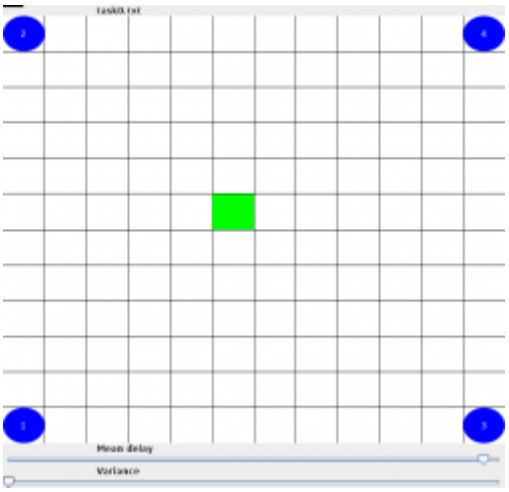
Assignment #1 – Multiagent system

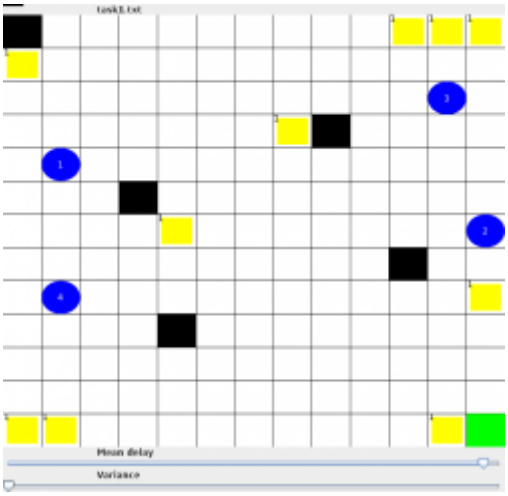
Assignment

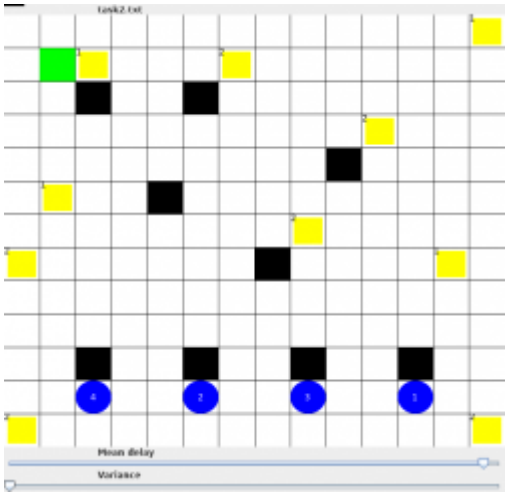
A group of four miners is facing an uneasy task. It is deployed to a gold mine and the miners have to collect all gold stones scattered around the mine and bring them all to one of the company depots. The company is saving money wherever it can and the stones they have to carry are becoming more and more heavy. At this point, these stones have become that heavy that no single one of them can lift them alone. Every time a miner wants to lift a stone, he must call some friend to help him. Will you help them to collect all the gold stones in time?

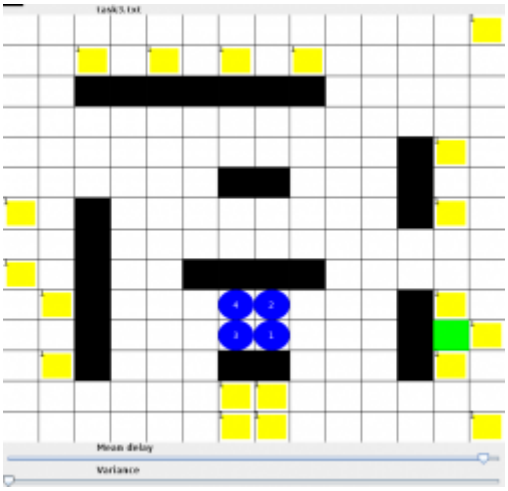
Scenario 1: On day one, they were quite lucky. They have arrived in a tidy gold mine and they may go wherever they want. The only problem here is that the work is still in progress and new gold stones appear from time to time. Whenever a gold stone appears somewhere in the mine, miners must be able to find it and bring it to a depot. If the miners do succeed, they will be awarded **2 points**. (see `task0.txt`)

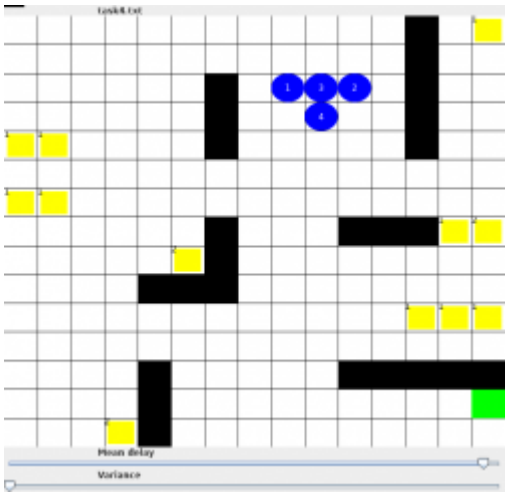


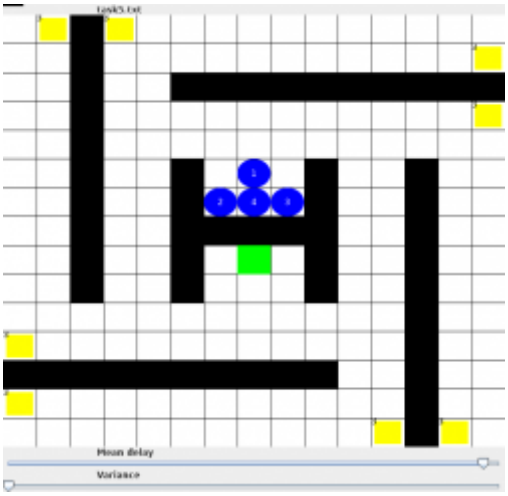
Scenarios 2–8: The conditions in the mine get more and more challenging on the subsequent days. Debris is scattered around the mine and the miners are still asked to perform what they are paid for – collecting the gold! Make your agent navigate through these mines obstructed by heavy machinery and other obstacles and make them succeed! Miners get **1 point** for every scenario they complete successfully. (see `task1.txt`-`task7.txt` – we will evaluate the performance of your mining team on slightly modified versions of these scenarios)

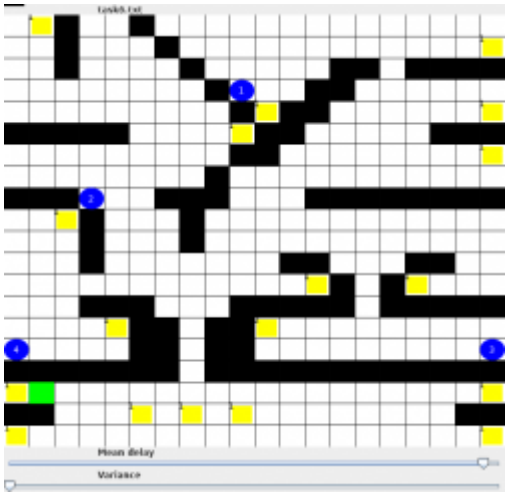

Mean delay
Variance

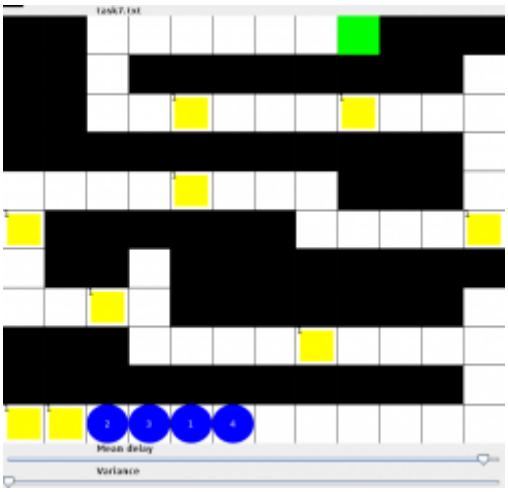

Mean delay
Variance


Mean delay
Variance


Mean delay
Variance







Competition: The group of the miners was proven to be highly competent and thus the managers have decided to send them to a mining competition. 25% fastest groups of miners will be awarded another **1 point**.

Mining research The conditions in the mining industry are getting worse every day. If you think your group of miners can overcome even greater difficulties, they may be awarded some **extra points** (after prior discussion with the tutor).

Rules:

- Miners see objects only in the 8-neighborhood of their position (i.e. you have to find the objects on the map).
- The pick action fails if there is no colleague in the 4-neighbourhood of the miner's position.
- A miner can carry at most one goldstone at a time.
- Position of depots are not known in advance and must be discovered.
- The percepts are issued to the agent only after each external action – consider using `sense()` if you are waiting for a certain percept.
- `sense()` action takes nearly no time regardless the action duration setting
- Miners can communicate information only via messages.

Download and implementation

Java project (applicable also for implementations in other languages!)

You should implement your solution inside `student.Agent` class (and possibly other classes of your need within the `student` package). You can test your solution by launching `mas.agents.SimulationCore [map file]` where `[map file]` is the name of scenario you want to test. If you want to be sure that your agents do not share any knowledge, you can use `mas.agents.SimulationCore [map file] ~` (this setting may complicate debugging).

If you are willing to implement your solution in other languages (e.g., Python, C/C++), please refer to the section below. Also, if you want to implement your solution in languages other than Python and C/C++, let us know about that beforehand (we need to verify that we can easily test your solution first).

Assessment

- **DEADLINE:** ~~6.11.2017 4:00~~ **7.11.2017 4:00**
- Maximum number of points for this assignment: **11 pts**
  - **max 2 pts** if the team succeeds in collecting all gold stones in Scenario 1 (i.e. dynamically changing mine with no obstacles)
  - **max 7 pts** if the team succeeds in collecting all gold stones in Scenarios 2-8 within the time limit
  - **1 point** if you place amongst 25% of fastest mining teams (in terms of execution time, 50ms per step) on Scenarios 3-5 (time will be taken as the average from runtimes on each of the scenarios).
  - **1 point** for a short report describing problems you have encountered and the ways you used to overcome them, e.g.
    - How have you found the gold stones and depots?
    - How have you solved synchronization problems?
- Except for the Scenario 1, teams should not be much slower than our reference solution (using 50ms duration of each action). Do not submit agents relying just on their luck, please
  - If your team fails to collect gold stones in time, but your team does reasonable things (i.e. does not wander randomly around the mine) you might still be awarded all the points. Do not forget to mention it in the report!
- For each scenario, several runs on varying mines will be executed. The points you will get will be proportional to the number of successfully solved instances.
- Your team must be able to work reliably with any setting of delay after execution of each step

Submitting your solution

Create a zip archive containing the content of the `student` package and your report in PDF format and submit it to the upload system [https://cw.felk.cvut.cz/upload/]. If you do not have access to the upload system, please send your files to karel.horak@agents.fel.cvut.cz.

Implementation in other language

WARNING: This is an experimental feature. You will be left in an uncharted territory. Use at your own risk! We will try to fix problems that may arise - but you are expected to be confident in the programming language of your choice.

To implement your agent, you are expected to write a program that accepts a single command-line argument (the ID of the agent) and communicates through standard input/output. Use standard output to send messages to other agents/environment, use standard input to receive messages from other agents.

Format of a message

`[recipient] [sender] [messageID] [queryData] : [content]`

- `[recipient]` is the ID of the agent who should receive your message (0 stands for the environment)
- `[sender]` is the ID of the agent sending the message
- `[messageID]` is the identifier of the message used in connection with `[queryData]` (see below)
- `[queryData]` indicates whether the message is a query (request for data from other agent), reply to a query or none of those. Possible options are:
  - `Q` if you expect an answer to your message
  - `R[messageID]` if the message is an answer to a message with ID `[messageID]`
  - `-` if you send a regular message
- `[content]` is the payload of the message (do not start it with `!`!)

Communication with the environment

To communicate with the environment and perform actions, send a message to agent 0. You will receive an answer containing your current percepts. Valid actions are: `!left`, `!right`, `!up`, `!down`, `!pick`, `!drop`, `!sense`.

Example message: `0 1 42 Q : !left` (agent 1 wants to move left)

The environment reply with a message: `1 0 839 R42 : !status [agentX] [agentY] [width]x[height] [percepts]`

- `[agentX]` and `[agentY]` denote agent's current position
- `[width]` and `[height]` indicate dimensions of the map
- `[percepts]` is a space delimited list of objects the agent sees at the moment. Each percept is a string `[type][x],[y]` where `[type]` is the type of object (`O` – obstacle, `D` – depot, `G` – gold, `A` – another agent) and `[x]` and `[y]` indicate the position of the object.

Example message: `1 0 839 R42 : !status 10 10 20×20 A9,10 O11,10 O10,11`

Initialization

Send `!ready` action to the environment (similarly to actions above). Start execution upon receiving `[agentID] 0 [messageID] - : !start`.

Using your agent within the environment

To evaluate your agent, run the class `SimulationCore` with one extra parameter: `mas.agents.SimulationCore [mapFile] [pathToAgentExecutable]`

Submission

Currently we plan to support Python and C / Cpp. To submit your solution, submit an archive containing `main.py` / `main.c` / `main.cpp` along with your PDF report.

Results of Competition

	High penalty for failure		Low penalty for failure	
	Login	**Avg. place**	**Login**	**Avg. place**
1	smolidan	3.4	smolidan	3.4
2	cejkama3	7.7	cejkama3	3.6
3	hilarjos	8.0	hilarjos	6.0
4	trollmil	12.2	trollmil	8.0
5	mishcnik	13.6	milecdav	8.1
6	hlavam28	13.7	hlavam28	8.2
7	milecdav	14.5	mishcnik	8.9
8	krynsdan	15.4	strasfra	10.1
9	spanemar	15.6	spanemar	10.2
10	strasfra	16.0	kozakj11	10.8
11	ekimoiva	17.3	ekimoiva	11.6
12	kozakj11	17.4	krynsdan	11.9
13	lukasond	18.3	srsenste	12.2
14	najmami2	19.3	vobecant	13.2
15	pavlaad2	19.3	lukasond	13.4
16	kiselmak	19.4	kiselmak	14.6
17	srsenste	20.5	petrmat1	15.5
18	kuchapav	21.1	doudaond	15.5
19	vobecant	22.3	zalouja4	15.8
20	louredio	22.5	pavlaad2	15.9
21	ondrala3	23.2	kuchapav	15.9
22	zvaramar	25.8	najmami2	16.7
23	zalouja4	25.9	zvaramar	16.8
24	rindtedu	28.0	petrija9	17.3
25	doudaond	28.1	rindtedu	17.4
26	petrmat1	28.5	silhapro	17.6
27	petrija9	28.9	louredio	17.8
28	silhapro	29.1	viazomyk	18.0
29	urbanm30	30.8	urbanm30	18.3
30	viazomyk	32.0	kriztom3	19.3
31	kriztom3	32.4	janisjar	19.4
32	janisjar	33.4	ondrala3	19.7
33	skoumond	34.8	niklmate	19.9
34	kalatoma	35.1	kalatoma	19.9
35	niklmate	35.6	strammar	20.1
36	strammar	35.8	vsetepet	20.3
37	vsetepet	36.0	rehacden	20.3
38	rehacden	36.0	hromalu1	20.4
39	hromalu1	36.0	skoumond	20.7
40	ciricste	37.0	ciricste	21.0
41	fiserto1	37.0	fiserto1	21.0
42	gabrimi5	37.0	gabrimi5	21.0
43	galinwil	37.0	galinwil	21.0
44	knakama1	37.0	knakama1	21.0
45	langrfil	37.0	langrfil	21.0
46	machvojt	37.0	machvojt	21.0
47	ondralad	37.0	ondralad	21.0
48	pavlisim	37.0	pavlisim	21.0
49	pograjak	37.0	pograjak	21.0

50	rozumden	37.0		rozumden	21.0
51	sedlazb1	37.0		sedlazb1	21.0

courses/be4m36mas/assignment1-miners.txt · Last modified: 2017/12/19 10:17 by horakka5