Navigating through Wybe

Gerard Tel
Dept Computer Science, Univ. Utrecht

Fall 2012

What happens inside a PND (Personal Navigation Device)? Bas den Heijer has programmed a small demo: *Wybe*, and this worksheet gives you some tasks that guide you through it. The demo was made for Dutch highschool students, so its controls and explanations are in Dutch.

1 En Route with Wybe

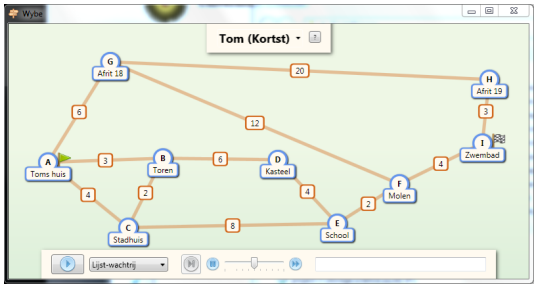
First get all the files you need.

1. **Get the files.** Program and example maps are in a folder called *RoutePlanner*. Download this folder from `http://www.staff.science.uu.nl/~tel00101/Wybe/`, and extract it on your Desktop or USB stick.

2. **What files are there?** Open the folder *RoutePlanner*. You should have *Wybe.exe* (icon: road sign). and a few with a file name ending in *.graph.txt*; these are Wybe's maps.

3. **Do you have a text editor?** Double click `negatief.graph.txt`; it should open in a text editor (like NotePad). Close the file again.

4. **What controls does Wybe have?** Start `Wybe.exe` (by double clicking), and see a screen as depicted here on the right.



Here are the controls:

- *Map selector:* Initially lists Tom (Kortst), choose between shipped maps or those you make yourself.
- *Info:* Background about Wybe (in Dutch).
- *Start button* and *Stop & Reset button:* Left bottom, starts route computation, and then stops and resets the computation.
- *Wachtrij (Queue):* Choose (Lijst, Boom, or Mens) the mechanism to select the next node to make definite.
- *Stepper and speed slider:* Slide the speed slide to the right for fast operation, to the left to operate step by step with the stepper button.
- *Uitleg (Explanation):* Wybe explains what it does (in Dutch).
- *Start flag:* Click start flag and then a node to change start point.
- *Destination flag:* Click destination flag and then a node to change destination point. (Start and destination can never be the same.)

2 Finding Routes

In the next steps, you will compute a few Wybe routes, and find out that the chosen route will depend on the map material.

5. **Shortest Route.** The map that Wybe starts with is "Tom (Kortst)", giving the length of each road segment in kilometers. What would be the shortest way from Tom's Home (Toms huis) to the Swimming Pool (Zwembad)?

   You can compute the shortest route by pressing the start button (left bottom). It runs a bit slow, but moving the speed slider to the right speeds up the process.

   Finally the selected route becomes black; is it that same as what you thought it would be?

6. **Why does Wybe stop?** Wybe also computed the distance from Tom's house to Afrit 18, Toren and Kasteel. Next to Afrit 19 you see now 26, whereas (via Zwembad) it is just 21km. Why is the computation not completed?

7. **What does a map look like?** Double click in the folder *RoutePlanner* the file *atomafstand.graph.txt* to open it in your editor. In the second line is the name of the map: Tom (Kortst). A bit lower, the list of locations (Nodes), each with name and coordinates. Then the list of road segments (Edges), each listing the points it connects and the length.

   Can you find the listed edge `0-6 6` in the map drawing?

8. **Adding and changing roads.** Add a road between Toren and School of length 12. (To see it, you must store the file, then in Wybe choose another map and return to Tom (Kortst) to load it again.)

   Did the shortest route change? How short should you make the edge between Toren and School to get a shorter route to Zwembad? Try to make it so short that the overall distance is 16.

9. **Route preferences.** In a PND you can usually select preferences: Will you walk, ride bike, or drive? Do you want fast, short, fuel economy? Do you want to take unpaved roads and ferries, or avoid them?

   Compare the maps Tom (Kortst), Tom (Snelst), Tom (Schoonst) and Tom (Wandel). They show, how a PND sees the map under different preferences. Kortst means Shortest and lists kilometers, Snelst means Fastest and lists minutes, Schoonst means Cleanest and lists grams of CO2 exhaust. In the map Wandel, which means Walk, the highway between Afrit 18 and Afrit 19 is completely absent.

   A real life PND uses a single map, with various informations per edge and node. The cost and availability of each edge is determined during the computation, depending on the preferences.

3 How does it work?

The next few steps teach you more about how route finding works with Dijksta's algorithm. If you would rather learn about what you can do with it, go to task 16.

Searching for routes does not follow the "logical", human process.

10. **How does Wybe search?** Open the map Klungelig. What is the route from A to AK? If you push Start, you see that it is discovered quickly that the distance to AK is 10000. Still, the right part of the network is searched completely before this 10000 is taken definitively. The computer program does not have a "sense of direction", and will search all points on the right to make sure there is no shorter path there.

11. **Definite and tentative nodes: Estimates.** During the computation, you see nodes becoming blue: we call them *definite* because for those nodes, the distance from the start point is known for sure. For the white nodes, the distance is not known for sure, even though some have number, an *estimate* next to them. The estimate is always: the best route through one of the definite (blue) neighbors.

12. **Change of estimate: Node N.** The estimate gives an indication of distance, but it may still improve as more neighbors become blue. Choose the map Rooster 1 and observe node N. After node H becomes blue, N gets estimate 11, but later when M becomes blue, it improves the estimate of N to 10.

    How can Wybe ever be certain that the estimate for some node will not improve any more?

13. **Dijkstra's invention: lowest estimate.** Edsger Wybe Dijkstra solved this question in 1959. The *lowest* estimate can not change anymore. So, Dijkstra's algorithm repeatedly selects the tentative node with lowest estimate, and makes it definite (it temporarily gets a red cloud behind it).

    If you choose Rooster 1 again, and shift the speed slider to the left, you can see that this happens. Repeatedly the white node with lowest estimate is chosen, then it is made blue (definite), after which the estimate of its neighbors is adapted.

14. **Choosing the next node.** For Rooster 1 this is easy to follow, but for Rooster 2 it becomes harder. Choose map Rooster 2, and switch the control below from Lijst-wachtrij to Mens-wachtrij; now you have to click the lowest node yourself!

    Push the start button and observe that next to A a 0 appears, but that's all. Click A to make it blue, and estimates appear next to B and V. Which is the lowest? Click it. Then there are two nodes that share lowest estimate 3, and it does noet matter if you click M or V first.

    Fortunately, in a real PND you don't have to do this! For a long route through Europe, millions of nodes may have to be visited. And if you click a wrong node once (a white node with non-minimal estimate) your route may be a detour!

15. **Lijst-wachtrij and Boom-wachtrij.** After blueing 10 to 20 nodes, it becomes rather a mess. In each step, you have to keep track of the entire "border zone" of the blue area. The white nodes with an estimate are called the *front*. In a computation in a map of Europe, the front may consist of thousands of nodes.

    The option lijst-wachtrij (list) keeps track of the front nodes, but without regard to their estimate order. In each step, the list is traversed completely to find the lowest. Even though computers are quite fast today, this idea would be too slow in a large map. The option Boom-wachtrij (tree) also keeps the front nodes, but additionally maintains their order, even under changes in estimates. It is more complicated, but faster: for each step, just a couple of dozens of nodes inspected.

4 Modelling with Wybe

Sometimes a route computation is more complicated, and you may think an adaptation of the algorithm is required. Fortunately, it is often possible to model requirements into the map, rather than making the algorithm more complicated.

16. **The Underground.** Open map Metro (Naief). You see the Underground network of a large Central European City, with driving time per edge. You are travelling from Zicin to Haje. Press Start (first choose lijst-wachtrij and bring the speed slider to the right) and see that you will change trains in Mustek and Muzeum.

    Stop the algorithm (Stop&Reset), put the start flag in Cerny Most (press green flag, then F) and the end point in Skalka. The route, with change-over in Florenc

and Muzeum, is not good! This is the route that has minimal time in the trains, but because changing train cost time as well, it is faster to continue to Mustek and change for Skalka there.

Can we adapt the planner to deal with changing time and still compute the fastest route always?

17. **Underground: Split stations.** Open map Metro (Gesplitst). This map has the changing stations twice: like Florenc with its Red and Yellow line, is represented by FlorencRe and FlorencGe.

    If you open `metrosplits.graph.txt` and `metronaief.graph.txt` in the editor, you see the differences. The naive map has one Florenc (node 3), but the split node has 3 FlorencGe and an extra node 17, FlorencGe. Under Edges you see the connection `3-17` with cost 4 minutes.

18. **VIA-point?** Open map Toms Quest. Tom wants to swim (Zwembad) but first needs money (from the PIN-machine). Unfortunately, the fastest route to Zwembad does not run through PIN-machine.

    How do you find the fastest route that runs through PIN-machine?

19. **Combinatorial route.** Tom has no money and he lost his swimsuit! He can get money from PIN-machine (in 1 minute), or borrow it from his friend (takes 13 minutes), or ask his grandma (Oma), who will feed him cookies for 22 minutes, but also can give him a swimsuit. He can search his swimsuit in his mess at home (22 minutes) or ask one from Grandma (22 min) and get money as well, or buy it in the Zeeman shop if he gets money first!

    Because of all the different ways to get money and swimsuit, the number of possible routes becomes very large. How many route computations would be required?

20. **Tom's Quest: Locations and Store.** We shall now produce a special "map" with all locations and materials that Tom needs, to solve his complete quest in a single route computation. Open map Toms Quest (Rugzak). The idea here is that Tom can not just be at each of ten locations (Huis, Oma, Toren, etc), but, at each of the locations, may possess: (0) nothing, (1) just money, (2) just swimsuit, (3) both money and swimsuit. So the map has 40 locations 9some drawn over each other).

    Open `ztomcombezit.graph.txt` in your editor. The start location is 0 (Tom is at Toms huis and has nothing) and the target is 38 (Tom is in Zwembad with money and swimsuit).

21. **Edges in Toms Quest.** There are four times as many nodes, but also, each original edge occurs four times. Originally, there was just one edge between PIN-machine and School, the line `3-4 6` in file `ztomcom.graph.txt`.

    Map `ztomcombezit.graph.txt` has this line as well, but here only mean to go from School to PIN-machine *without money and swimsuit*. Of course, you can do the same *with* money and/or swimsuit, so we need edges `13-14 6`, `23-24 6` and `33-34 6`.

    If you run Wybe on the map Toms Quest (Rugzak), you see that lines to Zwembad are drawn, but still Wybe reports: Doel is onbereikbaar (meaning: target is unreachable). Why?

22. **Getting things.** The map Toms Quest (Rugzak) allows Tom to reach Zwembad, but not to get money and swimsuit, because the ways to obtain them are not represented in the map.

    Tom can find his suit at home in 22 minutes: that is, from "home with nothing" (node 00) to "home with swimsuit" (knoop 20) in 22 minutes. Add to `ztomcombezit.graph.txt` under `[Edges]` a line `0-20 22`. At PIN-machine Tom can get money in 1 minute: add `3-13 1`.

    Could Tom reach his target now?? Save `ztomcombezit.graph.txt`, change maps in Wybe and go back to Toms Quest (Rugzak) to load the augmented map. Press start.

23. **Duplicating get edges.** Tom can not only find his suit at home when broke, but also if he has money already; so we need an edge: `10-30 22`. Tom can also get money from PIN-machine when he has the swimsuit; add the edge.

    Start Wybe again with the adapted map; Tom swims after 51,5 minutes.

24. **Further getting possibilities.** To compute the fastest solution for Tom, we shall add all possibilities for obtaining suit and money. The node numbering has a little logic in it: nodes 0 through 9 represent locations without money or suit, 10 through 19 repeat those locations with money, 20 through 29 repeat those with suit, and nodes 30 through 39 have the money and swimsuit both.

    From Grandma (Oma) Tom can get money and suit in 33 minutes; add line `6-36 33`.

From his friend (vriend) Tom can borrow money in 13 minutes (note: either *with* or *without* suit!).

At Zeeman he can buy a suit in 2 minutes (only with money!).

After adding all these edges, refresh Wybe's map again and observe that Tom can swim in 49.5 minutes.

(All edges you added in the last task are already there in `ztomcomruilen.graph.txt`.)

5 Limits of Wybe?

We hope you got some idea of what can be done with path planning software. Some of these things already exist, some may only be there in years to come.

25. **What should a PND do?** Maybe you have some creative ideas for what you would like a navigator to compute for you. And maybe you also have ideas about how it can be computed.

26. **One way straats.** Dijkstra's method also works in the same way for edges that you can traverse in only one direction. For simplicity of data entry, Wybe only knows two way streets.

27. **Longest route.** Sometimes you don't want as little, but as much as possible of something. Wouldn't a Castle Route be a path that views as many castles as possible? Suppose we store in the map, which edges run along a castle; could we ask for a "maximum castle route"?

    In general this is not possible, and also it isn't what you really want. You might be willing to ride a little bit further to see some nice castles, but a true "max-caste" route might lead you along thousands of castles all over Europe, just to go to a nearby place.

    This is how to implement a "max-castle" option. Of course, your map must have information about where the castles are. Then, you compute a route in the usual, "minimal" way, but edges along castles get a reduced price, which makes them more attractive for the algorithms to choose. This will probably give some extra castles, possibly against some extra riding cost, but not a guarantee of mathematically maximized castle view.

28. **Traveling Salesman.** In the classical *Traveling Salesman* problem, some cities (locations) must be visited by a traveler, after which he returns home. What order of the visits gives the shortest travel time?

    This question can be modeled as a single quest, and answered by a single computation of Wybe, applied to a suitably chosen map. Just state that Tom must pick up one goodie in each of the cities.

Unfortunately, with every goodie, the number of nodes in the network doubles. A detailed map of The Netherlands already has half a million nodes. With ten locations to visit, we would already have thousand times as many nodes, half a billion. With twenty cities, the Wybe netwerk has 500 billion nodes, which is the limit of even the fastest computers.