

Visualization of Mobile Search for a Black Hole in an Anonymous Ring

9th December 2015

OVERVIEW AND GOAL

Our visualization is based on algorithms from paper ‘Mobile Search for A Black Hole in an Anonymous Ring’. We have visualized algorithms of Divide, OptTime, TradeOff, Pairing and Gathering. Within the built-in function of move and ideal time calculations, end user can easily compare and analyze each algorithm’s complexity.

HOW TO ACHIEVE IT

- Framework and Language

We use HTML5 Canvas to handle drawing events correspond with JavaScript for implementing algorithms and agents moving.

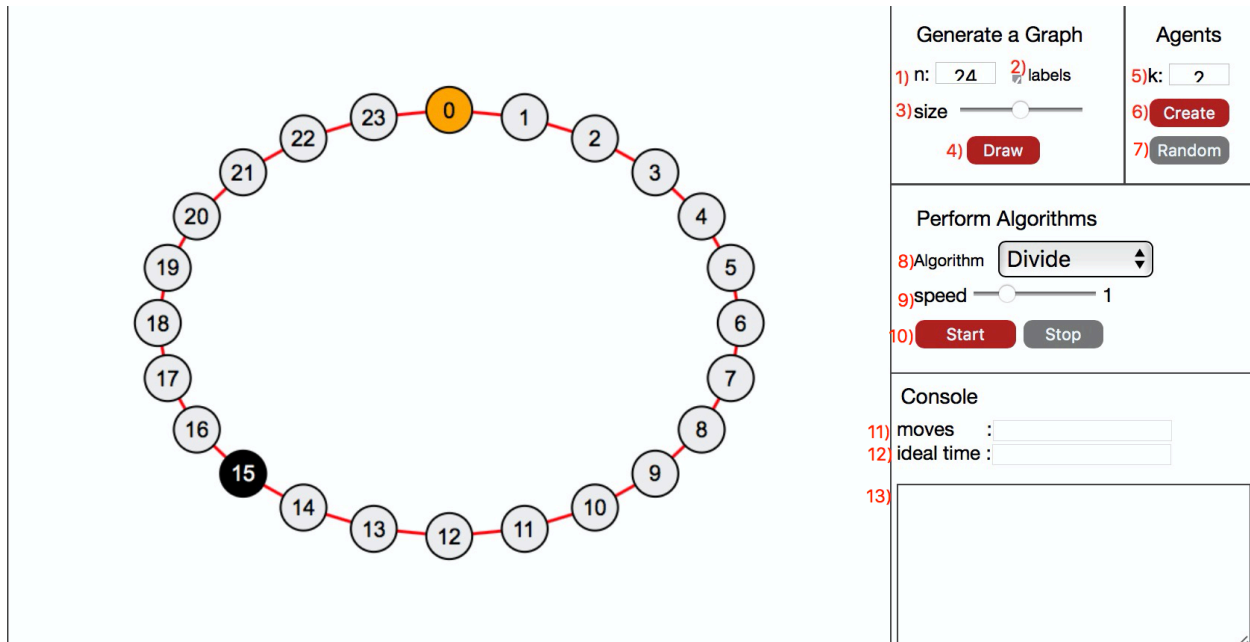
- Implementation Details

The visualization is based on the input from user’s option, then agent’s move is handled by cautious walk and its specific algorithm. The change of agents’ move is triggered by whenever agents reach to next target node.

State of an agent is used to control agent’s moving direction. Each time an agent arrives at a node, the collision detection mechanism will notify the agent to update its status according to the identity of a node and the state of a link when it passes through a node or a link.

Although agents on algorithms can move asynchronously, we only consider the synchronous scenarios. we generate the movement of agents sequentially by calling the callback function, eventually the visualization looks like a concurrent movement among agents and it is crucial for calculating ideal time complexity.

INSTRUCTION AND EXAMPLE



The image shows a circular graph with 24 nodes labeled 0 to 23. Node 0 is highlighted in yellow, and node 15 is highlighted in black. The graph is connected by red lines. To the right is a control panel with sections for 'Generate a Graph', 'Agents', 'Perform Algorithms', and 'Console'.

Generate a Graph

1) n: 2) labels

3) size

4)

Agents

5) k:

6)

7)

Perform Algorithms

8) Algorithm

9) speed

10)

Console

11) moves :

12) ideal time :

13)

Labels:

1. Number of nodes
2. Label IDs for each node
3. Adjust the graph size
4. Draw the graph with a random black hole position
5. Number of agents
6. Show the initial agents position randomly based on 'k' input
7. Show the initial agents position randomly with arbitrary 'k'
8. *Select an algorithm
9. Adjust agent moving speed before and during the execution
10. *Start execution
11. Real time updating for number of moves
12. Real time updating for number of moves
13. Logs for an algorithm

*Required field

Notice:

- it's better to show agents position for algorithms that has dispersed agents by clicking 'create' before execute even it's not required.
- After execution stops, by clicking the button 'start', it will restart the last time execution.

Figure below is an example for a result of an algorithm. The console box shows the crucial steps during the execution of an algorithm and finally 'moves' and 'ideal time' are indicators to show the current complexity.

The figure displays a circular graph with 24 nodes, numbered 1 to 24. An agent, labeled 'Agent 2' and represented by a Mario character, is positioned at node 1. Node 15 is highlighted in black, indicating a 'black hole'. The interface includes several control panels:

- Generate a Graph:** n: 24, labels, size slider, Draw button.
- Agents:** k: 2, Create button, Random button.
- Perform Algorithms:** Algorithm: Divide, speed slider (10), Start button, Stop button.
- Console:** moves :117, ideal time :92. The console shows the following steps:
 - 1) U right = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 }
U left = { 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 }
 - 2) U right = { 13, 14 }
U left = { 15 }
 - 3) Agent 2 writes on the white board of node 16.
 - 4) Agent 2 reports to the base that the black hole is at node 15 (algorithm terminates).

PROBLEMS ENCOUNTERED

Calculating Moves and Ideal time

Symptom :

- Ideal time calculation for algorithm TradeOff was not updated correctly.
- Moves calculation for algorithm TradeOff was not updated correctly.

Reason:

-
- When an agent notifies other agents, the delay of receiving notification is not avoidable since the algorithm is implemented sequentially so that the extra time delay (measured in unit time) which should never happen during synchronous scenario is actually accumulated several times.
 - Since the calculation of move is based on whether an agent reaches to a node, there is a case that chasing agents stays at the last safe node which is also counted by the calculation of move.

Agents' Moving Logic

Symptom :

- Agent was not followed through the right arc of path on the ring and even worse, agent may go out of the ring afterward.
- The moves calculation is calculated before agents move.

Reason:

The problem is due to the moving logic for our program that is implemented before the current agent moves therefore if we don't assign the next target's coordinates during agents' initialization, it leads to some strange behavior for agents' moving.

Moves calculation is also fixed by introducing a new temporary variable to count the number of moves before the next stage and we eventually update the final moves calculation when an algorithm terminates.