

Online Motion Planning MA-INF 1314

Graphexploration/Marker

Elmar Langetepe
University of Bonn

Repetition: CFS Algorithm Invariants Lemma

Execution CFS–Algorithm, properties hold: ■

- i) Any incomplete vertex belongs to a tree in \mathcal{T} . ■
- ii) There is always an incomplete vertex with $v \in V^*$ with $d_{G^*}(s, v) \leq r$, until $G^* \neq G$. ■
- iii) For any chosen root vertex s_i : $d_{G^*}(s, s_i) \leq r$. ■
- iv) After pruning T_i is fully explored by DFS. All trees $T \in \mathcal{T}$ have size $|T| \geq \frac{\alpha r}{4}$. ■
- v) All trees $T \in \mathcal{T}$ are disjoint (w.r.t. edges) ■

Proof: i) and v) simply hold by construction ■

Rep Analysis Theorem/Corollary

CFS-Algorithm known depth r $(4 + \frac{8}{\alpha})$ -competitive/cost

$\Theta(|E| + |V|/\alpha)$. ■

über Teilbäume T_R ■

- Subtree T_R , cost ■
- $K_1(T_R)$: path from s to s_i in G^* ■
- $K_2(T_R)$: DFS, $K_3(T_R)$: bDFS (Graph!) ■
- $\sum_{T_R} K_3(T_R) \leq 2 \cdot |E|$ bDFS global ■
- $\sum_{T_R} K_2(T_R) = \sum_{T_R} 2 \cdot |T_R| \leq 2 \cdot |E|$, DFS, disjoint ■
- $\sum_{T_R} K_1(T_R) \leq \sum_{T_R} 2r \leq \frac{8}{\alpha} \sum_{T_R} |T_R| \leq \frac{8}{\alpha} |E|$ ■

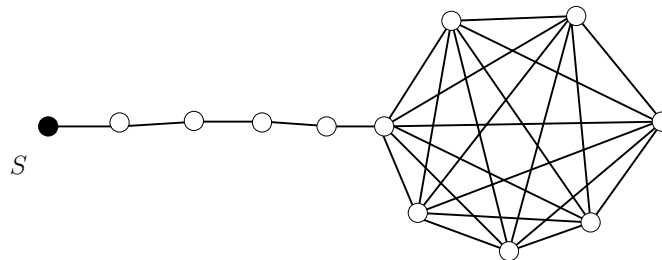
Rep: Graphexploration, unknown depth r

- Doubling-Heuristic: $O(|E| + \log r|V|)$ steps Schritte, **Corollary**■
- Adjust prune/explore with current value $d_{G^*}(s, s_i)$ ■
- **prune** $(T_i, s_i, \frac{\alpha d_{G^*}(s, s_i)}{4}, \frac{9\alpha d_{G^*}(s, s_i)}{16})$ ■
- **explore** $(\mathcal{T}, T_i, s_i, (1 + \alpha)d_{G^*}(s, s_i))$ ■
- **Lemma iv)**: Rest of T_i fully explored by DFS, all $T \in \mathcal{T}$ have size $|T| \geq \frac{d_{G^*}(s, T)\alpha}{4}$ ■
- **Theorem/Corollary** CFS–Algorithm unknown depth R is $(4 + \frac{8}{\alpha})$ –competitive/has cost $\Theta(|E| + |V|/\alpha)$ ■

Look-ahead $\alpha \cdot r$ necessary

Lower bound $\Omega(|E|^{1+\epsilon})$ Offline accumulator variant, if look-ahead is smaller than linear in r (constant). ■

- $2r$ is not sufficient: At least $2r + 1$! ■
- With $2r + o(r)$ not efficient! (small-o notation!) ■
- Graph: path and clique, beyond linear ■
- Accumulator size $n + f(n)$: $\Omega\left(\frac{n^3}{f(n)}\right)$ Schritte! ■
- $|E| \in C \cdot n^2$, $f(n) = n^{1-\epsilon}$ ■
- Conjecture: $r + o(r)$ is not sufficient for tether variant. Open!! ■



Look-ahead $\alpha \cdot r$ necessary

Lemma For the accumulator variant with accumulator size of $2r + d$ for constant d there are examples where $\Omega(|E|^{\frac{3}{2}})$ exploration steps are necessary. ■

Proof: Blackboard! ■

Note: It can still be competitive! ■

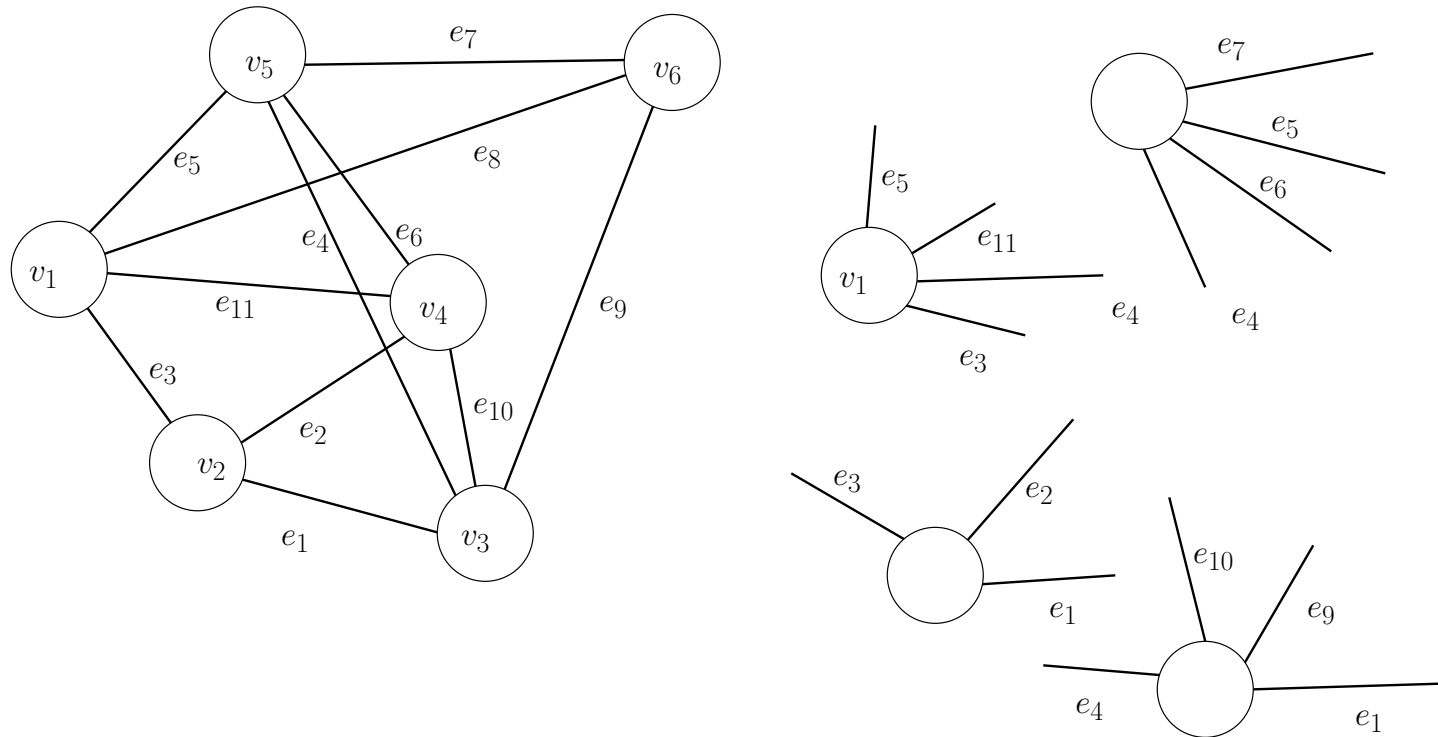
Offline cost?

- Mechanical cost/Computational cost■
- Build the spanning trees■
- Move along the shortest path■
- Merge the trees■
- DFS/bDFS■
- Not all linear■
- Exercise■

Different model

- Vertices/Edges have been marked■
- As a landmark■
- Assume: This is not possible! How to distinguish?■
- Vertices cannot be distinguished immediately!■
- Local order of the edges is given■
- May be not a planar embedding!■
- Given: $G = (V, E, S)$, S cyclic orders!■

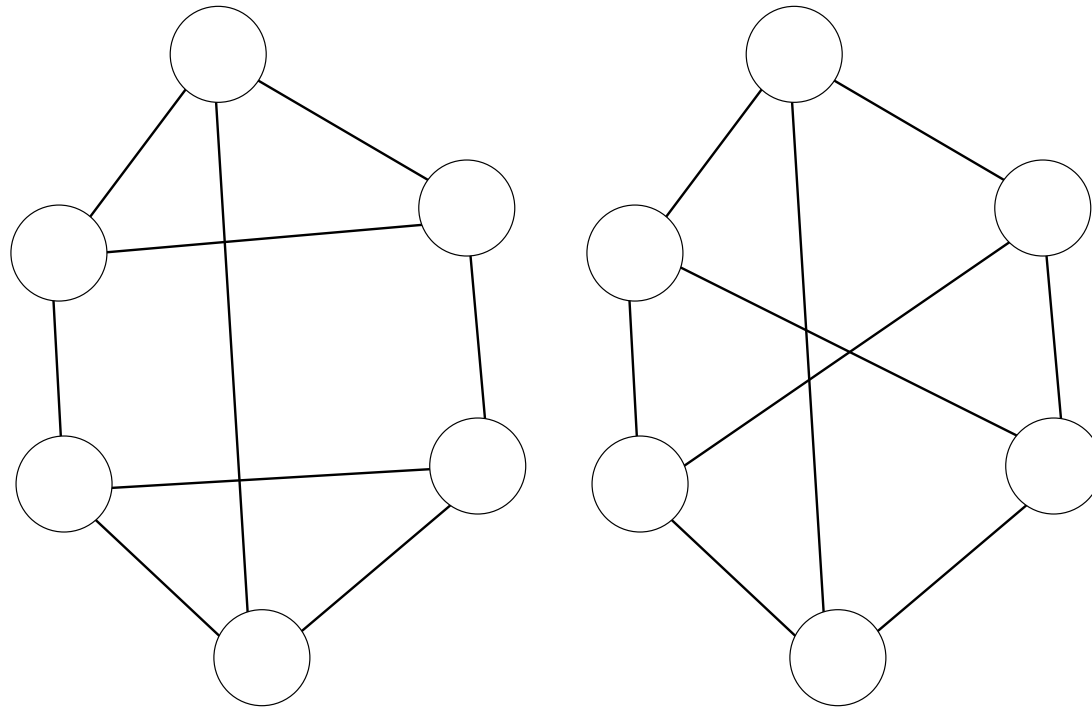
Different model, local order



From different vertices, permutation! Locally fixed

Mapping problem!

- Determine the graph (for navigation!)■
- Store all given information■
- Marker/pebble is necessary■



One-Marker Algorithm (Dudek et al.)

- Maintain known graph S ■
- List L of adjacent unknown edges ■
- Choose edge $e \in L$ from some $b \in S$ ■
- Visits vertex u ■
- Put pebble/marker at u ■
- Search in S from b for the pebble ■
- If marker was not found, add edge(b, u) and vertex u to S ■
- Insert the adjacent edges from u into L ■
- If marker has been found at known vertex $v = u$, try to search for the edge $e = (b, v)$ by the order from b ■
- For this: Place marker onto b , move to b and then in S back to $v = u$ along shortest path ■

- Check the outgoing edges for
- One will be the right one! Update S !
- Pseudocode! Exercise!

Analysis: One-Marker Algorithmus

- Mechanical cost: Number of steps!!
- ● Assumption: No loops!
- Set the marker $O(1)$
- Search for the marker: DFS on vertices $2|V_S|$
- Bring the marker back, move back: $2|V_S|$
- Do this for all possible edges: $O(|E| \times |V|)$

Analysis: One-Marker Algorithm

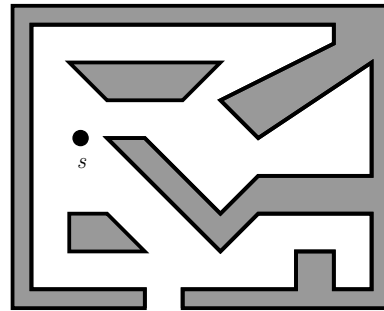
- Computational cost: Offline!■
- Shortest path in graphs■
- Dijkstra: $O(|E_S| + |V_S| \log |V_S|)$ ■
- For any edge■
- $O(|E|^2 + |E||V| \log |V|)$ ■

Graph-exploration

- Labyrinths, grid-graphs, gridpolygons, general graphs
- Graph-exploration: DFS and LB of 2
- Gridpolygons: Simple/general
- SmartDFS $\frac{4}{3}$, LB $\frac{7}{6}$
- STC Alg. $|C| + |B|$
- Tether/Accumulator/Depth variants: $\Theta(|E| + |V|/\alpha)$
- Marker Algorithm
- Online TSP for planar graphs!

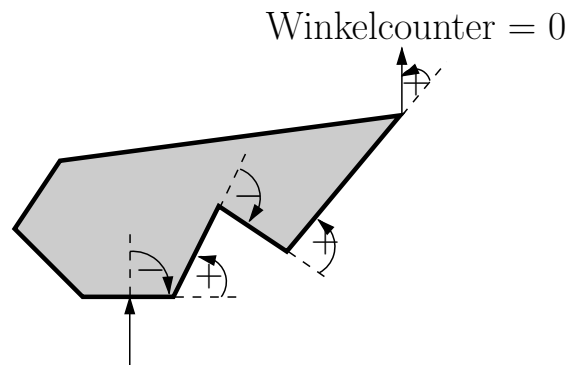
Kap. 2: Polygonal environments

- Set of disjoint simple polygons in the plane einfachen Polygonen
- Boundary polygon
- Different tasks: Searching for a goal/escape from a labyrinth
- Different sensor models
- First: Touch sensor, precise odometrie, escape from a labyrinth



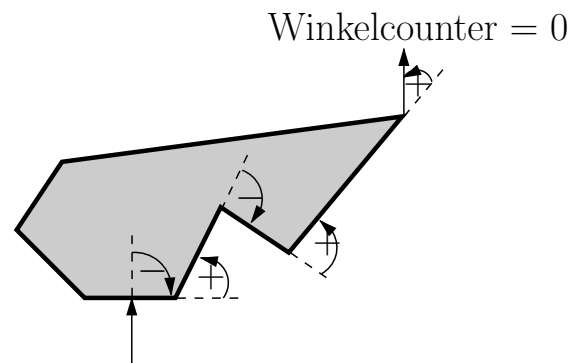
Escape from a labyrinth: Model

- Point-shaped agent
- Touch sensor
- Follow the wall
- Follow a direction (exact)
- Count rotational angles, in total
- No further memory



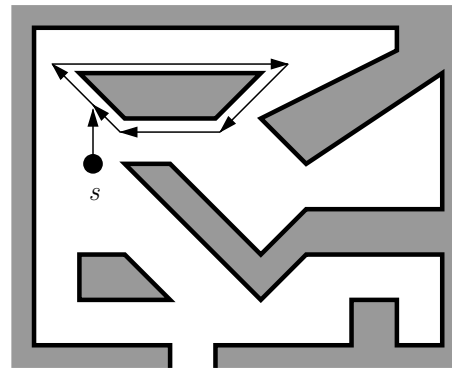
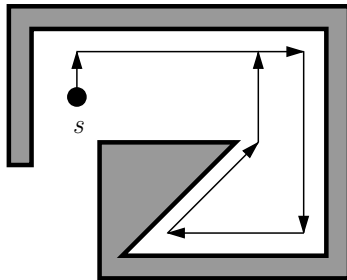
Pledge Algorithm

1. Choose angle φ , rotate agent heading in this direction.■
2. Move into direction φ , until agent reaches the boundary.■
3. Move right and keep in contact with the wall, Left-Hand.■
4. Follow the wall by Left-Hand-Rule and sum up the rotational angles, until the **overall rotational angle** attains value zero, now GOTO (2).■



Pledge Algorithm

- Angular counter mod $2\pi = 0$, not sufficient
- Only Left-Hand-Rule not sufficient
-



Correctness, structural properties, non-negative counter

Lemma The angular counter of the Pledge Algorithm is never positive. ■

Proof: ■

- Zero at the beginning ■
- Zero, when the boundary is left ■
- Right turn after hitting the boundary \Rightarrow negative ■
- Continuous change, zero \Rightarrow movement is possible ■

Correctness, no-success, finite path repeated

Lemma If the agent does not leave the labyrinth, after a while the agent repeatedly follows the same finite path, Π_o , again and again. ■

Proof:■

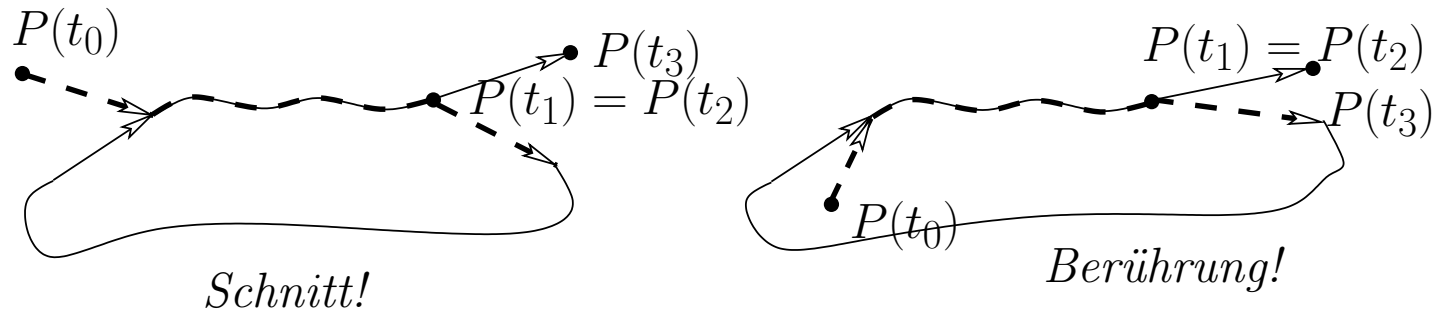
- Path is a polygonal chain■
- Vertices I: Vertices of the polygons■
- Vertices II: Hit-Points on the edges ■
- Correspond to vertices of type I■
- Finite set S of possible vertices of the path■
- The same counter value at the same vertex \Rightarrow the same path again and again■
- Assume: Never the same value■

- Case 1: After a while, keeping on the boundary \Rightarrow always the same path along one polygon ■
- Case 2: Leaving more than $|S|$ times (infinitely often) ■
- \Rightarrow at least twice with the same value 0 at the same vertex, contradiction! ■

Correctness: Π_o no self-intersection

Lemma Assume the agent does not leave the labyrinth by Pledge and let Π_o be the repeated path. Π_o has no self-intersections. ■

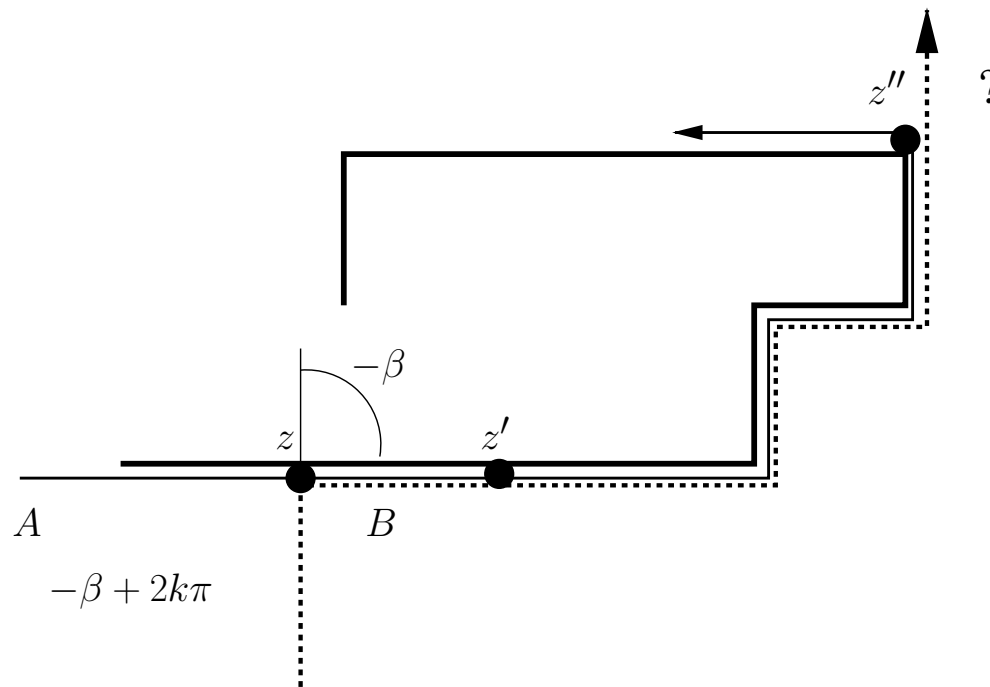
Difference: Intersection/Touching ■



Intersection only at the boundary! All *free paths* run in parallel! ■

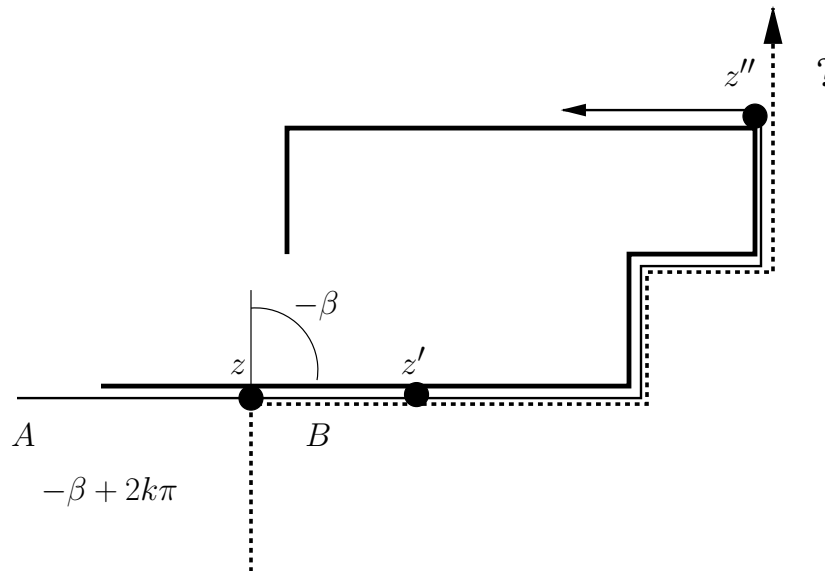
Correctness: Π_0 no self-intersection

- Proof: Ass. Intersection! **Two parts one of which is free, say B**
- Shortly behind z angular counter $C_A(z')$, $C_B(z')$
- $C_B(z') = -\beta$ and $C_A(z') = -\beta + 2k\pi$ for $k \in \mathbb{Z}$



Correctness: Π_o no self-intersection

- $C_B(z') = -\beta$ and $C_A(z') = -\beta + 2k\pi$ for $k \in \mathbb{Z}$
- $k = 0$? A and B are the same! Contradiction!■
- $k > 0$? Lemma, $C_A(z')$ negative■
- Means $k < 0$ and $C_A(p) < C_B(p)$ for all p from z' to z'' ■
- Path B leaves the obstacle first, no intersection!!!■



Correctness proof

Theorem For any labyrinth and any starting position the pledge-algorithm will leave the labyrinth, if this is possible. ■

Proof:■

- Ass.: Agent does not reach the boundary■
- **Lemma** Path Π_o again and again■
- **Lemma** No intersections■
- Orientations of Π_o : 1) cw-order 2) ccw-order■
- 2) $+2\pi$ per full round, finally positive, contradiction■
- Means 1) -2π per full round■
- Remains negative after a while. Moves around obstacle!■
- Orientation: cw-order, Left-Hand \Rightarrow Enclosed!■

Pledge algorithm with sensor errors

- Possible errors? ■
- Left-Hand-Rule, stable! ■
- Counting rotational angles! ■
- Hold the direction in the free space! ■
- For example: Compass! ■
- Full turns ok, but not precisely! ■
- Leave the obstacle slightly too early or too late! ■
- The main direction can be hold! ■
- Still correct? ■

