

AGENT DISPERSION

Ankit Kumar Srivastava

<ankitks@mail.utexas.edu>

Abstract

This paper presents impact of implementing agent dispersion in a multiagent domain, namely simulated RoboCupRescue (*RCR*). Agent dispersion in this context refers to spreading agents by dividing the disaster space into a certain number of sections and assigning a certain number of agents to each section. The scope of an agent is limited to its allocated section thus reducing the search space of an individual agent. I partnered with Byung Kang <byungkon5@gmail.com> to research on this topic as part of a final project for the course CS 378: Autonomous Multiagent Systems.

1. Introduction

A massive earthquake measuring 6.9 on the Richter scale has hit Kobe, Japan's most important port. The ground shakes and tremors create a great shear force toppling a large number of buildings. Broken gas lines cause fire to break out. *Civilians* are trapped in the debris choking streets. Public utilities are damaged, railway lines are broken and roads are blocked by massive wreckage. The major lifelines to the city center suffer severe damage, greatly delaying rescue efforts of *fire brigades* and *ambulances*. *Police officers* clear blocked routes to enable expeditious arrival of the rescue teams at disaster sites.

The RCR simulator models the afore-mentioned scenario. The simulated city with the layout map displaying current positions of each agent is called the *world*. Consider the world is divided into an x number of sections. Agents are distributed among these

sectors such that each agent functions within its own section only. Thus each individual agent will have a smaller portion of the world to explore, i.e. its own unit. The project implements this mechanism of spreading agents across the disaster space and aims to determine the impact of agent dispersion on a multiagent domain like the RoboCupRescue.

Section 2 of the paper states the team construction and an overview of the code used. Section 3 describes the dispersion algorithm in detail, followed by merits and deficiencies in section 4. Section 5 contains some experimental tests and analysis results, ending with the conclusion in section 6. A bibliography of research papers and online links comprises the last section 7.

2. Team Construction

As mentioned in the proposal and the progress report, the Java files from programming assignment 4 have been used as a base code for this project. The existing files in the sample package (shown in Figure 1 below) have been modified in addition to adding a couple of new classes. The rest of the code base has been left intact.

Initially the dispersion algorithm described in the next section was implemented only on the police agents. The motivation for this choice of agents was the fact that the action of a police agent, i.e. clearing blocked roads, is a vital step in the rescue process. When a fire brigade or an ambulance, on its journey to a disaster site, encounters an impassable road, the block can be removed by a police agent only. Until such time that a police agent answers the blockade report, the fire brigades and ambulance vehicles are stuck at the spot lest they find an alternate route to the site. Therefore it was concluded

that the functions of the fire brigades and ambulances are directly proportional to the efficiency of police agents.

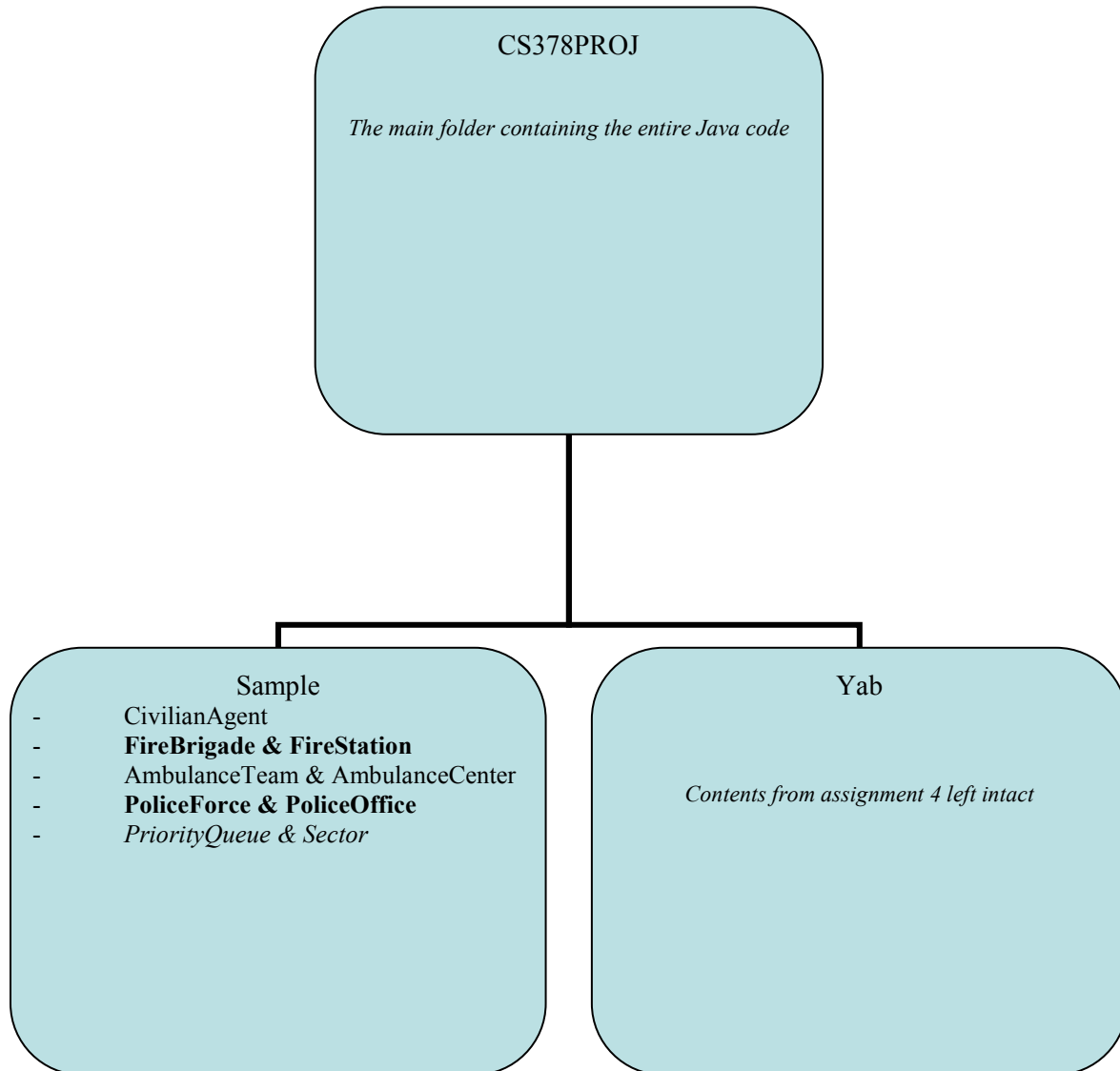


FIGURE 1: Overview of the Java code; Bold text indicates modified classes and italicized text indicates new classes.

After implementing dispersion in the police agents, it was observed that the ambulances (code from assignment 4) were very inefficient and affecting the end result.

It was concluded from observing around fifteen simulations that:

- a. At least two ambulances moved together with the result that both would get stuck on encountering a road block.
- b. Ambulances sometimes froze. Without any apparent reason they just stayed at the spot.

These behaviors resulted in a decrease in the number of civilians rescued, thus affecting the end score of the simulation. The formula for the end score is specified in a box in section three.

Note that these discrepancies were noticed only after most of the coding and the progress report were written. So the dispersion algorithm had already been implemented and tested on the police agents. Thus after some test print statements, i.e. printing data values at various stages of the ambulance process, it was decided to reuse the dispersion algorithm on the ambulance agents. This took care of problem number a. The world was divided into sectors and each ambulance agent out of the five (default value) was assigned one separate sector. Thus each ambulance roamed in its own sector avoiding flocking behavior. An important point is that the division of the world for ambulance agents is different from the division of the world for police agents. This is because by default the number of police agents differs from the total number of ambulance agents.

To solve problem number b, a stalling status check was introduced. If an ambulance has remained at the same motionless position for 3 cycles, the current destination building is removed from its list and the ambulance moves to the next

destination building in sector. The pros and cons of this strategy have been discussed in section 4. Note that the value of 3 cycles was arbitrarily chosen without any logical basis. If more time were available experiments could have been conducted to determine an optimum value for the number of cycles of delay.

As evident from Figure 1, two new classes have been added. These consist of:

- A. PRIORITY QUEUE: This class implements a simple version of priority queue using linked lists as the containing data structure.
- B. SECTOR: This class describes a sector, i.e., the units into which the world is divided. It maintains a list of roads (in case of police agents) or buildings (in case of ambulance agents) contained inside its boundaries.

3. Dispersion Algorithm – Motivation and Description

Each RCR simulation runs for a duration of 300 cycles. The simulator viewer also displays a “score” alongside the cycle length. The score calculation is described below.

$$\text{SCORE} = (P + \text{hpF} / \text{hpI}) * \sqrt{(\text{nbF} / \text{nbI})} ;$$

P = number of agents

hpI = \sum (health point of a Humanoid (agent / civilian) at time = 0)

hpF = \sum (health point of a Humanoid (agent / civilian) at time = 300)

nbI = \sum (area of a building at time = 0)

nbF = \sum (area of a building (with fieryness @ 0) at time = 300)

This score at the end of a simulation (time = 300) will be used as a yardstick for efficiency in the experiments in section 5. The higher the score, the better is the performance of the RCR simulation. The more number of humanoids and buildings saved, the higher is the value in the numerator leading to a higher score.

As mentioned previously, the project in its final form has dispersion implemented on both police and ambulance agents. Recall the primary function of police agents is to

clear blocked roads and that of the ambulance agents is to rescue people trapped under debris of buildings, etcetera.

As is evident from the formula of the score, the primary aim of any strategy intended to improve the performance of a RCR domain should be to save as many people and buildings as possible. And in order to save the buildings and people, roads along the route must be free of blockades. The world has 820 roads, 730 buildings, 10 police agents and 5 ambulance agents by default in the base code. Given the sheer size of the world, that is the number of roads and buildings in contrast to the number of police and ambulance agents, how can one ensure maximum performance?

Simulation of road blocks and weakening of civilians is random. However at any given cycle more than one instance of either casualty can exist at any location in the world. Also note that the number of cycles required to clear a road block, extinguish a building or load and unload a civilian is directly proportional to the amount of time elapsed since its inception or time of instantiation. Hence a police agent, fire brigade agent or an ambulance agent must reach the intended destination as quickly as possible to maximize performance.

The problem is thus reduced to implementation of an efficient search algorithm. The world with its collection of roads, buildings and civilians is the search space. The efficiency of the search algorithm will be based on the speed with which an agent can search for a disaster site and move from its current location to the target.

The classical strategy of *divide et impera*, i.e. divide and conquer, comes to mind. Consider the world is divided into many smaller sections known as *sectors* and an agent assigned to a particular sector is confined within the boundaries.

This infers:

- A. A shorter list of buildings or roads to be searched and traversed.
- B. Reduced time to reach a target because all targets will be reachable within a definite limit, less than distances in the undivided world.

Thus it was formulated that one way to maximize performance of a RCR simulator will be to implement agent dispersion that is spreading the agents by dividing the world into sectors and restricting movement of agents to their assigned sector.

The world is divided into 'n' sectors. This value of 'n' is further discussed in section 5 under experiments. Initially though, the simulation was run for value of 'n' varying from 1 to 10. Each sector is of length 'L', where 'L' is equal to the number of nodes contained.

Note

- A. The total number of sectors can at most be equal to the total number of agents.
- B. 1 sector implies the default case, when the world is undivided.
- C. In accordance with the default number of police and ambulance agents, the number of sectors in case of police is ten and that in case of ambulance is five.
- D. The nodes in case of police imply a list of roads, while that in case of ambulance imply a list of buildings.

The assignment of nodes (roads or buildings) to a sector is executed as follows. The entire list of nodes, (buildings or roads as the case may be), is traversed. Given the x and y coordinates of a node, a sector number (in the range of 1 to 'n' where 'n' <= # of agents) is calculated using the algorithm given below.

```

// Given x , y (the coordinates of the node) and div (the total number of sectors)
getSector () {
    1. Calculate width and height, the span of the world
    2. If there are even sectors then  $xv = div / 2$  (# divisions along x direction)
       and  $yv = 2$  (# divisions along y direction). Else  $xv = div$  and  $yv = 1$ 
    3. Change the base of the node,  $x = x - minx$  and  $y = y - minY$ 
    4.  $dx = width / xv$  and  $dy = height / yv$ 
    5.  $t1 = x / dx$  and  $t2 y / dy$ 
    6. Sector number =  $t1 + t2 * xv$ 
}

```

Once the sectors are created, the police office will allocate police agents and ambulance center will allocate ambulance teams. Initially, each sector gets assigned to one agent. If there are more agents than sectors, then the remaining agents will be deputed to high priority sectors (i.e. sectors having more number of nodes).

Thus the world has been segmented into sectors and agents have been dispersed. The agents assigned to a particular sector can only move within its enclosed boundaries. This is ensured by keeping track of which nodes belong to which sector. An additional feature was added to enhance the performance. If at any time, an agent's list of unseen nodes becomes empty, it will set its status to "available" and send a specific message to its center (police office or ambulance center). Thus the center can assign the free agent to assist another sector. The details of this mechanism are given in the box below.

1. If Agent's status is "done", get its id.
2. Get the busiest sector, i.e. one having highest number of nodes
3. Assign this sector to the agent
4. Set the Agent's status to "busy"

4. Merits and Deficiencies

The agent dispersion increases the efficiency of the simulation. However there are certain factors still unaccounted for. For example, the ambulances, despite being dispersed into sectors still malfunction occasionally. Given time, we would have come up with a better algorithm for the ambulance agents. However our primary purpose was to increase the efficiency of police agents and we were successful in that.

5. Experimentation and Analysis

In our experiment, we adopted this score at the end of a simulation (time = 300) as a yardstick. The higher the score, the better is the performance of the RCR simulation. In order to examine the effect of dividing the world into 'n' sectors on the efficiency of a simulation measured in terms of its score, the following **procedure** was carried out.

```

FOR ( n = 1 TO 10 )
{
    Run_Simulation;
    At time = 300, note the score;
    n = n + 1;
}

```

Thus ten scores were recorded and tabulated as shown below.

(N,SCORE)	(1,19.31)	(3,22.22)	(5,20.41)	(7,20.21)	(9,30.06)
(N,SCORE)	(2,33.18)	(4,30.75)	(6,33.17)	(8,28.82)	(10,31.94)

Statistical tests were then performed on these observations to formulate and verify two statements. It was discovered that a world with even number of precincts invariably

performs better than a world with odd number of precincts. This might be by chance as we were not able to come up with any logical reasoning. Secondly any world with more than one sector performs better than the default case of 1 sector; i.e. when there is no dispersion of agents.

To illustrate the former statement, student's t-test was performed. The above table was divided into two sets. Sample A consisted of the scores of odd number of sectors (top row) and sample B consisted of the scores of even number of sectors (bottom row).

STUDENT'S T-TEST

SAMPLE A = {19.31, 22.22, 20.41, 20.21, 30.06} <= odd

SAMPLE B = {33.18, 30.75, 33.17, 28.82, 31.94} <= even

AIM: To perform a significance test to compare the mean scores of the 2 samples.

NULL HYPOTHESIS: m_A (Mean of Sample A) = m_B (Mean of Sample B)

ALTERNATE HYPOTHESIS: $m_A \neq m_B$

RESULTS: The calculations were performed by the t-test calculator at

http://www.physics.csbsju.edu/stats/t-test_NROW_form.html

t = 4.29

degrees of freedom = 8

p-value = 0.003

CONCLUSION: The probability of the null hypothesis being true is 0.3 %

NOTE: Therefore, there is a 99.7 % chance that $m_A \neq m_B$. Now, $m_A = 22.4$ and $m_B = 31.6$. Hence **sample B, or even sectors has a better mean score and performance.**

The box given below demonstrates the truth of the second statement. When there is no dispersion, i.e. the world is just one sector, the simulation performs the worst.

DATA = {19.31, 33.18, 22.22, 30.75, 20.41, 33.17, 20.21, 28.82, 30.06, 31.94}

MEAN = 27.007

STD. DEV. = 5.76

MEDIAN = 29.44

MIN. SCORE = 19.31

MAX. SCORE = 33.18

SCORE FOR (n=1) = 19.31

CONCLUSION: In the list of scores corresponding to the number of sectors varying from 1 to 10, the score of the world with 1 sector corresponds to the lowest score of the data sample. Hence 1 sector world or **world with no dispersion implies worst performance.**

The above experiment of running the simulation for 10 different n values, performing the t-test, and summarizing the statistics was carried out two more times. Both reruns were consistent with the above conclusions.

6. Conclusion

Agent dispersion as described in this paper is an effective evaluation benchmark for rescue strategies. We concluded that the number of sectors must be fixed to number of agents if number of agents is even, otherwise one less than the number of agents.

7. Bibliography

1. RoboCupRescue website

[<http://www.rescuesystem.org/robocuprescue/index.html>]

2. Earthquake effects in Kobe, Japan

[<http://www.seismo.unr.edu/ftp/pub/louie/class/100/effects-kobe.html>]

3. Sycara, K., Multi Agent Systems, AI Magazine 1998

4. Scerri P. and Tambe M., The DEFACTO System: Training Tool for Incident Commanders, AAAI 2005