In a route search, the user specifies the types of entity she wishes to visit. Yet, only upon arrival at an entity, she knows if the entity satisfies her. Thus, a probably of success in satisfying the user is attached to each entity. The goal is to compute a route that goes via relevant entities of all the specified types while minimizing the overall travel distance.

**1. A user formulates a route search query**

**2. An initial route is computed**

**3. Upon arrival at an entity, the user provides a feedback**

**4. If feedback is negative, the system computes a new route**

**Why is it Difficult?**

- Already in the non-interactive case, computing the shortest route is **NP-hard**.
- What is better, a near object with low rank or a far object with high rank?
- When computing an initial route, how do we take into account all the possible feedbacks?

**Algorithms**

- \( d_w(o_1, o_2) = \frac{p(o_2)}{d(o_1, o_2)} \)
- \( b\text{Greedy}/w\text{Greedy} \) — starting at \( s \), Greedily minimize \( d_w(l, o') + d_w(s, o') + d_w(o', t) \)
- \( w\text{Optimistic} \) — assumes that all the objects are relevant and use a variation of Dijkstra’s algorithm (\( d_w \) is used instead of \( d \)).
- **Minimizing the Expected Distance (MED)** — Aim at selecting the next objects that will minimize the expected length of the route.

**Tests results:**

The length difference (in meters) w.r.t. the “perfect” route.

We used 3 datasets containing low, medium and high probabilities.

**Conclusions:**

- Current technology enables a new generation of routing – namely **Interactive Routing**.
- Three approaches for interactive routing are described.
- **MED** is best at finding the actual shortest routes (Greedy is the fastest).