Intelligent distributed systems (BMEVIMIAC02)
Homework documentation

Logistics organization of hospital patient care and accommodation with intelligent systems

Márton Bendegúz Bankó (AF1R7P)
Márk Balázs Hain (YZMF1G)
Tamás Lukács (H4LUX2)

2021. 05. 14.

Content

Task description

The intelligent space of the task is the phase of the patient's medical history in a hospital, which is defined by the patient's arrival at the hospital and their placement and care in the appropriate department.

According to the basic problem, agents who detect and work in this space help to optimally and as quickly as possible place the incoming and already diagnosed patient according to his problem.

Patients in need of hospital treatment are prioritized within the institution according to the type and condition of their illness.

The hospital environment consists of a reception desk and several wards. The reception represents the entrance, where newly arrived patients start their life journey within the hospital. Hospital wards are the places where patients are cared for, and the goal of patients arriving at the reception is to be admitted to the appropriate department.

Patients' complaints are classified according to the departments present in the hospital, and the patient can be treated in the appropriate department. There will always be a department suitable for the care of the patient. The patient's recovery is automatic, after a certain period of time spent in the ward, he recovers, he leaves the ward without further intervention, and his place is freed up.

The system keeps track of the occupancy of individual hospital departments, the location of patients and the distances between the hospital departments, the hospital reception and the available agents. With the help of these, he ensures the optimal utilization of the facility.

Solution summary

We tried to solve the task in a modular way, to distribute the responsibilities necessary to handle the problem equally between the agents and their environment.

We have created two types of agents: the "carrier" agent that transports the patients, and the "manager" agent that assigns tasks to the transporters.

The manager's main task is to detect newly arriving patients and assign a carrier agent to take the patient to the right place. The fact of perception is obtained from the environment, the environment gives a signal that a new patient is waiting for transport at the reception. After that, by running an auction protocol, it decides which cappoints an arrier agent to pick up the patient, transport him to the appropriate hospital ward and place him there.

The carrier agent is the agent who transports the patients from the reception to the appropriate hospital department. Its main functions include moving around the environment, as well as picking up and placing patients in the given position. It takes into account the position of the reception and individual departments of the hospital, and can participate in the auction call when a new patient arrives by the manager with the distances calculated from this and his own position. The auction bid is recalculated for each auction, as the agent's position in the environment can constantly change.

The biggest part of the connection between the two agents is the auction protocol. To solve this task, we implemented the Contract Net Protocol (CNP). This protocol is suitable for distributing tasks in a multi-agent system.

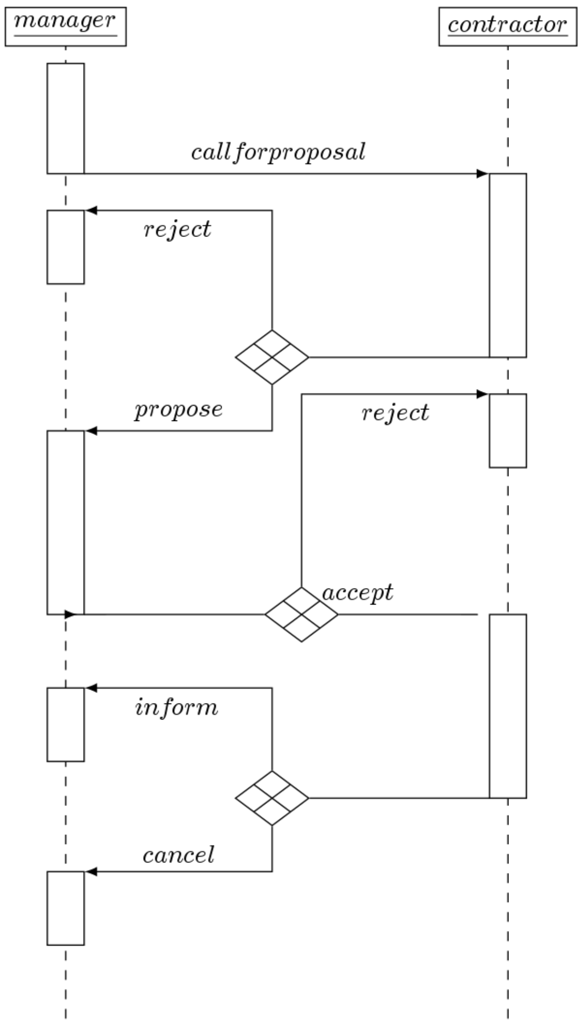

Figure 1: Diagram of CNP run
source: Wikipedia

When the environment is created, before the auction protocols run, the delivery agents introduce themselves to the manager, announce their existence and their names. This is necessary for the CNP to function.

The auction protocol of the CNP starts with the manager agent, upon the arrival of the new patient, he notifies the agents registered with him (who have introduced themselves) that a new patient is available, and also sends the necessary and influencing information about the patient to calculate the bid (to which department it should be transported).

As a result of the invitation, the delivery agents calculate the value they will bid on, depending on their own situation and the tasks they have already undertaken. This bid will be the agent's distance from the front desk if he has completed all of his committed tasks.

CNP allows an agent to decline to participate in the auction, not to bid and not be awarded the task. We did not use this in our solution, all carrier agents will bid during each auction event.

After the advertisement, the manager waits for the response bids to arrive from the agents in a window of time. When all bids have been made, it runs the calculation, which selects the best bid, which in our case corresponds to the bid with the smallest distance.

At the end of the counting, the manager announces the result of the auction. It sends a message to the agent that placed the best bid that its bid has won and assigns him the task, and it notifies the other agents of the fact that another agent has won the task.

The auction protocol of the solution implemented by us ends here, but in the CNP it is still stated that the agent who wins the auction responds to the manager. We did not implement this part, as we can more easily check the successful running of the auction with other means. In addition, we also see that in more complex systems this is a good tool, or a necessary part of the protocol.

Development Summary

Jason version

To develop the agents, we used the Jason library, currently the latest version 2.6.3.

ASL components

After properly configuring Jedit, we developed our two agents separately, and when they were close to completion, we only tested them in a common environment. All the logic required for the manager agent was written in AgentSpeak, since the task of this component is only to advertise and manage bids, for which AgentSpeak is perfectly suited. In the carrier agents, the communication with the manager, the calculation of the bid value, and the follow-up of the tasks won, some of the more important units of the process of carrying, were implemented at the ASL level. The latter means that the elementary steps of transporting a patient, such as picking up, putting down the patient, moving to the next field, are of course done with function calls on the model, but the sequence of these calls is defined in ASL, and whether the conditions required for the given call have been fulfilled conditions.

Java environment

The java environment follows the usual MVC (model - view -controller) design pattern, relying on the GridWorldModel, GridWorldView and Environment classes provided by the Jason library.
The model component of the program, the HospitalModel class, initializes the space of the problem. Creates different patient care wards and places them in a randomized location on the map. The data structure under the HospitalElements (= departments + reception) is a FIFO, in which the admitted patients are stored. Each HospitalElement has a maximum capacity, so many patients can be stored there at the same time. Patients can be parameterized with an age and a disease indicator.
20 patients are initially generated at the reception with limited randomized parameters (meaning: realistically), and later new patients can be added through user interactions. Patients can be picked up/dropped off at individual locations. After the placement of a patient, a counter starts to simulate the healing process. The value of the counter depends on age, its magnitude is 10-100 seconds. After recovery, patients leave the ward on their own two feet. (There is no bidding for them.)
Then there are 4 arbitrarily placed walls on the map, which demonstrate the operation of the routing algorithm. Model class places the reception.

The HospitalEnvironment department identifies and places patients at the reception desk. Notifies the Manager class (by updating beliefs) of new patients waiting to be announced. The reception also implements a waiting line, so that if a patient has left and they are still in the queue, it notifies the Manager agent about the next available patient. The environment is also responsible for calculating the shortest route between 2 points, which is done with the BFS algorithm. The environment also plays a role in the movement of the agents, namely at such a level that the agent is aware of its goal, but the corresponding route is calculated in the environment and assigned to the agent. The actions performed by the agent ("I arrived", "I have to go here", "I picked up the patient", "I put the patient down") are registered by the environment and their effect is validated on the model.

HospitalView implements the display of the user interface, provides real-time information about the status of agents / patients / departments, and manages user interactions.

Program description

Manager BDI description

The manager agent has the following rule:

- **all_proposals_recieved(CNPId)** With this rule, we can check that all agents who introduced themselves at the beginning of the run either bid for the given CNPId auction or rejected it.

Its goals and the associated implementation plans are as follows:

- **+newPatient(CNPId, PatientId, LocTo)** A plan that runs when a new patient arrives. A new patient is activated by adding the environment,
- **@r1 +propose(CNPId, Offer): auction_state(CNPId,propose) & all_proposals_received(CNPId)** If a bid has been received, and it has already been received from everyone else (so it was the last one), then we start counting the bids with this plan
- **@r2 +refuse(CNPId): auction_state(CNPId,propose) & all_proposals_received(CNPId)** Similar to the previous one, in this case the last agent refuses to bid
- **+!contract(CNPId): auction_state(CNPId, propose)** If we are in the "propose" state with the given auction, we run the calculation for selecting the best bid. After that, the results will be announced.
- **+!contract(CNPId)** In this case, we are not in the "propose" state, something probably went wrong.
- **-!contract(CNPId)** Plan for error indication, the auction process was executed incorrectly
- **+!announce_result(CNPId,[offer(O,WAg)|T], WAg)** We will accept the winning bidder's bid and notify them that they have won.
- **+!announce_result(CNPId,[offer(O,LAg)|T], WAg)** Non-winning bidders are informed that they did not win.

- **+!announce_result(_,[],_)** If there are no more participants, the announcement of results will stop

Carrier BDI description

A carrier agent has the following initial knowledge:

- Your own position (**pos(carrier1,5,6)**)
- The distance of the patient care departments from the reception (**pos("Cardiology",12)**)
- Location of patient care departments (**pos("Cardiology",x,y)**)
- How much can you bid initially (how far is it from the reception,**bid(5)**)
- How many bids have you won (**queue(0)**)
- Is there currently one at a given location L (**at(L)**)

Its goals and the associated implementation plans are as follows:

- **+plays(initiator,In)** He announces himself to the manager as a bidder
- **@c1 +bidPatient(CNPId,PatientId, LocTo)[source(A)] : plays(initiator,A) & price(CNPId,Offer)** Bid on an advertised task: your bid is always the current one**bid(x)** belief value. This value increases with the bid value of each task won and decreases with each step. For example, initially this is the distance between the agent and the reception, but if the agent wins a transfer to cardiology, he adds twice the distance from cardiology to the reception, since he can admit a new patient at the reception in that many steps. By definition, the value of the bid decreases by one with each step.
- **@r1 +accept_proposal(CNPId): proposal(CNPId,PatientId,LocTo,Offer)** If he won, he accepts and modifies the values of the queue and bid beliefs, or notes one**!handle(PatientId, LocTo)** goal.
- **+at(L)** This goal means that you have to go to a specific point, so in the implementation you take a step towards the point, as well as reduce the value of the bid belief.
- **+!at(L): at(L)** If it has arrived at the specified destination, it notifies the environment. If the given place is the reception, it picks up a patient, if it is not, it drops it and reduces the value of the queue belief.

- **+!handle(PatientId, LocTo)** In the framework of this plan, the transport of a patient is carried out: he notes that he has started a task so that when new bids are won, he does not start them, he goes to the reception, where he picks up the patient, and then goes to the LocTo patient care department, where he is dropped off.
- **!updateBid(L)** As a correction for shifts in the smaller bid belief value, when you are not taking anyone and there are no patients waiting for you, you set the bid to the current distance from the reception.

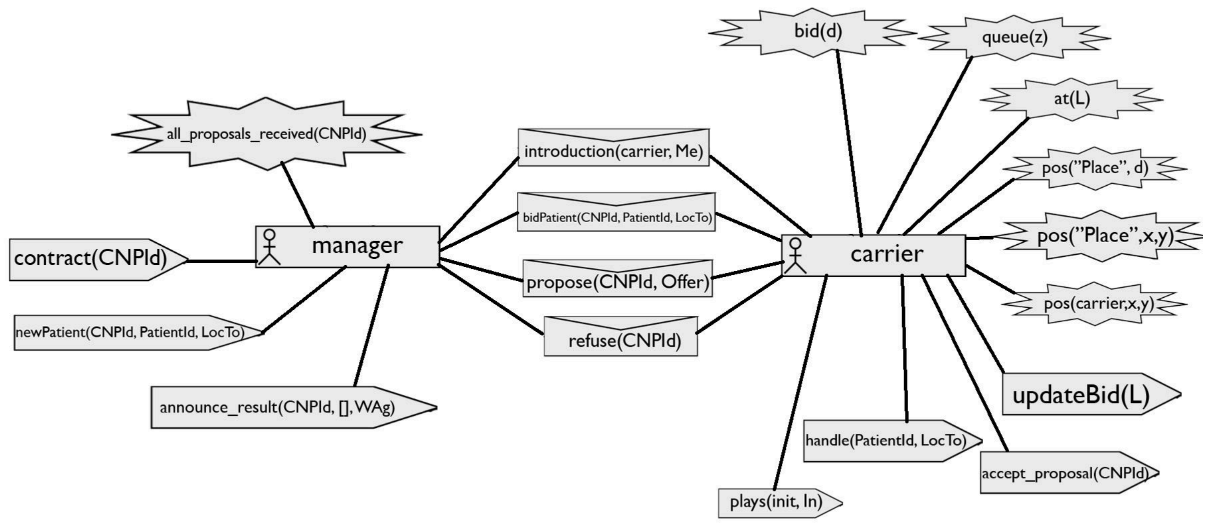

Figure 2: Relationship between agents

The program's user interface

The user interface uses the Java Swing framework. The main screen is realized by a layout with 3 components.

In the view on the left, the user can keep an eye on the properties representing the current state of the hospital in real time. These include: what patient care departments are present in the hospital, the current capacity of each department, and what agents work in the hospital and their condition. The user can testify about the organization of the agents' tasks and their detailed implementation on the LOG panel provided by Jason.

The bottom view is used to trigger user interactions, as the user has the opportunity to add a new patient to the problem space, with the disease of his choice, so he can visually validate that the system really strives to maintain the carrying optimum and keeps it. Additional parameters (age) of the patient added in this way are randomly generated.

In the view on the right, the plan of the hospital is displayed on a grid of 24x24 units. The patient care departments are displayed as yellow squares, with the abbreviation representing the department also appearing above them. The carrying agents appear as a blue circle, on which the ID assigned to the agent appears as a white number, and if the agent is currently carrying someone, a red circle with the patient's ID appears in the lower right corner. The 4x1 walls on the map appear as a multitude of black blocks, which the agents avoid. The reception is displayed as a red block.

One of the great strengths of the interface is that the elements that make up the hospital are always generated in a new, randomly chosen location, while paying close attention to complying with the basic physical constraints. In this way, the problem can always be represented from a new perspective, emphasizing the versatility of the task and the answers to its pivotal questions.

Actions performed by agents are 0.1 seconds long.

Summary of multi-agent system

As a result of the properties described at length earlier, the system can operate with any number of carrier agents (currently it is initialized with 5 + 1 manager agent), and the communication, autonomous operation and problem solving between them really enables the actual optimal execution of hospital patient transport. .

Video on how it works

- https://drive.google.com/file/d/1tAxlV3DOepaDVsP7LoqTHfxzUg2nqZz-/view?usp=sharing