taking the sum of all costs associated with σ.

2.5 Distributed Constraint Optimization Problem (DCOP)

The distributed constraint optimization problem (DCOP) adds a decentralized message passing solution to the COP. It makes sure a near optimal solution can be found even if the central communication is not existent.

A DCOP is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$ where $\mathcal{X}$, $\mathcal{D}$ and $\mathcal{F}$ have the same function as in the COP [5]. $\mathcal{A}$ and α represent the distributed elements. $\mathcal{A}$ is a set of autonomous agents $\{a_1, \ldots, a_n\}$ and $\alpha : \mathcal{X} \to \mathcal{A}$, that assigns the variables $x \in \mathcal{X}$ to an agent $\alpha(x)$.

The formal definition of a DCOP is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{F}, \alpha \rangle$ [5] [17] where:

- $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is a set of agents, where a_i is an agent.
- $\mathcal{X} = \{x_1, x_2, ..., x_m\}$ is a set of variables and x_i represents a task selected by an agent $a_i \in \mathcal{A}$.
- $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$ is a set of domains for variable $\mathcal{X}$, where $\mathcal{D}_i$ is a set of possible values for x_i.
- $\mathcal{F} = \{f_1, f_2, ..., f_k\}$ is a set of cost functions, where $f_i \in \mathcal{F}$ denotes the cost of the connection of value x_i of variable $x_i \in \mathcal{X}$.
- $\alpha : \mathbf{X} \to \mathbf{A}$ is the assignment of variable $x \in \mathcal{X}$ to an agent $\alpha(x)$

In a DCOP, the goal it to minimise the cost while finding an optimal solution to problem. This can be summarized in the following function:

$$\sigma^* := \arg\min_{\sigma \in \Sigma} \sum_{f_i \in \mathcal{F}} f_i(\sigma_{x^i}) \tag{1}$$

where:

- Σ is the state space.
- $f_i(\sigma_{x^i})$ is the cost function for assignment σ.

Over the last decade DCOPs have evolved making them suitable for a wide range of problems. One of the developments, what makes it suitable for the problems in the RRS, is the adaptation to dynamic, complex and real-time environments.

2.6 Max-Sum Algorithm

There are many algorithms that can solve a DCOP. However, not every algorithm is suitable for the dynamic environment of the RRS and the challenges is poses.

The Max-Sum algorithm is an algorithm with the characteristics to tackle the target allocation problems in the RRS [5]. It is an incomplete algorithm, meaning that it stops searching when a solution is found although it might not be the most optimal solution. Moreover, the Max-Sum algorithm has a synchronous feature, which forces agents to communicate with other agents before they make a decision. Lastly, the algorithm is inference-based on belief propagation, which allows agents to explore the structure of the constraints so cost can be reduced. These features make the Max-Sum algorithm a good algorithm for tackling the target allocation problems in the RRS [5].

2.6.1 Factor Graphs

The Max-Sum algorithm can be better explained using a factor graph for a better understanding [18]. A factor graph is a bipartite graph that contains two kinds of nodes: factor nodes and variable nodes [18]. The two types of nodes can be connected with each other. However, a node cannot be connected to a node of the same type. An example of a factor graph is given in figure 4.

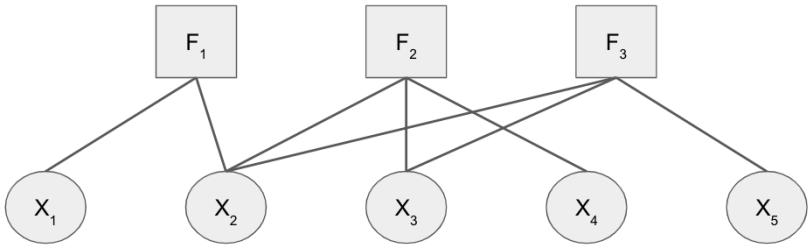


Figure 4: A factor graph of the Max-Sum algorithm: $max(f_1(x_1, x_2) + f_2(x_2, x_3, x_4) + f_3(x_2, x_3, x_5))$

2.6.2 Passing Messages

The Max-Sum algorithm is the derivative of the Sum-Product algorithm. This is a message passing algorithm that sends message back and forth over the edges of a factor graph. Depending on the direction of a message, a special function is used to calculate the message. When a message is send from a variable node to a factor node, it is denoted by $\mu_{x\to f}(x)$:

$$\mu_{x\to f}(x) = \sum_{g\in N(x)/f(x)} \mu_{g\to x}(x) \tag{2}$$

and when a message is send from a factor node to a variable node, is denoted by $\mu_{f\to x}(x)$:

$$\mu_{f\to x}(x) = \max_{Y}(f(x, Y) + \sum_{y\in Y} \mu_{y\to f}(y)) \tag{3}$$

The propagation of messages is a recursive process that repeats itself multiple times. These repetitions or cycles are needed to calculate a near optimal solution to the problem.

2.6.3 Loopy Belief Propagation

Factor graphs of the Max-Sum algorithm can have cycles in itself (see Figure 4). Due to these cycles, information flows around endlessly, while an optimal solution is not guaranteed. This problem is also known a loopy belief propagation [18].

Loopy belief propagation causes logistical challenges in the passing of messages in the factor graph. Therefore, a message passing schedule is needed, so that messages do not accidentally mix up with other messages. Some messages need to queue for some time whilst other can already be send. The problem is that in some cases the algorithm cannot terminate since there are always pending message. Therefore it sometime necessary to force the algorithm to terminate manually, that can be done by setting a maximum number of iterations for sending messages. The manual termination of the algorithm can have effect on the costs of the final solution [18].

3 Approach

The following chapter will discuss some of the problems ambulance agents can come across in the RRS. Moreover, it will be presented how these problems can be modelled as a DCOP. Finally, the benefits of using the Max-Sum algorithm and the AIT-extension as solutions to these problems will be discussed.

3.1 Ambulance Team Challenges

In the RRS, ambulance agents are able to rescue injured people and transport them to safe shelters and hospitals. In most events, the number of injured people will outnumber the number of ambulance teams available, since teams can only carry a maximum of one victim each time. It often happens that victims are left behind in the dangerous environment when the simulator ends. If these victims are still alive, the simulator will award points to every civilian. This element should be taken into account in the decision making process.

An ambulance agent is capable of several actions: it can search the environment for victims, dig out a victim, pick up a victim or drop a victim off. To determine what is the best decision to make, several variables are available to the agent: distance to refuge, distance to victim, distance to unexplored area, victims on board, victims in range.

3.2 Task assignment as DCOP

To approach an optimal solution in the allocation of tasks to rescue agents in the field, the Max-Sum algorithm uses a combination of communication and evaluation [5].

Evaluation happens with a special evaluation function that generates a score based on the efficiency of the task allocated to the agents in the field. If agents can be used in a more efficient way, tasks can be reassigned. Such reassignments can only be executed if the new task will increase the sum of the evaluation score.

As the Max-Sum algorithm is based on a DCOP, the modelling of such task assignment can be set out as follows:

- $\mathbf{A} = \{a_1, ..., a_n\}$ is a set of ambulance agents where a_i is a single agent.
- $\mathbf{X} = \{x_1, ..., x_n\}$ is a set of variables where x_i represents a task selected by an agent.
- $\mathbf{D} = \{D_1, ..., D_m\}$ is a set of tasks that an agent a_i can select from D_j. As previously discussed, ambulance agents can perform three main tasks: pick up a civilian, drop a civilian off, dig out a civilian and search for casualties.
- $\mathbf{F} = \{f_1, ..., f_k\}$ is a set of cost functions that is used for the evaluation of task assignments. The cost of an assignment needs to be calculated to determine if the assignment is an optimal assignment with a low set of costs. The final number of civilians alive is the most important parameter. It is

thus necessary to keep the costs of rescuing a civilian as low as possible. The objective function $\mathbf{F}_g(\mathbf{X})$ illustrates the total costs made and consist of two parts:

1. The calculation of the distance to a victim and thus time required to reach a victim. This is done by taking the coordinates of the victim and the agent, and calculate the direct distance using the Pythagorean theorem. This value is than divided by a predefined constant τ, that is an estimate for the distance an agent can move per time-step. In the event that an agent performs a search task, the cost are set to zero.

$$C(a,d) = \begin{cases} \frac{\sqrt{(X_a - X_d)^2 + (Y_a - Y_d)^2}}{\tau} & \text{(if } d \text{ is a civilian rescue task)} \\ 0 & \text{(if } d \text{ is a situation search task)} \end{cases} \tag{4}$$

2. To prevent that agents are constantly reassigned, an assignment penalty is required. A special constant ρ is used to represent this penalty.

$$P(d,n) = \begin{cases} \rho\left\{1 - \left(\frac{min(REQ(d),n)}{REQ(d)}\right)^2\right\} & \text{(if } d \text{ is a civilian rescue task)} \\ 0 & \text{(if } d \text{ is a situation search task)} \end{cases} \tag{5}$$

In the function above, $REQ(d)$ consists of a combination of the buried depth of the civilian (BD_d), the loss rate of physical strength of the civilian (DT_d) and the remaining physical strength of the civilian (HP_d). This function is defined as follows:

$$REQ(d) = \frac{BD_d \times DT_d}{HP_d} + 1 \tag{6}$$

$C(a,d)$ and $P(d,n)$ are combined in function 7. In the first part of this equation, the sum of the travel costs to all civilians is used. In the second part of the equation, all assignment costs are taken together. These two separate parts are than added up and together make up the cost function.

$$F_g(x) = \sum_{x_i \in X} C(\alpha(x_i), x_i) + \sum_{d \in \bigcup_{i=1}^{n} D_i} P(d, |\{x_i | x_i = d \wedge x_i \in X\}| \tag{7}$$

- $\alpha : \mathbf{X} \rightarrow \mathbf{A}$ defines the function $a_i \in \mathbf{A}$ that manages variable $x_i \in \mathbf{X}$. This makes sure an agent can only be assigned to one task only.

3.3 Limitations

In the standard simulation, each agent in the field has the ability to communicate one message with another agent at the time. In such case, it is assumed that the message receiving agent is located within the communication range of the message sending agent. The AIT-extension bypasses the single message per time-step and provides an individual agent with the ability to communicate an infinite amount of messages with all other agents that are located within the communication range of the agent itself.

The AIT-extension by [12], at the moment of writing, does not provide support for pseudo-communication by the central agent. The extension only provides support for pseudo-communication by agents individually, meaning that the Max-Sum algorithm can only be used in a decentralized manner. Additionally, due to the architecture of the program, a factor graph must be made for each individual agent resulting in high computational cost and RAM usage. Testing several maps exceeds a RAM usage of 25GB. Due to the limitation of the testing machine used in this study, maps exceeding 25GB are not tested.

3.4 Max-Sum Implementation

The communication ability between agents defines which nodes are connected in the biparite graph and which are not. The emergency services are represented as variable nodes and utility functions as factor nodes. Al emergency services which are able to communicate directly or indirectly with each other are connected with a single factor node. An example of this is shown in figures 5 and 6.

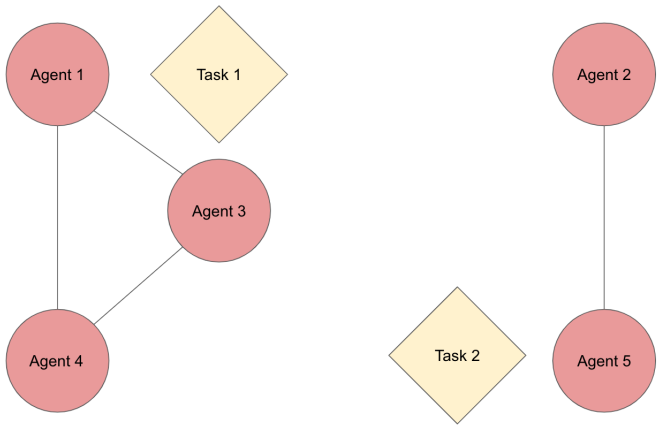


Figure 5: Example of communication availability between agents

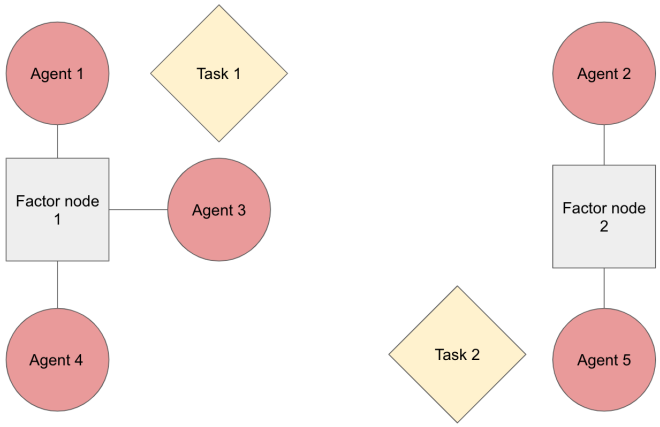


Figure 6: Factor graph representation of problem from figure 5

In the example of figure 5 and 6, two separate groups of agents are represented. The two groups of agents are unable to communicate with each other, but internally agents are able to share information.

For example, agent 1 and 3 are able to detect task 1 and communicate this toward agent 4. As such, they are able to calculate which agent will rescue task 1. To communicate, they are connected via one factor node. The same applies for agent 2 and 5, where agent 5 detects task 2 and is able to communicate this towards agent 2.

In this study, the Max-Sum algorithm is implemented by creating a DCOPHumanDetector. For every agent, DCOPHumanDetector is called repeatedly and for each agent a factor graph is created to find a solution to the DCOP. Messages are send and received repeatedly until the maximum number of iterations is reached or when there are no more pending messages available. The maximum number of iterations is a predefined parameter that prevents endless loopy belief propagation, a situation where there are always pending messages available.

4 Experiments

In this section, experiments are described and discussed to test the effectiveness of the approach. First, a baseline is set using the submitted code by AIT for the 2018 RRS competition where AIT uses a Max-Sum algorithm for the target allocation of agents. Second, the performance of the decentralized Max-Sum algorithm, discussed in this thesis, is measured in terms of points awarded by the simulation. Third, the maximum number of iterations the Max-Sum algorithm can use to converge to a solution is checked. Finally, performance is measured when the communication range of agents changes so a conclusion can be made about the effectiveness of the communication range of agents on the Max-Sum algorithm.

Each simulation is done on the RRS without pre-computation and repeated three times to reduce the variability that occurs in each simulation. This variability is a result of the build in noise generated by the simulator, and the decision making process after that. For example, a different task can be assigned to an agent due to noise that disrupted another message.

The scenario's chosen for the experiments are **SydneyS2**, **SF** and **Berlin**. Each scenario differs in population density, number of emergency services and severity of the simulated disaster. Below, each scenario is described and the most important characteristics are mentioned.

4.1 Scenario's

As shown below in table 1 and figure 7a, **SydneyS2** has civilians and emergency forces scattered around the city with a relative high population density. Additionally, refuges and water-refill points are available and distributed relatively equally what implies that agents in field are not always obligated to travel major distances to their destination.

Intial Score	Civilians	Ambulance teams	Refuges
352	351	25	3

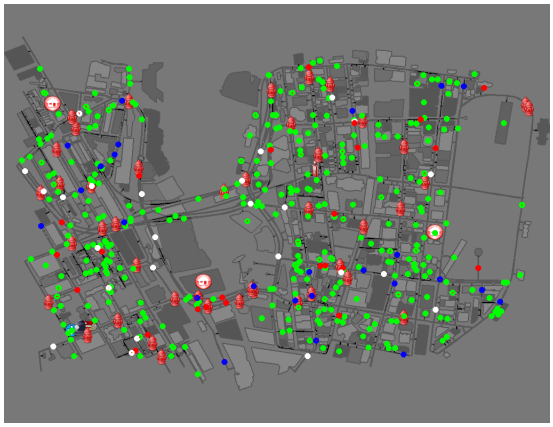
Table 1: Parameters of **SydneyS2**

SF is a map with a similar size to **SydneyS2** but is has 130 civilians, a much smaller population. This makes the map less dense populated and might result in fewer communications of field agents since agents will be out of range of each other unable to communicate.

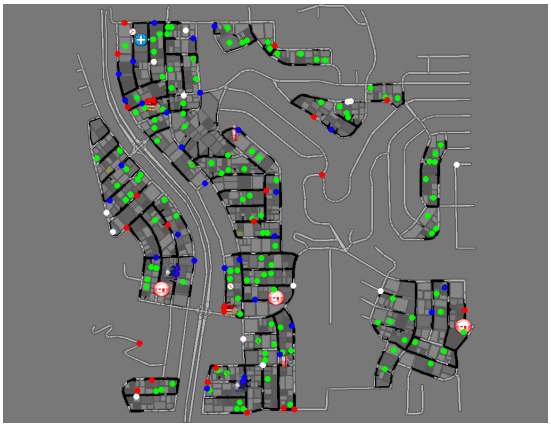
Intial Score	Civilians	Ambulance teams	Refuges
131	130	14	3

Table 2: Parameters of **SF**

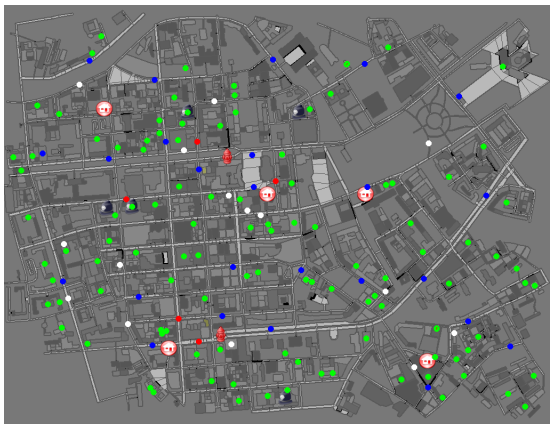
Berlin is a lightly populated map with only 111 civilians scattered around the city. However, this map simulates only a light disaster when compared to **SydneyS2** and **SF**.



(a) Snapshot of **SydneyS2** initial state



(b) Snapshot of **SF** initial state



(c) Snapshot of **Berlin** initial state

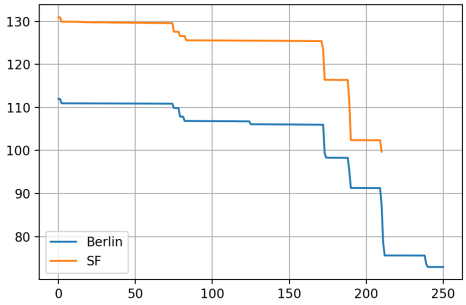
Figure 7: Initial state of each map

Intial Score	Civilians	Ambulance teams	Refuges
112	111	14	5

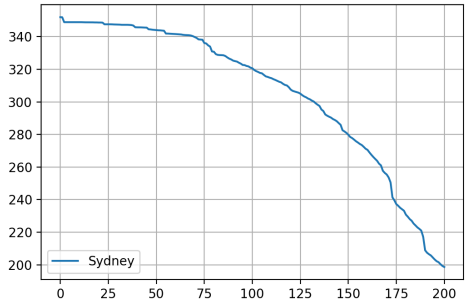
Table 3: Parameters of **Berlin**

4.2 Results

Figure 8 illustrates the performance of the centralized Max-Sum algorithm in the selected scenarios. In figure 9 and 10, the scores for decentralized Max-Sum algorithm are plotted for each map with three different communication ranges: limited (1000), normal (100.000) and extended (10.000.000). The variability is illustrated in figure 12, to provide a better understanding of the these scores. The variability is determined by repeating the scenarios with normal communication range 10 times. After these simulations the standard deviation is taken over the resulting scores.

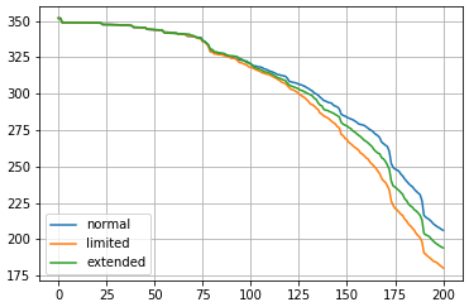


(a) Scores **Berlin** and **SF**

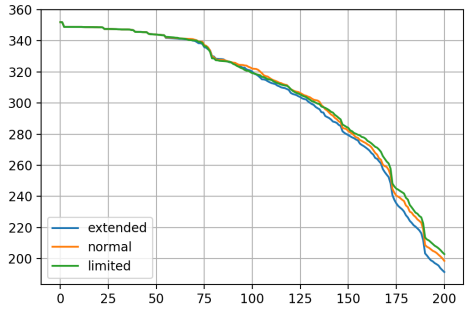


(b) Scores **SydneyS2**

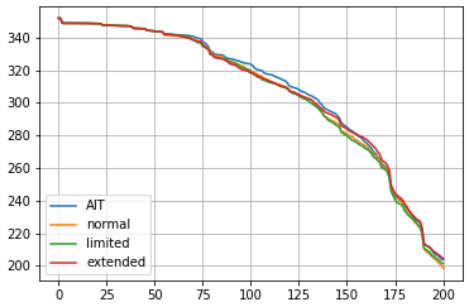
Figure 8: Baseline results of centralized Max-Sum algorithm



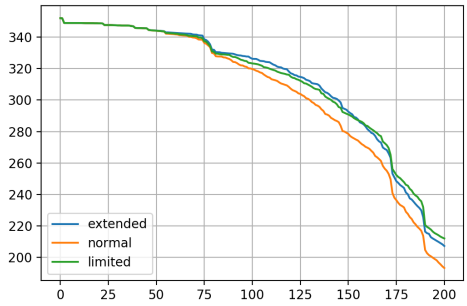
(a) **SydneyS2** 50 Iterations



(b) **SydneyS2** 100 Iterations



(c) **SydneyS2** 150 Iterations



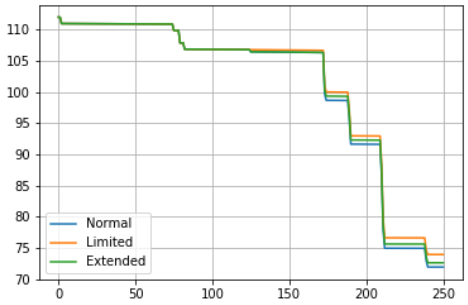
(d) **SydneyS2** 200 Iterations

Figure 9: Score test results with different communication ranges for the agents **SydneyS2**

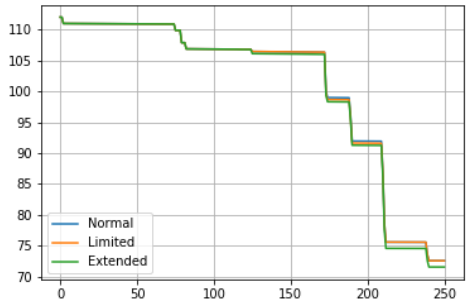
4.3 Discussion

In figure 9 the average end scores of **SydneyS2** are between 190 and 210 points. The centralized Max-Sum scores of **SydneyS2** in figure 8b have an end result of 200 points. So the end score results of centralized Max-Sum and decentralized Max-Sum are roughly the same. This is also the case in figure 10 where **Berlin** averages at 73 points. This is a 1 point difference between the baseline and Max-Sum. **SF** has a 2 points difference between centralized and decentralized Max-Sum.

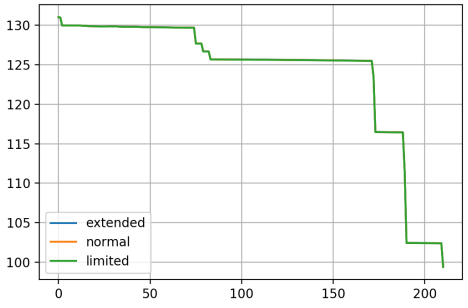
In figure 9a, a difference in the final score per communication range can be seen. A normal communi-



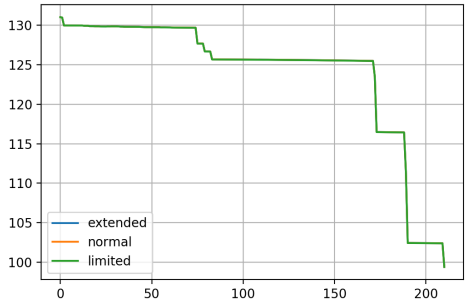
(a) **Berlin** 50 Iterations



(b) **Berlin** 150 Iterations



(c) **SF** 50 Iterations



(d) **SF** 150 Iterations

Figure 10: Score test results with different communication ranges for the agents

cation range is awarded more points than the limited communication range. This limited communication range performs worst. A possible explanation for this might be that in the simulations of the limited communication range, agents with useful information are located outside their range, preventing communication and thus generating a less optimal solution. Agents also perform worse than normal communication range when they have extended communication. This might be due to the large communication range that includes distraction communication with agents that do not have useful information because of their distance to problems the agent face.

The hypothesis of this thesis was that when the number of iterations in Max-Sum is increased, the variability in the simulations would decrease and as such the scores would improve. This cannot be be confirmed by the results gathered from **SydneyS2**. Although the results differ per simulation, there is no clear trend visible in the decrease of variability. In figure 12 it can be seen that in every simulation the variability in **SydneyS2** is roughly the same regardless of the number of iterations. Therefore, it could be argued that the results in figure 9 may be partly random.

Although the scenario of **Berlin** has not undergone an equally extensive test as **SydneyS2**, the scores in figure 10 can also be the result of variability in the simulations. This argument is partly supported by figure 12 which shows error bars that are equally as wide as the difference in scores in figure 10a and 10b.

Lastly, the scenario of **SF** has undergone the same testing as **SydneyS2**, but results from these tests did not show any variability. This might have to do with the limited noise in the scenario. The results

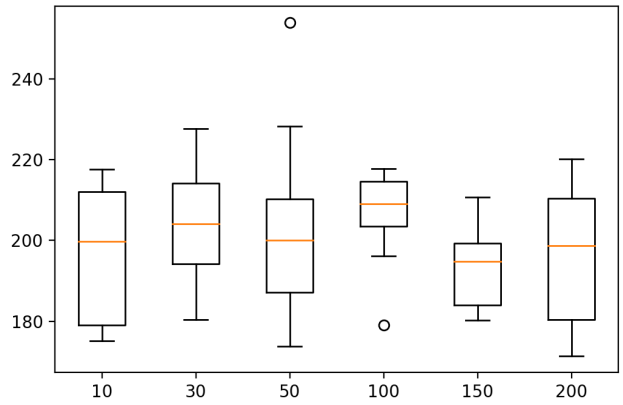


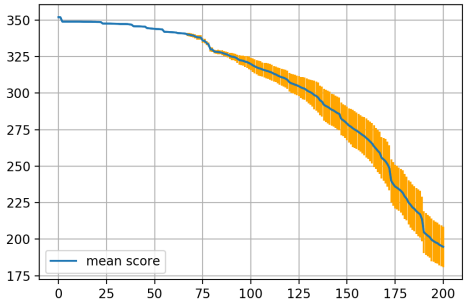
Figure 11: Final scores with various maximum number of iterations simulated at **SydneyS2**

of the limited noise can be found in figure 10. Here only one line is visible, since all mean scores of every communication range is plotted over each other.

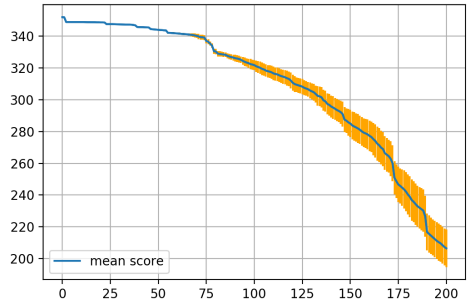
The initial expectation before testing was that results would change after adjustments in the communication range of the agent would be made. However, results in figure 9 do not support this hypothesis, since the limited and extended communication range perform better than the normal communication range when the number of iterations is increased (see figure 9d). Further research is needed to determine how the communication ranges influence the outcomes of the Max-Sum simulation.

It should be noted that the scenarios in this research are selected by hand, predominantly based on RAM usage. To confirm the effectiveness of the Max-Sum algorithm, scenarios should be tested where more emergency services and civilians are simulated than present in the scenarios discussed in this thesis.

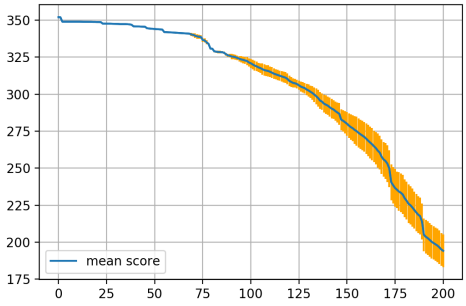
Finally, in figure 11 the variation in final scores is presented for **SydneyS2**. Here, iterations 10 and 30 have been tested to confirm the effectiveness of less iterations. As can be seen in the figure, **SydneyS2** with 100 iterations seems to perform best and has the least variation in final outcomes. However further research is needed to confirm the effectiveness of the number of iterations especially when those are increased.



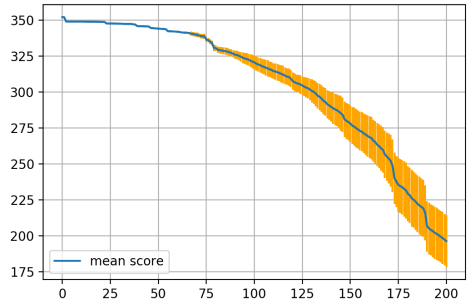
(a) **Sydney** 50 Iterations



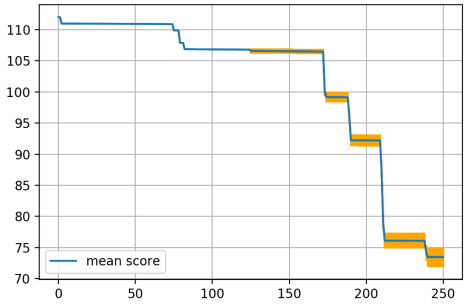
(b) **Sydney** 100 Iterations



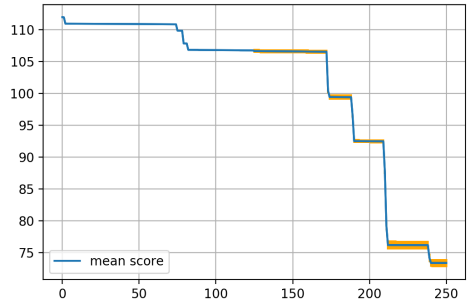
(c) **Sydney** 150 Iterations



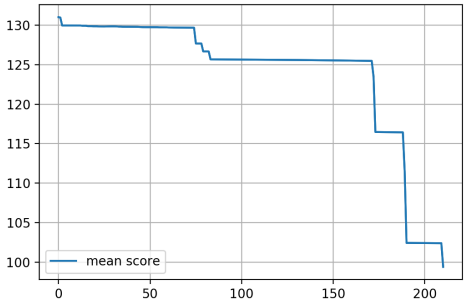
(d) **Sydney** 200 Iterations



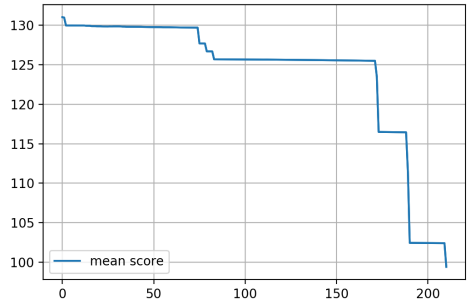
(e) **Berlin** 50 Iterations



(f) **Berlin** 150 Iterations



(g) **SF** 50 Iterations



(h) **SF** 150 Iterations

Figure 12: Variability test, simulated with normal range

5 Conclusion

In this thesis, the effectiveness of a decentralized Max-Sum algorithm has been tested on the RoboRescue Agent Simulation (RRS). The AIT-extension has been used, enabling agents in the field to communicate an infinite amount per time-step if needed. Moreover, two main factors have been investigated. First, the communication range of agents in the field and secondly, the maximum number of iterations the Max-Sum algorithm is given to converge to a solution.

After testing the communication range for agents, the conclusion can be drawn that the communication range does not influence the outcome of the final score of the RRS. As previously discussed, a possible factor in the outcome of the testing process could be that the number of iterations tested was too small. To confirm the above, further research is thus needed to test how the number of iterations run in a test influences the end results.

Overall it can be concluded from the test results that the decentralized Max-Sum algorithm used for the RRS has similar performance to a centralized version of the Max-Sum algorithm. This makes the task of the central-agent in some cases invaluable.

References

[1] T. Ukai, "The Great Hanshin-Awaji Earthquake and the Problems with Emergency Medical Care," *Renal Failure*, vol. 19, no. 5, pp. 633–645, 1997. PMID: 9380882.

[2] *RoboCup Rescue Simulation League*, 2020. Available at https://rescuesim.robocup.org/.

[3] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings, "Decentralized Coordination in RoboCup Rescue," *The Computer Journal*, vol. 53, no. 9, pp. 1447–1461, 2010.

[4] H. Kitano and S. Tadokoro, "Robocup rescue: A Grand Challenge for Multiagent and Intelligent Systems," *AI Magazine*, vol. 22, p. 39, Mar. 2001.

[5] F. Fioretto, E. Pontelli, and W. Yeoh, "Distributed Constraint Optimization Problems and Applications: A Survey," *Journal of Artificial Intelligence Research*, vol. 61, pp. 623–698, 2018.

[6] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised Coordination of Low-Power Embedded Devices using the Max-Sum Algorithm," 2008.

[7] J. Parker, A. Farinelli, and M. Gini, "Max-Sum for Allocation of Changing Cost Tasks," in *International Conference on Intelligent Autonomous Systems*, pp. 629–642, Springer, 2016.

[8] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. Rodríguez-Aguilar, "Efficient Inter-Team Task Allocation in RoboCup Rescue," vol. 1, pp. 413–421, 01 2015.

[9] J. Parker, A. Farinelli, and M. Gini, "Lazy Max-Sum for Allocation of Tasks with Growing Costs," *Robotics and Autonomous Systems*, vol. 110, pp. 44 – 56, 2018.

[10] A. Rogers, A. Farinelli, R. Stranders, and N. Jennings, "Bounded Approximate Decentralised Coordination via the Max-Sum Algorithm," *Artificial Intelligence*, vol. 175, no. 2, pp. 730 – 759, 2011.

[11] E. Rollon and J. Larrosa, "Improved Bounded Max-Sum for Distributed Constraint Optimization," in *International Conference on Principles and Practice of Constraint Programming*, pp. 624–632, Springer, 2012.

[12] Y. Miyamoto, T. Kusaka, Y. Okado, K. Iwata, and N. Ito, "An Approach for Distributed Constraint Optimization Problems in Rescue Simulation," in *RoboCup 2019: Robot World Cup XXIII* (S. Chalup, T. Niemueller, J. Suthakorn, and M.-A. Williams, eds.), (Cham), pp. 578–590, Springer International Publishing, 2019.

[13] M. Wooldridge, *An Introduction to Multiagent Systems*. John Wiley & Sons, 2009.

[14] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of Constraint Programming*. Elsevier, 2006.

[15] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais, "Boosting Systematic Search by Weighting Constraints," in *ECAI*, vol. 16, p. 146, 2004.

[16] K. Ghédira, *Constraint Satisfaction Problems: CSP Formalisms and Techniques.* John Wiley & Sons, 2013.

[17] A. Gershman, A. Meisels, and R. Zivan, "Asynchronous Forward Bounding for Distributed COPs," *Journal of Artificial Intelligence Research*, vol. 34, pp. 61–88, 2009.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.