A GOAL Multi-Agent Elevator Manager

Koen V. Hindriks, Wouter Pasman

September 18, 2013

1 Introduction

The objective of this assignment is to develop an intelligent multi-elevator controller. The goal is to optimize the performance of the elevators by minimizing the total time needed to move all people to the floors they want to go. Each elevator is controlled by an agent and a manager agent is supposed to manage each of the individual agents. The main goal of this multi-agent system (MAS) thus is to bring everyone as quickly as possible to his or her target floor.

To complete this assignment, you will need to modify a GOAL multi-agent system in an elevator simulator (see Figure 1), and hand in a working `elevator.mas2g` file with associated `.goal` files that meets the requirements discussed below.

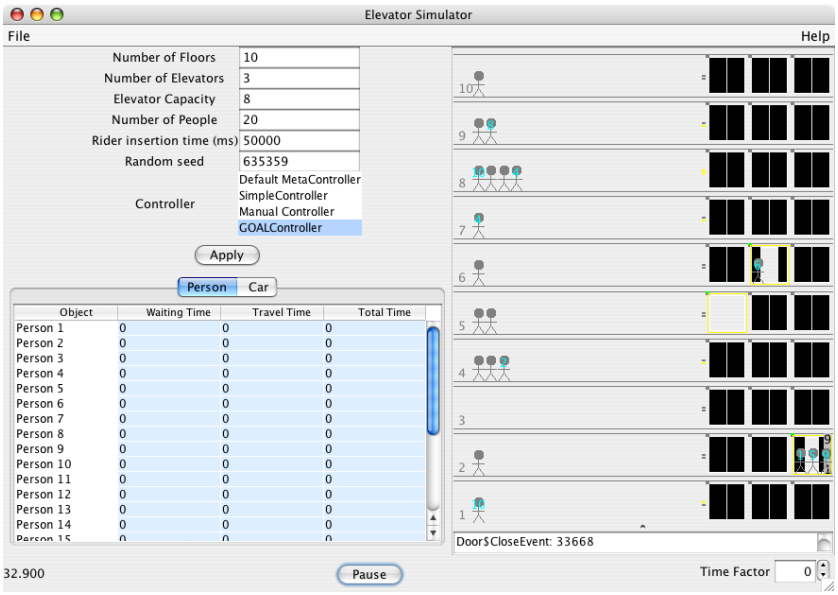

Figure 1: Elevator Simulator

The following materials on GOAL and the elevator simulator are available to you that you will need to complete the assignment:

- The GOAL *Programming Guide [1]*,
- the GOAL *IDE User Manual [2]*, and
- the *Elevator Environment Documentation [3]*.

In order to be able to analyse the performance of your MAS, you can use the built-in statistics functionality of the elevator simulator that computes the waiting time, travel time and total time

for each person and car separately, and the total, minimum, maximum and average times as well (see Figure 2). These statistics are calculated when the simulator is paused.

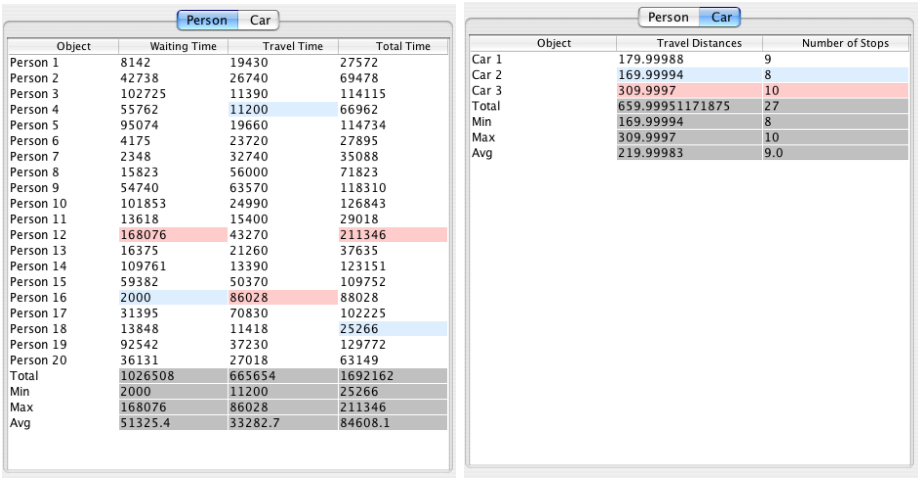

Figure 2: Statistics of a simulation session

2 Detailed Assignment Description

The assignment should be completed either individually or in pairs. In the sections below the objectives, deliverables, requirements, submitting of the solution, and the detailed assignment description are presented.

2.1 Objectives

- To learn to design and program an agent that uses logic to represent knowledge about its environment by building agents for an elevator simulator.
- To learn to represent and reason about a domain by means of first-order logic.

2.2 Deliverables

- A programmed solution provided in the form of a GOAL program: A MAS file `<name>.mas2g` and a set of GOAL files that are part of the MAS.
- A report documenting your solution, including an explanation and discussion of:
  - The main ideas used and implemented in the code.
  - Explanations and answers to questions listed in the Assignment section below.
  - An explanation and motivation of the choices made in the design of the logical rules and use of GOAL features.

The report need not be lengthy (nor more than 10 A4 pages).

2.3 Requirements

- You are asked to modify the multi-agent system specified in the `elevator.mas2g` file that is distributed with GOAL. The MAS should run in the GOAL interpreter and the provided elevator simulator using the GOAL controller (an option in the simulator).

- *The* GOAL *files should contain explanatory comments which provide enough information to make the file understandable to third parties not involved in programming the MAS.* This means that you should use comments (`%put your comment here`) to explain your code where needed.
- All agents should be implemented by programming a GOAL agent (see also [1]) to control the agent's choice of actions. The agent should be goal-directed and try to minimize the average total travelling time of the people in the simulator.
- The agent should be able to handle 3 to 6 elevators, 20 to 200 persons, 10 to 23 floors and a capacity of 4 to 20 persons per elevator. We are assuming the "Random Rider Insertion" behaviour of people in the simulator (an option in the simulator).
- The programs should contain rules to handle all percepts and should facilitate the derivation of relevant environment information from these percepts (see [3]).
- The report should include:
  - an introduction to the assignment,
  - a high-level description of the MAS design, including the communication protocols,
  - an explanation of the main rules that the agents use to decide what to do next; the report should help the reader to understand the organization of the source code but details should be commented on in the source code itself,
  - answers to the questions posed in the assignment, and
  - argued conclusions about the usefulness of logic for implementing the agents, in particular as a tool to represent and reason about the elevator world and to implement the agents' strategies for acting in the elevator simulator.

2.4 Submitting Assignment Solution

Submit your report in PDF format and the source code (`.mas2g` and `.goal` files) in a `<name>.zip` file by email to `ai@mmi.tudelft.nl`. Do not submit incomplete assignment solutions; only a complete assignment solution containing all deliverables will be accepted. *The deadline for submitting the assignment is Monday November 11, 23:59.*

2.5 Assignment

To complete the three parts of this assignment you will need to understand and modify the GOAL multi-agent system for the elevator simulator that is distributed with GOAL. By answering the initial questions below you should gain an understanding of what the agents in the provided elevator environment do. You will be requested to edit and modify these rules to improve the behavior of the agents. The updated GOAL and MAS files should be returned as part of the assignment solution.

2.5.1 Assignment part A

1. In order to try and test your GOAL code, if you did not do so already, please download the GOAL programming language installer from `http://mmi.tudelft.nl/trac/goal`.
2. To get you started the GOAL distribution already contains a simple MAS for controlling the elevator. This MAS contains a GOAL agent for controlling an elevator. You can load the example file by using the file menu in the GOAL IDE.

3. Load `elevator.mas2g` (see Figure 3) and run it. Use seed 6353 instead of the default seed (you can set the seed with the `Random seed` field in the Elevator simulator, see Figure 1). The other values should be kept at default (Floors=10, Cars=3, Capacity=8, People=20, InsertTime=50000). After the environment is set up, you can start (un-pause) the agents by running the MAS. All elevators will simultaneously start to move to floor 9 after some time. Explain why all elevators move to floor 9, and explain why they do not stop to pick up other people along the way.

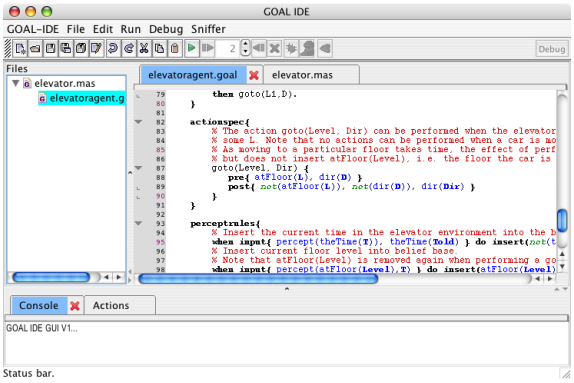

Figure 3: The IDE editor window showing `elevatoragent.goal` being edited.

4. Explain why the elevator that brings the person from the 9th to the 1st floor indicates that it will go down when it reaches the ground floor (which is not possible anyway), i.e. explain why the agent performs the action `goto(1,down)`.

2.5.2 Assignment part B

Add a new GOAL agent that is the manager deciding which elevator will handle which requests. Modify the existing code to make the elevator agents follow the instructions of the manager instead of deciding for themselves which floor to visit. The design of your agent needs to conform with the requirements listed below. Please make sure you read the complete assignment first before you start working on it.

1. Make sure you have loaded the original `elevator.mas2g` into the GOAL IDE. Add a new `.goal` file called `cnpmgr.goal` (using File menu). Change the `.mas` file accordingly, by adding a line for an agent called `manager` that uses the `cnpmgr.goal` file but is not connected to the environment.

2. The manager agent is not connected to the environment and therefore it does not receive `fButtonOn` percepts. To inform the manager, add code to the elevator agents that will ensure that the manager agent is informed of floor requests. Use the `send` action to communicate the relevant information and make sure that the manager agent has the up-to-date floor requests in its beliefbase.

3. Write code to make an elevator agent inform the manager agent when the elevator arrives at a floor. You may also consider adding code to inform the manager when the elevator leaves a floor. The aim of this code is to ensure that the manager agent can make a reasonable decision of which agent should be awarded which task.

4. Design elevator scheduling code. This scheduler plans elevator movements such that the floors are serviced optimally. This design is critical for optimal elevator behaviour and should be carefully designed and discussed. Discuss in what sense your elevators will be running optimally.

5. Implement the scheduling code. If you wish you can use the Contract Net protocol (see below) for the implementation, but this is not mandatory. You need to document in your report how interaction between the agents has been designed.

6. Modify the `elevator.goal` file such that an elevator agent chooses to perform `goto(X,Y)` actions only if it has been contracted by the manager to serve a particular floor.

7. Make sure that an elevator agent informs the manager agent that it completed a contracted task.

8. Test your implementation and discuss the performance.

2.5.3 Assignment part C

Please provide two concrete feedback comments on the GOAL system, either things you would like to have been available but are not (yet) provided by the current system, suggestions to improve the system, or other comments that may help us to improve GOAL. *Note: the points associated with this exercise are assigned if you provide at least 2 concrete comments regardless of the specific content of your comments.*

3 Contract Net Protocol

Contract Net is a communication protocol that was proposed by R. G. Smith [4]. We discuss the protocol briefly here. You can use this protocol or modify it to your needs, but this is not a requirement for this assignment.

The basic metaphor used in this protocol is contracting. A node that has a task announces existence of that task to other nodes in the net. Other nodes then can then submit a bid on that task. The bid contains the capabilities of the bidder that are relevant to the execution of the task. A manager receives the bids, selects the best one and gives that bidder an award message. After the task is completed, the contractor sends a report to the manager.

The four basic steps in the protocol are (see [4]):

1. Task announcement processing. On receipt of a task announcement, an agent decides if it is *eligible* for the task. It does this by looking at the *eligibility specification* contained in the announcement. If it is eligible, then details of the task are stored, and the agent will subsequently bid for the task.

2. Bid processing. Details of the bid from would-be contractors are stored by (would-be) managers until some deadline is reached. The manager then awards the task to a single bidder.

3. Award processing. Agents that bid for a task, but fail to be awarded it, simply delete details of the task. The succesful bidder must attempt to expedite the task (which may mean generating new sub-tasks).

4. Request and inform processing. These messasges are the simplest to handle. A request simply causes an inform message to be sent to the requestor, containing the required information, but only if that information is immediately available. (Otherwise, the requestee informs the requestor that the information is unknown.) An inform message causes its content to be added to the recipient's database. It is assumed that at the conclusion of the task, a contractor will send an information message to the manager, ddetailing the result of the expedited task.

This protocol can be simplified where appropriate, so you may opt to leave out parts of the protocol for efficiency or convenience.

4 Evaluation

Assignments are evaluated based on several criteria. All assignments need to be complete and satisfy the requirements stated in the detailed assignment description section above. *Incomplete assignments are not evaluated.* That is, if any of the deliverables is incomplete or missing (or your `.mas2g` file fails to run in the elevator simulator), then your assignment will not be evaluated.

The assignment will be evaluated using the following evaluation criteria:

- *Performance of your Agent*: Your agent will be run with a number of different test setups, picked randomly within the bounds given in Section 2.3. *All tests will be done with time factor 0.* The average total traveling time of the persons is measured and averaged over the runs.

  This is done for all groups. The group with the lowest total average traveling time scores a 10. The sample agent provided with this exercise scores a 6. Your score is linearly interpolated/extrapolated using these two scores. The minimum score for a round is 1.

  If your solution did not deliver people to their target floor at the end of the run, your solution will receive a score of 1 for that round (corresponding to an infinite average delivery time). We are aware that in some cases people do not press the floor button, and these cases will be ignored.

  The actual test situations (the parameters) will be published only after the submission deadline.

- *Quality of the Deliverables*: Overall explanation and motivation of the design of your logical agent; Quality and completeness of answers and explanation provided to the questions posed in the assignment; Explanatory comments in source code, quality of documentation.

- *Originality*: Any additional, original features of your agent or solutions to questions posed are evaluated positively.

Your final grade is the 50%-50% average of your agent performance rating and the judged quality of your deliverables and originality.

Note that you are required to submit original source code designed and written by yourselves. Source code that is arguably the same to that of, e.g., other students will not be evaluated and be judged as insufficient. Detection of fraud will be reported to the administration.

References

[1] K. V. Hindriks. The GOAL programming guide, 2010. Available at `www.mmi.tudelft.nl/trac/goal`.

[2] K.V. Hindriks and W. Pasman. GOAL User Manual, 2010. Available at `www.mmi.tudelft.nl/trac/goal`.

[3] W. Pasman and K.V. Hindriks. Elevator environment, 2009. Comes with the GOAL installer at `www.mmi.tudelft.nl/trac/goal`.

[4] R. G. Smith. The contract net protocol: High-level control communication and control in a distribution problem solver0. *IEEE Transactions On Computers*, 1980. Available on Blackboard.