Database Driven RoboCup Rescue Server for RoboCup 2010

Rahul Sarika, Harith Siddhartha, and Kamalakar Karlapalem

International Institute of Information Technology, Hyderabad,
Gachibowli, Hyderabad-500032, India
{rahul_sarika,harith}@research.iiit.ac.in
kamal@iiit.ac.in
http://mas.iiit.ac.in

Abstract. We build up on Database Driven RoboCup Rescue Server (DDRRS) v1.0 to improve its speed, scalability, reliability and robustness to give DDRRS v1.1. We also talk about the Score Vector, the evaluation scheme to be used in this year's competition. We then conclude with the idea of an evolving Score Vector and a distributed DDRRS.

1 DDRRS v1.1

In the year 2008, we emphasised on the need to scale up the number of agents and sizes of the maps used to make the simulations realistic. We also gave a database centric solution to this problem, Distributed Database RoboCup Rescue Server (DDRRS) [5]. DDRRS version 1.0, which was demonstrated during RoboCup 2008 held at Suzhou, successfully scales up to 500 rescue agents with 1500 civilians on standard maps like Kobe, Foligno et cetera.

One of the major drawbacks in the design of the original RoboCup Rescue Simulation Server [1], RCRSS v49.*, is the redundancy and unusually large overhead in messages passed between the kernel and other simulators. The kernel sends the same updated information to every simulator regardless of its type. For example, suppose a fire fighting agent submits a fire extinguish command. The traffic simulator only simulates the motion of the agents. The traffic simulator does not need the information of the fire fighter which has submitted an extinguish request. Similar redundancies can be observed during a kernel-simulator conversation.

This problem was solved by DDRRS with the help of a database. A snapshot of the database schema as given in Tables 2, 3, 4 was designed such that any information required by a simulator was just one query away. At the beginning of the simulation, all the simulators are given certain database privileges which ensure that no simulator can go 'rogue'. In the case of the fire fighting agent example which we took earlier, the traffic simulator has no permission to view or update the information related to the AK_EXTINGUISH request submitted by the fire fighting agent. Each simulator requests the database for information it really needs.

During a few experiments, we observed that the time taken to update the tables was longer than estimated. It so happens that when a field in a certain row in a table is updated, all the fields in that particular row are accessed. Table MovingObjects (Table 2) which contains the information of all the agents, Buildings (Table 3) which contains the information of all the buildings and Roads (Table 4) which contains the information of all the roads contain fourteen, eighteen and eighteen columns respectively. In the table Roads, the fields actively accessed and updated are roadID (ID of the road) and block (width of the road block). In case of the the table Buildings, buildingID (ID of the building), temperature (temperature of the building) and a couple of other attributes are accessed and updated frequently. One can imagine how slow a simulator's performance becomes during an update of the state of mere hundred buildings. A small example here will help us see the problem very clearly. The traffic simulator (1) processes an agent's AK_MOVE command, (2) simulates the motion of the agent and (3) updates the new position, position_history and position_extra of that agent. In order to do this, the simulator extracts the current position (position) of the agent from the table MovingObjects (Table 2) and after it simulates the AK_MOVE command submitted by that agent, the simulator updates the fields position (current position of the agent), positionHistory (path taken the by agent from the start of the game till now) and positionExtra (offset in the position of the agent if it is on a road) in the table MovingObjects.

To overcome this problem, we fragmented the database schema. Table MovingObjects (Table 2) was fragmented into Agents (Table 5), Agents_Position (Table 6), Agents_State (Table 7), Agents_Other_Attributes (Table 8). Agent_Position (Table 6) now contains information related to only the motion of the agents which is used by the traffic simulator. Similarily, the tables Buildings and Roads have also been fragmented into Buildings (Table 11), Buildings_Position (Table 12), Buildings_Other_Attributes (Table 13) and Roads (Table 9), Roads_Other_Attributes (Table 10 respectively.

This modification is the first of the two major features of DDRRS v1.1, the next version of DDRRS v1.0. With this feature, the number of agents that can be simulated can now go beyond thousand, thus bringing us a little closer to our goal to simulate millions of civilians and thousands of agents in larger cities. Fragmentation of the database in this manner also paves way towards the implementation of a distributed version of DDRRS, DDRRS v2.0.

2 Score Vector

Score Vector [6], which we introduced in DDRRS v1.0 in the year 2008, has been selected to be used as the official evaluation scheme for the 2009 RoboCup Rescue Competition. It neatly captures numerous inter-related aspects such as progress of civilian state of health, rescue operations, et cetera. for evaluating a team. Without taking these inter-related aspects into consideration, as in the current scoring method, the scoring is rendered poorly as it cannot capture the true picture of the state of the teams' performance. These aspects can help the

organizers of the competition set new challenges and shape the existing ones into a rigid form. The participants can make use of the same to investigate and rectify any defects present in their algorithms.

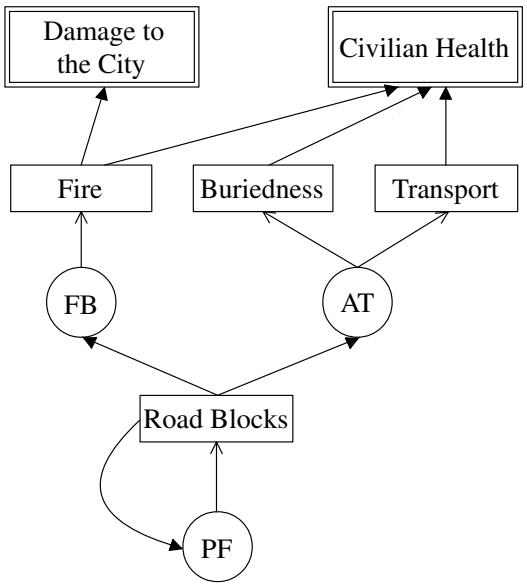

Fig. 1. Score Dependancies

Figure 1 shows the dependencies among the factors that affect the score of any RCRSC. **Damage to the City** and **Civilian Health** at the top are the two most important factors as mentioned in the Section 2. However, **Damage to the City** is caused by ***Fire*** which is taken care of by the fire brigade (FB). There are two issues for FB, namely ***Road Blocks*** and limited capacity to carry water. ***Road Blocks*** are cleared by the police force (PF) and lack of coordination between FB and PF results in FB travelling longer distances than required to extinguish fires and refill tanks. This results in a wastage of time and fuel, and additional damage to city. A similar argument can be put forth in the case of the ambulance team (AT). **Civilian Health** is affected by ***Fire***, ***Buriedness of Civilian*** and ***Transport of Civilian to Refuge***. AT's job is to rescue civilians from the debris and transport them to the refuge which can get delayed due to the presence of ***Road Blocks***.

With so many factors influencing the game, it is not correct to decide the winner with just a scalar quantity. A vector defined by these factors gives clarity to the approach of a team towards the game and also gives confidence to decide the deserving winner. Preliminary factors we needed to consider for better evaluation of the teams so as to enhance the level of competition are given in Table 1.

Table 1. Factors influencing the performance of a rescue team and the type of influence on the score

No	Factor	Influence on the score	Objective for teams
A	Agents in the following categories:		
	1. Dead (0<=HP<=10)	Negative	Minimize
	2. Critical (11<=HP<=40)	Negative	Minimize
	3. Average (41<=HP<=70)	Positive	Maximize
	4. Healthy (71<=HP<=100)	Positive	Maximize
B	Time spent by a rescue agent travelling in the city	Negative	Minimize
C	Average number of messages passed amongst rescue agents	Negative	Optimize
D	Ratio of civilians in refuge	Positive	Maximize
E	Ratio of civilians rescued	Positive	Maximize
F	Percentage of building area destroyed	Negative	Minimize
G	Ratio of fires extinguished	Positive	Maximize
H	Average time taken to		
	1. Rescue a civilian	Negative	Minimize
	2. Extinguish a fire	Negative	Minimize
	3. Transport a civilian to a refuge	Negative	Minimize

Unlike a regular scalar quantity currently used, the Score Vector provides multiple prespectives on behaviour and performance of agent teams. In fact, contradictory and seemingly impossible tasks which require agent teams to make tradeoffs will be measured by the Score Vector. It is truly a challenge for an agent team to use a single strategy or approach to optimize the parameters of Score Vector. The agent competition can be designed such that agent teams cannot predict which parameters are important and need to be considered. Further, the parameters themselves can be weighed to rank the teams based on the difficulty levels of different performance aspects of the agent teams. With the Score Vector, teams can identify which aspects of their strategy are good and which are not. This gives them a better way to analyze and compare their performances with others during a game as well as after the game.

3 Future Work and Conclusion

After RoboCup 2008, we worked on enhancing DDRRS v1.0 and Score Vector. We have succeeded in speeding up DDRRS in the form of DDRRS v1.1 and building a better evaluation scheme. We intend to give a solid demonstration at Graz this year and convince the RoboCup Rescue community to use DDRRS for RoboCup 2010.

Our future work is to build a distributed DDRRS which is essential for a massive simulation scenario and an self evolving Score Vector which adjusts the weights given to parameters by observing the performance of the agent teams during the course of the game.

Table 2. MovingObjects

Field	Type
objectID	int
position	int
positionExtra	int
stamina	int
hp	int
damage	int
buriedness	int
direction	int
water	int
objectType	int
contents	int
moves	text
positionHistory	text
lastUpdatedTime	int

Table 3. Buildings

Field	Type
buildingID	int
buildingType	int
posX	int
posY	int
floors	int
attributes	int
ignitionFlag	int
fieryness	int
brokenness	int
entrances	int
apexes	int
groundArea	int
totalArea	int
buildingCode	int
importance	int
temperature	int
lastUpdatedTime	int
objectsInRange	text

Table 4. Roads

Field	Type
roadID	int
head	int
tail	int
length	int
roadKind	int
carsToHead	int
carsToTail	int
humansToHead	int
humansToTail	int
width	int
block	int
repairCost	int
medianStrip	int
linesToHead	int
linesToTail	int
widthForWalkers	int
lastUpdatedTime	int
objectsInRange	text

Table 5. Agents

Field	Type
id	INT
stamina	INT
direction	INT
agentType	INT
lastUpdatedTime	INT

Table 6. Agents_Position

Field	Type
id	INT
position	INT
positionExtra	INT
positionHistory	text
path	text
lastUpdatedTime	int

Table 7. Agents_State

Field	Type
id	INT
hp	INT
damage	INT
buriedness	INT
lastUpdatedTime	INT

Table 8. Agents_Other_Attributes

Field	Type
id	INT
contents	INT
water	INT
lastUpdatedTime	INT

Table 9. Roads

Field	Type
id	INT
roadKind	INT
carsToHead	INT
carsToTail	INT
humansToHead	INT
humansToTail	INT
medianStrip	INT
linesToHead	INT
linesToTail	INT
widthForWalkers	INT
lastUpdatedTime	INT

Table 10. Roads_Other_Attributes

Field	Type
id	INT
head	INT
tail	INT
length	INT
width	INT
block	INT
repairCost	INT
lastUpdatedTime	INT

Table 11. Buildings

Field	Type
id	int
buildingType	int
floors	int
ignitionFlag	int
entrances	int
apexes	int
groundArea	int
totalArea	int
buildingCode	int
importance	int
lastUpdatedTime	int

Table 12. Buildings_Position

Field	Type
id	INT
posX	INT
posY	INT
attributes	INT
lastUpdatedTime	INT

Table 13. Buildings_other_Attributes

Field	Type
id	int
fieryness	int
brokenness	int
temperature	int
lastUpdatedTime	int

References

1. T. Takahashi, et al. : Agent Based Approach in Disaster Rescue Simulation - From Test-Bed of Multiagent System to Practical Application, RoboCup (2001)
2. T. Takahashi : RoboCupRescue Simulation League, RoboCup (2002)
3. RoboCup Rescue Homepage: http://www.RoboCupcuprescue.org
4. RoboCup Rescue Wiki: http://www.robocuprescue.org/wiki/index.php?title=RSL2009
5. Rahul Sarika : Database Driven RoboCup Rescue Server, RoboCup (2008)
6. Harith Siddhartha : Retrospective Analysis of RoboCup Rescue Simulation Agent Teams, AAMAS (2008)