448.057 Context-Aware Computing (UE)
Exercise 4 – Coverage Problem in WSNs

Task description

The task for the fourth assignment of this course consists in modelling a wireless sensor network solving the coverage problem using NetLogo. Wireless sensor networks consist of several nodes equipped with micro-controller, memory, radio transceiver, and different sensors (e.g., temperature, humidity, and radiated sunlight) used to perceive the surrounding environment. Each wireless sensor node is typically battery-powered, and its resources are highly-constrained due to the limited size and cost. Therefore, it is important to minimize the energy consumption as much as possible.

The coverage problem consists in minimizing the number of wireless sensor nodes that actively measure the environment using their sensors, such that every point in a given area is observed by at least one node. Nodes that do not actively measure the environment lie dormant (i.e., remain inactive) and can replace nodes whose battery is exhausted at a later point in time. In practice, the coverage problem consists in assigning the roles `ON` or `OFF` to each wireless sensor node in such a way that (i) the environment is (almost) fully covered and (ii) the amount of nodes with role `ON` is minimal over time. When a node that is `ON` runs out of battery, some areas may become uncovered, hence other nodes will need to be switched `ON` to cover such areas.

The goal of this exercise is to implement in NetLogo a self-organizing coverage algorithm for wireless sensor networks similar to the one presented in [1, 2]. In the NetLogo model to be implemented, wireless sensor nodes are randomly dispersed in a black field modelled as a 60x60 torus, and have a circular shape with size 1. The number of nodes is controllable by the user through the input variable `total-nodes` (range $[1 - 350]$). Every node has a communication radius specified by the user-defined variable `antenna-range` (range $[1 - 20]$) and forms a bidirectional link with any neighbouring node lying within this communication range. Nodes are assumed to be static (i.e., their position remains fixed over time), and can only use these bidirectional links to communicate with their neighbouring nodes. Furthermore, every sensor node has a sensing radius that is specified by the user-defined `sensing-range` variable (range $[1 - 10]$). For simplicity, it is assumed that wireless sensor nodes have both a sensing and communication range that is circular, i.e., a sensor can observe any point (patch) in the environment or communicate to any neighbour (turtle) within a given radius.

Every node is battery-powered. Although every alkaline battery has a nominal capacity of 100 units, manufacturers warn that, due to prolonged shelf-storage life, a battery can only offer between 85 and 100% of its nominal capacity. Hence, every node has a different initial battery capacity specified by the `battery-level` variable randomly distributed between 85 and 100 units. Each node also has a current role: `ON` or `OFF`. Nodes that are `ON` should be displayed in green color, whereas `OFF` nodes should be displayed in gray. For nodes with role `ON`, the `battery-level` should be decreased by 1 unit every tick. Nodes having an `OFF` role are assumed to have a negligible (zero) power consumption. A node that reaches a `battery-level` equal to zero should be displayed in red and excluded from the network.

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

The self-organizing coverage algorithm should follow these specifications:

1. A node should be assigned role `ON` if and only if the following holds:
   - `battery-level` > 0;
   - The node has no neighbouring node with role `ON` within a distance ($\alpha \cdot$ `sensing-range`), where α is a user-specified variable (range $[0-2]$ with 0.01 increments). Please note that a node is only neighbour if antenna-range is greater or equal than ($\alpha \cdot$`sensing-range`).
2. Otherwise, the node is assigned role `OFF`.

Over time, each node queries the nodes in its neighbourhood to check their current role, and then decides which role to take according to the above specifications. The self-organizing algorithm should adapt to changes in the structure of the network (e.g., nodes powered off because of depleted batteries) and therefore the NetLogo simulation should show the evolution of the wireless sensor networks over time. Towards this goal, the simulation should contain two buttons: `setup` and `go`, associated to the `setup` and `go` procedures, respectively. The `setup` procedure creates the field (`setup-ground` sub-procedure) and generates an initial amount of nodes dispersed randomly (`setup-nodes` sub-procedure). The `setup` procedure should further initialize the variables of each wireless sensor node (`setup-variables` sub-procedure) and link nodes within communication radius as neighbours (`connect-nodes` sub-procedure). The skeleton of the `setup` procedure is shown in Listing 4.

Listing 1: Skeleton for the `setup` procedure.

```
to setup
        clear-all
        reset-ticks
        setup-ground
        setup-nodes
        setup-variables
        connect-nodes
end
```

The `go` procedure consists of three sub-procedures: `assign-role`, `draw-coverage`, as well as `update-battery-level`. These procedures are repeated over time. In the `assign-role method`, each node select its role: this has to be done every tick because of changing battery levels and role values from the neighbourhood. The coverage area of all nodes with role `ON` should be drawn with a gray shade in the `draw-coverage` procedure. The `update-battery-level` method computes and updates the residual battery of each node. The skeleton of the `go` procedure is shown in Listing 2. To break the symmetry, a node should start the `assign-role` operation only after a random amount of initial rounds of the `go` procedure have been completed (i.e., each node should wait a different number of rounds).

Listing 2: Skeleton for the `go` procedure.

```
to go
        assign-role
        draw-coverage
        update-battery-level
```


Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

```
        tick
end
```

In order to have a clearer view of the ongoing activities in the wireless sensor network, three plots should be created. The first should display the percentage of covered space; the second one should display the percentage of `OFF` nodes. The ultimate goal of the algorithm is to maximize the covered area and the number of `OFF` nodes. Hence, the third plot should display the sum of the percentage of `OFF` nodes and the percentage of covered area.

Basic functionality ***(3 points).*** Create a NetLogo model that follows the specifications above and make sure that the program does not enter an infinite loop. The submitted code should be properly commented and each team member should be able to explain it.

System behaviour ***(3 points).*** Discuss the high-level behaviour of the system by answering the following questions (please respect the space constraints):

1. What happens if all nodes update their role simultaneously and all nodes initially have role `OFF`?

   Answer:

2. What is the impact of the value of the variable α given the following configuration:

   - `total-nodes` = 200;
   - `antenna-range` = 10;
   - `sensing-range` = 5.

   Answer:

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

3. (c) Which value of α (approximately) maximizes the value of the third plot? Why?

Answer:

Flooding *(4 points)*. Wireless communication technologies such as ZigBee and Bluetooth Low Energy use flooding algorithm to deliver messages between nodes in networks [3]. For this task, we ask you to implement and investigate the performance of such communication schemes. For example, in BLE only designated nodes which have a power supply are re-broadcasting messages, these nodes are called relay nodes. In this task, you should use the role selection algorithm from the previous tasks to select the relay nodes (i.e., "ON" means that it is a relay node) and simulate the message passing through the network. All nodes that are "OFF" can still receive messages, but do not rebroadcast them. During each run of the simulation (run != tick), only a single node A initiates the message exchange, this should simplify the implementation. Moreover, for this task we ignore the battery value, so once the role assignment phase has stopped the nodes stick with their role. Before Node A sends its message, all nodes have to select their role by running the role assignment algorithm. The variable `start_msg_tick` specifies after how many iterations the role assignment algorithm stops, and node A sends its message.

The following functionality should be implemented.

- Use a switch to activate the flooding functionality, such that the previous task still works.
- A Node A should be able to broadcast a message that is intended for Node B. All relay nodes re-broadcasts the message. The simulation stops if the message reaches node B, or if no more messages are exchanged. Make sure that a node only rebroadcasts the message during the next tick.
- Set the colour of each node depending of the hop-count such that you can see how messages are spreading in the network.
- Count and plot the total number of messages received and sent by all nodes.
- Calculate the shortest path, i.e., the hop-count of the message when it is received at node B.
- Calculate the packet error rate, i.e., how often a message from node A does not reach node B. You should do this offline in Excel / Python as described later.

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

You should compare the number of messages received / send as well as the packet error rate and the hop-count for different values of the sensing range. A convenient way to accomplish this is by using the `BehaviorSpace-Tool` of the Netlogo environment. It can be found under `Tool`. Essentially, this allows you to run a Netlogo program numerous times with different parameters, and save the results into a `CSV` file.

To setup your experiments you have to click ”Tools”-¿”New” and enter the following details:

- *”Repetitions”*: `100`
- *”Measure runs at every step”*: False (Un-tick)

Listing 3: Vary variables:

```
["antenna_dB" 7] ; Set antenna_db to 7
["sensing_range" [0 0.3 10]] ; Sweep sensing_range
                             ; [ start value , increment , end value ]
                             ; For higher resolution set a lower
                             ; increment value
["total_nodes" 150] ; Set total number of nodes 150
["start_msg_tick" 15]
```

Listing 4: Measure runs using these reporters

```
msg_reached_goal ; Reporters as specified
number_of_msg_send ;
number_of_received ;
```

The reporter `msg_reached_goal` should return 1 if the message was received by node B and 0 if not. The reporter `number_of_msg_send` should return the total number of sent messages. The reporter `number_of_received` should return the total number of received messages. After you configured the experiment, you can run it by clicking `Run`. In the pop-up you have to tick `Table output` and un-tick all other options. Afterward, you are asked to specify the location of the `CSV` file. Apart from some headers, the CSV file contains the output for each of the reporters specified in `Measure runs using these reporters` for each run. To analyze the output you can use Excel, OpenOffice or Python. We provide an example for Python and Pandas.

1. How does the `sensing_range` influences the hop count, the total number of sent, and the total number of received messages?

   Answer:

2. What happens if the sensing_range is chosen too high? At what value for the sensing range do you expect the packet error rate to increase steeply? Why is that?

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

Answer:

3. What is the optimal value for the sensing range, i.e., low packet error rate and number of send messages?

   Answer:

Implementation Hints: After the role assignment you may delete all existing links and only establish unidirectional links between relay nodes and relay nodes, as well as relay nodes and non-relay nodes. This might help to see if networks are split into two or more groups.

Bonus task *(3 points)*. Lets assume that the message relaying is not very reliable, i.e., the reception of a message fails with a certain probability. Investigate how the packet error rate evolves for different failure probabilities. Do this for a very low `sensing_range` (0.7) and a *sensing_range* near to your previously selected optimal value.

1. How is the packet error rate influenced by the sensing range? Explain why.

   Answer:

References

[1] C. Frank and K. Römer, *Algorithms for generic role assignment in wireless sensor networks*. In Proceedings of the 3rd Conference on Embedded Networked Sensor Systems (SenSys). 2005. `http://www.vs.inf.ethz.ch/publ/papers/sensys05.roleassignment.pdf`

[2] K. Römer, C. Frank, P.J. Marrón, and C. Becker, *Generic role assignment for wireless sensor networks*. In Proceedings of the 11th ACM SIGOPS Workshop. 2004. `http://www.vs.inf.ethz.ch/publ/papers/sigops.roleassignment.pdf`

[3] Wikipedia Flooding `https://en.wikipedia.org/wiki/Flooding_(computer_networking)`

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>

Please submit the filled PDF and your NetLogo source code to the TeachCenter as illustrated in the lecture within **Sunday, 06.01.2019, 23:59 CET**.

Group ID:

Stud. Names:

Finalize Form

Michael Stocker, Robert Thomann <michael.stocker@tugraz.at, r.thomann@tugraz.at>