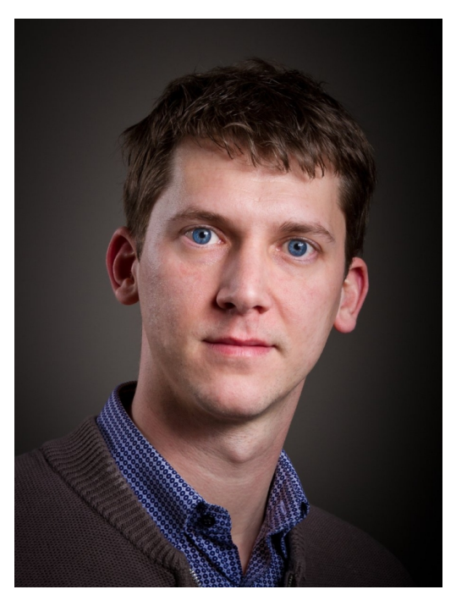
Universiteit Utrecht

Wouter van Toll, Roland Geraerts
W.G.vanToll@uu.nl
uu.nl/staff/WGvanToll

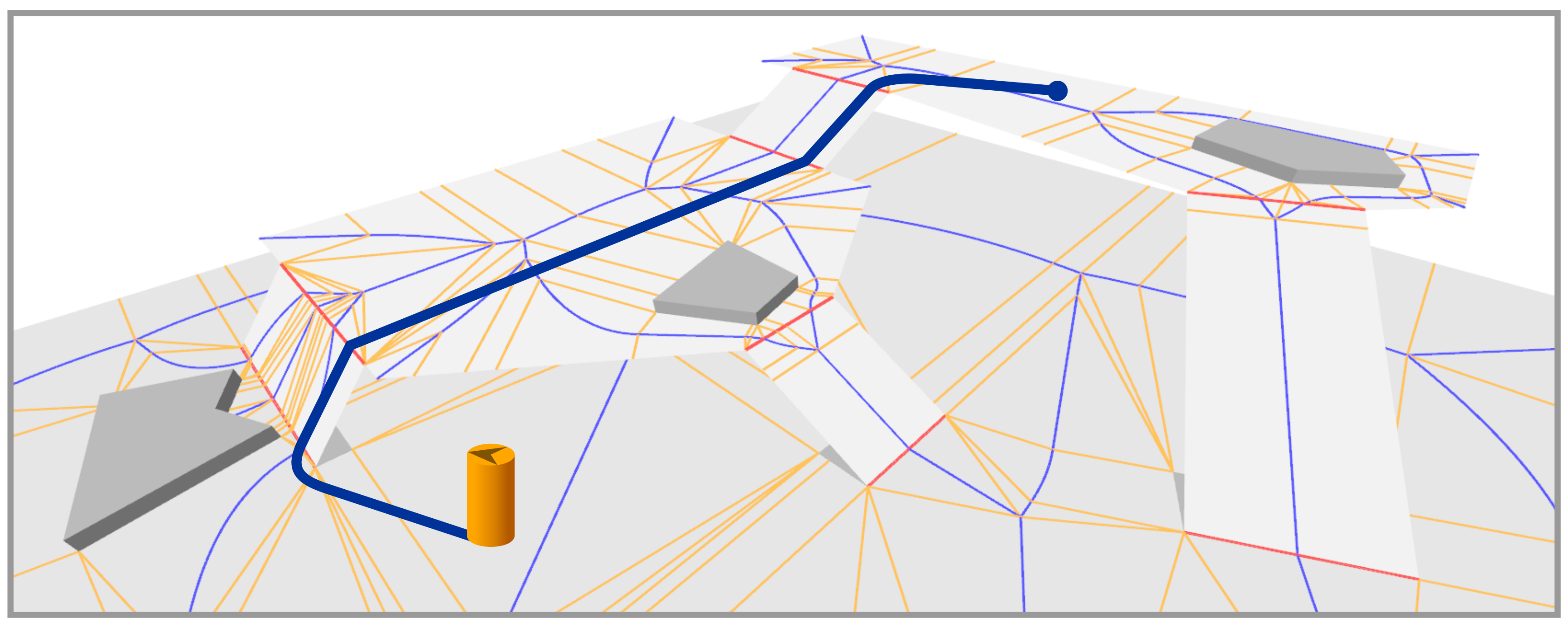
DPA*: Dynamically Pruned A*
for Re-planning in Navigation Meshes



In simulations and games, AI-controlled agents can use the A* search algorithm on a **navigation mesh** to plan paths through an environment. When an obstacle is inserted or deleted, the mesh changes and agents should re-plan their paths. We present DPA*, an extension of A* that **re-plans an optimal path** for an agent based on the old path and the dynamic obstacle. DPA* is fast, memory-friendly, and suitable for real-time **crowd simulation**.

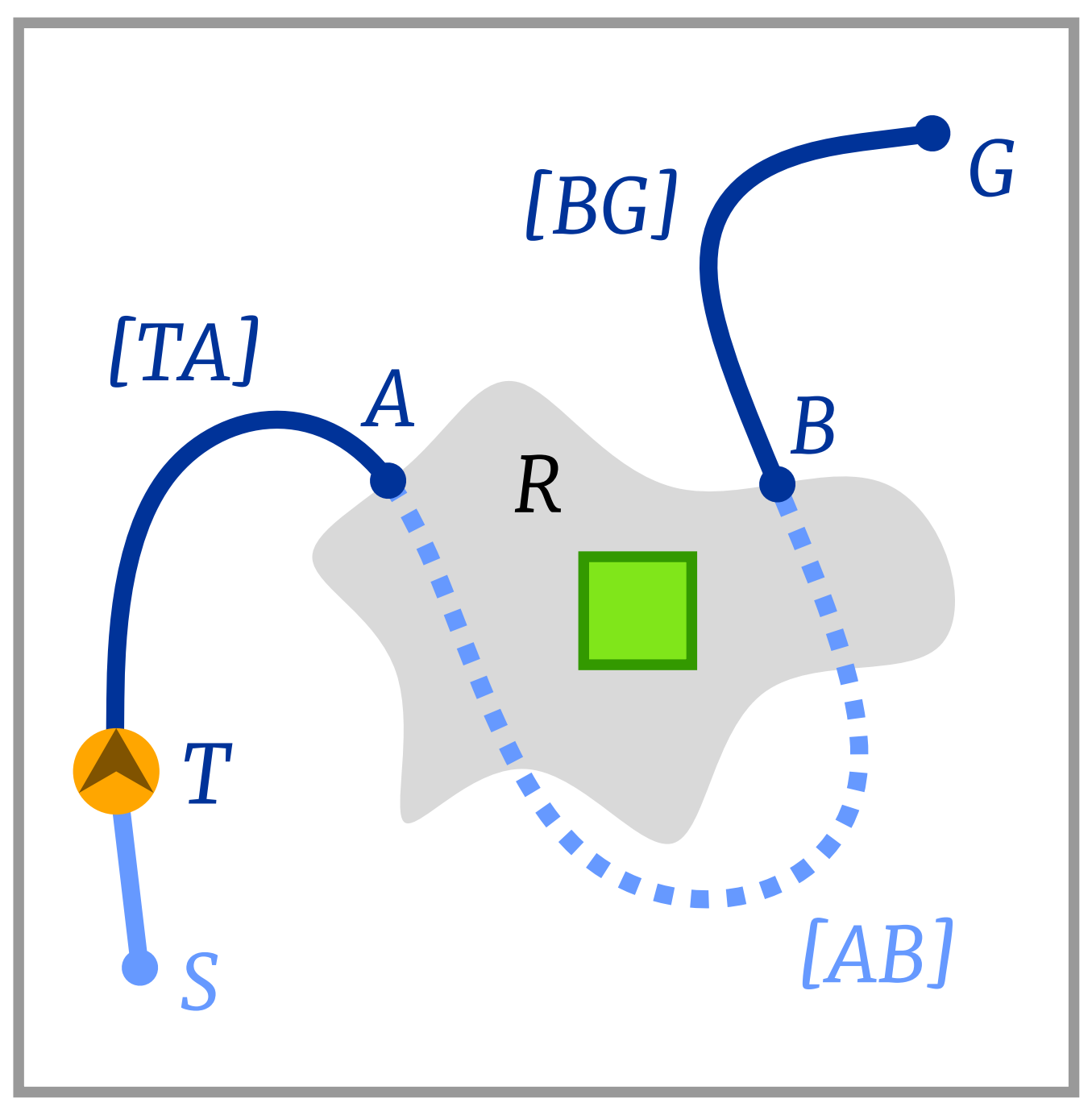
Path Planning in Navigation Meshes

A **navigation mesh** subdivides the walkable space of an environment into regions. An agent uses the **dual graph** of this mesh to plan a path to its goal.



Dynamic Events

When an obstacle (dis)appears, the navigation mesh and its graph **change**. The agent's path may now be **invalid**, or a **better path** may have appeared.



When finding a new optimal path, parts of the old path can typically be **re-used**.

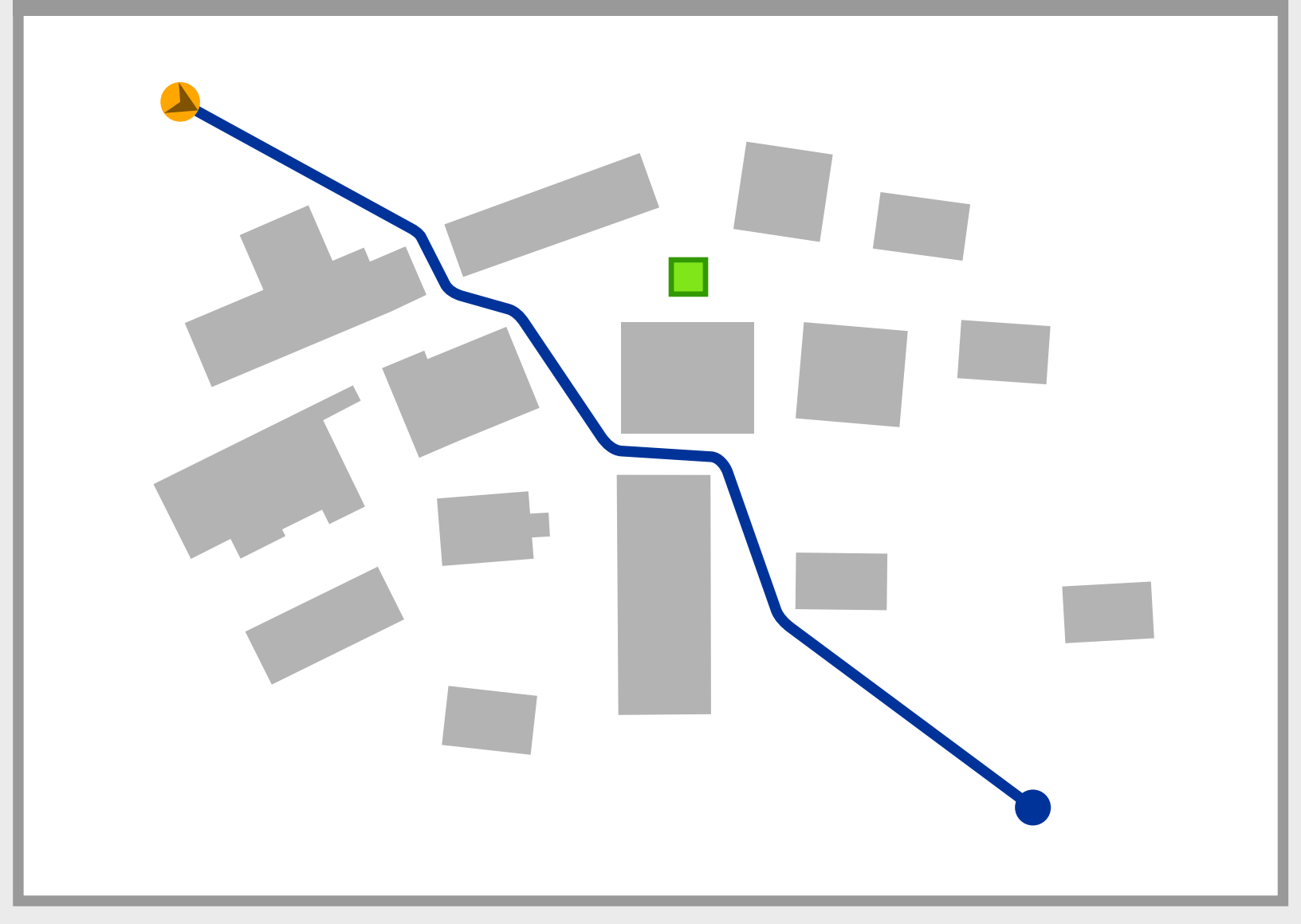
Most re-planning algorithms remember the entire previous search and are too **memory-intensive** for crowds.

DPA* uses only the old path and its relation to the **affected region** *R*. It always computes an optimal path.

Dynamically Pruned A*

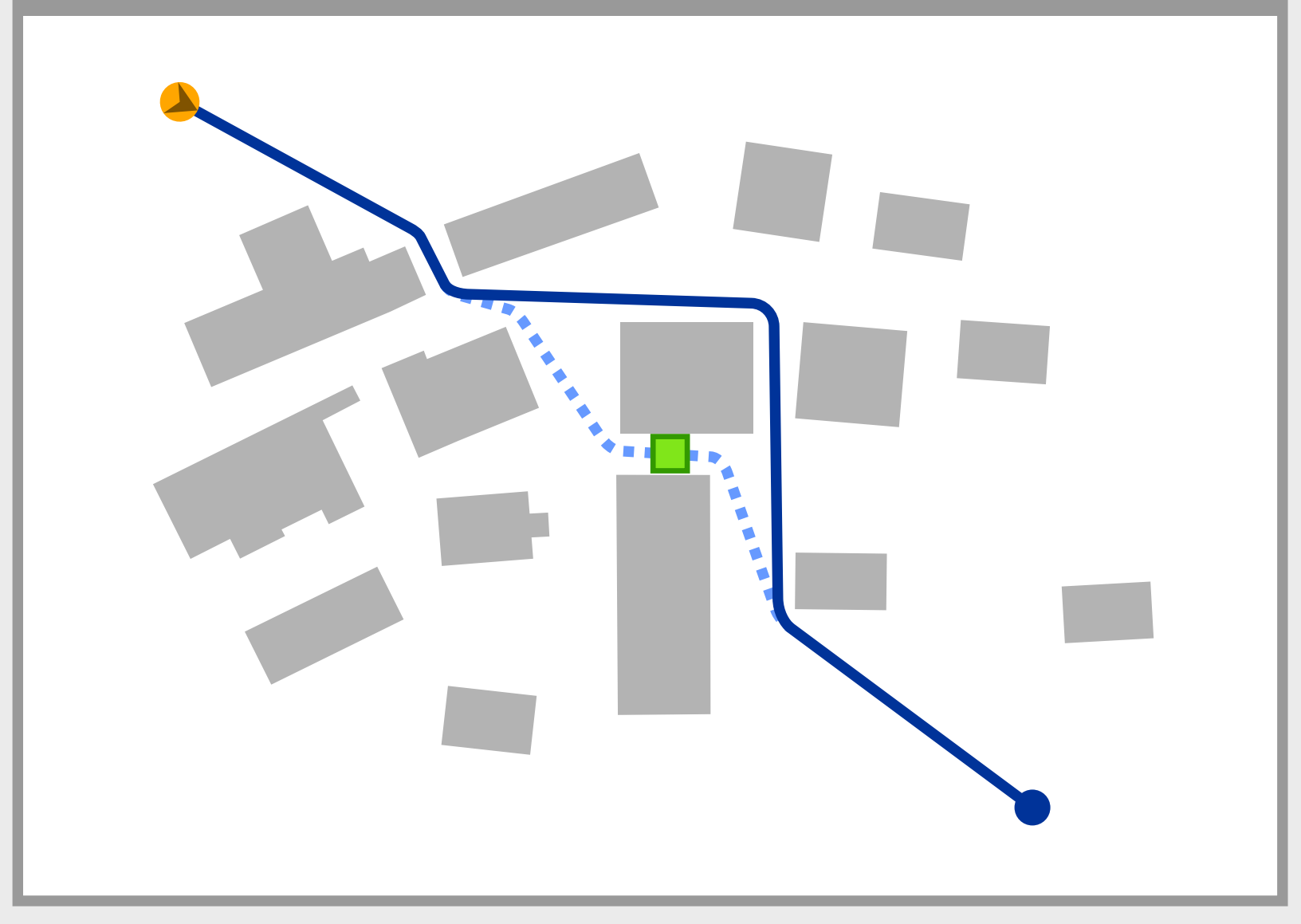
DPA* adds **pruning rules** to the standard A* search algorithm. These rules are different for each of the four possible **re-planning scenarios**.

1. Insertion, old path not in *R*



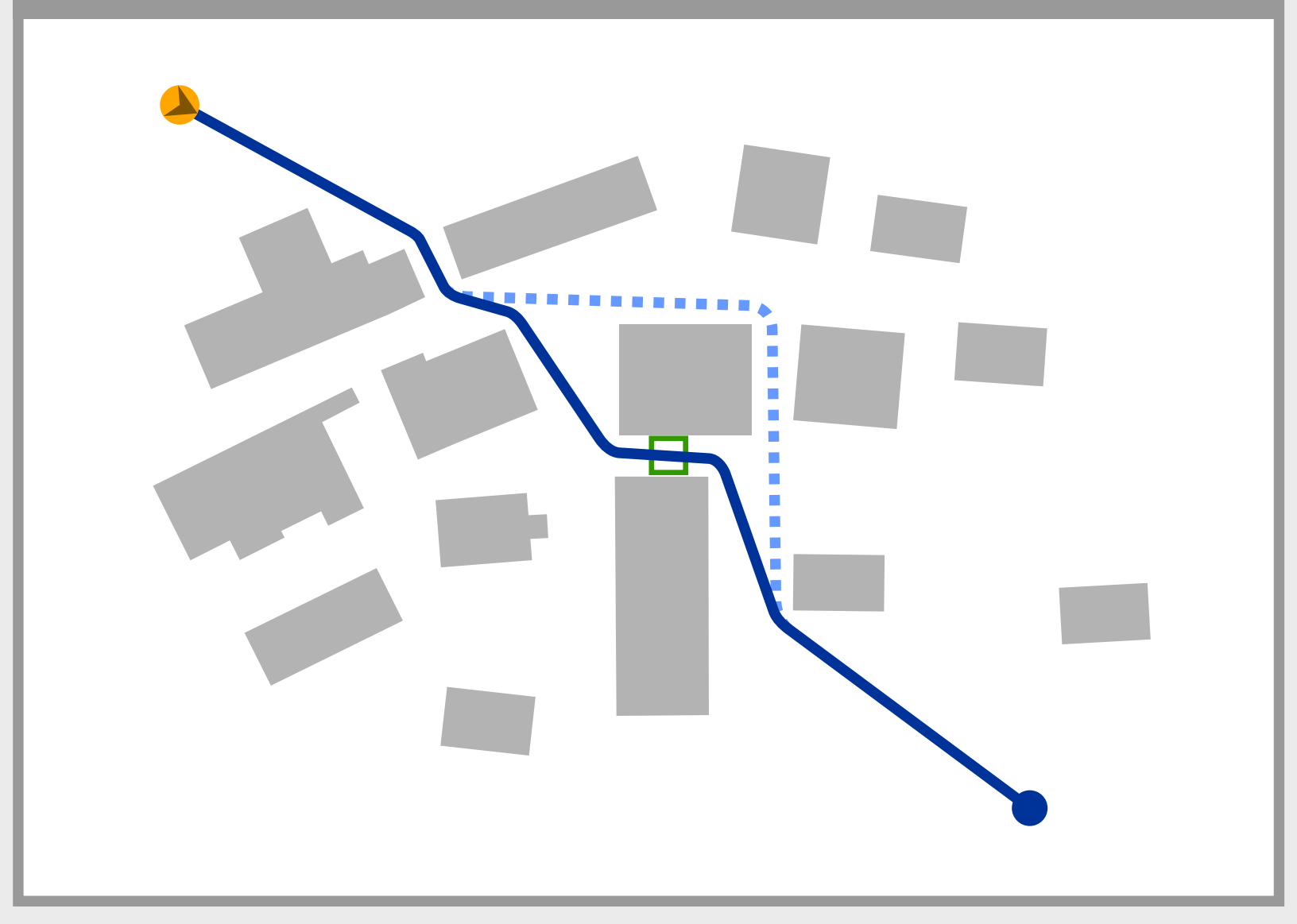
The old path *[TG]* is still optimal. Re-planning is not necessary.

2. Insertion, old path in *R*



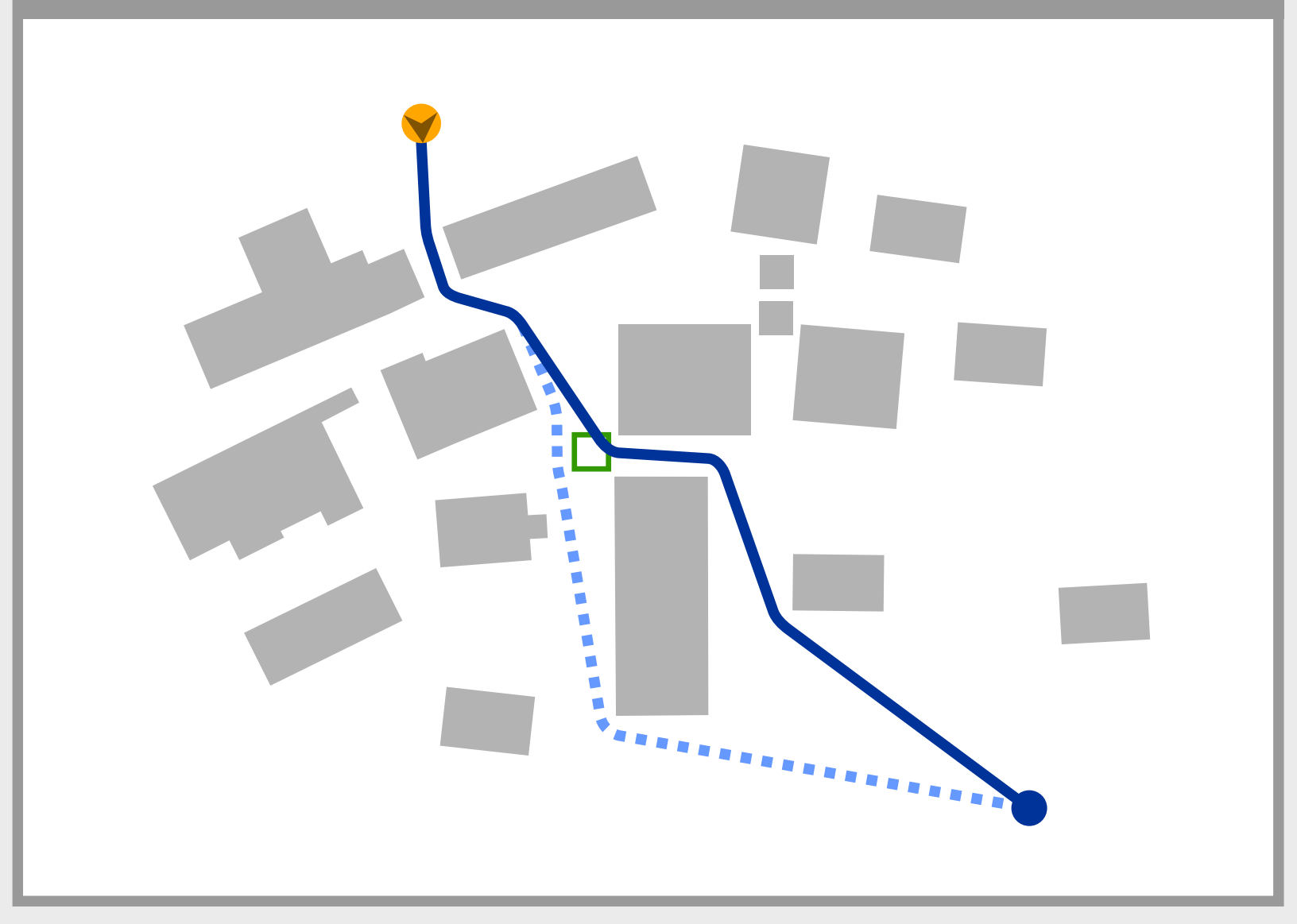
For any vertex on subpath *[BG]*, we know what its successor will be.

3. Deletion, old path not in *R*



If there is a better path than *[TG]*, then it must visit *R* at least once. Prune the search at every vertex that is too far away from *R*.

4. Deletion, old path in *R*

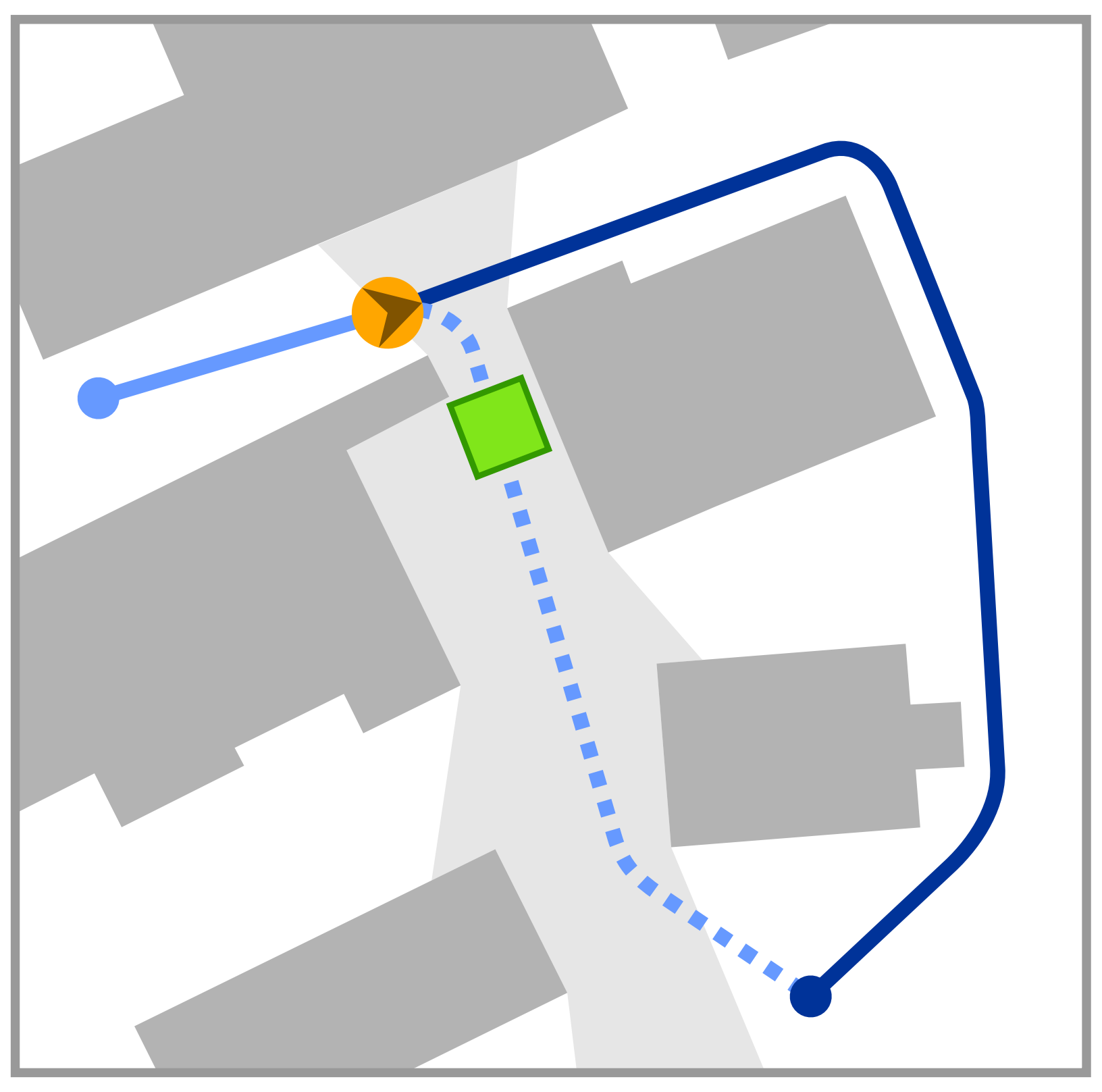


The old path is obstacle-free, but *[AB]* no longer exists in the graph. As in scenario 3, the new optimal path must visit *R* at least once.

Experiments and Results

We have implemented DPA* and tested it on the **Explicit Corridor Map** (ECM) navigation mesh. Our algorithm is particularly efficient when:

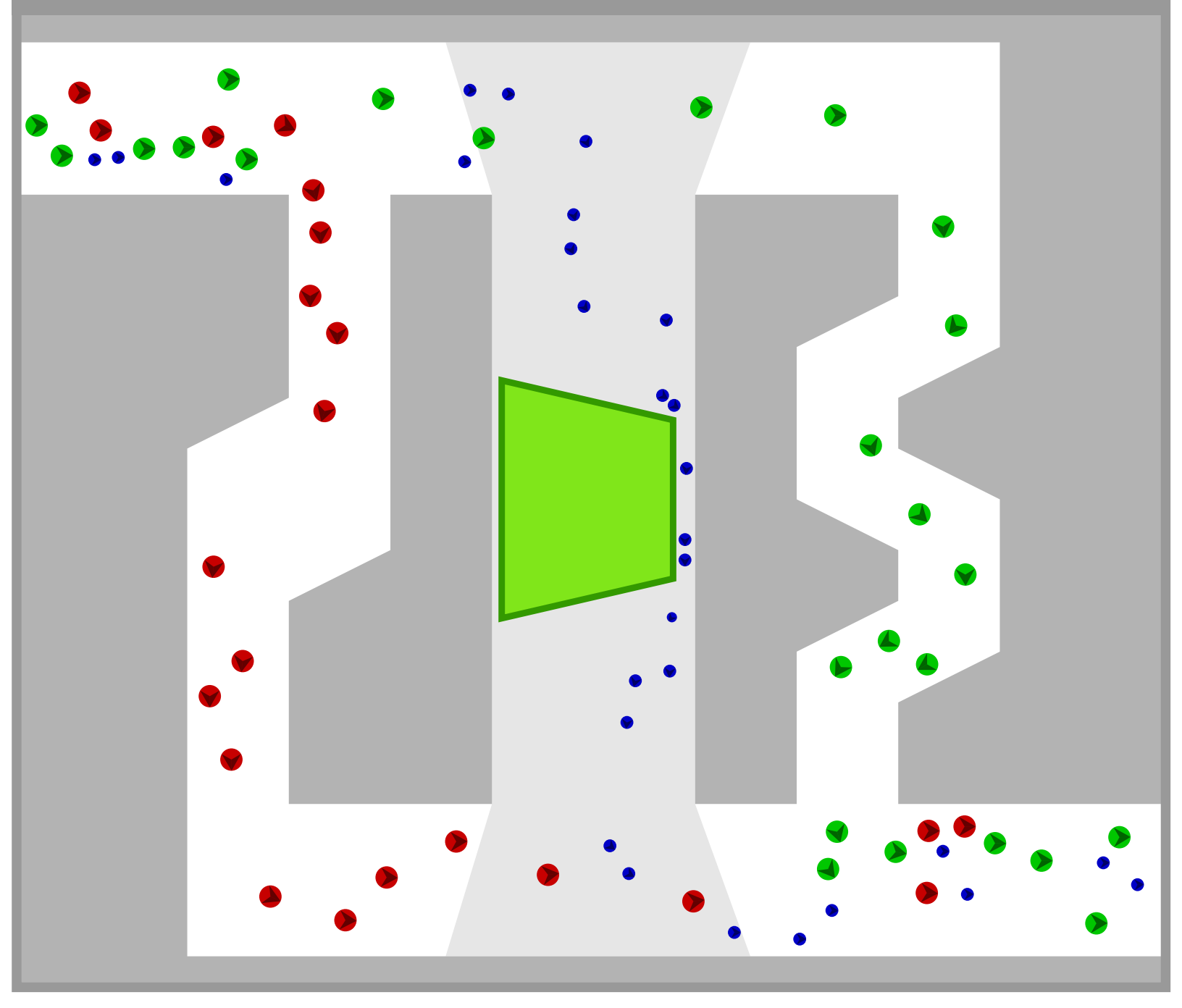
- the graph is large,
- the path is long,
- a dynamic deletion is **far away** from the agent,
- a dynamic insertion is close to the agent (e.g. the agent can **see** the new obstacle).



DPA* can be **over 60% faster** at re-planning than regular A*.

Our **crowd simulation** software can model tens of thousands of agents in real-time. DPA* enables efficient re-planning in dynamic environments.

Different agent behaviors



Integration with game engines

This research has been supported by the COMMIT/ project.