

# Re: Finding simple cycles approximating target distance

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2006-01/msg01219.html>

---

- *From:* [israel@xxxxxxxxxxx](mailto:israel@xxxxxxxxxxx) (Robert Israel)
  - *Date:* 10 Jan 2006 20:07:12 GMT
- 

In article <1136885432.085264.66410@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>, KoenM <rockinfewl@xxxxxxxx> wrote:

>Hi all,

>

>Thinking about developing a bicycle route finder, I'm not sure how to  
>best solve the following problem relating graph theory.

>

>Given a directed weighted graph, and a specific vertex as starting  
>point, how do I find simple cycles approximating some target distance,  
>most efficiently?

>>>From the graph at hand, I expect result cycles will consist of about 10  
>to 100 edges.

>

>I guess I'm after 'simple' cycles here, to make sure I get  
>'interesting' cycles to bike, and not heavily zigzagging ones that pass  
>the same point multiple times?

>

>Any pointers or algorithms you have in mind?

Assuming the weights are all positive and you want the total weight in the interval [a,b], you can do a depth-first search using the following pseudo-code.

I assume neighbours(x) for node x returns the set of nodes y such that x -> y is an arc of your digraph.

```
searchit:= proc(s,t,A,lo,hi)
# find a path from s to t of total weight in [lo,hi], avoiding
# vertices in A
for y in neighbours(s) minus A do
w := weight(s -> y)
if y = t then
if w >= lo and w <= hi then return (s,t)
else if w <= hi then
temp := searchit(y, t, A union {y}, lo - w, hi - w)
if temp <> NULL then return (s,temp)
return NULL
```

And call it with

Re: Finding simple cycles approximating target distance

searchit(startpoint,startpoint,{ },a,b)

Robert Israel israel@xxxxxxxxxxx  
Department of Mathematics <http://www.math.ubc.ca/~israel>  
University of British Columbia Vancouver, BC, Canada

- 
- *Follow-Ups:*
    - ◆ *Re: Finding simple cycles approximating target distance*  
◇ From: KoenM
  - *References:*
    - ◆ *Finding simple cycles approximating target distance*  
◇ From: KoenM
  - Prev by Date: *Re:  $x^x = 1/4$*
  - Next by Date: *Re:  $x^x = 1/4$*
  - Previous by t