**LEARNING MACHINE > POOA ALGORITHM**

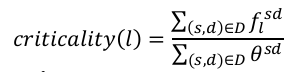
**Parameters:**

|  |  |
| --- | --- |
| \_K | Quantity of candidates (k shortest paths) of each i-e pair  When order path continously remains \_K times, stop learning stage to start post learning stage.  Else, reset race. |

**Tóm tắt:**

POOA gồm 2 pha:

* Offline: Thực hiện trước khi request đến
  + Tìm \_K đường đi ngắn nhất cho mỗi cặp nguồn đích lưu vào \_AllKPaths.
  + Khởi tạo giá trị đua cho mỗi đường (thường khởi tạo = 0) \_XValues.
  + Tính criticality cho link l theo công thức



*∑(s,d)  flsd là tổng giá trị subflow của link l sau khi tăng luồng đạt cực đại thuộc các path của các cặp s - d. Trong đó, link l có thể thuộc 1 hoặc nhiều path. Giá trị Subflow là giá trị link l sau khi đã tăng luồng đạt đến cực đại.*

*∑(s,d)  θ sd là tổng giá trị maxflow của tất cả cặp s-d.*

* Online: gồm 2 giai đoạn (lưu ý là tại mỗi request chỉ tính toán cho cặp nguồn đích tương ứng với request)
  + Learning stage:
    - Tính toán cost cho mỗi đường tìm được trong \_AllKPathi-epair .
    - Chọn trong số các đường đã tính toán, đường có Cost nhỏ nhất và đáp ứng được nhu cầu request .
    - Tăng giá trị đua cho đường được chọn đó.
  + Post-Learning stage: thực hiện sau khi thứ tự đua của các đường duy trì \_K lần giống nhau.
    - Chọn đường có giá trị đua lớn nhất thoả nhu cầu request làm kết quả.

Độ đáp ứng thuật toán phụ thuộc vào các tham số của bài toán đã đề cập trên.

Chi tiết từng bước đã được mô tả bên dưới.

Find k shortest path s foreach i-e pair.

\_AllKPaths[i-epair] = YenAlgorithm (i-epair, k)

**OFFLINE PHASE**

Set foreach path found in previous step x value = 0

\_Xvalue[path] = 0

**ONLINE PHASE**

**\_RemainingKTime [i-epair] = \_K**

Post-Learning Stage

true

Learning Stage

Compute criticality foreach link

First request came

**LEARNING MACHINE > RRATE ALGORITHM > LEARNING STAGE**

i-epair = (Request.Source, Request .Destination)

Foreach path in \_AllKPathi-epair

Foreach link in path

\_PathCost[path] = \_Criticality[link] / link.ResidualBandwidth

Find resultPath

*resultPath = min(costpath∈\_AllKPaths[i-epair],path.Bandwidth>=demand)*

Increase value of candidate

\_Xvalue[resultPath]++

Sort \_Xvalue[iepair]

true

\_RemainingKTime[iepair]++

If Path’s orders remain

Reset race

\_RemainingKTime[iepair] = 0

\_Xvalues[iepair][i] = 0 (∀ i : 0 🡪 \_Xvalues[iepair].Count)

**LEARNING MACHINE > RRATE ALGORITHM > POST - LEARNING STAGE**

**Find resultPath**

*resultPath = Max(\_XValuepath∈\_AllKPaths[i-epair],path.Bandwidth>=demand)*