

ĐẠI HỌC BÁCH KHOA HÀ NỘI

LỚP KỸ SƯ TÀI NĂNG - CÔNG NGHỆ THÔNG TIN K62

Tìm đường thoát hiểm

Ngày 8 tháng 5 năm 2019

Author

1 Tiêu chuẩn để đánh giá trọng số con đường

Xét đoạn đường với chiều dài L , chiều ngang H , chỉ số an toàn T , và hiện tại có N người trên đoạn đường đó. Giả sử trong điều kiện bình thường, đoạn đường không có người, độ an toàn $T = 1$, thì với một người di chuyển vào con đường đó sẽ có vận tốc là V . Nhưng với ngoại cảnh cụ thể, thì vận tốc người đó sẽ nhỏ hơn, có thể coi như vận tốc thực tế của người đó tỉ lệ thuận với độ an toàn và tỉ lệ nghịch với mật độ người trên con đường đó. Ta có thể xét

$$v = \frac{V}{F(T, D)}$$

$D = \frac{L * H}{X * Y}$ là mật độ người trên đoạn đường, $F(T, D)$ ta gọi là hàm ngữ cảnh, đặc trưng cho sự ảnh hưởng của ngoại cảnh tới vận tốc di chuyển của con người. Như vậy, thời gian để những người đó đi hết đoạn đường là:

$$t = \frac{L}{v} \Leftrightarrow t = \frac{L * F(T, D)}{V}$$

Đặt trọng số $W = L * F(T, D)$ thì $t = \frac{W}{V}$.

Như vậy, trong công thức trên, nếu V là vận tốc chuẩn của một người thì w giống như chuẩn độ dài của con đường trong ngoại cảnh cụ thể, và thời gian đi qua con đường sẽ được tính bởi hai đơn vị trên.

Ta sẽ giả sử tất cả mọi người trong tòa nhà đều có cùng vận tốc trung bình là v_{tb} , và khi đó thì để so sánh xem hai con đường nào tốt hơn, tức là ta phải so sánh thời gian thoát hiểm của hai con đường đó, đồng nghĩa với việc ta phải so sánh trọng số w của quãng đường.

Ở đây, trong mô phỏng, ta sử dụng hàm:

$$F(T, D) = \frac{1}{T * (1.0001 - D)} \quad (1)$$

2 Ý tưởng thuật toán

Xét đoạn đường đi từ một Indicator u trong tòa nhà tới một exit node e , gồm các corridor theo thứ tự là $p = \langle p_1, p_2, \dots, p_k \rangle$. Với mỗi corridor p_i sẽ có tương ứng n_i người.

Trọng số w của đoạn đường được tính: $w = \sum w_i$ với $w_i = \frac{L_i * g(D_i)}{f(T_i)}$ là trọng số của quãng đường tương ứng tại thời điểm hiện tại. Với công thức như trên thì ta thấy trọng số của đường đi từ Indicator u sẽ phụ thuộc vào mật độ lượng người trên mỗi cạnh tại thời điểm tính, giả sử là t_1 . Ta có nhận xét rằng khi mà nhóm người A ở cạnh p_{i1} tới được cạnh p_{i2} ($i2 > i1$) thì những người ở cạnh p_{i2} đã di chuyển sang cạnh khác, tức là họ không ảnh hưởng tới thời gian di chuyển của nhóm người A trên cạnh p_{i2} . Những thành phần ảnh hưởng trực tiếp tới thời gian di chuyển trên cạnh p_{i2} của nhóm người A là điều kiện môi trường và lượng người di chuyển trên đó cùng thời điểm nhóm A có di chuyển trên p_{i2} . Như vậy, nếu ta muốn lấy trọng số w để đặc trưng cho thời gian thoát hiểm của một nhóm người trên một đoạn đường thì không thể tính bởi công thức như trên. Để có thể tính chi li, xét dãy $t = \langle t_1, t_2, \dots, t_k \rangle$, t_i là thời điểm mà số người đó bắt đầu đi vào quãng đường p_i , thì trọng số w sẽ được tính

$$w = \sum w_i \quad (2)$$

với w_i là trọng số của quãng đường p_i tại thời điểm t_i .

Vấn đề là với mỗi đường đi khác nhau sẽ có cách chia thời gian khác nhau, và để tính toán với từng đoạn đường như vậy là không thể về khía cạnh thời gian tính toán.

Với ý tưởng tương tự trên, ta sẽ chia đường đi làm 2 phần. Phần đầu tiên là quãng đường đi được trong $t(s)$ tiếp theo, phần tiếp theo là đoạn còn lại. Giả sử trong $t(s)$ tiếp theo thì những người ở Indicator u đi đến được đoạn p_i .

Đặt $path1 = \langle p_1, \dots, p_{i-1} \rangle$, $path2 = \langle p_i, \dots, p_k \rangle$. Trọng số của đoạn đường p sẽ được tính:

$$w = w1 + w2 \quad (3)$$

với

- $w1 = \sum w_j, j < i$, các đoạn w_j được tính với số người trên đoạn là n_1 người, hay là số người đang ở p_1 tại thời điểm hiện tại.
- $w2 = \sum w_j, i \leq j \leq k$, các đoạn w_j được tính với số người tới được p_i trong t giây tiếp theo.

2.1 Mô hình tòa nhà

Tòa nhà sẽ có mô hình giống như mô hình tòa nhà trong thuật toán LCDT, gồm các Indicator nhận thông tin từ máy chủ của tòa nhà và chỉ hướng cho những người gần đó trong trường hợp di tản. Nó sẽ vẫn chứa dữ liệu backup dùng cho trường hợp mất kết nối với máy chủ, sẽ là hướng chỉ đến đỉnh tiếp theo trong cây khung đường đi ngắn nhất đến các lối ra.

3 Nội dung thuật toán

3.1 Mô hình chung

Để tiện cho việc tính toán, ta coi các Indicator như là các Node trong đồ thị, nối các Corridor được biểu diễn như các Edge nối giữa các Node.

Ta áp dụng thuật toán Dijkstra để tìm đường đi có trọng số nhỏ nhất của mỗi node tới exitNode-lối ra của tòa nhà. Ta đưa ra hàm tính trọng số của một cạnh như sau:

$$W(edge) = L * F(edge.trustiness, edge.density) = \frac{L}{T * (1.01 - D)} \quad (4)$$

Xét với một nhóm người A trên một cạnh e đang đi trên đoạn đường path từ đó về tới exit node. Với ý tưởng chia đoạn đường từ đỉnh e đến exit node thành hai đoạn, path1 và path2, trong đó:

- path1: là đoạn đường gồm các đỉnh mà nhóm người A đi được trong thời gian τ . Xét S là đỉnh cuối cùng trong path1, tức là S là đỉnh xa nhất mà nhóm người A đi được trong thời gian τ trên path. Ta xét rằng khi nhóm người A đi đến các đoạn đường P tiếp theo trên path1 thì những người trước đó ở P đã di chuyển sang con đường khác. Như vậy thì các yếu tố ảnh hưởng tới thời gian di chuyển của những người A trên path1 chỉ là các yếu tố vật lý, độ tin cậy và số lượng người trong nhóm A di chuyển trên đường. Do đó ta sẽ tính trọng số của đoạn đường path1 tương ứng với số lượng người A, chứ không phải là số lượng người trên cạnh của từng đoạn đường tại thời điểm tính toán.
- path2: là đoạn đường bắt đầu từ đỉnh S tới exit node trên con đường path. Khi nhóm người A di chuyển trên path1, thì những nhóm người khác ở các đỉnh xung quanh cũng đang di chuyển. Như vậy khi nhóm người A tới được S thì cũng sẽ có các nhóm người khác tới được đỉnh S cùng thời điểm với nhóm người A. Như vậy từ đỉnh S, cũng với lập luận rằng những người ở đoạn đường phía sau đoạn chứa nhóm người A sẽ di chuyển đến các đoạn tiếp theo khi A tới được đó, như vậy thì lượng người trên các đoạn đường tiếp theo dọc theo path mà vào các thời điểm nhóm người A đi qua thì sẽ có lượng người trên đó bằng số lượng người tới được đỉnh S trong thời gian τ . Do đó ta sẽ tính trọng số của đoạn đường path2 cho lượng người A theo công thức trước đó với số lượng người trên mỗi đoạn là lượng người đến được S trong thời gian τ .

Một điều quan trọng trong cách tính trọng số trên là làm sao biết được có bao nhiêu người tại đỉnh S sau τ giây tiếp theo.

Trong thuật toán dijkstra, các đỉnh được gán nhãn sau là các đỉnh có trọng số đường đi ngắn nhất tới đỉnh nguồn lớn hơn với các đỉnh được gán nhãn trước đó. Với công thức tính trọng số đoạn đường mà đặc trưng được cho thời gian để con người đi hết đoạn đường đó thì những đỉnh nào được gán nhãn trước tức là những người ở gần đó sẽ tới exit nhanh hơn, hay có nghĩa là việc họ di chuyển sẽ không ảnh hưởng tới những người ở đằng sau khá xa họ. Và trong thuật toán này, khi một đỉnh đã tìm được đường đi ngắn nhất cho nó, thì nó sẽ thông báo cho các đỉnh khác mà vẫn đang tìm đường biết được quãng đường của nó, để các đỉnh khác có thể tính toán lượng người cùng di chuyển trong tương lai.

Node:

- *next*: đỉnh được lựa chọn là hướng đi đến.
- *nextEdge*: cạnh được lựa chọn là hướng đi đến.
- *adjacences*: Danh sách struct gồm có:
 - *node*: đỉnh kề
 - *edge*: cạnh kề
 - *passingWeight*: trọng số của path mà đi qua đỉnh kề
 - *reaching*: đỉnh tới được sau t(s) mà đi qua đỉnh kề
- *weight*: trọng số của đường đi tới root.
- *nComingPeople*: số người sẽ đến sau t(s).
- *tReachedNode*: đỉnh sẽ tới được sau t(s).
- *tComingNodes*: danh sách đỉnh tới được sau t(s).
- *label*: nhãn của đỉnh.

Edge

- *to*: đỉnh tới.
- *Length*, *Width*, *Trustness*: các thông số.
- *nPeople*: số người trên cạnh.
- *density*: mật độ.
- *weight*: trọng số của cạnh.

Trong thuật toán 1, từ dòng 7-10 thể hiện rằng khi đỉnh u được gán nhãn, thì nó sẽ cập nhật cho đỉnh S biết rằng nó sẽ tới được đó trong τ giây nữa. Sau khi cập nhật thông tin cho đỉnh S thì đỉnh S có nhiệm vụ báo lại cho những đỉnh đằng trước mà đang muốn đến S biết các thông tin để có thể tính toán lại trọng số khi mà chọn con đường đi qua S(dòng 10).

Như ta đã biết thì trong thuật toán dijkstra thì trọng số của các đỉnh sau khi được gán nhãn chỉ phụ thuộc vào đường đi của nó tới nguồn trên cây khung dijkstra. Như vậy những người ở những con đường không nằm trên cây khung dijkstra sẽ không được xét đến. Nhưng họ vẫn sẽ di chuyển, và sẽ làm thay đổi mật độ người ở những đoạn đường họ tới được. Vì vậy, ngay sau khi gán nhãn cho một đỉnh, ta phải duyệt những cạnh nằm giữa đỉnh đó và các đỉnh xung quanh đã gán nhãn, cập nhật lượng người cho các đỉnh phía trước, bởi vì trong thuật toán dijkstra những cạnh này sẽ không được duyệt tới sau này nữa(Dòng 11-12).

Algorithm 1 Algorithm caption

Input: Đồ thị có trọng số ứng với các Indicator và Node

Output: Đường đi ngắn nhất từ root đến mọi đỉnh trong đồ thị

heap: cấu trúc heap min với phần tử là các đỉnh và so sánh dựa trên weight.

```
1: procedure MAIN ALGORITHM
2:   heap.push(root)
3:   root.label  $\leftarrow$  true
4:   while heap.size > 0 do
5:     u  $\leftarrow$  heap.pop()
6:     u.label = true
7:     s  $\leftarrow$  u.tReachedNode
8:     s.nComingPeople  $\leftarrow$  s.nComingPeople + u.nextEdge.nPeople
9:     s.tComingNodes  $\leftarrow$  u
10:    UpdateComingNode(s)
11:    for v in u.adjacences and v.node.label = true do
12:      UpdateComingPeople(v.edge)
13:      for v in u.adjacences and v.node.label = false do
14:        s = FindCrossNode(v.node, v.edge)
15:        s.nComingPeople  $\leftarrow$  s.nComingPeople + v.edge.nPeople
16:        w1  $\leftarrow$  CalculateWeight(u, s, v.edge.nPeople)
17:        w2  $\leftarrow$  CalculateWeight(s, root, s.nComingPeople)
18:        newW  $\leftarrow$  v.edge.weight + w1 + w2
19:        ad in v.node.adjacences which ad.node = u
20:        ad.edgeWeight  $\leftarrow$  newW
21:        ad.reachedNode  $\leftarrow$  s
22:        if newW < v.node.weight then
23:          v.node.weight  $\leftarrow$  newW
24:          v.node.next  $\leftarrow$  u
25:          v.node.tReachedNode  $\leftarrow$  s
26:          heap.push(v.node)
27:        end if
28:        s.nComingPeople  $\leftarrow$  s.nComingPeople - E[v.node,u].nPeople
      end for
    end while
```

Khi một đỉnh *u* được gán nhãn, ta sẽ tính toán trọng số đường đi rồi đưa chúng vào heap. Chú ý là một đỉnh *x* bất kì có *k* đỉnh kề với nó thì nó có thể chọn đi qua *k* đường ngắn nhất của các đỉnh kề để về tới nguồn. Ta sẽ lưu lại *k* giá trị trọng số của *k* đường này. Ta sẽ làm như vậy do khi một đỉnh tìm được đường "ngắn nhất" cho nó thì sẽ cập nhật lượng người cho đỉnh *S* tới được sau τ giây, như vậy lượng người tại *S* trong tương lai sẽ tăng lên. Chính vì vậy mà trọng số con đường của các đỉnh tới được *S* trong tương lai sẽ thay đổi. Ta phải cập nhật lại cho những đỉnh đó, và như vậy trọng số sẽ thay đổi, nên sẽ phải so sánh tìm lại con đường tốt nhất trong những thẳng kề nó.

Algorithm 2 UpdateComingPeople

Input: Edge $edge$, Node $node$

Cạnh nằm giữa hai đỉnh đã được gán nhãn và chưa được cập nhật.

Output:: Cập nhật số người ở cạnh đó cho đỉnh nó tới được sau $t(s)$

```
procedure MY PROCEDURE
   $s \leftarrow \text{FindCrossNode}(node, edge)$ 
   $s.nComingPeople \leftarrow s.nComingPeople + edge.nPeople$ 
  UpdateComingNode( $s$ )
```

Algorithm 3 UpdateComingNode

Input: Node s

Output: Cập nhật lại trọng số của các node chưa được gán nhãn mà sẽ tới được s sau thời gian τ .

```
procedure MY PROCEDURE
  for  $u$  in  $s.tComingNodes$  and  $u.label = false$  do
     $v$  in  $u.adjacences$  which  $v.reaching = s$ 
     $w1 \leftarrow \text{CalculateWeight}(v.node, s, v.edge.nPeople)$ 
     $w2 \leftarrow \text{CalculateWeight}(s, root, s.nComingPeople)$ 
     $v.passingWeight \leftarrow v.edge.weight + w1 + w2$ 
    GetNextNode( $u$ )
    heap.Rebuild( $u$ )
  end for
procedure GETNEXTNODE( $u$ ) ▷ Đặt lại đỉnh kề tốt nhất cho  $u$ 
  for  $v$  in  $u.adjacences$  and  $v.node.label = true$  do
    if  $v.edgeWeight < u.weight$  then
       $u.weight = v.edgeWeight$ 
       $u.next = v.node$ 
       $u.tReachedNode = v.reachedNode$ 
    end if
```

Trong thuật toán 4, do s và u được gán nhãn nên con đường đi từ u đến s là đoạn đường trên cây khung dijkstra.

Algorithm 4 FindCrossNode

Input: *Node* node, *Edge* edge

Output: Trả lại đỉnh xa nhất trên đường đi ngắn nhất của u mà v đi tới được trong thời gian t .

```

procedure MY PROCEDURE
     $sumWeight \leftarrow v_{tb} * t$ 
     $nPeople \leftarrow edge.nPeople$ 
     $from \leftarrow node$ 
    while  $sumWeight > 0$  do
         $sumWeight \leftarrow sumWeight - \mathbf{GetWeight}(edge, nPeople)$ 
         $from \leftarrow edge.to$ 
         $edge \leftarrow from.nextEdge$ 
        if  $from = root$  then break
    end while
    return  $from$ 

```

Algorithm 5 CalculateWeight

Input: *Node* u , *Node* s , int $nPeople$

Output: Trả lại trọng số của quãng đường đi từ u tới s với $nPeople$ người đi.

```

procedure MY PROCEDURE
     $weight \leftarrow 0$ 
    while  $u \neq s$  do
         $edge \leftarrow u.nextEdge$ 
         $u \leftarrow u.next$ 
         $density \leftarrow \mathbf{GetDensity}(edge, nPeople)$ 
         $weight \leftarrow weight + L * \mathbf{F}(edge.Trustness, density)$ 
    return  $weight$ 

```
