Robocup 2018 – TDP Rescue Agent Simulation
M.I.C (Iran)

Reza Vaziri Nejad, Amirhossein Zare, Reza Iranbakhsh

Tehran, Iran

{Vaziri.79.reza, 00amirhosseinzare, reza.iranb}@gmail.com

Abstract

This paper describes an epitome of M.I.C team researches, which has been conducted to achieve the optimum and most efficient solutions to solve rescue simulation agent issues. Population density is what our clustering algorithm is based on, which helps agents to position themselves in the most proper location. We have also provided a detailed comparison between common path planning algorithms, and for a better result, we used the “Ant Colony Optimization” algorithm to prove the agent’s transportations. Also, we proved our agent’s cooperation using communication methods to have more and better accuracy and control of the situations.

As all we know, for having a proper result on the non-communication situations, we have to prove our agent self-Decision making. To achieve this goal, we redesigned our previous target selection and the search methods.

1 Introduction

Natural disasters are one of the most catastrophic accidents such as earthquakes, which lead people to die and cause enormous financial losses. The purpose of the Rescue Simulation League (RSL) is to decrease life and economic losses caused by natural disasters such as earthquakes. The M.I.C team is a multi-agent rescue simulation team that is trying to find and improve better algorithms to solve the agent's problems.

The overall purpose of simulating a natural disaster and implementing an utterly independent intelligence to run on this simulation is to be able to perform the most reasonable rescue operations as soon as an actual disaster happens. Fortunately, The ADF does include the vital requirements that are needed to accomplish a comprehensive and functional intelligence. For instance, this platform allows you to choose the optimum path to reach (path planning) or to categorize the entire entities of the map to improve the performance of rescue operations, which relies on multi-tasking (clustering). In RoboCup Asia-Pacific 2017 competition, we decided to develop our strategies mostly on agent modules. On the other hand, it is not necessary to mention that other modules such as path planning and clustering can extremely affect the outcome of the rescue operation and to achieve this goal, we decided to focus mainly on path planning and clustering algorithms. As already mentioned in the previous Team Description Papers the M.I.C clustering method was based on the sample “K-Means,” but for this competition, we decided to use our new clustering algorithm which is similar to DBSCAN clustering algorithm, and we have fully explained it at the clustering part.

It is obvious, rescuing more civilians and search more efficiently, needs a suitable path planning algorithm. We need the shortest path available to reach our destination. but it is not reliable unless we would know the important blockades positions. To achieve this goal, we used the “Ant Colony Optimization” algorithm to use the more reliable routs.

2 Modules

The main purpose of implementing a clustering algorithm is to improve the execution of multi-tasking processes between agents, and as it could act very well, it could make the overall score much worse. Almost every target that fire agents or ambulance agents choose is related to buildings. Fire brigades should extinguish the fire and prevent the fire transfers. and ambulance agents may pick an injured human from inside of a building. That said, buildings are a good option to implement our clustering algorithm based on.

If we take a closer look at agents' target selection, we can conclude that in a specific area, the more the number of buildings is, the more it is probable that agents may choose a target from that area. Then we think that the best indicator to consider in our clustering algorithm is the density of entities.

As for path-finding algorithms, we need to keep in mind that the performance of any algorithm can be different whenever the size of our graph changes. An algorithm might act well in small graphs, but as the graph becomes larger, the performance becomes worse. We have explained this subject further.

2.1 Clustering

According to the basic K-Means clustering algorithm, we need to provide a variable *k* that contains a numeric value. This value will be the number of clusters. Being able to assign the number of clusters you need is a good feature but the necessity of having a specific value to provide, leads us to a static process of clustering, while rescue simulation is a dynamic environment. Initializing a dynamic clustering algorithm will help the entire simulation to be as real as possible and also we can gain more possible scores.

Unlike ambulance team, fire brigades need to shut off the fire with their collaboration. That means that agents should work together with a partly low distance between them. Although, the procedure of ambulance team is to spread out throughout the entire map to rescue damaged humans. Since the state of each human changes every cycle, the best way to allocate entities of each cluster is to use population diversity of buildings. Due to this point, a same number of agents will be assigned to each cluster, and as a result, every agent will have a limited area, and they will find their target much easier and faster.

We decided to use DBSCAN as our clustering algorithm. The algorithm gets two values ε and m, where ε stands for radius and m, stands for minimum points included in radius. The overall procedure of DBSCAN is to find the neighbours of every point within ε radius, identify the core points with more than m neighbours and finally consider non-core points as noises. In this case, we choose a random building as a temporary-core point, get building-type objects in its ε range and then calculate a density rate based on the number of buildings in range. If density rate of each cluster is greater than m, that building and also nearby buildings in ε range will be included in a single cluster.

Besides creating these clusters, another important point is the way we handle noise points. We have explained this in "Strategies" section.

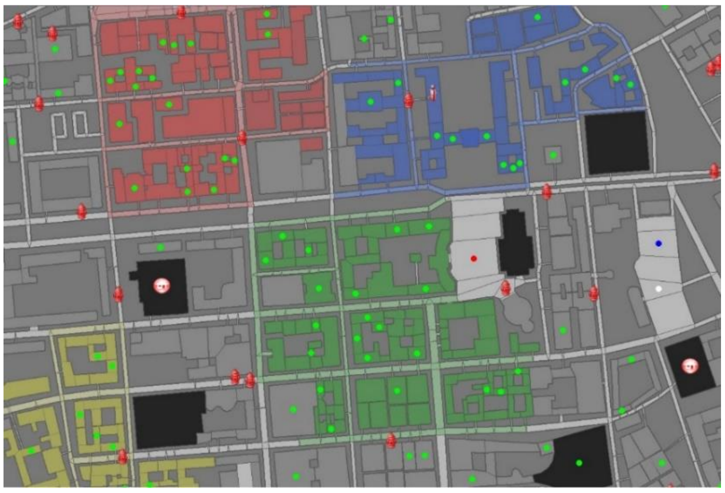


Fig.1. *DBSCAN performance (black buildings have been calculated as noises)*

2.2 Path Planning

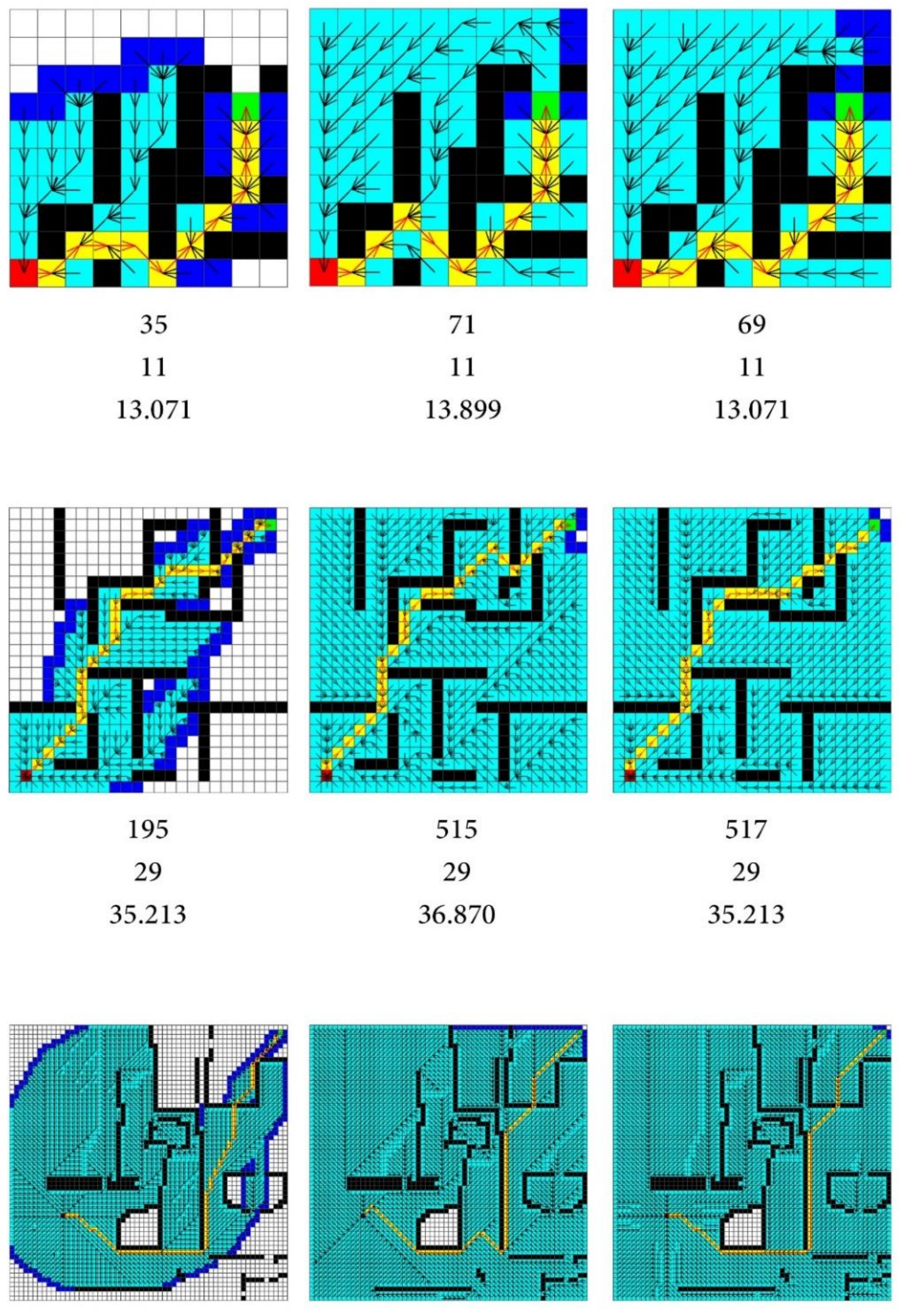
2.2.1 Primary Path Planning

Path planning algorithms calculate the shortest path from a specific point to another. Naturally, rescue simulator includes different buildings and roads with dynamic properties. In order to have a general picture of the map, we convert it to a graph. This graph gives us the ability to calculate weight (distance) between nodes. Most of the current path planning algorithms are designed just to identify the shortest path. However, in rescue simulation, the shortest path is not always the best path but also blockades and fire zones on the route make many differences.

A-Star is a common path-finding algorithm that is used in rescue simulation. For examinations, we also considered BFS and Dijkstra as other options we could choose rather than A-Star.

- Experimental comparison

  According to our examinations, A-Star acts perfectly in small graphs where we would have the shortest path in the shortest time. But as the graph becomes larger, A-Star performs more like BFS algorithm and finally, as the graph becomes larger again, A-Star, BFS and Dijkstra will perform the same as each other. However, A-Star delivers much faster performance than other algorithms in any situation. In order to consider fire zones and blockades in our path calculation, we allocate more weight to some nodes based on the type of agent and its target, and automatically A-Star will not consider that node to calculate the best path (*Fig 2*).



		A*	Breadth First Search (BFS)	Dijkstra
10x10	Nodes expanded	35	71	69
	Steps	11	11	11
	Distance	13.071	13.899	13.071
25x25	Nodes expanded	195	515	517
	Steps	29	29	29
	Distance	35.213	36.870	35.213
60x60	Nodes expanded	2177	3153	3181
	Steps	78	78	78
	Distance	88.355	93.326	88.355

🟦 Closed Set 🟨 Chosen Path 🟥 Start Point 🟩 Target 🟦 Frontier ⬛ Obstacle

Fig.2. *Path planning algorithms comparison*

- Algorithmic comparison

1. Dijkstra

   Dijkstra algorithm primary duty is to find the shortest paths between the nods in a graph. The way that the Dijkstra algorithm work is as follow:

   1- At first, one of the nodes of the graph will be called as “initial node.”
   2- Build a list of unvisited nods, and named them unvisited set.
   3- For the current node, it will consider all of its neighbours.
   4- Calculate the tentative distance between them.
   5- The goal is to find the shortest way, so it will compare the distances, and it will select the smallest one.
   6- After considering its all neighbours, it will mark the current node as visited and it will remove it from the unvisited set.
   7- This will be done as long as checking unvisited nodes.
   8- Now it will select the best path, using the smallest index.

   But this algorithm has a critical problem because it uses sophisticated computing, it will take a long time to find the path, and it causes a problem in the agent's operations.

2. Breadth-first search (BFS)

   Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. This non-recursive implementation is similar to the non-recursive implementation of depth-first search, but differs from it in two ways:

   1- It uses a queue instead of a stack (First In Last Out).

   2- It checks whether a vertex has been discovered before enqueueing the vertex rather than delaying this check until the vertex is dequeued from the queue.

   But this algorithm (BFS) has a little critical problem that will cause a delay in the agent's works because, in addition to checking the main routes, it will also review the sub-routes(paths that end in the other buildings). Because of that, we need a faster algorithm, so we decided to choose the “A*” algorithm.

3. A-Star

   “A-Star” pathfinding algorithm, is one of the best path planning algorithm because of its speed and accuracy to find the best path. A-Star uses the best-first search algorithm, and it finds the shortest path to the target by searching among of the possible paths.

   It uses the function below to determine which of its partial paths to expand into one or more paths.

$$f(x) = h(x) + g(x)$$

g(x) is the cost of the path from the start node to n.

h(x) is a heuristic that estimates the cost of the cheapest path from n to the goal.(“Ant Colony”)

As you could easily see in ***Fig.2***, A-Star path planning is the path planning for the rescue simulation. It could easily find the best path by approaching itself to the blockade. When the blockade is one of its neighbours, it will just take a step back, and choose the other way to approach to its target.

A* is known as its fast pathfinding, so after our examinations, we decided to use the A-Star pathfinding algorithm for our path planning.

A-Star is good alone, but it could be better with another algorithm that helps the agents to choose the best path to their targets. Therefore, we decided to use Ant Colony Optimization, to increase the result of our work.

2.2.2 Ant Colony Optimization

In the natural world, ants of some species and randomly, and upon finding food return to their group while laying down pheromone paths. If other ants find such a way, they are likely not to keep moving at random, but instead to follow the trail, returning and reinforcing it if they eventually see food. The overall result is that when one ant finds a right path from the colony to a food source, other ants are more likely to follow that way, and positive feedback eventually leads to all the ants following a single track. The idea of the ant colony algorithm is to mimic this behaviour with "simulated ants" walking around the graph representing the problem to solve.

Ant Colony Optimization is a useful algorithm that helps our path planning system to estimate the fastest, cleanest and the best way for the agent's movement.

For Implementing the ant colony optimization algorithm on the A-Star path planning algorithm, we use the structure below:

$$f(x) = h(x) + g(x)$$

$$h(x) = (\frac{a(x) \mp 10}{10})$$

a(x) is the effect of trail pheromone in ant colony; the range is 0 to 1.
g(x) is the cost of the path from the start node to *n*.

Every time when an agent traces a path, *a(x)* goes 1, and in each cycle, it will be multiplied by $\frac{1}{2}$, so it will never reach zero and we never lose those paths that the agents pass. *p(x)* just increase and decrease.

Each of the agents has their own *a(x)* table. Using the ADF communication structure, agents are sharing their own *a(x)* table to each other. Agents store their path and the *a(x)* value, so they can easily know the other agents used path and the cycle that they used that route.

For making the best use of this method, each agent has a separate function:

- Ambulance Team and Fire Brigade agents:

$$h_A(x) = (\frac{a(x) - 10}{10})$$

Since Fire brigade and ambulance team agents must reach their targets, as fast as possible, using this function, known roads with no blockades, have a high probability of tracing.

- Police Force agents:

$$h_P(x) = (\frac{a(x) + 10}{10})$$

Because police forces must trace the whole map for the chance of finding the blockades, value of $h_A(x)$ will make path with the highest chance of the clean way. Therefore, we dicided to increase the $h(x)$, to have more chance to clear the other ways that those are not cleaned yet.

3 Strategies

3.1 Police Force

Time-Consuming is essential for all agent, and having time-consuming will save civilians, turning off the fires, and healing each agent as fast as it can. Police agents have the responsibility to clear blocked routes throughout the map to make the movement more comfortable for the ambulance team and fire brigade agents.

At the first of the game, some roads will be blocked. Ambulances and fire brigades will need to access all the roads for rescuing and saving the life of the civilians and turn off fires of the buildings.

To have a better result on clearing the blockade, we used an algorithm that is similar to Ant Colony Optimization (ACO), and it works like the opposite of the ACO. The algorithm set a value to each road that the police force has been clear or already visited that road. And whenever the police visit that road again, it will add a number to the road value. The police force agent must visit and clean the roads with the fewer value. We used this algorithm, and we saw that more than 90 percent of the situations, police force cleared all possible roads.

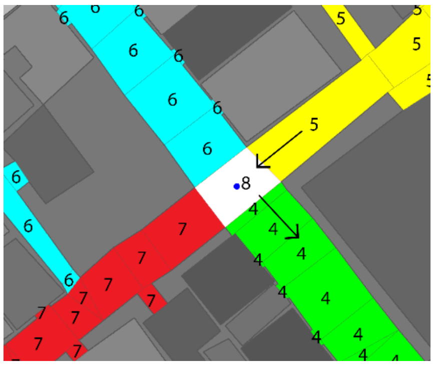


Fig.3. *An example of our clustering clearing method. As you can see, the police force agent has already visited the blue, red and yellow roads and it is going to visit and clear the green roads. In this method, the road which is a neighbour of a building, will not get any value more than its neighbour road; and the agent can pass the higher valued roads if one of its neighbours has less value. If in a road, were more than one police force agent, the extra police force agent, will search the world for fire, civilians and blockade.*

Our police force agents will clear the blocked roads in three groups:

– Fire zones: For bringing the fire under our control, fire brigade agents need to have access to buildings which are on fire. Therefore, we should open their ways as fast as possible. Our police force agents only clear the routes to the building which are near or in the fire zones.

– Human: After cleaning all possible roads, these agents search for any alive human whether if a human is stuck in a blockade or it is just behind the blockade. We use communication to be aware of the exact position of stuck agents.

– Refuge: Since refuges are one of the most vital elements in rescue simulation, clearing all refuges' entrances is another way to improve the overall performance of the rescue operations. Clearing refuges' entrances unintentionally, would cause significant problems for every agent with its procedure related to refuges. For instance, fire brigades would not be able to refill their water tank, or ambulance team could not release their victim properly. We assign each refuge to one police agent. When simulation begins, police forces move immediately towards their assigned refuge and clear its entrance. Finally, As soon as all entrances are cleared, these police forces will join "Human" and "Fire zones"agents to help them out.

There is also a probability that there would be just a building near a refuge and there are no roads nearby. In this situation, we need to get the joint edges of refuge and buildings near that refuge. We do this repeatedly until there is no nearby building left and we have reached the entrance. it is shown in figure 3; we also used an algorithm similar to DBSCAN. When simulation begins, agent checks if there are more than one refuges near each other. If this has happened, those refuges will be assigned to only one police force. If the map does not have any refuge, the police force will clear the way end to the hydrants.

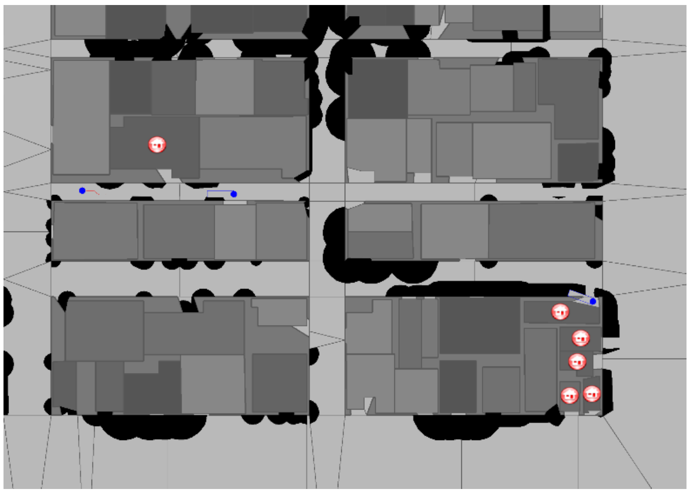


Fig.4. *Police forces have cleared refuges' entrances.*

We also developed another feature in order to minimize the wasted time on path clearing. Suppose that there are two types of roads exist on the map:

- Primary roads
- Building Entrances

We did some examinations on the elapsed time of clearing all roads in the map, and we concluded that the elapsed time remarkably decreases when police forces are not planning to clear buildings' entrances. This is because of the RS concept, where usually agent must move to the building's entrance to clear the blockade. In order to have the least wasted time in the process of clearing roads, Except for the police forces that have to clear refuges' entrances, we do not pay attention to any building and its entrance, until the ambulance agents command the police force agents to clear the blockade.

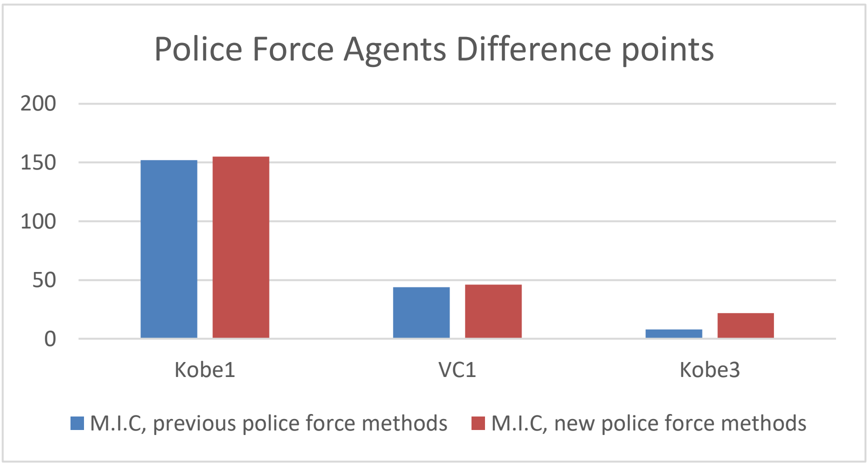


Fig.5. *Police force difference scores on the Kobe1, VC1 and the Kobe3 of the RoboCup Asia-Pacific 2017. The blue column is our previous code and methods, but the red column is our new police force algorithm with the other former methods (The difference point is only based on our police force agents).*

3.2 Ambulance Team

3.2.1 Ambulance Main Task

The primary responsibility of ambulance team is finding the exact position of injured civilians and remove their buriedness if needed and rescue a maximum possible number of humans until the end of the simulation.

As soon as clustering calculations are done and finished, the same amount of agents will be assigned to each cluster. For instance, if there are *n* agents and *k* clusters on the map, each cluster will have $\frac{n}{k}$ agents, but as this method might not perform as we expect. Then at the time we are writing this paper, we are actually working on a functionality which brings us a proper divider algorithm in order to assign the optimum number of agents to each cluster.

Ambulance agent needs to visit all buildings to have the best-updated world model of the civilians, to save time for the ambulance team agents, we worked on an algorithm which adds each building that the agent sees in a list of all possible entities that are in that building; and the other agents will not check those buildings again.

Ambulance agents keep rescuing humans in its allocated cluster until there is no remaining target left in the cluster. In this case, the agent will move to rescue the noise points which are not included in any cluster, or it will search the world (whole map) for civilian, blockade and fire to communicate with other agents and update the world model.

For no communication environments, we applied a self-organized task allocation strategy which will be a good help to the ambulance team rescuing operations. The allocation of the ambulance team agents to victims is performed based on particular specifications, like “death time,” burriedness of the civilians, and the distance between the agent and the target.

3.2.2 Death Time

The cycle that a civilian dies in is called “Death Time.” We have already designed a new algorithm for our team, called “Death Time” which estimate the cycles that may cause a civilian die.

For estimating the death time, we are considering the target burriedness, health point, estimated burriedness removing, exact position URN and the distance to the refugee (if the map doesn't have any refugee, this variable will equal to 1).

After finding the humans death time, the ambulance team agents will select the best civilians or agents to rescue them. In some cases, if the death time estimator gives us the information that a civilian will undoubtedly die in the rescue operation, we will assign more than one ambulance agent to that specific civilian to rescue more humans and earn a better result.

Agents have the highest priority for the ambulance team, because in some situations, there is one ambulance free in the roads, and the rest of them are in a building, and they have burriedness, so the best strategy is rescuing the ambulances, fire brigades, and police forces at first, and then rescue the civilians.

3.3 Fire Brigade

The primary duty of fire brigade agents is to control the fire not to be extended and save other humans by extinguishing the fire near them.

Fire search is designed to satisfy the following requirement:

- Updating the shape of the fire
- Finding new fire zones

Fire search is done according to our fire estimator and other parameters such as distance to fire zones and etc. After finding a new fire zone, fire search tried to update the shape of the fire, and any other pieces of information of the fire zone by visiting the fire buildings near the newly born fire zone.

For finding a new fire zone, we're using a clustering algorithm (DBSCAN) and a formula for searching the necessary roads, and areas.

We use DBSCAN clustering algorithm to find the building density, but in some cases like the VC map, because the buildings are entirely stuck together, the building density won't work. Therefore we use a method to fix this bug. This technique first acquires essential roads using the formula below, and then it will find the important buildings that they could quickly ignite, and then the fire brigade agents will search that area to find fire and extinguish them.

Important roads are those roads that have at least: a gas station, small buildings, buildings with wood material and more than five agents were on that road.

To achieve this goal, we made a formula to search for the important roads.

$$T = \frac{T_1 + T_2 + T_3 + \cdots + T_n}{F - 1}$$

(V = Final road value, T = Important targets in a road, N = Number of fire brigade agents in that Road)

Each of the important targets in roads has the same value as the other targets except the gas station. For example, each gas station value is equal to 10 wooden buildings. After setting the value to each of the roads, agents will start searching the roads from the highest value to the lowest value. Each of the high priority roads will be searched by 2 fire brigade agents; then they will search the other roads until they find a target.

According to this formula, we can find important roads and search them for fire ignition. After finding the fire, we will calculate the size of the fire zone, using the number of the buildings, the size of each building and the number of the neighbour roads. After finding fire zones, the fire brigade agents will divide into groups, and they will spread into their clusters *(in here, cluster means the fire zones)*. Also, we will create unstable zones using the formula that contains the gas stations and their neighbours, it will help us to have more accuracy on them when they are on fire.

After finding fire zones and calculate the size of the fire, the most important thing to do is to, reach to that area and extinguish the on fire buildings. If the on fire building won't be reachable for the fire brigade agents (because of the roads blockade), they will command the police force to clear that blockade and if took a long time to remove the blockade, or because of the simulator problems, they stuck in there, they will try to prevent the fire transfers by filling up the nearest buildings to that fire zone.

Almost always, there are multiple buildings, that a fire brigade agent must extinguish it, using the valuing system that includes the table below, fire brigade can easily select the building, that must be extinguished at first. The valuing system is as the below:

Type	Building Information / Building Values					
	Entity	Value$_{\times 10^4}$	Entity	Value$_{\times 10^4}$	Entity	Value$_{\times 10^4}$
Yellow	Civilian	75	Agent	76	Empty	60
Orange	Civilian	44	Agent	46	Empty	47
Red	Civilian	30	Agent	20	Empty	10

Table.1. *Fire brigade valuing system. In some situations, we have agent and civilian in a building, so the output numbers will increase. For example, in a yellow building, we have 2 agents and a civilian, the output number will be* 227×10^4.

$$V = \frac{A \times W \times 10^3}{S \times D}$$

(V = Final building value, A = Agent valuing system for buildings (*using the table*), W = The amount of water inside the fire brigade water thank, S = Size of the building, D = The distance between the agent and the building)

This valuing system will run for each on fire building, to choose the best choice for extinguishing.

Now we are working on a method, which will calculate and estimate the number of the needed fire brigade agents and the amount of the water that the building needs to be extinguished, using the building size, number of the building floors, material, fieryness, temperature and also agents water tank size, amount of the water that they have, distance between the each building and the agents, water pressure and etcetera. Using communication, agents will announce the building value and its information to each other, and the number of the needed agents will move to the target, and they will do their task.

There are some problems in extinguishing the fire that we try to solve. For instance, there are a lot of situations where the fire starts from a point and spreads around. Previously, we did not have any specific method to prevent the fire from spreading, but now, we developed a new algorithm to use on target allocation modules which helps us to achieve this purpose. In order to prevent the fire to be extended, we need to extinguish the fire from the outer-level of fire buildings. Regarding the fact that outer-level buildings have less fieryness value, so the first priority is the outer building because, while we are extinguishing them, we are preventing the fire transfers (*Fig. 6*).

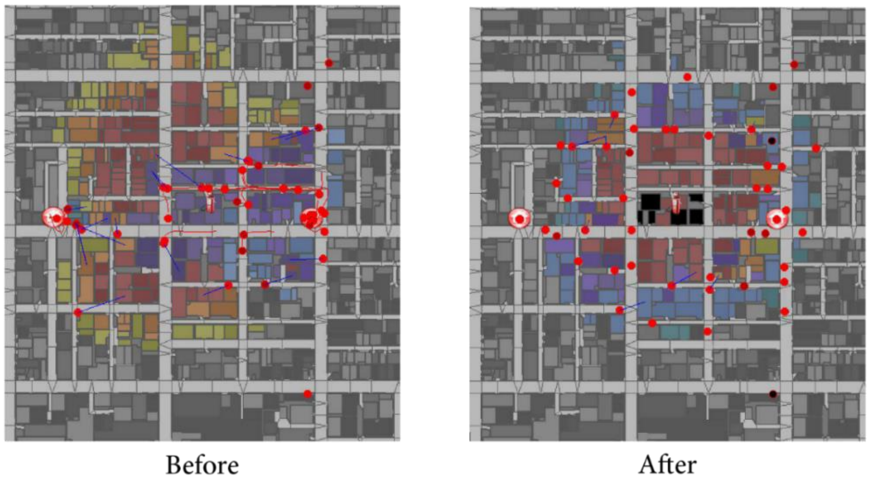


Fig.6. *Fire Brigades are preventing the fire from spreading by extinguishing the outer buildings. Fire brigade at first, will extinguish all yellow buildings after that the orange ones and then the red buildings. Because the high temperature of the fire, some of the fire brigade will fill up the neighbour buildings of the yellow buildings to prevent the fire transfer.*

We have found that, if there were not any target for a long time to extinguish, the agent should fill up the neighbour buildings of the gas stations. It will reduce our score but it will decrease the chance of gas station explosion in the future.

4 Preliminary Results

The table below represents the RoboCup Asia-Pacific 2017 preliminary day results. The M.I.C team took the first place in this contest. We also provide the score of our new implementations.

Team	Map			
	Kobe1	**VC1**	**Eindhoven1**	**Paris1**
M.I.C (new version)	186.12	80.03	133.47	238.19
M.I.C	152.62	44.38	111.98	235.28
Naito Rescue	153.96	43.44	91.82	207.60
Apollo Rescue	119.80	27.86	64.78	164.45

5 Conclusions

One of the most effective approaches to improve the overall performance of the team is the cooperation between agents in no-communication situations. For this purpose, there is a need for custom and dynamic clustering algorithms and a path planning system which can change the final path if needed. By implementing the new algorithms and methods which have been mentioned in this paper, an overhaul in team's performance was witnessed. You may see our results before using our improvements and after using it. We have made more than 36 percent progress in our final results on the no-communication situations, and that could be effective in the communication mode too.

References

[1] R. Vaziri Nejad, A. Zare, R. Iranbakhsh
RoboCup IranOpen 2018, M.I.C Team Description Paper. 2018

[2] R. Vaziri Nejad, P. Haghighat, A. Zare, B. Barazandeh, S. Taheri, R. Iranbakhsh
RoboCup Asia-Pacific 2017, M.I.C Team Description Paper. 2017

[3] Daniel Smilkov and Shan Carter.
Neural Network Playground; *playground.tensorflow.org*

[4] M. Ghahramanpour, A. Absalan, A. Kandeh
RoboCup 2017, Aura Team Description Paper. 2017

[5] Nikos Kanargias. *Maze 5.2 (Java application). 2017*

[6] Red Blob Games. *Introduction to A**. 2014

[7] en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

[8] en.wikipedia.org/wiki/A*_search_algorithm

[9] en.wikipedia.org/wiki/Breadth-first_search

[10] en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[11] Y. Alipour, M. Farshbaf Nadi
RoboCup 2016, Poseidon Team Description Paper. 2016

[12] www.rescuesim.robocup.org