# Locally optimal dissimilar paths in road networks

Stéphanie Vanhove     Veerle Fack

Combinatorial Algorithms and Algorithmic Graph Theory
Department of Applied Mathematics and Computer Science
Ghent University

LBS 2011 Vienna

## Work in progress

## Outline

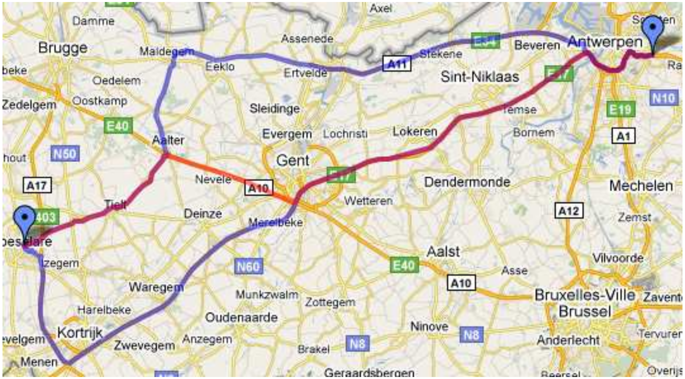
1 Dissimilar paths

2 Our method

3 Preliminary results

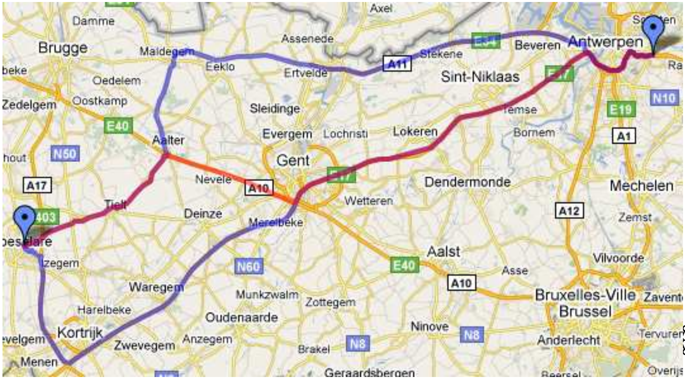
4 Conclusion

## Dissimilar paths



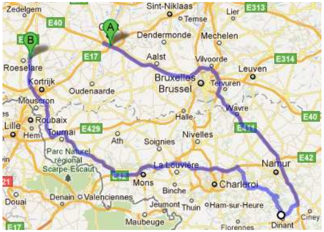
## Why dissimilar paths?

- Alternative routes
- Spreading transportation of hazardous materials

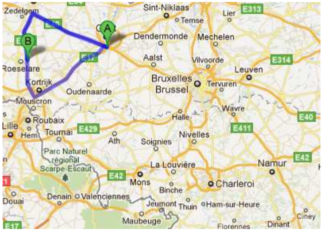


## What to avoid

We may get:



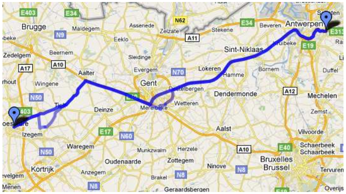
We want:



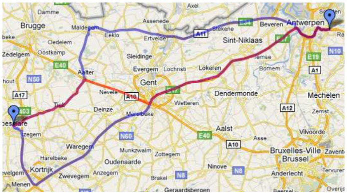
Paths should have *acceptable weights*.

## What to avoid

We may get:



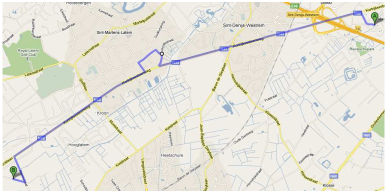
We want:



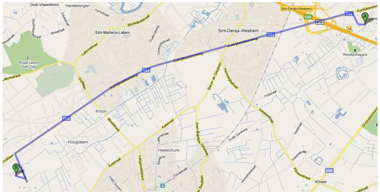
Paths should be *dissimilar*.

## What to avoid

We may get:



We want:



Paths should be *locally optimal*.

## Our goal

Our goal: develop an algorithm which finds a set of paths such that

- the paths are *dissimilar*
- the paths are *locally optimal*
- the paths have acceptable weights
- the calculation can be performed fast

## Outline

1 Dissimilar paths

2 Our method

3 Preliminary results

4 Conclusion

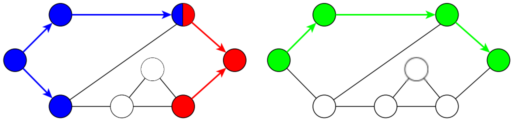
## Our method

1. Generate many paths (e.g. 1000)
   - with a certain maximum path weight
2. Select a dissimilar subset (e.g. 3 paths)
3. Make the chosen paths locally optimal

## 1. Generate many paths

Grow a *forward search tree* from start node and a *backward search tree* from target node.

- Add a new path whenever both searches meet (if not too long)
- Continue until enough paths found or no more paths can be found



## 2. Select dissimilar paths

Definition of dissimilarity $D$ between 2 paths $P_i$ and $P_j$:

**Definition**

$$D(P_i, P_j) = 1 - [L(P_i \cap P_j)/L(P_i) + L(P_i \cap P_j)/L(P_j)]/2$$

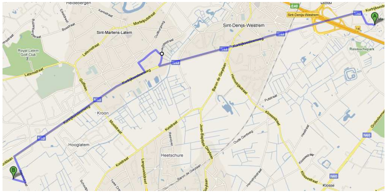
- Assigns a value between 0 and 1
- $0 \rightarrow$ the paths coincide completely
- $1 \rightarrow$ the paths have no arcs in common

## Select dissimilar paths: heuristic

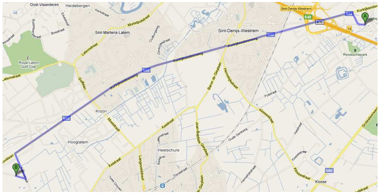
1. Select the shortest path.
2. Out of all remaining paths:
   select path most dissimilar to shortest path.
3. Out of all remaining paths:
   select path most dissimilar to both paths already chosen.
4. ...

## 3. Make them locally optimal

Before:



After:



## 3. Make them locally optimal

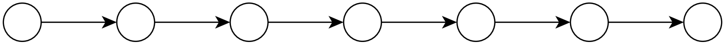
**Definition**

A path is locally optimal if every "short" subpath is a shortest path.

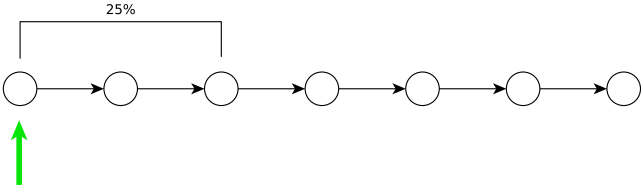
"short" = less than e.g. 25% of the shortest path weight

Method: whenever a "short" subpath is *not* a shortest path, replace it by the shortest path. Repeat until locally optimal.

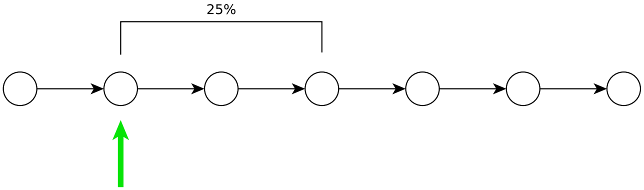
## 3. Make them locally optimal



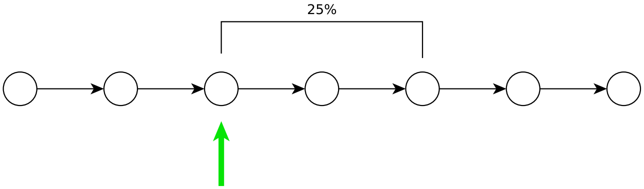
## 3. Make them locally optimal



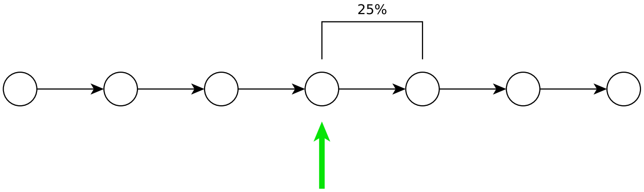
## 3. Make them locally optimal



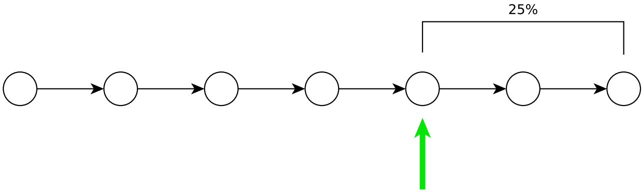
## 3. Make them locally optimal



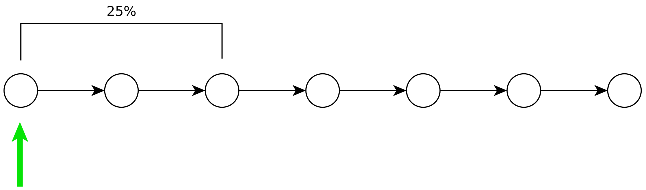
## 3. Make them locally optimal



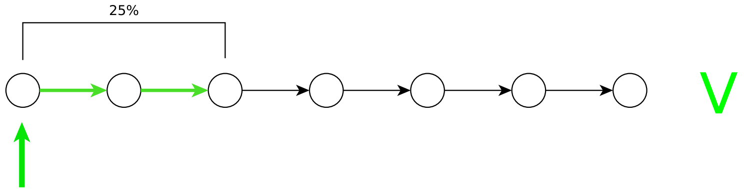
## 3. Make them locally optimal



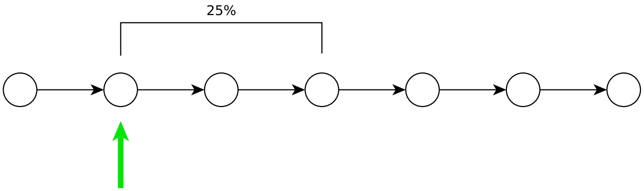
## 3. Make them locally optimal



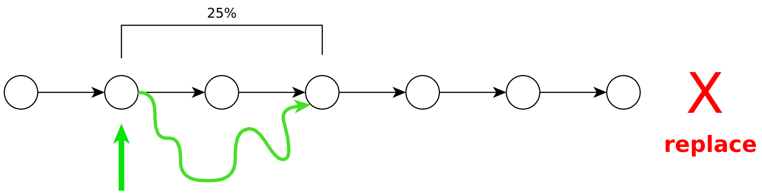
## 3. Make them locally optimal



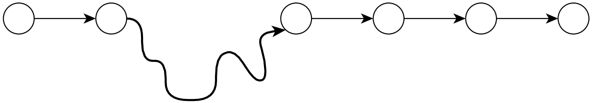
## 3. Make them locally optimal



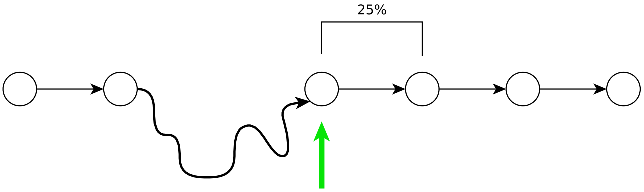
## 3. Make them locally optimal



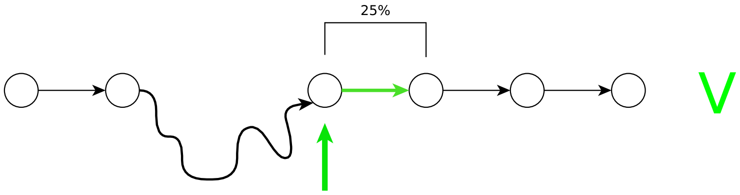
## 3. Make them locally optimal



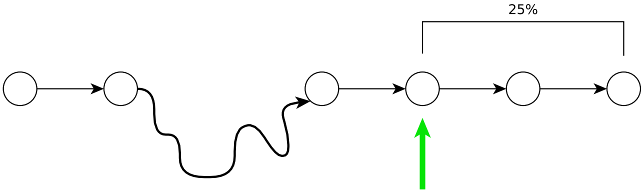
## 3. Make them locally optimal



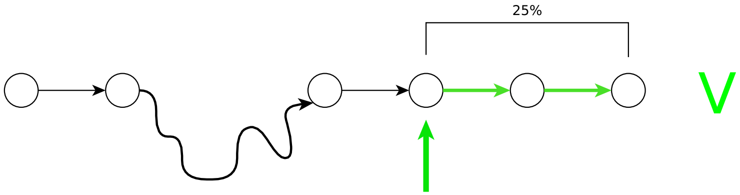
## 3. Make them locally optimal



## 3. Make them locally optimal



## 3. Make them locally optimal



## Outline

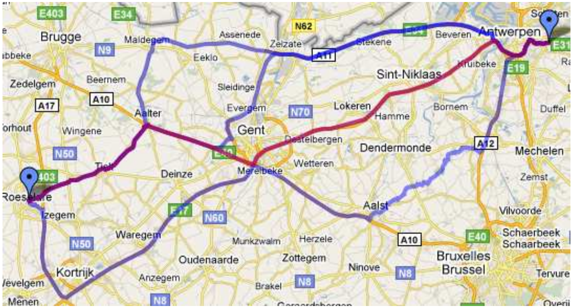
1 Dissimilar paths

2 Our method

3 Preliminary results

4 Conclusion

## Results



- Alternatives 4%, 9%, 15%, 27% longer than shortest path
- All paths are locally optimal for $\alpha = 25\%$
- Calculation time: between a few seconds and a few minutes

## Outline

1 Dissimilar paths
2 Our method
3 Preliminary results
4 Conclusion

## Conclusion

- Quality of the results is satisfying.
- Algorithm could be faster.
- Future work:
    - Speed up the algorithm.
    - Generate paths which are locally optimal immediately.
    - Perform detailed experiments.

Thank you for your attention!

Questions?