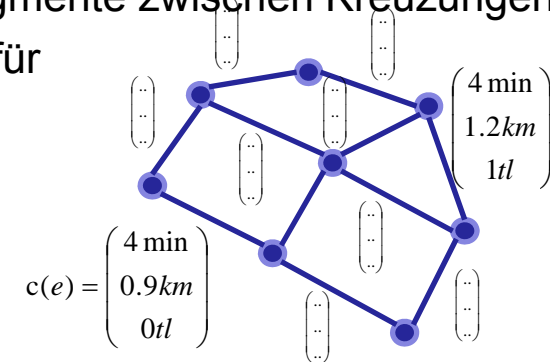


4.5 Routen-Suche in Multiattributs-Verkehrsnetzwerken

– Gegeben:

- Verkehrsnetzwerk
- Straßensegmente haben mehrere Kosten-Attribute, wie z.B.:
Weglänge, Fahrzeit, Anzahl der Ampeln, Benzinverbrauch (abhängig vom Fahrzeugtyp), ...
- Graph $G(V, E, c)$:
 - V : Menge von Knoten \leftrightarrow Kreuzungen, Sackgassen
 - $E \in V \times V$: Menge von Kanten \leftrightarrow Strassensegmente zwischen Kreuzungen
 - $c: E \rightarrow \mathbb{R}^d$: Kostenfunktion \leftrightarrow Kostenvektor für Kanten mit d Kostenattributen
- Annahmen:
 - Keine **negativen** Kosten (**Warum?**)
 - Alle Kostenattribute sind additiv
(Gilt für Kosten eines Pfades (Route))



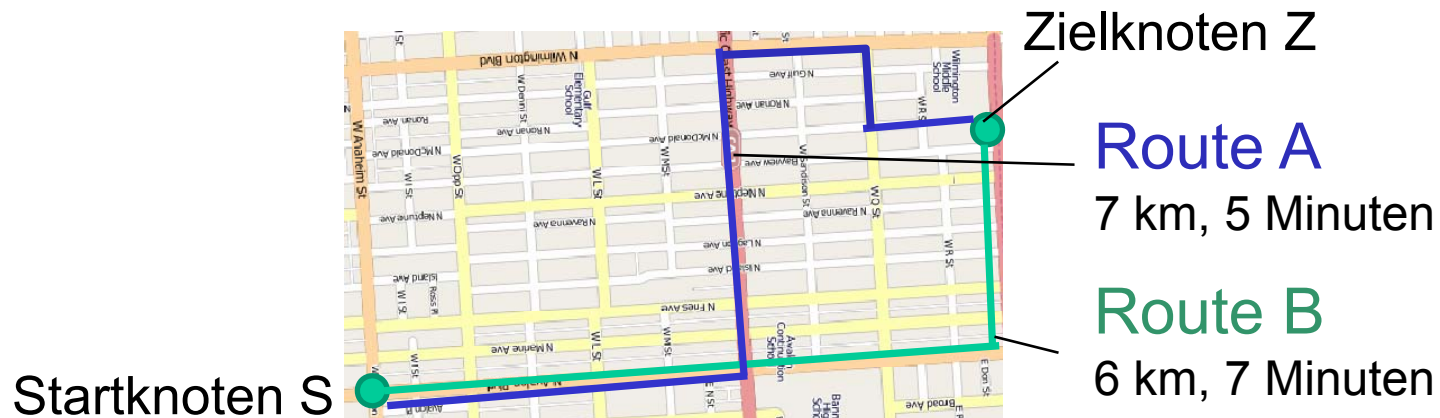
- Route:
 - $p = \langle v_1, \dots, v_n \rangle$ mit $(v_i, v_{i+1}) \in E, \forall 1 \leq i < n$
 - ohne Zyklus, d.h. $\forall 1 \leq i < n, \forall 1 \leq j < n, i \neq j, v_i \neq v_j$
 - Kosten einer Route: $\text{cost}(p) = \sum_{e \in p} c(e) = \begin{pmatrix} \text{cost}(p)_1 \\ \vdots \\ \text{cost}(p)_d \end{pmatrix}$
- Gesucht:
 - “Bester” Pfad von S nach Z



:= Pfad mit minimalen Kosten $\sum_{i=1}^d w_i \cdot \text{cost}(p)_i$?

– Problem

- Mehrere (Pfad-)Attribute (Kostenattribute) zu berücksichtigen (i.d.R. eine Auswahl an Kostenattribute)



Welcher Pfad ist optimal A oder B?

- Kostenattribute schwierig zu vergleichen:
#Ampeln \leftrightarrow Strecke?
- Präferenzen sind abhängig vom Benutzer und Anwendung:
Geschäftlich oder privat unterwegs?
- Geeignete Gewichtung der Attribute schwer zu ermitteln

– Ansatz:

Ausgabe aller **pareto-optimaler** Pfade → **Route Skyline**

– Anfrage:

- Finde die Menge RS alle Pfade zwischen S und Z, sodaß für jeden Pfad $p \in RS$ kein Pfad p' zwischen S und Z existiert mit der folgenden Eigenschaft:

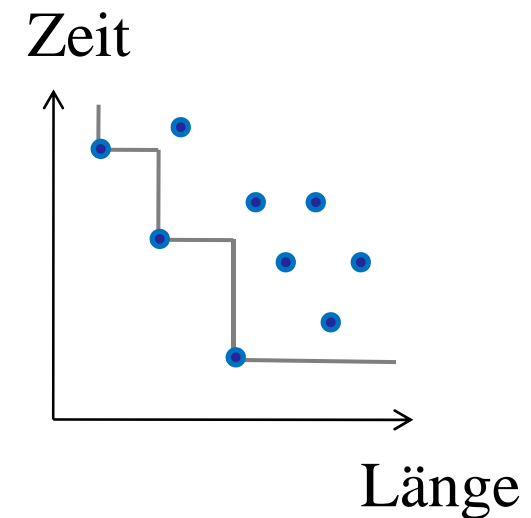
$$\forall 1 \leq i \leq d : \text{cost}(p')_i \geq \text{cost}(p)_i$$

$$\exists 1 \leq i \leq d : \text{cost}(p')_i > \text{cost}(p)_i$$

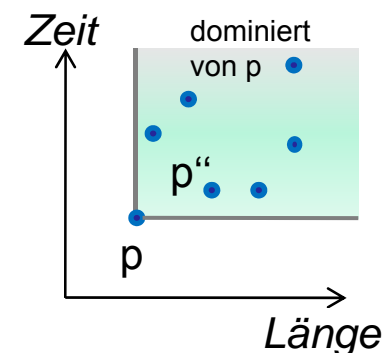
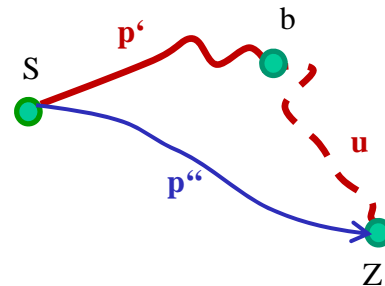
- Man spricht: p wird **nicht** von p' dominiert

– Probleme:

- Anzahl potentieller Pfade extrem hoch
 - Materialisierung der Kosten aller Pfade extrem teuer
- => Skylineanfragemethoden für Vektorobjekte nicht (effizient) anwendbar



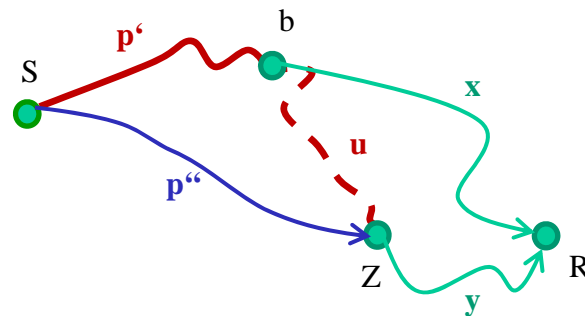
- Effiziente Route-Skyline Berechnung: [KRS 10]
- Idee:
 - Graphdurchlauf bei Startknoten S starten analog zu Dijkstra/A*-Suche
 - Erweitere eine Route p' falls p' Teilroute einer Skylineroute p sein kann, d.h. erweitere p' nicht wenn abgeschätzt werden kann dass p von einer anderen Route p'' dominiert wird
 - Stoppe falls keine Route mehr erweitert werden kann
- Gegeben eine Teilroute $p' = \langle S, \dots, b \rangle$,



- Wie kann man abschätzen ob p' Teil einer Skylineroute ist? => **Pruningkriterien**

– Pruningkriterium I: Pruning mittels Vorwärtsabschätzung

- Basiert auf Kostenabschätzung mittels Referenzknoten (siehe Kap. zu Graph Embedding)
- Schätze Gesamtkosten (Kostenvektor) $\text{cost}(p)$ einer potentiellen Route (über einen Referenzknoten R) ab
- Falls bereits ein Pfad p'' existiert, der den Vektor $\text{cost}(p)$ dominiert, dann kann p' nicht zu einer Skylineroute erweitert werden



$$\text{cost}(u)_i \geq |\text{cost}(x)_i - \text{cost}(y)_i|$$

- Funktioniert solange die Kostenabschätzung die untere Schrankeneigenschaft erfüllt

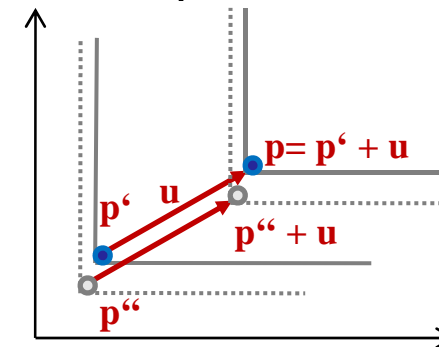
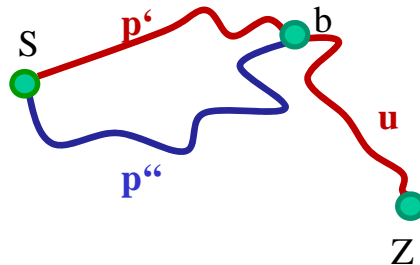
– Pruningkriterium II: Pruning unter Verwendung der Monotonieeigenschaft der Dominanzbeziehung

Theorem:

Gegeben sei eine Skyline Route $p = \langle S, \dots, b, \dots, Z \rangle$, d.h. p wird von keiner anderen Route zwischen S und Z dominiert. Dann wird jede Teilroute $p' = \langle S, \dots, b \rangle$ von p von keiner anderen Teilroute $p'' = \langle S, \dots, b \rangle$ dominiert.

Beweis:

Annahme es gibt eine Route $p'' = \langle S, \dots, b \rangle$, die die Teilroute $p' = \langle S, \dots, b \rangle \neq p''$ von p dominiert. Dann würde die Erweiterung von p'' über die Teilroute $u = \langle b, \dots, Z \rangle$ von p zu einer Route führen, die die Route p dominiert \Rightarrow Widerspruch zur Annahme daß p Skylinerroute ist.



– Skyline-Route-Algorithmus:

- Idee: wie A*-Suche, aber verwalte bei jedem Knoten n eine Skyline (Skyline bzgl. aller Pfade $p' = \langle S, \dots, n \rangle$)

- Algorithmus:

Input: Start S, Ziel Z, Graph(V,E,L) (mit Embedding)

Output: alle Skyline-Routen zwischen S und D

initialisiere Routen-Heap (queue);

queue.insert(S);

while (queue is not empty)

 aktRoute = queue.top();

// erweitere alle Skyline Routen von Knoten aktNode

 cand = extend(aktRoute.last_node.SkR);

for all $c \in \text{cand}$

if $\exists p \in Z. \text{SkR}, p$ dominiert $\text{cost}_{\text{est}}(c)$, **then** lösche c aus cand;

else

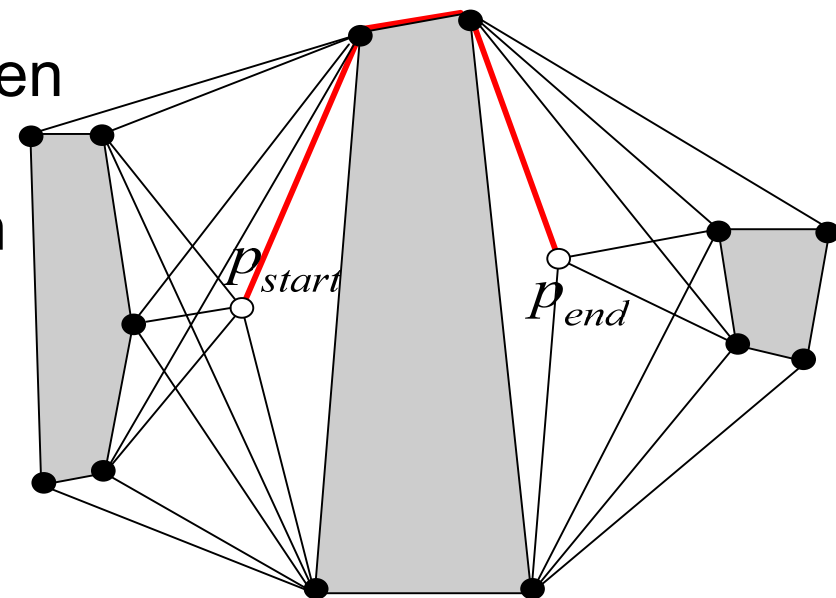
 update(c.last_node.SkylineRoutes,c);

 queue.insert(c);

report(Z.SkR);

4.6 Anfragen im Euklidischen Raum mit Hindernissen [ZPMZ04]

- Ziel: Anfragebearbeitung (z.B. NN-Anfragen) im euklidischen Raum bei Anwesenheit von Hindernissen (Häuser, Seen, ...)
- Mögliche Pfade (mit kürzesten Distanzen) können mittels Sichtbarkeitsgraphen modelliert werden.
- Für die Ermittlung der kürzesten Distanz zwischen zwei Punkten können Graphalgorithmen (z.B. Dijkstra) verwendet werden.



4.6.1 Sichtbarkeitsgraph

- Dient der Modellierung der kürzesten Pfade zwischen zwei beliebigen Punkten im Raum mit Hindernissen
- Hindernisse werden als Polygone modelliert
- Knoten: Eckpunkte von Hindernissen, Objekte (p_i)

Gilt nur im 2-dimensionalen Raum !!!

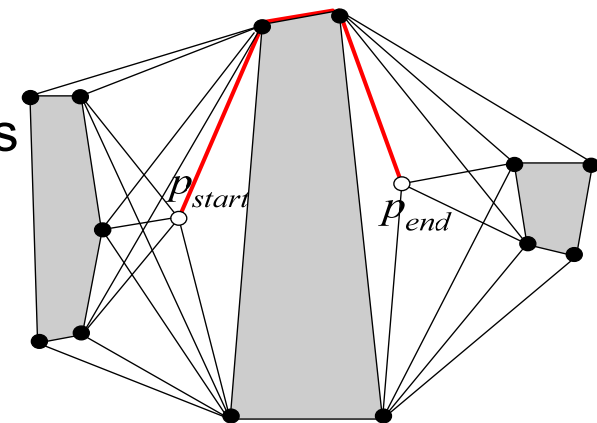
- Kanten: Verbinden Knoten die gegenseitig „sichtbar“ sind (d.h. dürfen kein Hindernis kreuzen)

- Ermittlung des Sichtbarkeitsgraphen:

- Suche für jeden Eckpunkt eines Polygons die Eckpunkte anderer Polygone die direkt „sichtbar“ sind.

(siehe Computational Geometry [BKOS97])

- Naive Suche: $O(n^3)$
- Rotational plane-sweep: $O(n^2 \log n)$ [SS84], $O(n^2)$ [W85, AGH86]

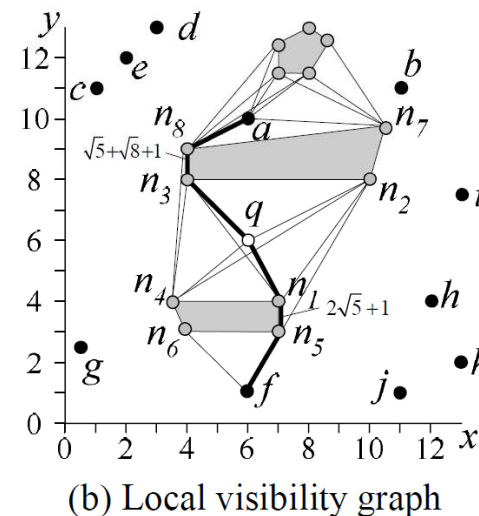
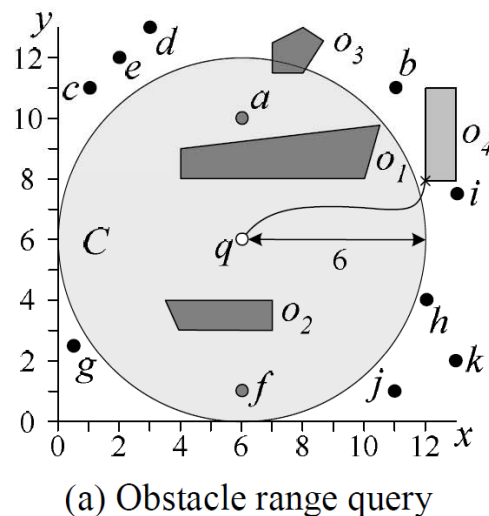


4.6.2 Idee bei Nachbarschaftsanfragen in Räumen mit Hindernis

- Berechnung des gesamten Sichtbarkeitsgraphen zu aufwändig.
- Deshalb Reduktion auf relevante Teilgraphen, bei denen nur relevante Hindernisse betrachtet werden.
- Welche Hindernisse sind für die Anfragebearbeitung relevant?
- Euklidische Distanz zwischen zwei Punkten bildet eine untere Schranke für die Distanz in Räumen mit Hindernissen.
- Ermittlung der relevanten Hindernisse über die Verwendung der Euklidischen Distanz (Filterschritt)

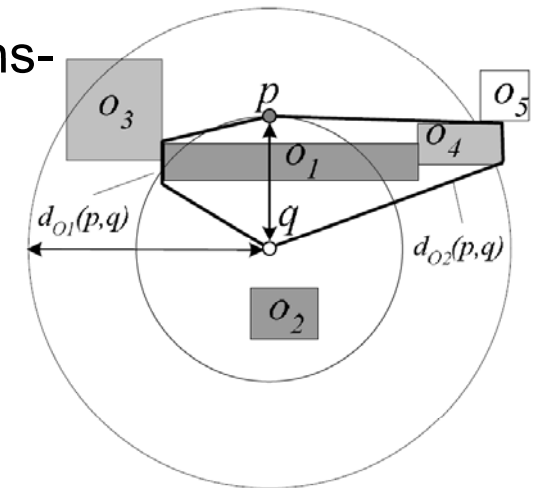
4.6.2 Bereichsanfrage mit Hindernissen

1. Euklidische Bereichsanfrage auf Ergebniskandidaten P von Anfragepunkt q mit Distanz e (filter)
2. Suche von für die Anfrage relevanten Hindernissen O . Diese schneiden die Fläche der Bereichsanfrage.
3. Lokalen Sichtbarkeitsgraph über P und O aufbauen
4. False Hits aus P über Sichtbarkeitsgraph entfernen



4.6.2 NN-Anfrage mit Hindernissen

- Entspricht dem IER-Algorithmus, wobei jede Runde der entsprechende Sichtbarkeitsgraph erzeugt werden muss, um $\text{dist}_{\text{NET}}(p,q)$ zu berechnen.
- Berechnung der Netzwerkdistanz $\text{dist}_{\text{NET}}(p,q)$ über den Sichtbarkeitsgraphen:
 1. Berechne Hindernisse O , die $[p,q]$ schneiden
 2. Berechne den Sichtbarkeitsgraph über $O \cup \{p,q\}$ und daraus $\text{dist}'_{\text{NET}}(p,q)$
 3. Frage alle Hindernisse O an, die die Bereichsanfrage von q bzgl. $\text{dist}'_{\text{NET}}(p,q)$ schneiden
 4. Führe 3. und 4. so lange durch, bis O oder $\text{dist}'_{\text{NET}}(p,q)$ sich nicht mehr verändert.
Es gilt: $\text{dist}_{\text{NET}}(p,q) = \text{dist}'_{\text{NET}}(p,q)$



[HNR68]: Hart, Nilsson, Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In IEEE Transactions of Systems Science and Cybernetics, 1968.

[KRS10]: Kriegel, Renz, Schubert: Route Skyline Queries: A Multi-Preference Path Planning Approach. In Proc. of ICDE, 2010.

[KKKRS08]: Kriegel, Kröger, Kunath, Renz, Schmidt: Efficient Query Processing in Large Traffic Networks. In Proc. of ICDE, 2008.

[PZMT03]: Papadias, Zhang, Mamoulis, Tao. Query Processing in Spatial Network Databases. In Proc. of VLDB, 2003.

[BKOS97] de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O. Computational Geometry. pp. 305-315, Springer, 1997.

[SS84] Sharir, M., Schorr, A. On Shortest Paths in Polyhedral Spaces. STOC, 1984.

[W85] Welzl, E. Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time, Information Processing Letters 20, 167-171, 1985.

[AGHI86] Asano, T., Guibas, L., Hershberger, J., Imai, H. Visibility of Disjoint Polygons. Algorithmica 1, 49-63, 1986.

[ZPMZ04]: Zhang, Papadias, Mouratidis, Zhou. Spatial Queries in the Presence of Obstacles. In Proc. of EDBT, 2004.