Algorithms 2500 – Section 1A
First Project
Deadlines: 10/24 First report, 10/31 Second report

Greedy algorithms for mission assignment in Wireless Sensor Networks

Maximum points 100

General information

Maximum group size 3.

Any programming language can be used for the implementation.

The project reports should be submitted in hardcopy to the Instructor the day of the deadline. Additionally, the reports and the code should be submitted to the TA electronically the day of the deadline.

The project should be discussed with the TA within 1 week from the deadline of the second report.

Wireless Sensor Networks

A Wireless Sensor Network (WSN) is composed by small devices endowed with sensing and communication capabilities. Modern devices can collect physical quantities such as temperature, pressure or light intensity, as well as images and videos.

Application of WSNs include hazard and inaccessible fields, where sensors cannot be deployed manually. As a result, sensors are generally thrown from an aircraft over the Area of Interest (AoI), generating a random deployment.

Sensing missions are usually assigned to the network. A mission could be generated as a consequence of an event of interest that arise in a zone of the network, and requires some sensors to be assigned to it to perform the sensing task. As an example, a forest fire may arise and we want our sensors to monitor the fire.

The goal of this project is to investigate the performance of greedy algorithms for mission assignment in WSNs.

Network and mission model

Sensors are randomly deployed in an Area of Interest (AoI). For each sensor s we keep track of its position \((x_s, y_s)\) and of its energy level \(e_s\). Initially, \(e_s = 1000\) for each sensor. The number of sensors deployed is \(N\).

Missions are submitted to the network and represent sensing tasks. A sensor can be assigned to only one mission at a time. A mission \(m\) has an epicenter \((x_m, y_m)\), a radius \(R_m\), a number of sensors required \(N_m\), and a start time \(S_m\) and an end time \(E_m\).
As soon as a mission $m$ arrives in the network (i.e. the time is equal to $S_m$), a mission assignment algorithm selects the $N_m$ sensors to serve the mission. Such sensors need to be at most at a distance $R_m$, from the mission epicenter.

When a sensor is assigned to a mission, it is considered busy and cannot be assigned to other missions. Additionally, a sensor consumes energy when assigned to a mission, and in particular it consumes 1 unit of energy per unit time. As a result, to complete a mission $m$ a sensor needs at least $E_m S_m$ energy units.

Summarizing, a sensor can be assigned to a mission $m$ only if (i) it is at most at a distance $R_m$ from the mission epicenter, (ii) it is not busy (i.e. not assigned to other missions), (iii) it has at least at least $T_m$ energy units.

A mission $m$ is considered satisfied if the algorithm is able to assign $N_m$ sensors which are at most at a distance $R_m$ from the epicenter, and that have sufficient energy. Missions that cannot be satisfied (unsatisfied missions) are discarded and no sensor is busy for a mission that cannot be satisfied.

A sensor assigned to a mission is again available after the mission is terminated. When a sensor consumes all its energy is considered dead and cannot be assigned to any mission.

We denote by $M = \{m_1, ..., m_k\}$ the set of missions. We assume that missions in $M$ are sorted by start time.

Algorithm design

The project requires to design two greedy algorithms, that will be compared with a random base line approach. The goal of these algorithms is to determine for each mission the set of sensors that will be assigned to that mission. If an algorithm cannot find a suitable set of sensors, the mission is unsatisfied.

**Online greedy:** This algorithm considers one mission at a time in the order of arrival, i.e. it has not knowledge of future incoming missions. The algorithm takes as input the sensor locations, their current energy level, their assignment (busy or available) and the mission requirements. As soon as a mission arrives, the algorithm selects the set of sensors to serve the mission. The greedy selection criteria used is at your discretion an should be discussed in the report.

**Offline greedy:** This algorithm considers the entire set of all mission $M$, i.e. it is aware of all incoming missions. The algorithm takes as input the sensor locations and initial energy, and it considers the requirements of all missions in $M$. The algorithm selects which mission should be satisfied, and from which sensors. The greedy selection criteria used is at your discretion an should be discussed in the report.

**Random:** This algorithm considers one mission at a time in the order of arrival. For each mission, the algorithm keeps selecting a sensor at random in the set of available sensors that can serve the mission, until enough sensors have been selected.

We are interested in comparing the performance of the algorithms in different operative settings as described below.
Implementation and evaluation

The AoI can be assumed to be a square of size 50 x 50m². Assign to each node a random position (x,y) by selecting a random value for both x and y in the interval (0,50).

Generate a set of K = 1000 missions. The epicenter of each mission should be selected at random in the AoI. The radius $R_n$ of each mission is set to 5m. The mission start and end time can be generated randomly, the interval in which these values are picked should be chosen to have meaningful results. The mission duration $T_m = E_m - S_m$, number of sensors required for a mission $N_m$, and the number of sensors deployed $N$, may vary as explained in the following.

Increasing the number of sensors

This set of experiments wants to study the performance of the algorithms increasing the number $N$ of available sensors. We assume the duration of each mission is $T_m = 10$ and the number of required sensors per mission $N_m = 5$. The number of sensors should be increased in an interval that generates meaningful results, an example could be from 10 to 1000 sensors, increasing each step by 50 sensors.

Two graphs should be generated comparing the results of the three algorithms as a function of $N$:

- Percentage of satisfied missions
- Percentage of nodes with less than 10 units of remaining energy

Increasing the mission duration

This set of experiments wants to study the performance of the algorithms increasing the mission duration $T_m$. We assume that $N = 500$ sensors are deployed, and the number of required sensors per mission $N_m = 5$. The mission duration should be increased in an interval that generates meaningful results, an example could be from 1 to 100, increasing each step by 10.

Two graphs should be generated comparing the results of the three algorithms as a function of $N$:

- Percentage of satisfied missions
- Percentage of nodes with less than 10 units of remaining energy

Increasing number of sensors per mission

This set of experiments wants to study the performance of the algorithms increasing the number of sensors per mission $N_m$. We assume that $N = 500$ sensors are deployed, and the duration of each mission is $T_m = 10$. The number of sensors per mission should be increased in an interval that generates meaningful results, an example could be from 1 to 20.

Two graphs should be generated comparing the results of the three algorithms as a function of $N$:

- Percentage of satisfied missions
- Percentage of nodes with less than 10 units of remaining energy
Reports

Two reports should be prepared. The first report does not require implementation and should include at least the following sections:

- Abstract: Summary of the following sections
- Proposed solutions: Pseudo-code, description and complexity analysis of the proposed algorithms
- Plan of experiments: Description of the methodology which is planned to use in the experiments, significance and expected results.

The second report requires implementation and describes the implementation and results. It should include at least the following sections:

- Abstract: Summary of the following sections and results
- Implementation: Description of the methodology used for implementing the proposed solutions (e.g. relevant classes and data structures, structure of the program, etc.).
- Experiments: Description of the experiments performed and obtained results. Results should be represented in a graphical form as well as discussed in the writing.
- Conclusions: Summary of the work and final considerations.