Agent Simulation Competition Rules
RoboCup 2025

RoboCup Rescue Simulation Organizers

Version 2.1, February 07, 2023

Table of Contents

1. Purpose

This document describes the rules of the RoboCup Rescue Agent Simulation competition for the RoboCup 2025.

2. Major Changes

There is not any major changes to the RCRS Simulator since RoboCup 2022. However, we encourage teams to develop their code using the new RCRS Core Python in order to take advantage of state-of-the-art Machine Learning algorithms, such as SciKit, TensorFlow, PyTorch, among others.

3. Agent Simulation Competition

The RoboCup Rescue Agent Simulation Competition evaluates the performance of teams of agents in regards to their distributed coordination and planning algorithms for rescuing civilians in a city after an earthquake.

The competition is divided in three rounds (i.e., preliminary, semifinal, and final rounds), see Section 3.1. At each round, the participating teams provide their agent team's code that will be executed on a set of different disaster scenarios and a score calculated to each scenario. At the end of the preliminary and semifinal rounds, the set of agent teams with the highest sum of scores are selected to move to the next round of the competition. At the end of the final round, the participant team whose agent team accumulates the highest sum of scores is proclaimed the winner.

3.1. Rules

1. **Agents** Teams shall implement all types of agents using the RCRS ADF Framework (a sample implementation is available at RCRS ADF Sample) or the RCRS Core Python (a sample implementation is available at RCRS Python Sample) that will be executed on the RoboCup Rescue Agent Simulator.

   Teams using the RCRS Core Python have no constraints in implementing their agents. However, teams using the RCRS ADF Framework can only implement their own code to replace or extend the following RCRS ADF classes:

   ```
   adf.core.component.centralized.CommandExecutor
   adf.core.component.centralized.CommandPicker
   adf.core.component.communication.ChannelSubscriber
   adf.core.component.communication.MessageCoordinator
   adf.core.component.extaction.ExtAction
   adf.core.component.module.algorithm.Clustering
   adf.core.component.module.algorithm.PathPlanning
   adf.core.component.module.complex.TargetAllocator
   adf.core.component.module.complex.TargetDetector
   ```

   Participating teams are not allowed to change any other RCRS ADF classes, especially the

`Tactic` classes. Participating teams should report any bug in the RCRS ADF classes prior to the competition (see Bug Exploitation). It is the responsibility of the participating teams to ensure that their code connects the correct number of agents to the server.

Participating teams must implement their code in a package named after their team's name (`TEAM`) and competition year (`YEAR`).

```
TEAM_YEAR.centralized.CommandExecutor
TEAM_YEAR.centralized.CommandPicker
TEAM_YEAR.communication.ChannelSubscriber
TEAM_YEAR.communication.MessageCoordinator
TEAM_YEAR.extaction.ExtAction
TEAM_YEAR.algorithm.Clustering
TEAM_YEAR.algorithm.PathPlanning
TEAM_YEAR.complex.TargetAllocator
TEAM_YEAR.complex.TargetDetector
```

For instance, if the short name of the participating team is TEST and the competition year is 2025, the participating team shall provide a package containing the methods:

```
TEST_2025.centralized.CommandExecutor
TEST_2025.centralized.CommandPicker
TEST_2025.communication.ChannelSubscriber
TEST_2025.communication.MessageCoordinator
TEST_2025.extaction.ExtAction
TEST_2025.algorithm.Clustering
TEST_2025.algorithm.PathPlanning
TEST_2025.complex.TargetAllocator
TEST_2025.complex.TargetDetector
```

Participating teams must provide a configuration file containing information of the classes they have changed from the original RCRS ADF and a mapping between classes, package path, and file in their code.

2. **Modularity** Participant teams are forbidden to cast or make internal dependency between the classes listed in Agents. It is mandatory that these classes are usable independent of each other.
3. **Code Reuse** The reuse of code from agent teams of previous years is encouraged. But the following restriction applies:

   a. Teams are not allowed to have more than 50% of other agent team's code or logic in the classes

   ```
   adf.core.component.module.complex.TargetDetector
   adf.core.component.module.complex.TargetAllocator
   ```

   b. Teams are allowed to reuse without any change the classes

```
adf.core.component.centralized.CommandPicker
adf.core.component.centralized.CommandExecutor
adf.core.component.communication.ChannelSubscriber
adf.core.component.communication.MessageCoordinator
adf.core.component.extaction.ExtAction
adf.core.component.module.algorithm.Clustering
adf.core.component.module.algorithm.PathPlanning
```

c. The reuse of any code or module from another agent team must be explicitly reported in the `README` document and released in the source-code and shared with the Organizing and Technical Committee members before the first day of competition. In the `README` document, the team must inform what motivated the reuse of the code or module and, if they made any changes (small or large) to the code or module, these changes must be described in detail.

d. If the agent team uses a class or module from another agent team without any change, the package name must not be changed.

The Organizing or Technical Committee members will check the agent teams' implementation with agent teams' code from previous years to determine if the participating team complies with the Code Reuse rule. If the participating team does not comply with the rule, the agent team will be disqualified from the competition.

4. **External Libraries** It is allowed to use external libraries, but these libraries must be open source and the libraries must not violate the competition rules. If any library is not open source and the participating team wants to use it in the competition, the participating team must submit a request for approval at least 1 (one) month before the beginning of the competition. The request must be submitted to the Technical and Organizing Committees.
5. **Shared Memory** Agents cannot use any form of shared memory, including static memory accessible to all agents, direct function calls between agents, or writing files for use by other agents during the scenario simulation in the `Simulation phase`. The exception is the `Pre-Computation phase` when agents are allowed to write files (see Phases for details). The Organizing or Technical Committee may execute each agent of the agent team in a different virtual/physical machine if the agent team is suspected of violating this rule.
6. **Rounds** The competition is structured into three rounds: one preliminary round, one semifinal round, and one final round. The preliminary round will be executed in two consecutive days (first and second days of the competition), while the semifinal and final rounds are executed in one day each (third and fourth days of competition respectively).
7. **Sessions** Each round consists of several sessions. A session is comprised of a set of simulations in different scenarios. A member of the Organizing or Technical Committee will chair each session. The session chair is responsible for executing the simulations, collecting scores and logs, and handling any issues that arise during the session.
8. **Code Submission** All teams must submit the agent team' source-code (binary code will not be accepted) and the compilation scripts before the start of each round. The number and time of submissions as well as specific requirements will be explained during the competition setup time to the team leaders. The Organizing or Technical Committee has the authority to change

the time of submissions and to audit every submitted source-code.

9. **Scenarios** The scenarios will be provided by the Organizing or Technical Committee. Participating teams shall NOT know the disaster scenarios (i.e., map, random seeds, simulator configuration, parameter values, and phases of execution) before the start of the simulation. All conditions for a particular disaster scenario will be identical for all agent teams. A scenario is composed of a map, a set of rescue agents and civilians, and a set of configuration options for each of the simulator components.
10. **Maps** Each map is constrained to a maximum of **10,000 roads** and **10,000 buildings**. The building and road entrances are supposed to be fully connected. A validation tool will be used to check the full connectivity of roads and building entrances in each map. Participating teams do not have the right to complain in case roads or building entrances are not fully connected if evidenced that this was not detected by the validation tool.
11. **Phases** The scenario simulation may be performed in two phases of execution of the agent team's code: the `Pre-Computation phase` (Pre-Computation Phase) and the `Simulation phase` (Simulation Phase). The `Pre-Computation phase` is not mandatory for all scenarios and is assumed a configuration parameter of the scenario. Thus, the execution of the `Pre-Computation phase` will be defined as a configuration parameter of the scenario (see Scenarios).
12. **Pre-Computation Phase** The `Pre-Computation phase` allows an agent of each type to pre-process map- and scenario-specific data and store it into a file to use during the `Simulation phase`. Only one agent of each type can connect to the server and execute the pre-computation algorithm. This phase is limited to **2 minutes** and after the time is elapsed the server will be terminated. Pre-computation is allowed under the conditions:

    a. The data must be generated by a computer program with no human interaction or intervention.
    b. Data for all maps must be generated by a single computer program.
    c. The computer program should work for any new map.
    d. Agent must choose the file to store the pre-computing data.
    e. Agents must be able to work if no pre-computation data is present for the map.
    f. The source-code of the pre-computation program must be released after the competition.

13. **Simulation Phase** The `Simulation phase` corresponds to the agent team' simulation in the competition scenario. All agents have up to **3 minutes** to connect into the simulator kernel. The simulation of the scenario begins no later than **3 minutes** after the first agent begins its handshake with the simulator kernel. All file permissions, except read permission for previously written files, will be removed.
14. **Valid Map** The Organizing or Technical Committee members are entitled to define whether a map result is valid or invalid in a session. The decision is based on the results of the map, for example, it may be decided that a map is invalid when all the teams scores very close in that map.
15. **Valid Game** Participant teams will NOT be entitled to request the rerun of their agent team in most circumstances. In extreme circumstances participating teams may have the right to

request a single rerun. Circumstances that may result in a rerun are:

a. Power failure.
b. Accidental or deliberate termination of a kernel, simulator, or agent process.
c. Java Virtual Machine crash.

In the case of rerun, the last score is used as the official score of the agent team on that scenario. Examples of events that will NOT result in a rerun are:

a. Simulator crash.
b. Agents failing to fully connect before the simulation starts.
c. Agents crashing or failing to act during the run.
d. Apparently incorrect behavior by a simulator or the viewer.
e. Simulator or ADF bug.

Teams that wish to request a rerun must do so in writing. The request must include the participating team's name, the scenario's name, the description of the problem, and the reasons why the team feels a rerun is appropriate. The request must also state whether the request is for a rerun of the team or a full session rerun. Only one Java Virtual Machine crash rerun request is accepted for each session.

16. **Bugs** It is the responsibility of the participating teams to ensure that their code works correctly with the simulator. Although the Organizing and Technical Committee make every effort to provide a reliable simulation environment, they have no responsibility for any kind of software failure at during the competition. RCRS and RCRS ADF bugs are not sufficient grounds to request a rerun.
17. **Bugs Exploitation** Teams that exploit known RCRS or RCRS ADF bugs to gain advantage will be disqualified from the competition. Disqualification will be made only after consultation with the RoboCup Trustees.
18. **Team Leaders' Meetings** Every day of the competition, there will be a team leaders' meeting before the beginning and after the end of the competition day to discuss issues or provide information about the competition. All team leaders of the participating teams in that day shall participate in these meetings, if the team leader fails to participate the team's issues and opinion will be disregarded.
19. **Complains/Comments/Suggestions Only the team leader** of the participating teams can complain, comment, or make suggestions in writing to the Organizing or Technical Committee about the competition. Comments and suggestions of other team members will be disregarded. If these complains, comments, or suggestions are deemed derogatory or abusive then the matter will be referred to the RoboCup Trustees and may result in penalties for the team concerned. Penalties may include points reduction or, in the worst case, disqualification.
20. **Problem Resolution** If a problem arises during a session, **team leaders** can make a request in writing for the session chair to resolve the problem. The session chair can take the decision on the spot or can refer it to the committee. Chair decisions are final, but if a team strongly disagrees, the team can submit a written appeal to the committee. In order to allow the competition to continue, appeals will not be received during a round, but will be discussed by

the committee at the end of each competition day. The Organizing or Technical Committee make final decisions at any condition.

21. **Rule Dispute Resolution** If there is an ambiguity in the rule or any unexpected situation happens, a temporary committee composed of the Technical, Organizing, and Executive Committee members and the local chair have the power to take a decision regarding the issue. The temporary committee decision has the same effect as a rule.
22. **Open Source Policy** Source code files must be released open source immediately after the end of the competition to guarantee fair play and to encourage community activity after competition. Log files and related parameter files will be open access.

3.2. Presentation

The presentation aims to share the knowledge of the participating teams and improve the academic research aspects of the league. Each participating team will have **20 minutes to present** their implementation and another **10 minutes for questions and answers**. A presentation template is available at https://github.com/roborescue/rcrs-templates/blob/master/presentation/presentation_template.pptx.

The presentation will be evaluated by a panel of experts and the leader of the other teams. The final evaluation of the presentation will be incorporated into the score of the preliminary, semifinal and final rounds. The presentations will be evaluated according to a set of criteria:

1. **Relevance** [5 points] Evaluates how relevant is the participating team’s approach to the goals of the RoboCup Rescue. 0 means it is not relevant and 5 means it is very relevant.
2. **Originality** [5 points] Evaluates how original is the proposed participating team’s approach to RoboCup Rescue. 0 means it is not original and 5 means it is very original.
3. **Significance** [5 points) Evaluates how significant to the league is the proposed participating team’s approach. 0 means it is not significant and 5 means it is very significant.
4. **Slide Content** [5 points] Evaluates the quality and completeness of the presentation material with respect to the participating team' strategy and the readability. 0 means that the presentation slides are of poor quality with respect to the content and 5 means that it is very informative and complete.
5. **Slide Structure** [5 points] Evaluates the structure of the presentation material. 0 means that the presentation slides are poorly structured and 5 means that it is very well structured.
6. **Talk** [5 points] Evaluates how clear and easy to follow was the presentation and explanation, and whether the presenter had a positive attitude or not with respect to the presentation and the raised questions. 0 means the presentation is not clear or the presenter had a bad attitude and 5 means the presentation is clear and the presenter had good attitude.

Each team’s presentation score will be calculated taking into account the sum of points given by the other teams excluding the X best and the X worst scores, where X is defined based on the number of competing teams during the first team leaders' meeting, plus the points given by the 2 Committee members who are not member of any participating team on the competition.

The score of each participating team will then be included in the ranking calculation in Ranking as a scenario in all rounds that the participating team participates.

All participating teams will have the same number of evaluations and the same evaluators. In case some evaluator is not able to participate in the evaluation of all participating teams' presentation, his/her evaluation will be disregarded.

Team leaders or a representative assigned by the team leader must be present at all other participating teams presentation. The presentation session chair will check if the representative of each team is present at the beginning of each presentation, if there is no representative present the missing participating team’s presentation score will be set to 0 (zero) and all the scores assigned by that participating team will be disregarded.

3.3. Ranking

Each round is composed of several sessions (S), and at each session the participating teams receive an identification ranging from t_1 to t_n, where n represents the number of participating teams in that session.

Each session is comprised of a set of scenarios (M), and each scenario also receives an identification ranging from m_1 to m_p, where p represents the number of scenarios in that session.

A score SC^k_{ji} is assigned to each team $i \in T$ ($T = (t_1, \ldots, t_n)$) at each session $k \in S$ ($S = (s_1, \ldots, s_n)$) for each scenario $j \in M$ ($M = \left(m_1, \ldots, m_p\right)$).

For each session k and scenario j, the Selective Minimum (SM^k_j) is calculated as

$$SM^k_j = max\left(SC^k_{ji}\right) - \left(\left(max\left(SC^k_{ji}\right) - mean\left(SC^k_{ji}\right)\right) \times 2\right),$$

and the Maximum Score (MS^k_j) is calculated as

$$MS^k_j = n \times SDC,$$

where n is the number of participating teams on session k, and SDC is the coefficient indicating the step between points among teams (we will use $SDC = 2$ in RoboCup Rescue 2025 competition). The maximum value of each step is calculated as

$$MSS^k_{j\,/\,step\, \in \left(1,\, \ldots,\, MS^k_j\right)} = \frac{\left(\left(max\left(SC^k_j\right) - SM^k_j\right)\right)}{\left(MS^k_j \times \left(MS^k_j - step\right)\right)}$$

To each participating team is assigned the step value, whose $MSS^k_{j\,/\,step}$ value is lower than the participating team' score, but the $MSS^k_{j\,/\,step+1}$ value is greater than the participating team' score.

$$TP^k_{ji} = step \therefore MSS^k_{j\,/\,step} < SC^k_{ji} < MSS^k_{j\,/\,step+1}$$

The final team score for each scenario and participating team is calculated as

$$FTS^k_i = \sum_{j = m_1}^{m_p} TP^k_{ji}$$

The final team score is then used to generate a ranking of all the participating teams for that session. The participating team with the highest final team score is ranked as first, the second highest as second, and so on.

Appendix A: Parameters

The following tables show the simulator and scenario parameters and their accepted ranges for the competition. Note that agents are not able to access the values of all these parameters.

Table 1. Number of agents and refuge information.

Entity	Min	Max
Fire Brigade	0	100
Police Force	0	100
Ambulance Teams	0	100
Fire Station	0	1
Police Office	0	1
Ambulance Centre	0	1
Civilians	0	1000
Refuges	0	Unlimited
Refuge Capacity	0	Unlimited

Table 2. Ranges for simulation parameters common to all components.

Parameter	Description	Range
random.seed	Seed for random number generator	Any range

Table 3. Ranges for kernel parameters.

Parameter	Description	Range
kernel.timesteps	Number of simulation timesteps	100 - 1,000
kernel.agents.think-time	Number of milliseconds each agent has to send commands	500 - 3000

Table 4. Ranges for general communication channels configuration parameters.

Parameter comms.channels.*	Description	Range
count	Number of communication channels	1 - 20
max.platoon	Number of channels a platoon agent can subscribe to	0 - 10
max.centre	Number of channels a centre agent can subscribe to	0 - 20

Table 5. Ranges for voice channel parameters.

Parameter comms.channels.<x>.*, where <x> is channel number	**Description**	**Range**
type	Type of the channel	voice
range	Maximum range of a message in mm	0 - 300,000
message.size	Maximum size of a voice message in bytes	64 - 2,048
message.max	Maximum number of a voice message an agent can send per timestep	1 - 100

Table 6. Ranges for radio channel parameters.

Parameter comms.channels.<x>.*, where <x> is channel number	**Description**	**Range**
type	Type of the channel	radio
bandwidth	Maximum capacity of the channel in bytes per timestep	0 - 8,192

Table 7. Ranges for voice and radio channel parameters.[1]

Parameter comms.channels.<x>.*, where <x> is channel number	**Description**	**Range**
type	Type of the channel	radio | voice
noise.[input | output].failure.use	Whether or not to use failure noise on channel <x> in input or output	yes | no
noise.[input | output].failure.p	Probability of message failure	0 - 1
noise.[input | output].dropout.use	Whether or not to use dropout noise on channel <x> in input or output	yes | no
noise.[input | output].dropout.p	Probability of message dropout	0 - 1

Table 8. Ranges for perception parameters.

Parameter comms.channels.<x>.*, where <x> is channel number	**Description**	**Range**
perception.los.max-distance	Maximum distance an agent can perceive world changes	30,000 - 150,000

Table 9. Ranges for collapse parameters.

Parameter	**Description**	**Range**
collapse.wood.*	Proportion of wooden buildings with each degree of damage	0 - 1 (Sum = 1)
collapse.steel.*	Proportion of steel buildings with each degree of damage	0 - 1 (Sum = 1)
collapse.concrete.*	Proportion of concrete buildings with each degree of damage	0 - 1 (Sum = 1)

Table 10. Ranges for misc simulator parameters.

Parameter misc.* <type> = wood | steel | concrete	**Description**	**Range**
buriedness.<type>.severity.rate	Probability that an agent in a collapse building of type <type> with a degree of collapse severity will be buried	0 - 1
buriedness.<type>.severity.value	Initial buriedness value for a buried agent in a collapsed building of type <type> with a degree of collapse severity	0 - 200
injury.collapse.<type>.severity.slight	Probability that an agent inside a collapsing building of type <type> with a degree of collapse severity will receive a slight injury	0 - 1
injury.collapse.<type>.severity.serious	Probability that an agent inside a collapsing building of type <type> with a degree of collapse severity will receive a serious injury	0 - 1

injury.collapse.<type>.severity.critical	Probability that an agent inside a collapsing building of type <type> with a degree of collapse severity will receive a critical injury	0 - 1
injury.collapse.slight	Amount of damage that a slight injury due to collapse causes	0 - 10,000
injury.collapse.serious	Amount of damage that a serious injury due to collapse causes	0 - 10,000
injury.collapse.critical	Amount of damage that a critical injury due to collapse causes	0 - 10,000
injury.collapse.multiplier.<type>	Damage multiplier for an agent of type <type> due to collapse	0 - 1
injury.bury.<type>.severity.slight	Probability that an agent buried inside a building of type <type> with a degree of collapse severity will receive a slight injury	0 - 1
injury.bury.<type>.severity.serious	Probability that an agent buried inside a building of type <type> with a degree of collapse severity will receive a serious injury	0 - 1
injury.bury.<type>.severity.critical	Probability that an agent buried inside a building of type <type> with a degree of collapse severity will receive a critical injury	0 - 1
injury.bury.slight	Amount of damage that a slight injury due to buriedness causes	0 - 10,000
injury.bury.serious	Amount of damage that a serious injury due to buriedness causes	0 - 10,000
injury.bury.critical	Amount of damage that a critical injury due to buriedness causes	0 - 10,000
injury.bury.multiplier.<type>	Damage multiplier for an agent of type <type> due to buriedness	0 - 1

injury.fire.<type>.severity.slight	Probability that an agent inside a burning building of type <type> with a degree of collapse severity will receive a slight injury	0 - 1
injury.fire.<type>.severity.serious	Probability that an agent inside a burning building of type <type> with a degree of collapse severity will receive a serious injury	0 - 1
injury.fire.<type>.severity.critical	Probability that an agent inside a burning building of type <type> with a degree of collapse severity will receive a critical injury	0 - 1
injury.fire.slight	Amount of damage that a slight injury due to fire causes	0—10,000
injury.fire.serious	Amount of damage that a serious injury due to fire causes	0—10,000
injury.fire.critical	Amount of damage that a critical injury due to fire causes	0—10,000
injury.fire.multiplier.<type>	Damage multiplier for an agent of type <type> due to fire	0 - 1
injury.<type>.k	k parameter for the damage progression function for injury type <type> (collapse, bury, fire)	0 - 1
injury.<type>.noise.mean	Mean noise added to the damage progression function for injury type <type> (collapse, bury, fire)	0 - 1
injury.<type>.noise.sd	Standard deviation of noise added to the damage progression function for injury type <type> (collapse, bury, fire)	0 - 1

Table 11. Ranges for clear simulator parameters.

Parameter	**Description**	**Range**

clear.repair.rate	Rate of road clearing per police force agent in square m per timestep	0 - 50,000

Table 12. *Parameters agents are guaranteed to be able to query.*

Parameter	**Description**
kernel.agents.think-time	See Table 3
kernel.startup.connect-time	See Section 3.1 - (12) and (13)
comms.channels.count	See Table 4
comms.channels.<x>.type	See Table 5 and Table 6
comms.channels.<x>.range	See Table 5
comms.channels.<x>.messages.size	See Table 5
comms.channels.<x>.messages.max	See Table 5
comms.channels.<x>.bandwidth	See Table 6
clear.repair.rate	See Table 11
scenario.agents.fb	Number of Fire Brigades
scenario.agents.fs	Number of Fire Stations
scenario.agents.pf	Number of Police Forces
scenario.agents.po	Number of Police Offices
scenario.agents.at	Number of Ambulance Teams
scenario.agents.ac	Number of Ambulance Centres
kernel.communication-model	Communication model class name
kernel.perception	Perception model class name

[1] Noise can be specified as input or output (or both). Input noise is applied as the agent sends a message to the server; output noise is applied as an agent receives a message. Thus, input noise is identical for all receivers but output noise is unique to each receiver. There are two types of noise: failure noise and dropout noise. Failure noise means a message disappears completely with no notification to either the sender or the receiver. Dropout noise removes the content of a message but the receiver still receives a zero-length communication from the sender, i.e., the sender knows a message was sent but the content is lost.