RoboCupRescue 2023
TDP Agent Simulation AIT-Rescue (Japan)

Haruki Uehara[1], Hiroya Suzuki[1], Joe Fujisawa[1], Ryoya Maeda[1], Itsuki Matsunaga[1], Kaito Ito[1], Ai Okamoto[1], Keitaro Sasada[1], Yuki Shimada[1], Yuya Sugisaka[1], Ryosuke Suzuki[1], Keita Hayashi[1], Keisuke Ando[1], Takeshi Uchitane[1], Kazunori Iwata[2], and Nobuhiro Ito[1]

[1]Department of Information Science, Aichi Institute of Technology, Japan
[2]Department of Business Administration, Aichi University, Japan

Abstract. We introduced a layered distributed constraint optimization problem (L-DCOP) to the RRS task assignment problem. The experimental results confirmed that the new model was more efficient in rescue operations than that modeled on the distributed constraint optimization problem. However, we could not introduce these ideas because of communication limitations. Therefore, we used the agent's travel distance estimation used in the L-DCOP experiment and improved the search order determination method. This improved the agent's search efficiency and the accuracy of travel distance estimation, and enabled more efficient rescue operations. As a result, we achieved better results than our agents from last year.

1 Introduction

RRS tasks have time window constraints that must be completed within a certain amount of time. RRS tasks have ordering constraints that must be executed in order between tasks of different types. In recent years, our team has been developing a method to assign tasks in RRS task assignment as a distributed constraint optimization problem (DCOP). However, the DCOP cannot consider the time window and ordering constraints of tasks. Therefore, this year, we applied a layered DCOP (L-DCOP), which can take into account both the time window and ordering constraints of tasks, to the RRS task assignment problem. As a result, we confirmed that the rescue activities modeled by the L-DCOP were more efficient than those modeled by the DCOP.

Because of communication limitations in RRS, we could not apply the aforementioned research. Therefore, we used the agent's travel distance estimation used in the L-DCOP experiment. We also improved the agent's search order method, which we had not focused on previously. Using this implementation, we succeeded in surpassing our agent's score from last year in 4 out of 10 disaster scenarios.

In Section 2, we explain the DCOP, L-DCOP, modeling of the RRS task assignment problem by the L-DCOP, and experiments in which we compared the

rescue results of the RRS task assignment problem modeled using the L-DCOP with those modeled using the DCOP. In Section 3, we explain the modules used in AIT-Rescue 2023. In Section 4, we explain the strategy for each agent implemented in AIT-Rescue 2023. In Section 5, we report on the experiment conducted to evaluate the effectiveness of the implemented strategies. In Section 6, we summarize this TDP and describe issues that we will address before RoboCup2023.

2 Scientific challenge

In this section, we discuss the definitions of the DCOP and L-DCOP, and the modeling of the RRS task assignment problem using the L-DCOP. Then, we describe the experiment in which we compared the rescue results of the RRS task assignment problem modeled using the L-DCOP with those modeled using the DCOP.

An important problem in RRS is the task assignment problem. In the task assignment problem, an agent is assigned appropriately to a task according to the given conditions. A task is a problem to be solved by an agent. Such a problem can be represented as a DCOP in a multi-agent system. An attempt to solve the problem expressed as a DCOP can then be made using the DCOP algorithm.

2.1 Distributed constraint optimization problem

The DCOP is the problem of finding the optimal combination of variables that satisfies the constraints between agents and variables as a solution. In this study, the variables that constitute the problem are distributed and managed by multiple agents. The DCOP [1] is defined as follows:

- $\mathbf{A} = \{a_1, \ldots, a_m\}$
  is a finite set of agents, where m is the number of agents.
- $\mathbf{X} = \{x_1, \ldots, x_n\}$
  is a finite set of variables, where n is the number of variables. Each variable is managed by one of the agents. $m \leqslant n$ and the number of variables must always be greater than or equal to the number of agents.
- $\mathbf{D} = \{D_1, \ldots, D_n\}$
  is a family of sets that collects the range of variable $x \in \mathbf{X}$, where $1 \leqslant i \leqslant n$ and D_i denotes the range of the corresponding variable x_i. Each element of the range is called a candidate value for the variable and is represented by d. The combination of any candidate values is represented by σ and the combination of all candidate values is represented by Σ. Σ is defined as

$$\Sigma = \prod_{i=1}^{n} D_i \tag{1}$$

  where $\prod$ is a large operator that performs a direct product.

– $\mathbf{F} = \{f_1, \ldots, f_l\}$
  is a finite set of functions (called cost function) that represent constraints between variables. The cost function is defined as

$$f_j \colon \prod_{x_k \in \mathbf{X}^j} D_k \to \mathbb{R} \tag{2}$$

  where $\mathbf{X}^j$ is the set of variables whose constraint relation is expressed as f_j.
– $\alpha \colon \mathbf{X} \to \mathbf{A}$
  In the DCOP, the combination of candidate values is determined that satisfies the constraints and has the lowest total cost. Therefore, a correspondence always exists between any variable and any agent.

The objective function $\mathbf{F}_g(\sigma)$ for optimization is the sum of the output values of the cost function. The objective function $\mathbf{F}_g(\sigma)$ is defined as

$$\mathbf{F}_g(\sigma) = \sum_{i=1}^{k} f_i(\sigma_{f_i}) \tag{3}$$

where σ_{f_i} is a combination of only the candidate values corresponding to the cost function f_i from σ.

The optimal combination σ^* that minimizes the cost can be defined as

$$\sigma^* = \underset{\sigma \in \Sigma}{\operatorname{argmin}}\ \mathbf{F}_g(\sigma) \tag{4}$$

2.2 Layered DCOP

An extended DCOP called the L-DCOP is proposed to solve the task assignment problem with ordering and time window constraints. Under the ordering constraint, a task must be completed before the target task can be executed. Under the time window constraint, a target task must be completed within a certain time. In L-DCOP, tasks are stratified by priority. Additionally, the DCOP is solved for each layer, and the overall problem is solved based on the combination of the layers and the constraint conditions. The processing steps in the L-DCOP are as follows:

1. Tasks are assigned to each layer according to their priority, thereby creating a precedence graph.
2. A factor graph is created for each layer and the DCOP algorithm is executed for each layer.
3. Task assignment is performed according to the factor graph. After the agent has been assigned a task, a simple temporal network (STN) is created.
4. The most efficient execution order is determined by STN.

The precedence graph created in Step 1 is shown in Fig. 1. A precedence graph is a directed acyclic graph with nodes $\mathbf{T} = \{T_1, \ldots, T_n\}$ corresponding to tasks and edges $\mathbf{E}_p$ that represent the ordering constraints among tasks.

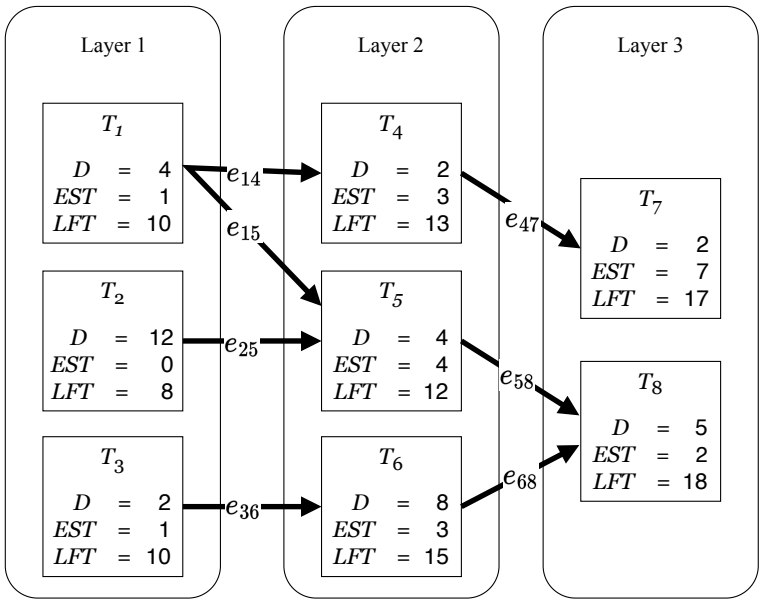

Fig. 1: Example of a precedence graph

Directed edge $e_{ij} \in \mathbf{E}_p$ indicates that task T_i should be completed before task T_j can be executed. For example, Fig. 1 shows that task T_1 must be completed before tasks T_4 and T_5 can be executed. Additionally, each task maintains information about the earliest start time (EST), latest finish time (LFT), and duration (D). Fig. 1 shows that task T_1 has time D of 4, time EST of 1, and time LFT of 10.

In the precedence graph, tasks are assigned in such a manner that there are no ordering constraints among tasks in the same layer. Therefore, tasks in the same layer can be executed independently.

An example of the factor graph created in Step 2 is shown in Fig. 2. In this TDP, $n = m$.

The variable node $\mathbf{X} = \{x_1, \ldots, x_m\}$ represents the set of agents and the function node $\mathbf{F} = \{f_1, \ldots, f_n\}$ represents the set of tasks. Task T_i and function node f_i have a one-to-one correspondence. An edge connecting a variable node and function node indicates that the agent can start the task. In Fig. 2, agent x_1 indicates that tasks f_1 and f_3 can be started. A factor graph is created for each layer, as shown in Fig. 2. The DCOP algorithm is then executed for each layer.

The STN created in Step 3 is the graph used by the agent to manage the schedule of its tasks. An example of an STN is shown in Fig. 3.

In Fig. 3, a node represents the start time of a task (ST) or the end time of a task (FT). ST_i and FT_i represent the start and end times of task T_i, respectively. The edge connecting the start time of the task (ST) to the end time of the task (FT) represents the time taken for task T_i (DT_i). The edge connecting the end

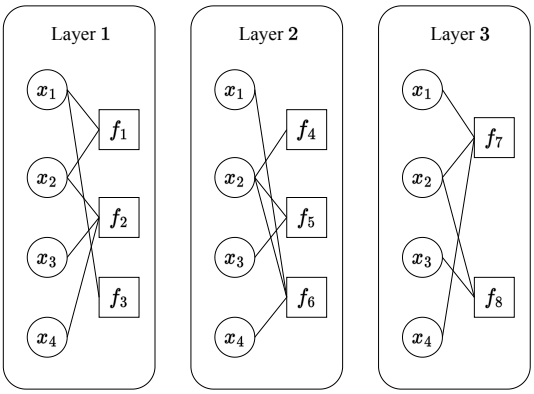

Fig. 2: Example of a factor graph in the L-DCOP

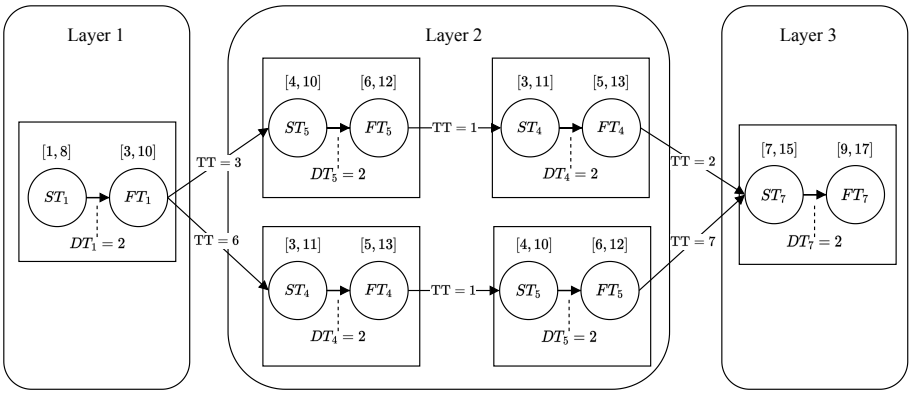

Fig. 3: Example of a STN

time of the task (FT) to the start time of the task (ST) represents the travel time between tasks (TT). In Fig. 3, the time taken for a task in task T_1 (DT_1) is 2, and the travel time from task T_1 to task T_5 (TT) is 3. Each node has a defined interval that represents the time window constraint of the task. If the number of agents executing task T_i is m, the interval of the start time is $\left[EST_{T_i}, LFT_{T_i} - \frac{DT_i}{m}\right]$ and the interval of the end time is $\left[EST_{T_i} + \frac{DT_i}{m}, LFT_{T_i}\right]$. In Fig. 3, the interval of the start time of task T_1 is $[1, 8]$ and the interval of the end time of task T_1 is $[3, 10]$.

After an STN is created for each agent, the task execution order is determined to minimize the sum of the travel time between tasks and task execution time. Each agent orders its tasks based on its STN.

2.3 Modeling the RRS task assignment problem with the L-DCOP

The L-DCOP is a DCOP framework that has been extended to enable the handling of task assignment problems in which ordering and time window constraints

exist. Therefore, to model the task assignment problem in RRS as the L-DCOP, needs to be extended by adding the following definitions to the task assignment problem modeled as the DCOP:

- time window and ordering constraints
- precedence graph
- STN.

Time window and ordering constraints of the task in RRS In RRS, Fire Brigades are responsible for the rescue task. In the rescue task, civilians are rescued from a collapsed building. Additionally, Ambulance Teams are responsible for the load task. In the load task, civilians are transported to a refuge. The targets of the rescue and load task are civilians. Injured civilians continue to suffer damage until they are transported to a refuge. Therefore, civilians must be rescued and transported to a refuge quickly. Additionally, a buried civilian in a building cannot be transported until the rescue is complete. Therefore, the time window and ordering constraints in the RRS task can be defined as follows:

Time window constraint: The rescue and load tasks must be completed while the target civilian is alive.
Ordering constraint: The load task cannot be executed until the rescue task is complete.

Definition of precedence graphs in the rescue and load tasks The precedence graphs for the rescue and load tasks are shown in Fig. 4.

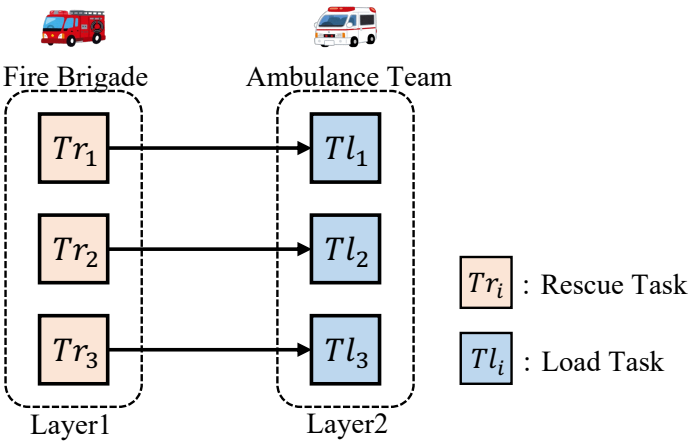

Fig. 4: Precedence graphs for the rescue and load tasks

The precedence graph assigns the rescue task to Layer 1 and the load task to Layer 2. The rescue task and load task target the same civilians.

The precedence graph consists of two types of tasks. The rescue task handled by the Fire Brigade is defined as $\mathbf{Tr} = \{Tr_1, ..., Tr_i, ..., Tr_m\}$. The load task handled by the Ambulance Team is defined as $\mathbf{Tl} = \{Tl_1, ..., Tl_i, ...Tl_m\}$.

Additionally, information about rescue task Tr_i and load task Tl_i can be defined as follows:

Rescue task Tr_i

- DTr_i (time required for rescue task Tr_i)
  This is the time needed to rescue civilians. When one agent performs a rescue operation in RRS, the degree of burying is reduced by a certain amount in one step. Therefore, the time required to complete the rescue depends on the degree of burying.
- EST_{Tr_i} (time when rescue task Tr_i can be started)
  This is the time when Ambulance Teams can begin to rescue civilians. In this TDP, EST is set to 1 in the rescue task. This is because no disaster occurs after the simulation starts. Therefore, all tasks can be started at the first step.
- LFT_{Tr_i} (time when the rescue task Tr_i must be complete)
  This is the time at which civilians are estimated to be in a dead state.

Load task Tl_i

- DTl_i (time required for load task Tl_i)
  This is the time required to transport civilians. Civilians are assumed to be transported to the nearest refuge.
- EST_{Tl_i} (time when load task Tl_i can be started)
  This is the time when Fire Brigades can begin to transport civilians. It is necessary to have completed the rescue task before the load task begins. Therefore, the load task can start when the rescue task is complete. This time depends on the schedule to which the rescue task is assigned.
- LFT_{Tl_i} (time when load task Tl_i must be complete)
  This is the time at which civilians are estimated to be in a dead state.

Definition of the STN in rescue and load tasks The STN determines the task that satisfies the time window constraint from the permutation of all tasks assigned to the agent. Then the STN determines the execution order with the shortest execution time. Therefore, when there are n tasks, we must calculate whether the time window constraint is satisfied for $n!$ combinations; that is, the number of combinations increases exponentially as the number of tasks increases.

We propose a method to create a schedule by inserting assigned tasks into the STN. The process of adding rescue task Tr_3 to the STN of agent a_k is shown in Fig. 5.

In Fig. 5, the execution order of Tr_1 followed by Tr_2 has already been determined. Inserting rescue task Tr_3 at the insertable position of this STN creates three different schedules. The agent extracts the schedule that satisfies the time

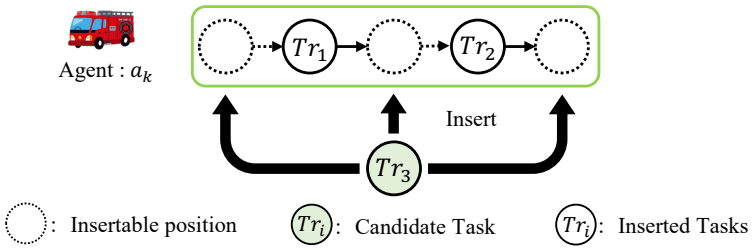

Fig. 5: Add rescue task Tr_3 to the STN of agent a_k

window constraints from the created schedule. Then the agent keeps the schedule with the shortest task execution time in the extracted schedules. Whenever a task in a layer is assigned, this operation is executed. At this time, the assigned task is a candidate task for insertion.

The method creates all schedules that satisfy the time window constraints of the inserted and candidate insertion tasks each time a task is assigned. If no schedule satisfies the time window constraint, then the candidate task is not inserted. It then compares all the generated schedules and retains only the schedule with the shortest task execution time.

The STN needs to ensure that each task satisfies its time window constraints. Additionally, the time window constraint must take into consideration not only the execution time of the task but also the travel time of the agent. Therefore, the travel time can be defined as follows:

- TTr_i
  This represents the travel time from the location of the Fire Brigade to the location of civilians targeted by rescue task Tr_i. The location of the Fire Brigade is the location of civilians targeted by the rescue task to be performed before Tr_i. If no task is to be executed before Tr_i, then it uses the initial position of the Fire Brigade.
- TTl_i
  This represents the travel time from the location of the Ambulance Team to the location of the civilians targeted by transport task Tl_i. The location of the Ambulance Team is the location of the shelter that has completed the transport task to be performed before Tl_i. If no task is to be executed before Tl_i, it uses the initial position of the Ambulance Team.

Each agent has an STN. The following is a description of the elements required for the calculations to be performed each time a task is assigned. The Fire Brigade checks whether the time window constraints are satisfied based on the following factors. Let Tr_i be the target task. Additionally, let Tl_i be a transport task that targets the same citizens as Tr_i. In this case, consider the following A, B, and C:

A: time taken to complete Tr_i $(TTr_i + DTr_i)$

B: time taken to complete Tl_i $(TTl_i + DTl_i)$
C: total time for A of tasks before Tr_i.

If $A + B + C$ does not exceed LFT_{Tr_i}, the Fire Brigade can be judged to have satisfied the time window constraint.

In the same manner, the Ambulance Team checks whether the time window constraints are satisfied based on the following factors. Let Tl_i be the target task to be checked to determine whether the time window constraint is satisfied. In this case, consider the following a and b:

a: time taken to complete $Tl_i.(TTl_i + DTl_i)$
b: EST_{Tl_i}.

If $a + b$ does not exceed LFT_{Tl_i}, then the Ambulance Team can be judged to have satisfied the time window constraint.

The STN keeps the schedule with the shortest task execution time. The elements required in this calculation are as follows:

Fire Brigade: total time of $TTr_i + DTr_i$ in the target execution sequence
Ambulance Team: total time of $TTl_i + DTl_i$ in the target execution sequence.

2.4 Implementation of the binary max-sum for the task assignment problem in RRS

We apply the binary max-sum (BMS) to the task assignment problem in RRS. The BMS is an algorithm for solving the DCOP. The L-DCOP executes the DCOP algorithm for each layer in the precedence graph. An example of a factor graph to be created is shown in Fig. 6.

First, task assignment and scheduling are repeated in Layer 1 until all tasks are assigned to agents. The schedule for each Fire Brigade obtained in Layer 1 is shared with Layer 2. The EST of the load task in Layer 2 is updated based on the received schedule. Then, task assignment and scheduling are performed in the same manner as in Layer 1. Finally, the simulation is run based on the Layer 1 and Layer 2 schedules.

Processing steps for iterative task assignment The iterative task assignment procedure for the BMS running on Layer 1 is as follows:

1. Execute task assignment using the BMS.
2. Adds the assigned task to the agent's own STN and maintains the shortest schedule that satisfies the time window constraints of the task.
3. The agent shares the tasks inserted in the STN using communication, and eliminates the tasks it has selected and received from the tasks in Layer 1.
4. If a task exists in Layer 1 and any agent can select a task in Layer 1, repeat steps 1–3.

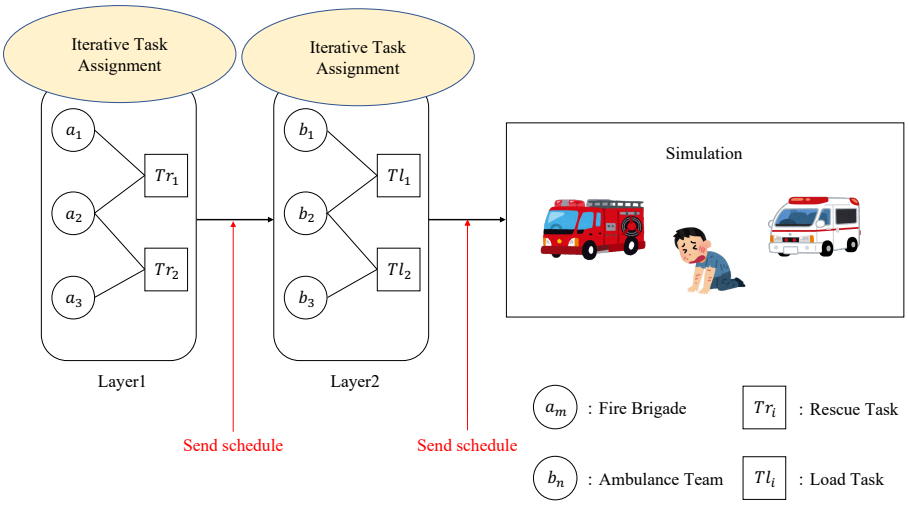

Fig. 6: Example of task assignment in the L-DCOP

In step 2, each agent keeps the shortest schedule that satisfies the time window constraints of the task. Therefore, the agent operates based on that schedule. Layers and tasks are kept by each agent. Therefore, it is necessary to synchronize the layer information each time a task is assigned. In Step 3, the layer information is synchronized by sharing the tasks inserted in the STN with other agents.

2.5 Experiment

In this experiment, we compared the rescue results of the following two types of agents:

- agents that model the RRS task assignment problem as a DCOP (DCOP Agent)
- agents that model the RRS task assignment problem as an L-DCOP (L-DCOP Agent).

In this experiment, we used the ADF extension environment for the DCOP [4]; that is, each agent could communicate multiple times during one step.

In this experiment, it was difficult to determine the travel time from the location of the Ambulance Team to the location of the civilian to be transported when the Fire Brigade was assigning tasks. Therefore, we computed the Ambulance Team's behavior virtually. In this experiment, the Ambulance Team did not perform the transport operation. The civilian was considered to have been transported to a refuge after the rescue by the Fire Brigade. If a civilian could be transported alive to the nearest refuge, the civilian was considered to be transported. We based the determination of whether a civilian could be transported

alive on the estimated travel time from the civilian's location to the nearest refuge and the civilian's estimated survival time.

We simulated a total of 12 disaster scenarios by combining four scenarios and three maps. In the simulation, we measured the number of civilians rescued and transported, and the behavior of the agents. The number of civilians rescued and transported in each disaster scenario is shown in Table 1. In Table 1, rescued represents the number of civilians rescued and transported represents the number of civilians transported to the refuge.

Table 1: Comparison of the Number of Civilians Rescued and Transported

map	disaster scenario(placement)		L-DCOP Agent		DCOP Agent	
	Agent	Task	rescued	transported	rescued	transported
Eindhoven	centralized	centralized	34	34	39	39
	centralized	distributed	26	24	30	30
	distributed	centralized	41	41	41	41
	distributed	distributed	41	37	31	31
SF	centralized	centralized	40	38	35	35
	centralized	distributed	36	34	33	33
	distributed	centralized	43	43	36	36
	distributed	distributed	44	43	36	36
VC	centralized	centralized	38	38	36	36
	centralized	distributed	40	38	42	42
	distributed	centralized	46	44	44	44
	distributed	distributed	45	43	42	42
Total			474	457	445	445

As shown in Table 1, the L-DCOP Agent transported and rescued more civilians than the DCOP Agent in 9 out of 12 disaster scenarios. The results also indicated that the L-DCOP Agent had a higher total number of rescued and transported civilians than the DCOP Agent. Therefore, the L-DCOP Agent proposed in this TDP performed efficient rescue activities. However, the proposed algorithm cannot yet be used in competition because it uses the ADF extension environment for the DCOP. Thus, we used the method used in the L-DCOP Agent experiment to estimate the distance traveled by the agent. We also improved the method for determining the order in which agents searched. These are discussed below.

3 Modules

3.1 Overview of this section

In this section, the TwoOpt and the BoundaryCenterPathLengthEstimator modules are described, which were newly developed and added this year. The TwoOpt module determines the order in which agents search for buildings. The BoundaryCenterPathLengthEstimator module calculates the distance between buildings. The other modules remain the same as those in AIT-Rescue 2022 [2].

3.2 TwoOpt module

Agents need to search each building to obtain information about civilians. To date, each agent has searched the buildings assigned to it using clustering and selected the nearest unsearched building at each step. If the distance required for the search is minimized, the search time can be reduced. Therefore, it is necessary to calculate the shortest search order for the group of buildings assigned to the agent during precomputation.

Therefore, we developed the TwoOpt module, which calculates the optimal search order for short routes. The TwoOpt module uses the 2-opt algorithm, which is an approximate solver for the traveling salesman problem, to determine the search order of buildings.

The 2-opt algorithm can be used to calculate the search order of buildings for the following reason. Consider a graph with buildings as vertices. Edges connect reachable buildings. Assume that roads do not become impassable because of blockades in this scenario, which results in a complete graph. The problem of finding the shortest building search order is equivalent to finding the shortest Hamiltonian path in this graph. The shortest Hamiltonian problem can be converted to a traveling salesman problem [3]. Hence, the 2-opt algorithm can be used to determine the search order of buildings.

The TwoOpt module executes the following process:

1. Generate a list of the initial search order from the input set of buildings.
2. Randomly select two buildings from the search order list.
3. Compare the distances of the search paths in the original search order with the search order in which the two buildings are replaced. If the distance is shorter, reverse the order of the two buildings and the building in between them on the list.
4. Repeats Steps 2 and 3 for an arbitrary number of times.
5. Output the list of buildings.

This module is used in the precomputation of the agent's search module to create a search order from the buildings assigned to the agent. This search order is then used in the Fire Brigade's search module.

3.3 BoundaryCenterPathLengthEstimator module

To estimate the time it takes for an agent to move, accurate travel speed and distance are required. To date, the Euclidean distance between the starting point and target point has been used as the travel distance of the agent. However, agents move along roads, and there may be a significant error between the Euclidean distance and actual travel distance. Therefore, it is necessary to obtain a value that closely approximates the actual travel distance.

Therefore, we developed a module to estimate the travel distance called the BoundaryCenterPathLengthEstimator module. The BoundaryCenterPathLengthEstimator module calculates the distance between the center of the entity that serves as the starting point, the midpoint of the boundary line between

entities, and the center of the entity that serves as the endpoint of the path. We obtained this list of entities from the pathfinding algorithm.

We used this module in the TwoOpt module to reference the distances between buildings when calculating the search order of buildings for the agent.

4 Strategies

4.1 Ambulance Team

Starting with the 2021 RoboCup, a bed set was added to the refuge. Ambulance Teams need to transport civilians to refuge with available beds. Therefore, the choice of refuge is important.

Communication with the command center To obtain information on the refuges, the Ambulance Team must perceive the refuges or receive information on the refuge from the command center. Therefore, ambulance teams do not always store new information on refuges.

When the Ambulance Team starts transporting a civilian, it compares the information it receives about the refuge with the most recent information about the refuge. The command center stores new information on all refuges and compares the information received with the most recent information for those refuges. If the number of available beds in a refuge is different, information on other optimal refuges is sent to that ambulance team. The Ambulance Team changes the designated transportation refuge when the optimal refuge designated by the command center differs from the refuge recommended for transportation.

Selection of transports targets considering agent's movement speed The Ambulance Team priotize the lives that can be saved as much as possible. Therefore, when a civilian that needs to be transported is found, Eq. (5) is used to determine whether or not transport is possible.

$agentToCivilian$ represents the straight-line distance from the current location to the target civilian. Similaryly, $civilianToRefuge$ represents the straight-line distance from the target civilian to the nearest refuge. Additionally, $agentAveMove$ is a parameter that represents the average speed of the agent movement. In the experiment, we determined the value of the agent movement speed and found that the average value was $40,000$. Therefore, the value of $agentAveMove$ should be set to $40,000$. Moreover, LEX represents the survival time of a civilian.

$$\frac{agentToCivilian + civilianToRefuge}{agentAvgMove} < LEX \tag{5}$$

4.2 Fire Brigade

The Fire Brigade is responsible for rescuing buried civilians and making them ready for transport by the Ambulance Team. In addition, to prioritize the lives

that can be saved as much as possible, it is important to select rescue targets considering the civilian's survival time, rescue time, and distance to the refuge.

Rescue target selection considering the agent's movement speed When the Fire Brigade finds a civilian who is buried, they use Eq. (6) to determine whether or not to rescue them. Eq. (7) represents the rescue time for a civilian. Futhermore, $fbNumbers$ represents the number of fire brigades involved in rescuing the civilians in question.

$$\frac{agentToCivilian + civilianToRefuge}{agentAvgMove} + rescueTime < LEX \tag{6}$$

$$rescueTime = \frac{buriedness}{fbNumbers} \tag{7}$$

4.3 Police Force

The police force is responsible for clearing debris from a blocked road and allowing other agents to move along the calculated path. It's also responsible for clearing the debris if other agents and citizens are buried in debris.

There are two main ways to clear debris. One is to remove the debris in a rectangular shape and the other is to remove the debris by shrinking the debris. The method of removing the debris in a rectangular shape can be efficiently removed when the specific road or building is blocked by debris. On the other hand, in the method of removing the debris by shrinking the debris, the debris shrinks toward the center point and eventually disappears. Since no debris remains on the road, agents can move smoothly.

Until now, the debris was basically removed using the method of removing the debris in a rectangular shape, and only when the police force itself was buried, the debris was removed using the method of removing the debris by shrinking the debris. The police force itself only removed debris by shrinking the debris when it was buried. However, when the debris is large, the method of removing the debris by shrinking the debris takes a long time to remove the debris, and the agent may die. Therefore, the number of steps required to remove the debris is calculated from the RepairCost of the debris and the ClearRepairRate of the agent. If the number of steps is long, the debris is removed using the method of removing the debris in a rectangular shape. If the number of steps is short, the debris is removed using the method of removing the debris by shrinking the debris.

5 Preliminary Results

We conducted comparison experiments using our agent from last year to examine the effectiveness of AIT-Rescue 2023. We used scenarios from the final round scenarios of RoboCup 2022. The experimental results are shown in Table 2.

Table 2: Experimental results for AIT-Rescue 2022 and AIT-Rescue 2023

Scenario	**Team**	
	AIT-Rescue 2022	**AIT-Rescue 2023**
kobe2	128.386	129.050
paris2	27.862	28.421
berlin2	8.147	7.995
vc2	43.875	45.624
sydney1	15.112	14.715
ny1	19.666	18.679
montreal1	48.397	47.487
eindhoven1	12.003	11.985
istanbul1	43.538	43.325
sf1	38.173	38.718

The experimental results show that our improved agent achieved a higher score than our agent from last year in 4 out of 10 scenarios. Additionally, the scores improved substantially for kobe2 and vc2. kobe2 and vc2 are small map sizes. Therefore, we believe that our agents performed effectively in scenarios in which the time required for agent movement was small. By contrast, the maps with significantly reduced scores were Sydney1 and montreal1, which were large maps. This may be because our implementation of TwoOpt did not account for debris on the road.

Thus, we consider that the modules and strategies that we described in this TDP worked effectively in RRS.

6 Conclusions

In this study, we modeled the RRS task assignment problem as an L-DCOP. Then we compared rescue activities modeled using the conventional DCOP and those modeled using the L-DCOP. As a result, we confirmed that the rescue activities modeled using the L-DCOP were more efficient than those modeled using the DCOP. However, this algorithm cannot be used in competition yet because it requires an ADF extended environment [4].

Therefore, we used the agent's travel distance estimation used in the L-DCOP experiment. We also improved the agent's search order method, which we had not focused on previously. This improved the agent's search efficiency and the accuracy of travel distance estimation. Using this implementation, we succeeded in surpassing our agent's score from last year in 4 out of 10 disaster scenarios.

We will improve the clustering of the agents' search ranges before RoboCup 2023. Current clustering divides entities so that the number of agents equals the number of clusters. However, in some scenarios, using a smaller number of divisions may allow for more efficient rescue operations. Therefore, we will implement a module to determine the number of divisions according to the scenario.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP17K00317 and JP21K12039; and the Kayamori Foundation of Information Science Advancement.

References

1. Fioretto, F., Pontelli, E., Yeoh, W.: Distributed constraint optimization problems and applications: A survey. CoRR **abs/1602.06347** (2016), `http://dblp.uni-trier.de/db/journals/corr/corr1602.html#FiorettoP016`
2. Hiroya, S., Akira, H., Haruki, U., Joe, F., Itsuki, M., Ryoya, M., Yuki, S., Iwata, K., Ito, N.: RoboCupRescue 2022 TDP Agent Simulation AIT-Rescue (Japan) (2022)
3. Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley (January 1991)
4. Miyamoto, Y., Kusaka, T., Okado, Y., Iwata, K., Ito, N.: RoboCupRescue 2019 TDP Infrastructure AIT-Rescue (Japan) (2019)