Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik
Arbeitsbereich SAV/BV (KOGS)

Image Processing 1 (IP1)
Bildverarbeitung 1

Lecture 12 – Grouping and Searching

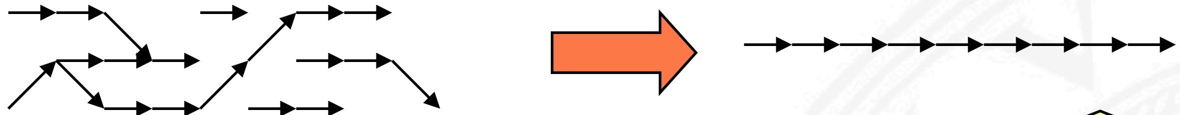
Winter Semester 2014/15

Dr. Benjamin Seppke
Prof. Siegfried Stiehl

Grouping

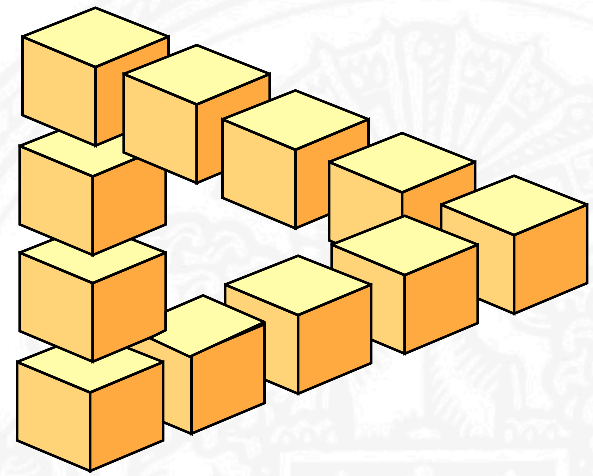
To make sense of image elements,
they first have to be grouped into larger structures.

Example: Grouping noisy edge elements into a straight edge



Essential problem:

Obtaining globally valid results by local decisions



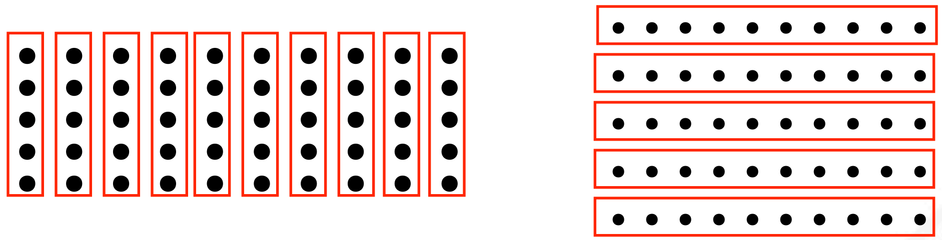
Important methods:

- Fitting
- Clustering
- Hough Transform
- Relaxation

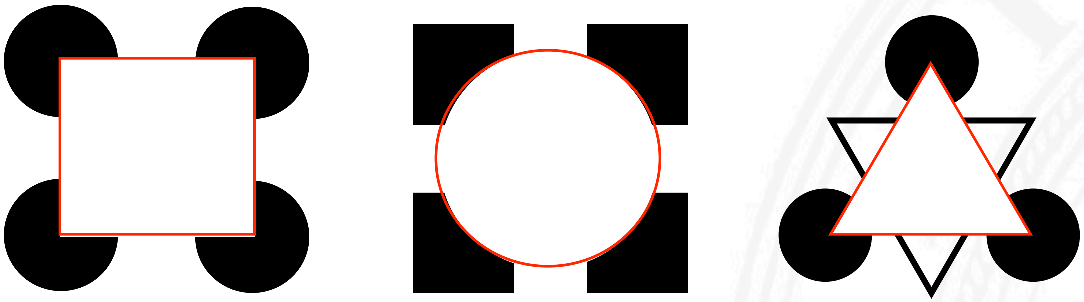
- locally compatible
- globally incompatible

Cognitive Grouping

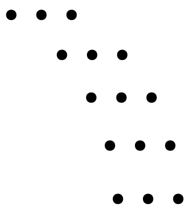
The human cognitive system shows remarkable grouping capabilities



grouping into rows or columns according to a distance criterion



grouping into virtual edges



grouping into virtual motion

It is worthwhile wondering which cognitive grouping rules should also be followed by machine vision

Fitting Straight Lines

Why do we want to discover straight edges or lines in images?

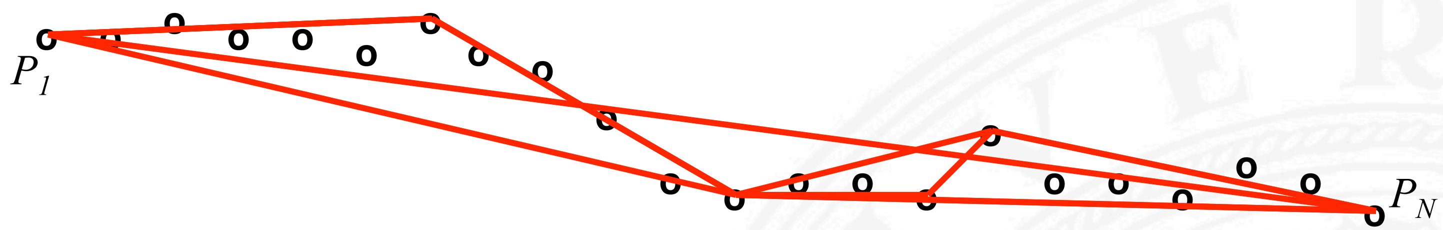
- Straight edges occur abundantly in the civilized world.
- Approximately straight edges are also important to model many natural phenomena, e.g. stems of plants, horizon at a distance.
- Straightness in scenes gives rise to straighness in images.
- Straightness discovery is an example of constancy detection which is at the heart of grouping (and maybe even interpretation).

We will treat several methods for fitting straight lines:

- Iterative refinement
- Mean-square minimization
- Eigenvector analysis
- Hough transform

Straight Line Fitting by Iterative Refinement

Example: Fitting straight segments to a given object motion trajectory

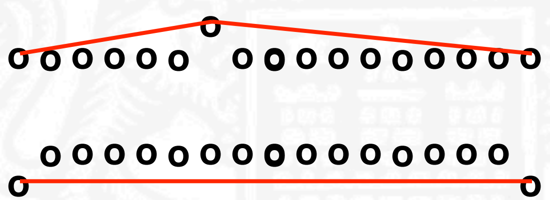


Algorithm:

1. First straight line is P_1P_N
2. Is there a straight line segment P_iP_k with an intermediate point P_j $(i < j < k)$ whose distance from P_iP_k is more than d? If no, then terminate.
3. Segment P_iP_k into P_iP_j and P_jP_k and go to (2).

Advantage: simple and fast

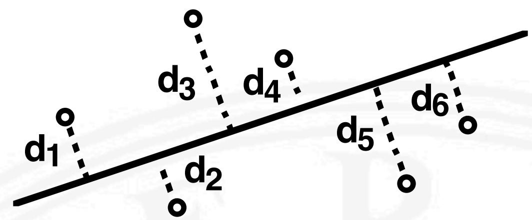
Disadvantages:
- strong effect of outliers
- not always optimal



Straight Line Fitting by Eigenvector Analysis I

Given: $(x_i\, y_i)\quad i = 1 \ldots N$

Wanted: Coefficients $c_0,\, c_1$ for straight line $y = c_0 + c_1 x$ which minimizes $\Sigma\, d_i^2$



The optimal straight line passes through the mean of the given points. Why?

Let (x´y´) be a coordinate system with the x´ axis parallel to the optimal straight line.

- optimal straight line $x' = x_0'$
- error $\Sigma\, d_i^2 = \Sigma\,(x_i' - x_0')^2$
- condition for optimum $\delta/\delta x_0\, \{\Sigma\,(x_i' - x_0')^2\} = -2\, \Sigma\,(x_i' - x_0') = 0$

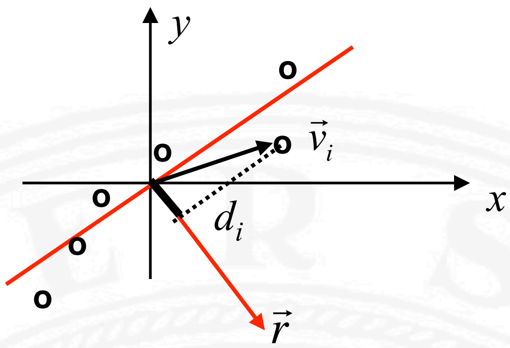
$x_0' = 1/N\, \Sigma\, x_i'$

A new coordinate system may be chosen with the origin at the mean of the given points:

$$x_j' = x_j - \frac{1}{N}\sum x_i \quad , \quad y_j' - = y_j - \frac{1}{N}\sum y_i$$

Optimal straight line passes through origin, only direction is unknown.

Straight Line Fitting by Eigenvector Analysis II



After coordinate transformation the new problem is:

Given: points $\vec{v}_i = (x_i \;\; y_i)^T$ *with* $\sum_{i=1}^{N} \vec{v}_i = 0$

Wanted: direction vector $\vec{r}$ which minimizes $\Sigma\, d_i^2$

Minimize

$$d^2 = \sum_{i=1}^{N} (d_i)^2 = \sum_{i=1}^{N} \left(\vec{r}^{\,T} \vec{v}_i\right)^2 = \sum_{i=1}^{N} \left(\vec{r}^{\,T} \vec{v}_i\right)\left(\vec{v}_i^{\,T} \vec{r}\right) = \vec{r}^{\,T} S \vec{r}$$

scatter matrix

Minimization with Lagrange multiplier λ:

$$\vec{r}^{\,T} S \vec{r} + \lambda \vec{r}^{\,T} \vec{r} \rightarrow \min \qquad \text{subject to} \quad \vec{r}^{\,T} \vec{r} = 1$$

Minimizing r is eigenvector of S, minimum is eigenvalue of S.

For a 2D scatter matrix there exist 2 orthogonal eigenvectors:

- r_{min} orthogonal to optimal straight line
- r_{max} parallel to optimal straight line

Straight Line Fitting by Eigenvector Analysis III

Computational procedure:

1. Determine mean of given points: $\vec{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$ $\quad \mu_x = \frac{1}{N}\sum x_i \quad , \quad \mu_y = \frac{1}{N}\sum y_i$

2. Determine scatter matrix:

$$S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} = \begin{pmatrix} \sum (x_i - \mu_x)^2 & \sum (x_i - \mu_x)(y_i - \mu_y) \\ \sum (x_i - \mu_x)(y_i - \mu_y) & \sum (y_i - \mu_y)^2 \end{pmatrix}$$

3. Determine maximal Eigenvalue $\quad \lambda_{\max} = \max\{\lambda_1, \lambda_2\}$

$$\lambda_{1,2} = \frac{S_{11} + S_{22}}{2} \pm \sqrt{\left(\frac{S_{11} + S_{22}}{2}\right)^2 - |S|}$$

4. Determine direction of eigenvector corresponding to λ_{max}

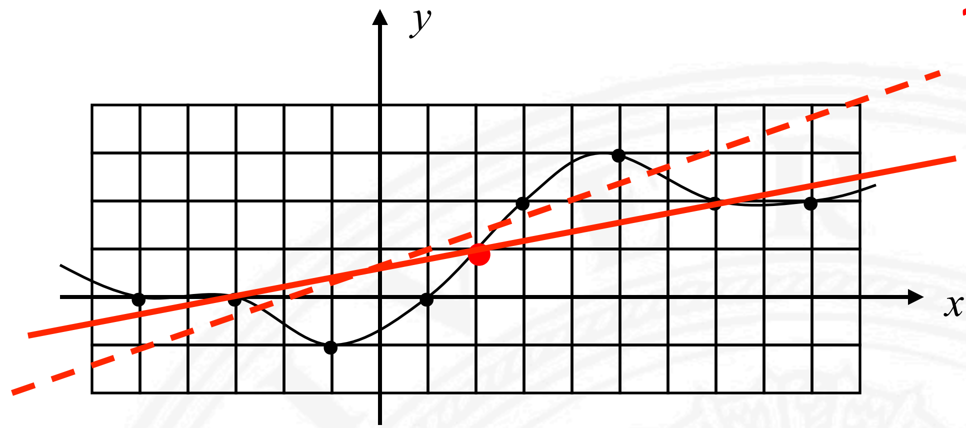
$S_{11} r_x + S_{12} r_y = \lambda_{\max} r_x$ by definition of eigenvector → r_y / r_x

5. Determine optimal straight line:

$$(y - \mu_y) = (x - \mu_x)\frac{r_y}{r_x} = (x - \mu_x)\frac{(\lambda_{\max} - S_{11})}{S_{12}}$$

Example for Straight Line Fitting by Eigenvector Analysis

What is the best straight-line approximation of the contour?



Given points: { (-5 0) (-3 0) (-1 -1) (1 0) (3 2) (5 3) (7 2) (9 2) }

Center of gravity: $m_x = 2 \quad m_y = 1$

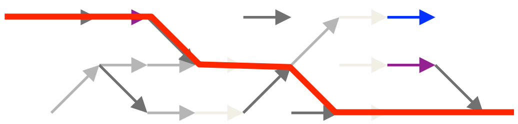
Scatter matrix: $S_{11} = 168 \;,\; S_{12} = S_{21} = 38 \;,\; S_{22} = 14$

Eigenvalues: $\lambda_1 = 176.87 \;,\; \lambda_2 = 5.13$

Direction of straight line: $r_y / r_x = 0.23$

Straight line equation: $y = 0.23\,x + 0.54$

Grouping by Search



What is the "best path" which could represent a boundary in a given field of edgels?

The problem can be formulated as a search problem:

- What is the best path from a starting point to an end point, given a cost function $c(x_1, x_2, \ldots, x_N)$?
- The variables $x_1 \ldots x_N$ are decision variables whose values determine the path.

Unfortunately, the total cost $c(x_1, \ldots, x_N)$ is in general not minimized by local minimal cost decisions min $c(x_i)$, e.g. following the path of maximal edgel strength.

Hence search for a global optimum is necessary, e.g.

- Dynamic Programming
- A* search
- Hill climbing

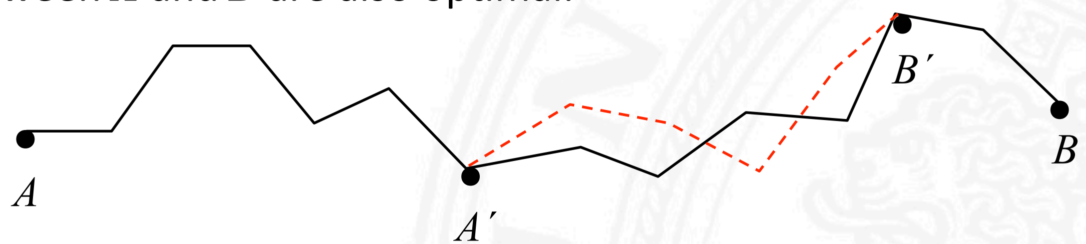
Dynamic Programming I

Dynamic Programming is an optimization method which can be applied if the global cost $c(x_1, x_2, \ldots, x_N)$ obeys the principle of optimality:

If $a_1, a_2, \ldots, a_N$ **minimize** $c(x_1, x_2, \ldots, x_N)$**,**
then $a_{i+1}, a_{i+1}, \ldots, a_{k-1}$ **minimize** $c(a_1 \ldots a_i, x_{i+1}, x_{i+2}, \ldots, x_{k-1}, a_{k \ldots} a_N)$

Hence, for a globally optimal path every subpath has to be optimal.

Example: In street traffic, an optimal path from A to B usually implies that all subpaths from A' to B' between A and B are also optimal.



- Dynamic Programming avoids cost computations for all value assignments for $x_1, x_2, \ldots, x_N$.
- If each x_i, $i = 1 \ldots N$, has K possible values, only $N \times K^2$ cost computations are required instead of K^N.

Dynamic Programming II

Suppose $c(x_1, x_2, \dots, x_N) = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{N-1}, x_N)$, then the optimality principle holds.

Dynamic Programming:

Step 1: Minimize		$c(x_1, x_2)$ over x_1	→	$f_1(x_2)$
Step 2: Minimize	$f_1(x_2) +$	$c(x_2, x_3)$ over x_2	→	$f_2(x_3)$
Step 3: Minimize	$f_2(x_3) +$	$c(x_3, x_4)$ over x_3	→	$f_3(x_4)$
•				
•				
•				
Step N: Minimize	$f_{N-1}(x_N) +$	$c(x_{N-1}, x_N)$ over x_N	→	$f_N = \min c(x_1, x_2, \dots, x_N)$

Example of a cost function for boundary search:

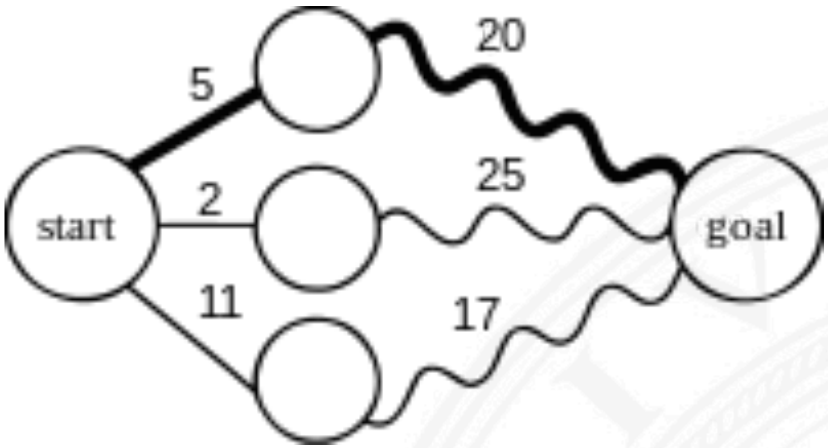
"Punish accumulated curvature and reward accumulated edge strengths"

$$c(x_1,\dots,x_N) = \sum_{k=1\dots N} (1 - s(x_k)) + \alpha \sum_{k=1\dots N-1} q(x_k, x_{k+1})$$

$s(x_k)$ edge strength

$q(x_k, x_{k+1})$ curvature

Dynamic Programming Illustration



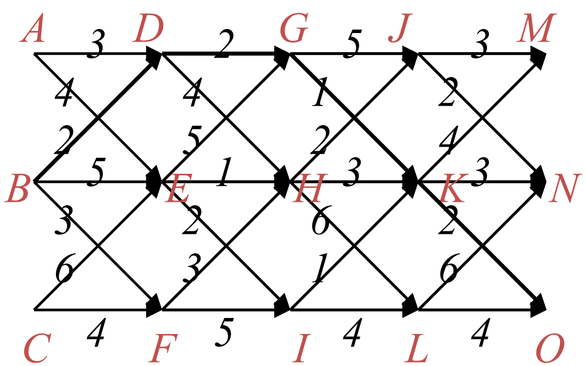
Finding the shortest path in a graph using optimal substructures:

- a straight line indicates a single edge
- a wavy line indicates a shortest path between the two vertices it connects (other nodes on these paths are not shown)
- the bold line is the overall shortest path from start to goal

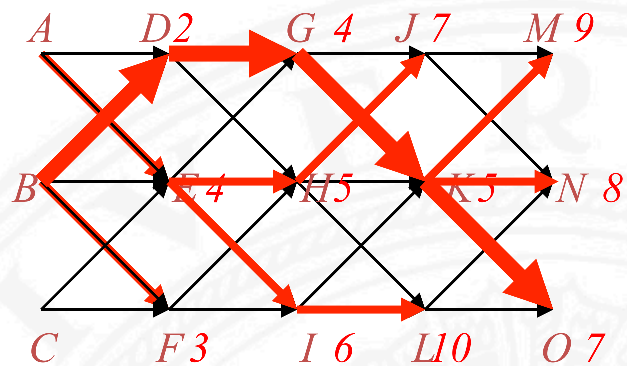
→leads to solving the optimization problem backwards

Dynamic Programming III

<u>Example</u>: Find optimal path from left to right



(costs at edges)

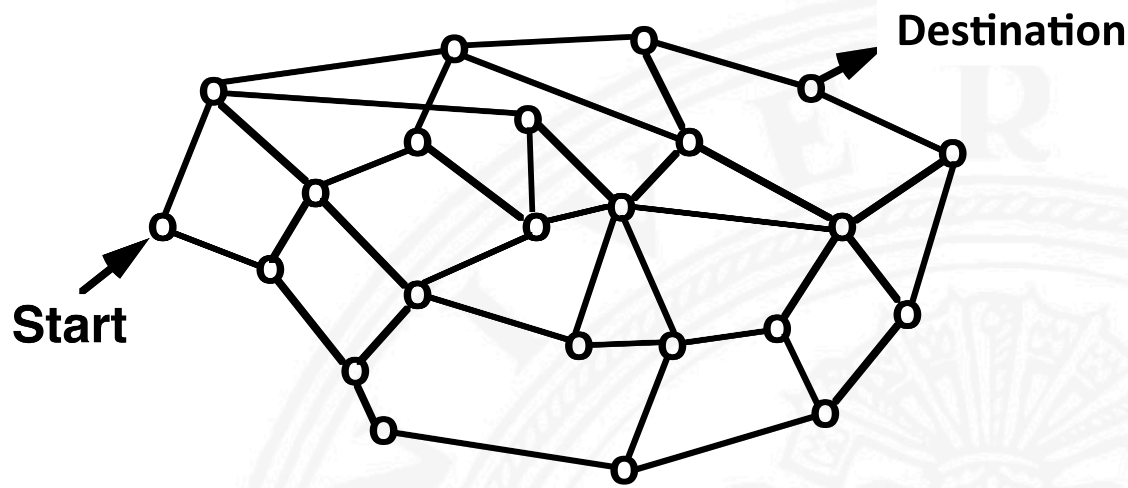


optimal path!

- Find best paths from *A, B, C* to *D, E, F*, record optimal costs at D, E, F
- Find best paths from *D, E, F* to *G, H, I*, record optimal costs at *G, H, I*

  etc.
- Trace back optimal path from right to left

Intelligent Search with the A* Algorithm

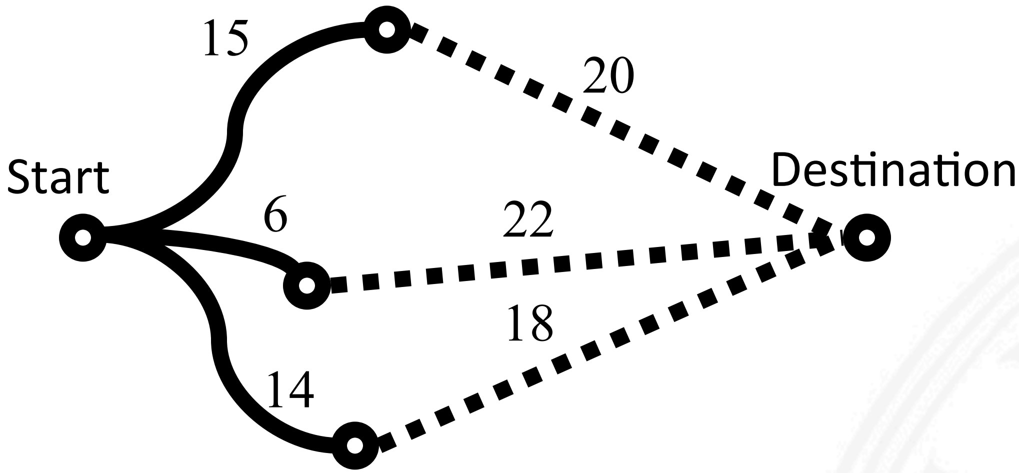


Example:

Find the best connection in local traffic

- each node is a transfer location
- each transfer costs some time
- each edge represents one or more traffic lines
- each traffic line takes a certain time of travel

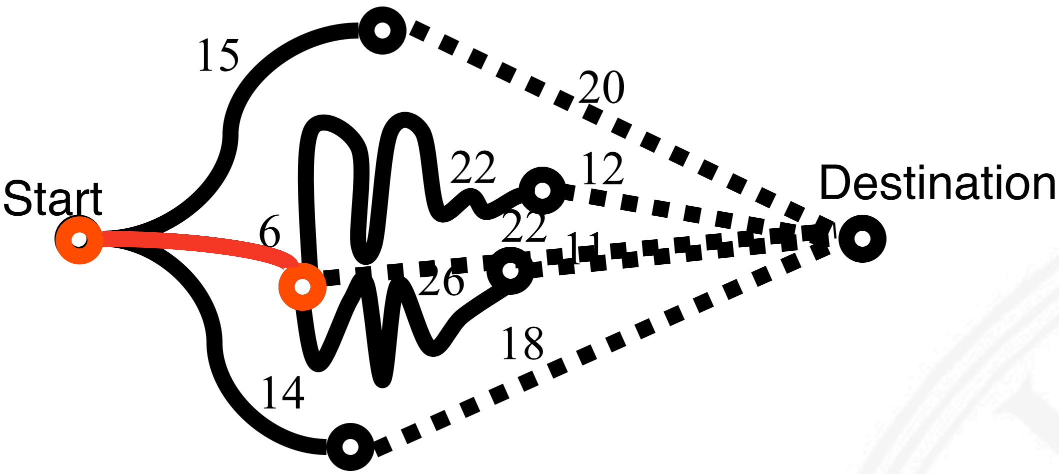
1. Search Step



Path	Estimated Costs
Path 1	15 + 20 = 35
Path 2	6 + 22 = 28
Path 3	14 + 18 = 32

- Determine alternative routes to the next branching points
- Determine costs for alternative routes to the next branching points
- Estimate remaining costs
- Determine estimated total costs

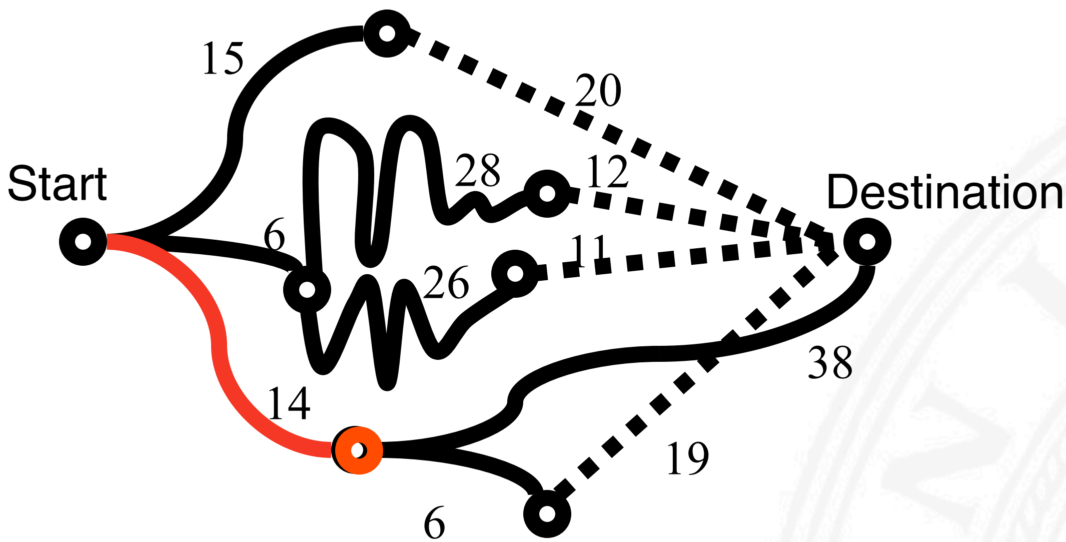
2. Search Step



Path	Estimated Costs
Path 1	15 + 20 = 35
Path 3	14 + 18 = 32
Path 4	6 + 22 + 12 = 40
Path 5	6 + 26 + 11 = 43

- Follow path with least estimated total costs
- Determine alternative routes to the next branching points
- Determine costs for alternative routes to the next branching points
- Estimate remaining costs
- Determine estimated total costs

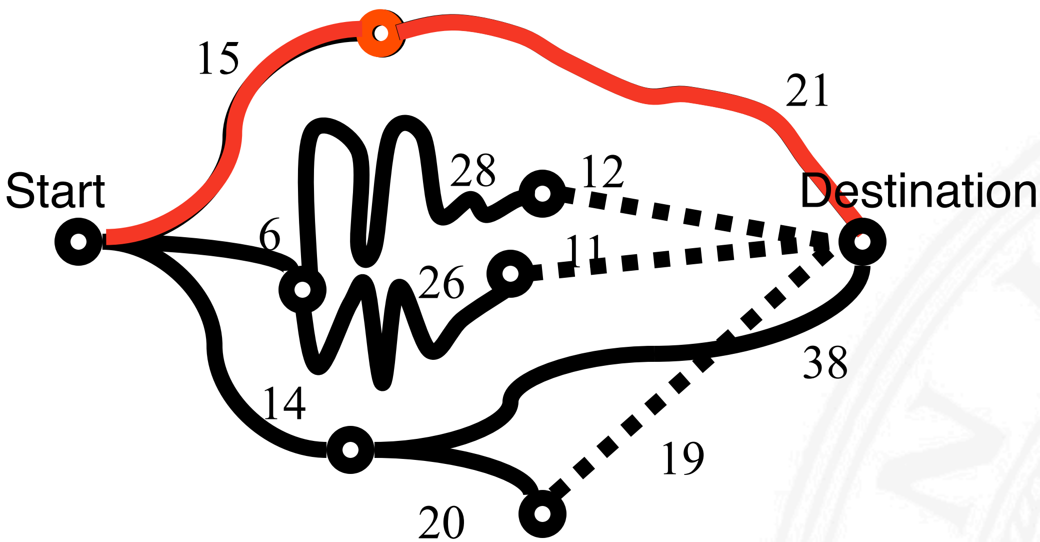
3. Search Step



Path	Estimated Costs
Path 1	15 + 20 = 35
Path 4	6 + 22 + 12 = 40
Path 5	6 + 26 + 11 = 43
Path 6	38
Path 7	14 + 6 + 19 = 39

Carry out the same steps as in Search Step 2, here for Path 3

4. Search Step



Path	Estimated Costs
Path 4	$6 + 22 + 12 = 40$
Path 5	$6 + 26 + 11 = 43$
Path 6	38
Path 7	$14 + 6 + 19 = 39$
Path 8	$15 + 21 = 36$

Carry out the same steps as in Search Step 3, here for Path 1

Path 8 is the shortest path.