CSE 478 Robot Autonomy
Lazy Search

Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

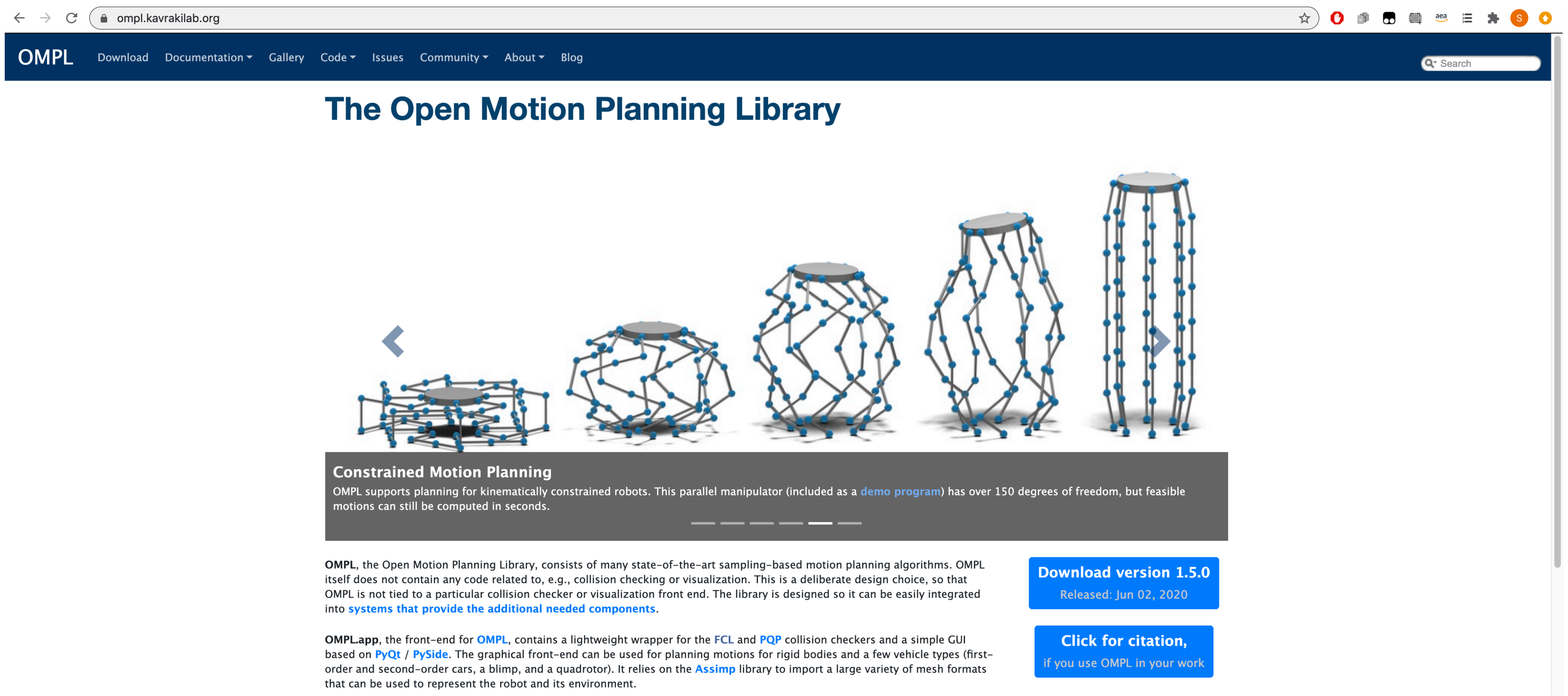
TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)

Motion Planning

OREO
OREO

253
250

Motion Planning is a technology



The Open Motion Planning Library

Constrained Motion Planning

OMPL supports planning for kinematically constrained robots. This parallel manipulator (included as a demo program) has over 150 degrees of freedom, but feasible motions can still be computed in seconds.

OMPL, the Open Motion Planning Library, consists of many state-of-the-art sampling-based motion planning algorithms. OMPL itself does not contain any code related to, e.g., collision checking or visualization. This is a deliberate design choice, so that OMPL is not tied to a particular collision checker or visualization front end. The library is designed so it can be easily integrated into systems that provide the additional needed components.

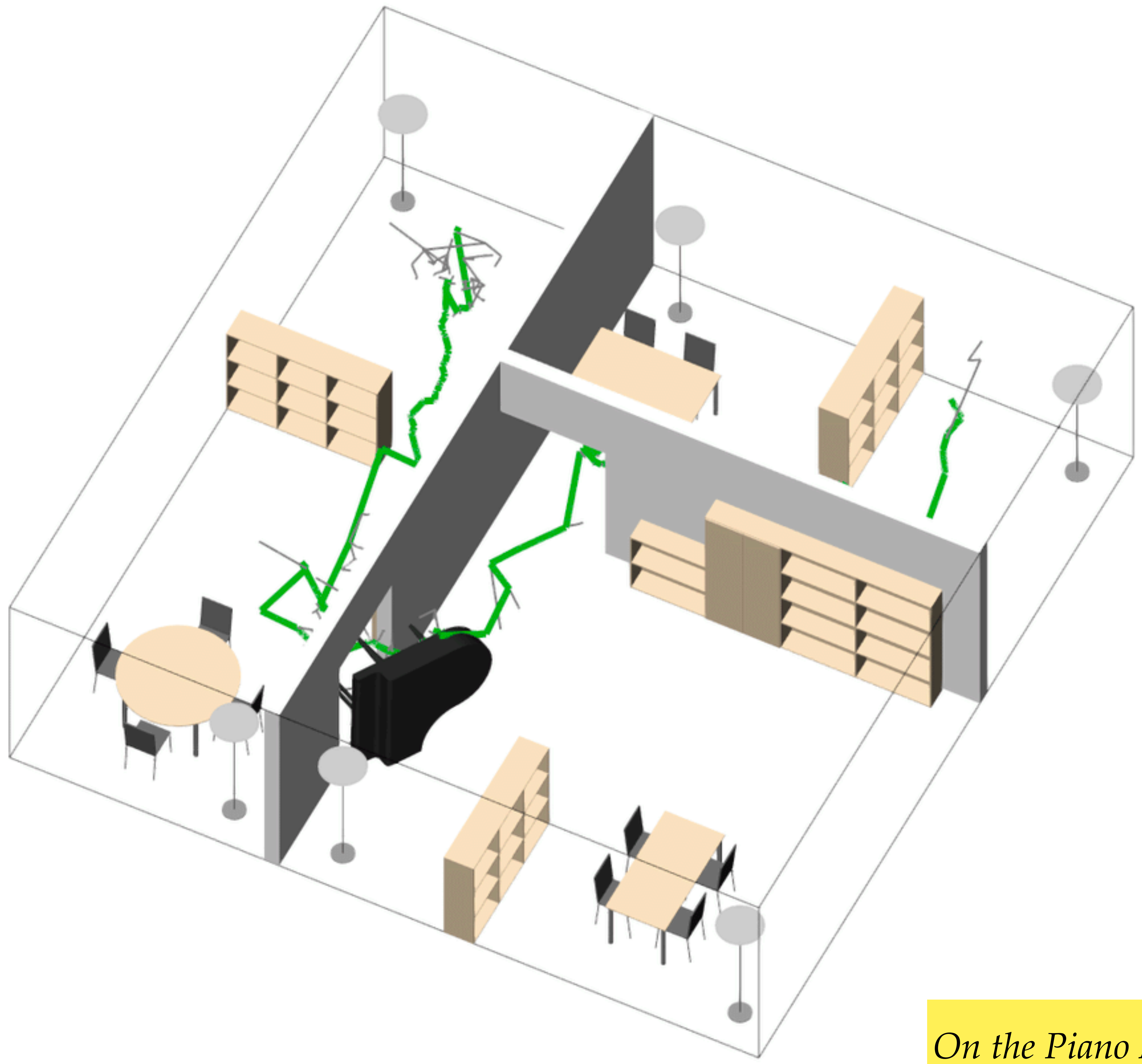
OMPL.app, the front-end for OMPL, contains a lightweight wrapper for the FCL and PQP collision checkers and a simple GUI based on PyQt / PySide. The graphical front-end can be used for planning motions for rigid bodies and a few vehicle types (first-order and second-order cars, a blimp, and a quadrotor). It relies on the Assimp library to import a large variety of mesh formats that can be used to represent the robot and its environment.

Download version 1.5.0
Released: Jun 02, 2020

Click for citation,
if you use OMPL in your work

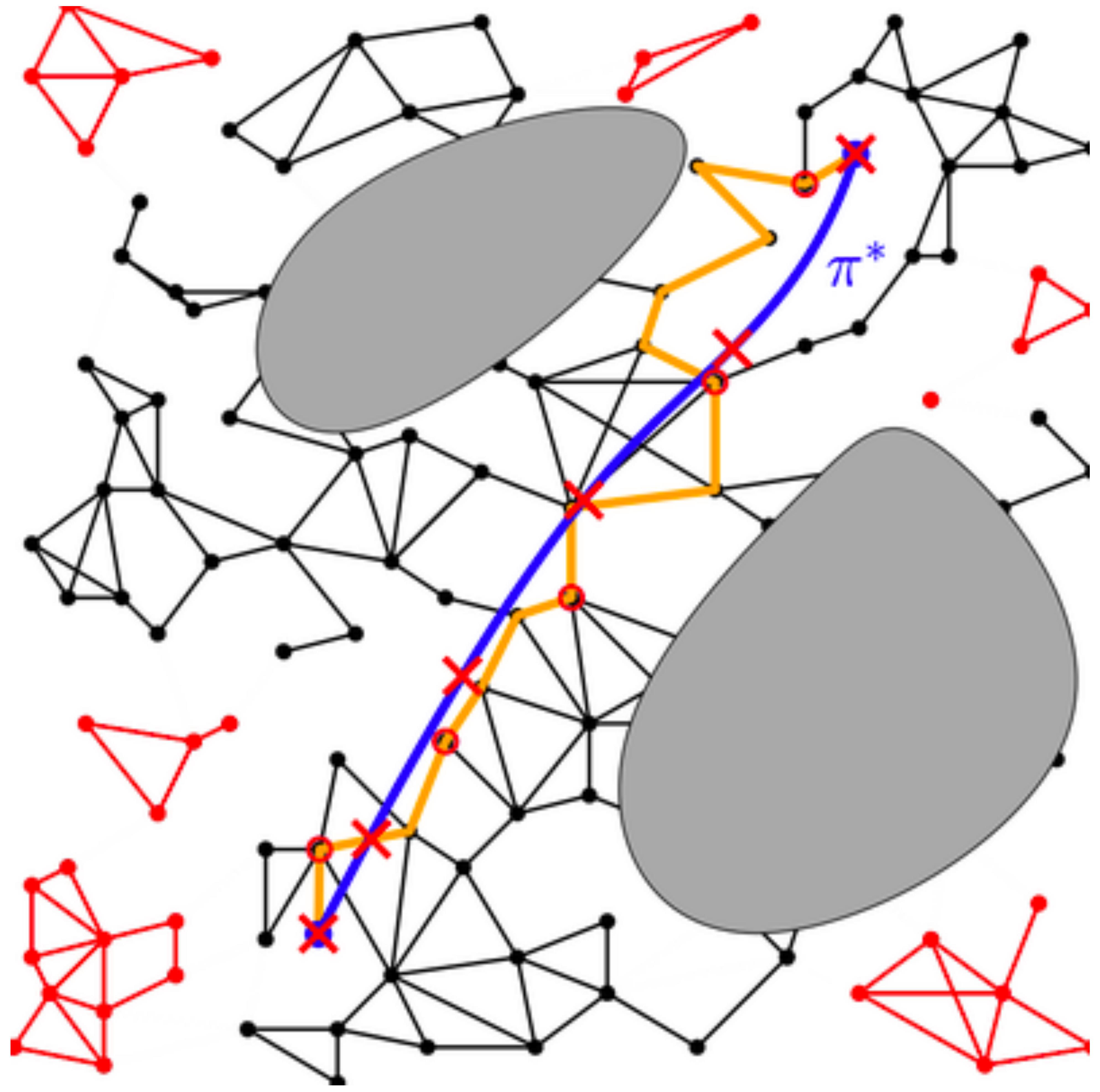
10-100X Improvement

The Piano Movers' Problem



On the Piano Movers problem. I-III, Schwartz and Sharir, Comm. on Pure and Applied Math., 1983

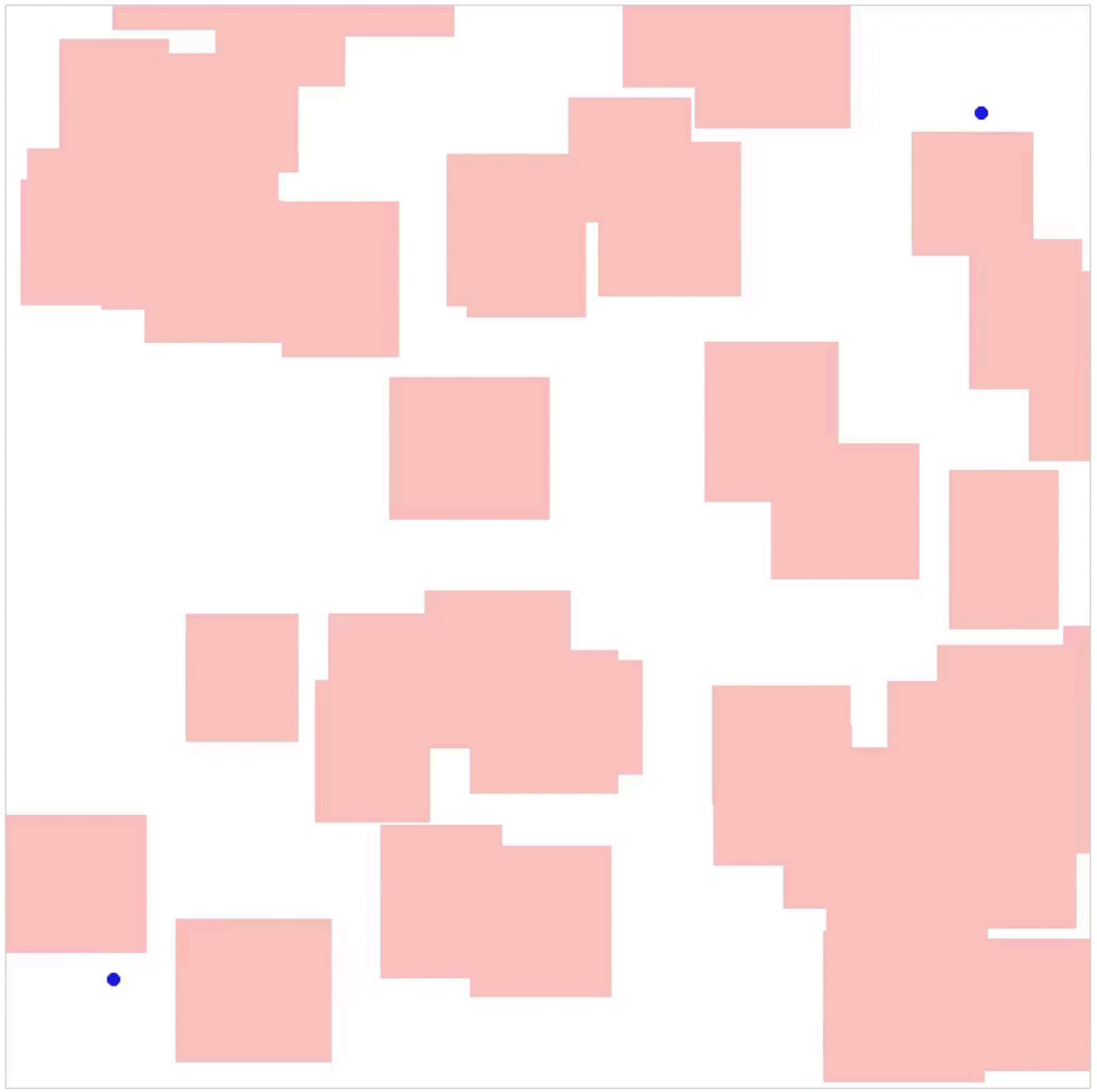
Roadmaps



$$\begin{tikzcd}\boxed{\text{Build Roadmap}} \arrow[d] \\ \boxed{\text{Plan on Roadmap}}\end{tikzcd}$$

Probabilistic roadmaps for path planning in high-dimensional configuration spaces, Kavraki et al., IEEE TRO, 1996.

A* Search

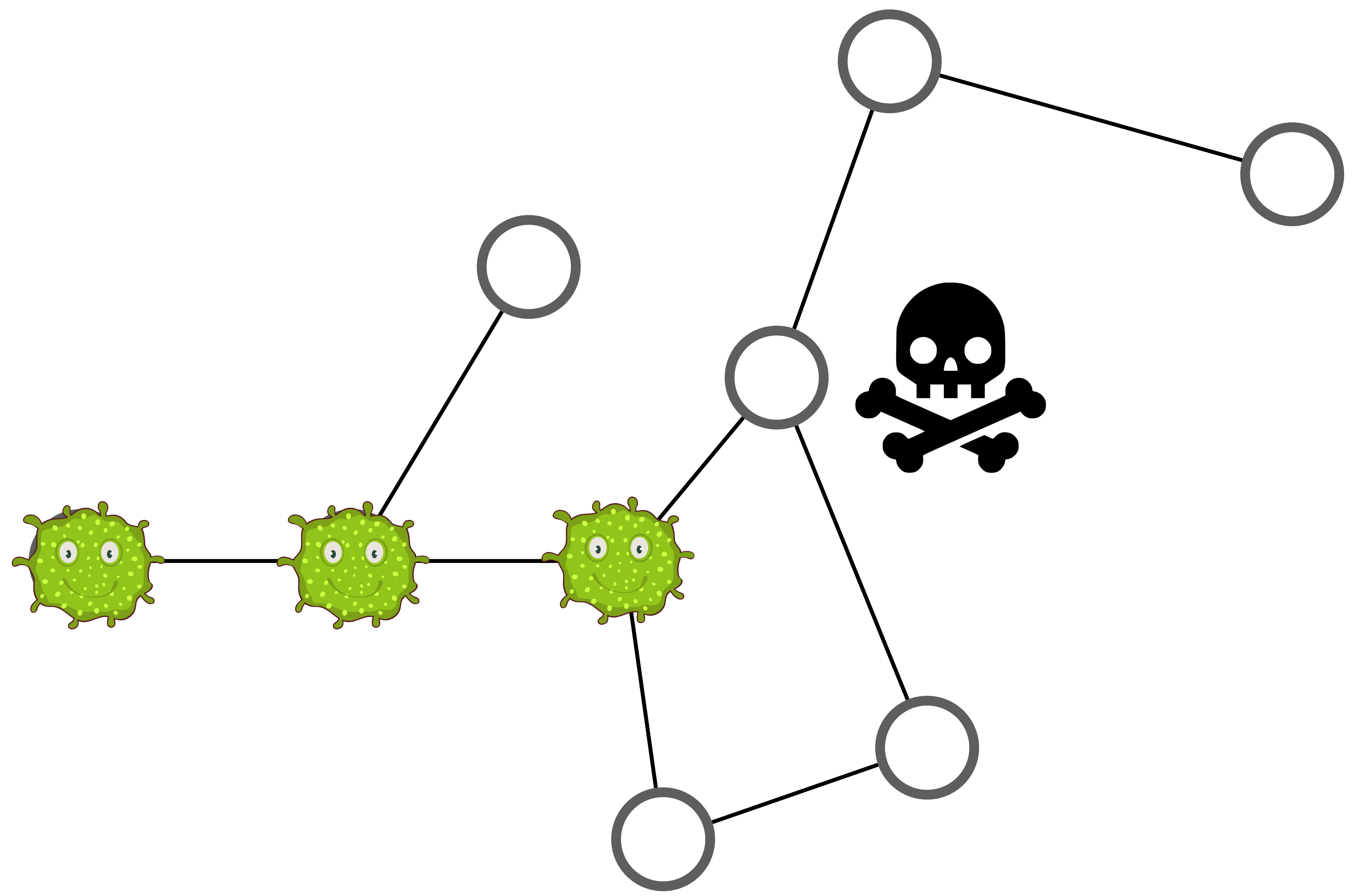


A* Search

OPTIMAL!!

Is it optimal over something we care about?

A* Search: Amoebas!



Optimal Substructure

$$f(a) < f(b) \implies f(a \circ x) < f(b \circ x) \forall x$$

You will never catch up.

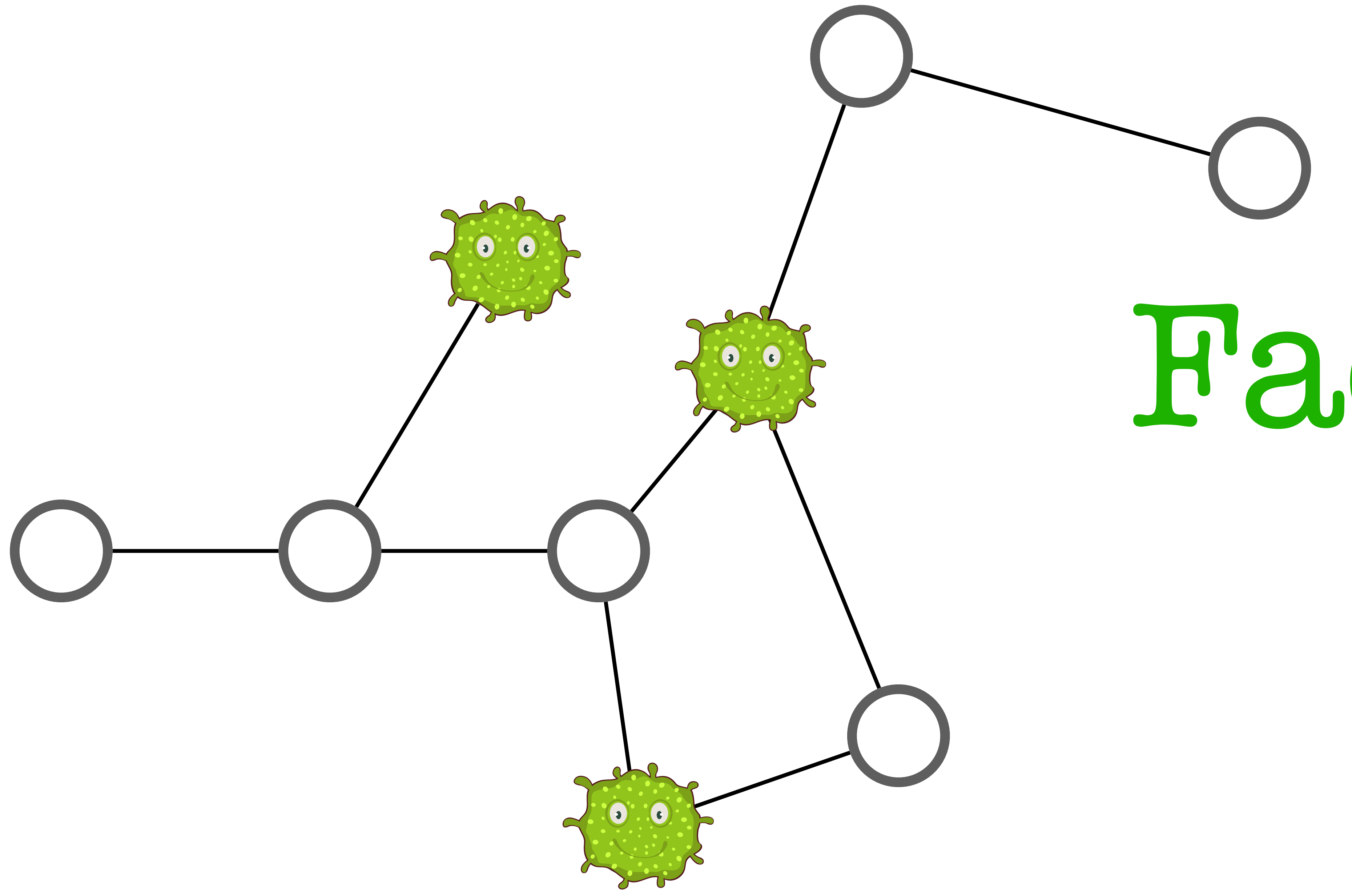
Bellman Condition

$$f^*(a) = \min_{x \in \mathrm{succ}} \{c(a,x) + f^*(x)\}$$

Be best, locally.

Bacteria Vectors by Vecteezy

A* Search: Favoritism



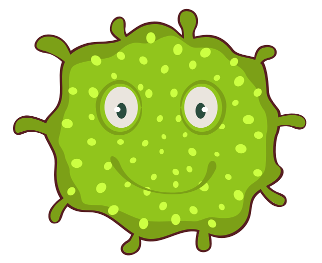
Optimism in the Face of Uncertainty (OFU)

$$\min_{x \in \text{open}} \; g(x) + h(x)$$

Always be optimistic under uncertainty.
You'll either be correct,
or learn something important if you're wrong.

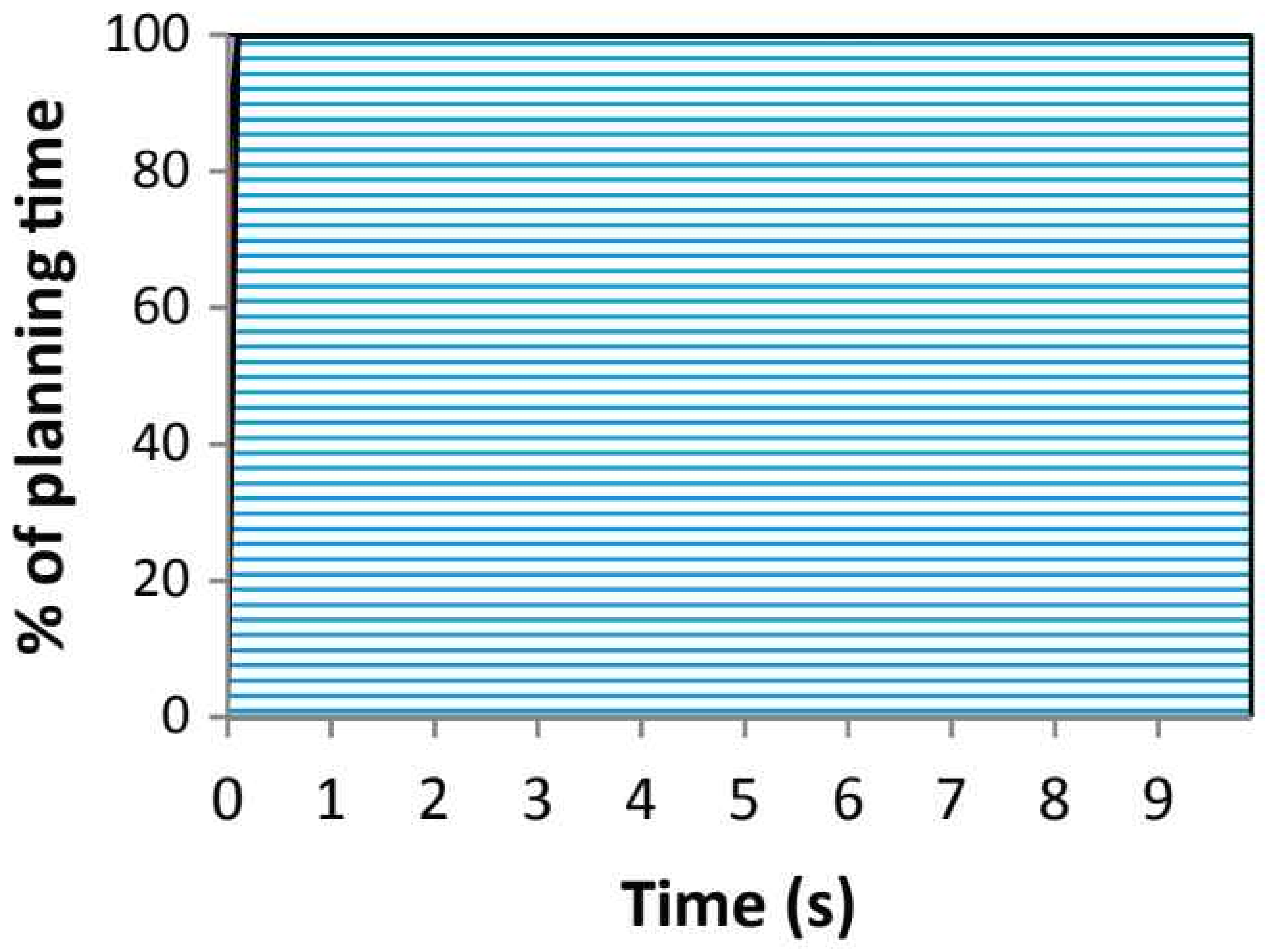
R-MAX: A general polynomial time algorithm for near-optimal reinforcement learning, Brafman and Tennenholtz, JMLR, 2002.

A* Search is Optimal …

Expands the Fewest Number of Vertices 

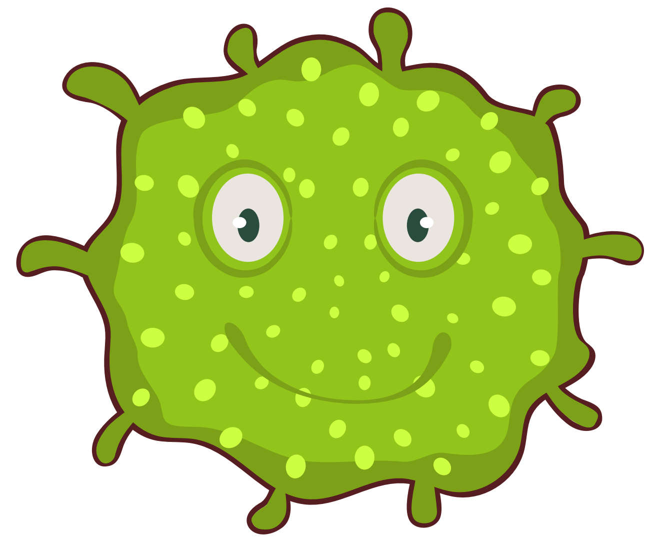
But is this what we *really* want in Motion Planning?

Edge Evaluation Dominates Planning Time



▤ **Edge Evaluations**

🟥 **Other**

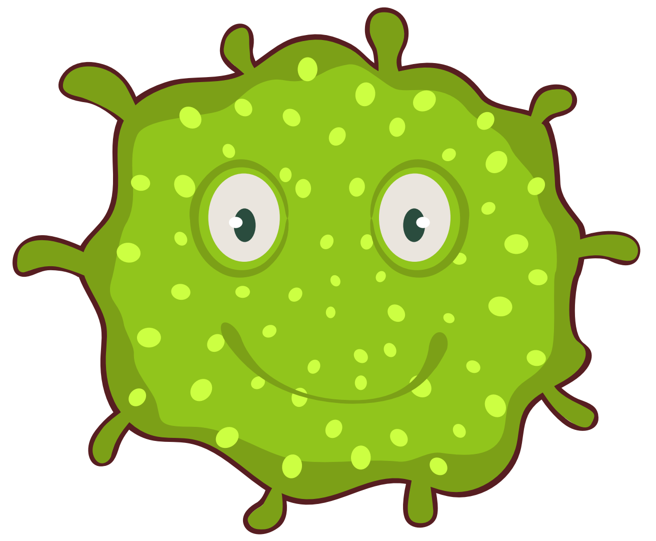


Amoebas are Cheap

Slime is Expensive

Lazy collision checking in asymptotically-optimal motion planning, Hauser, ICRA 2015.

Is there a Search Algorithm that Minimizes the Number of Edge Evaluations?



I don’t care about amoebas.
What algorithm minimizes slime?

LazySP

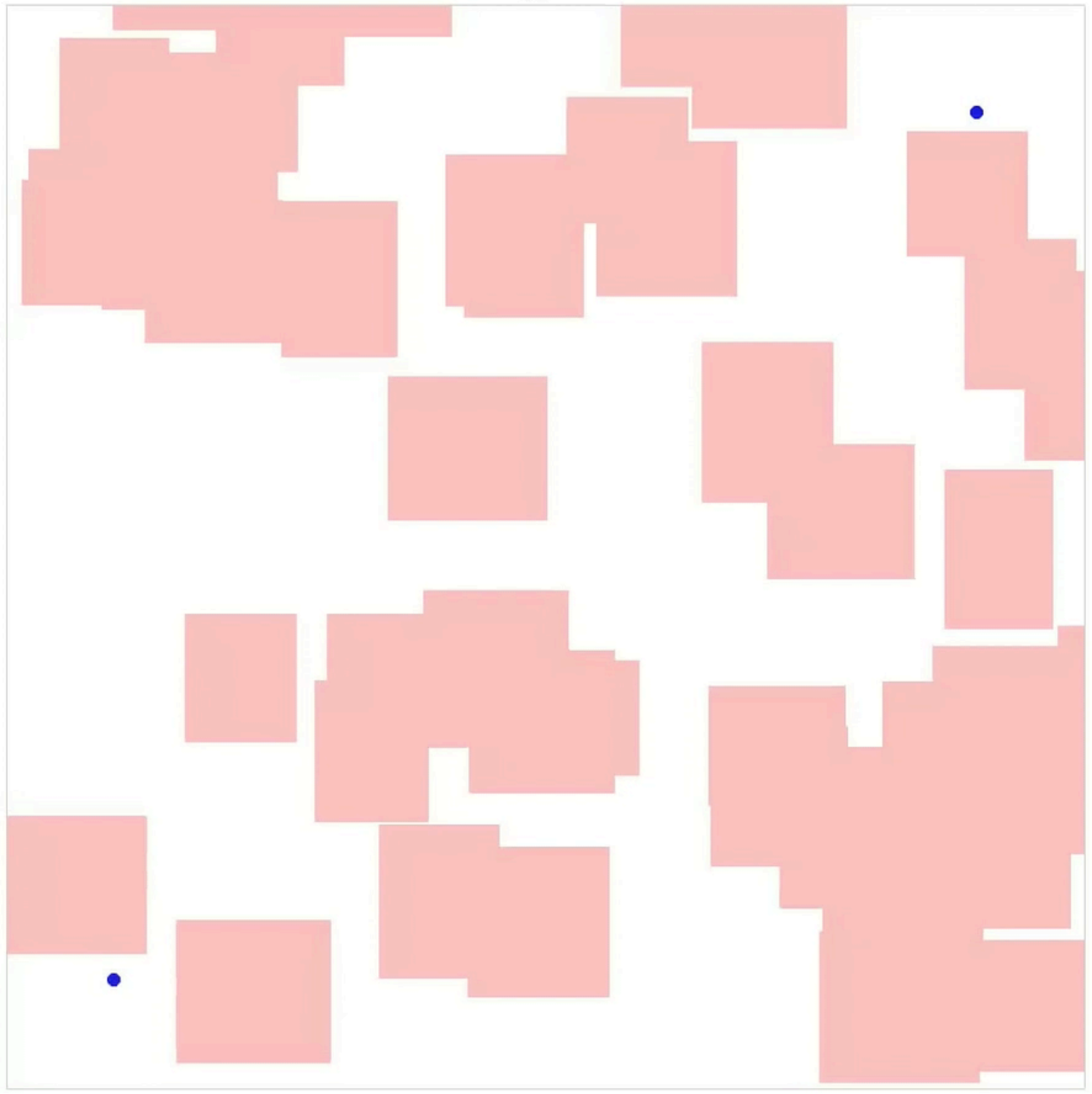
ICAPS 2018, 2019 [Best Conference Paper Award Winner]

First Provably Edge-Optimal A*-like Search Algorithm

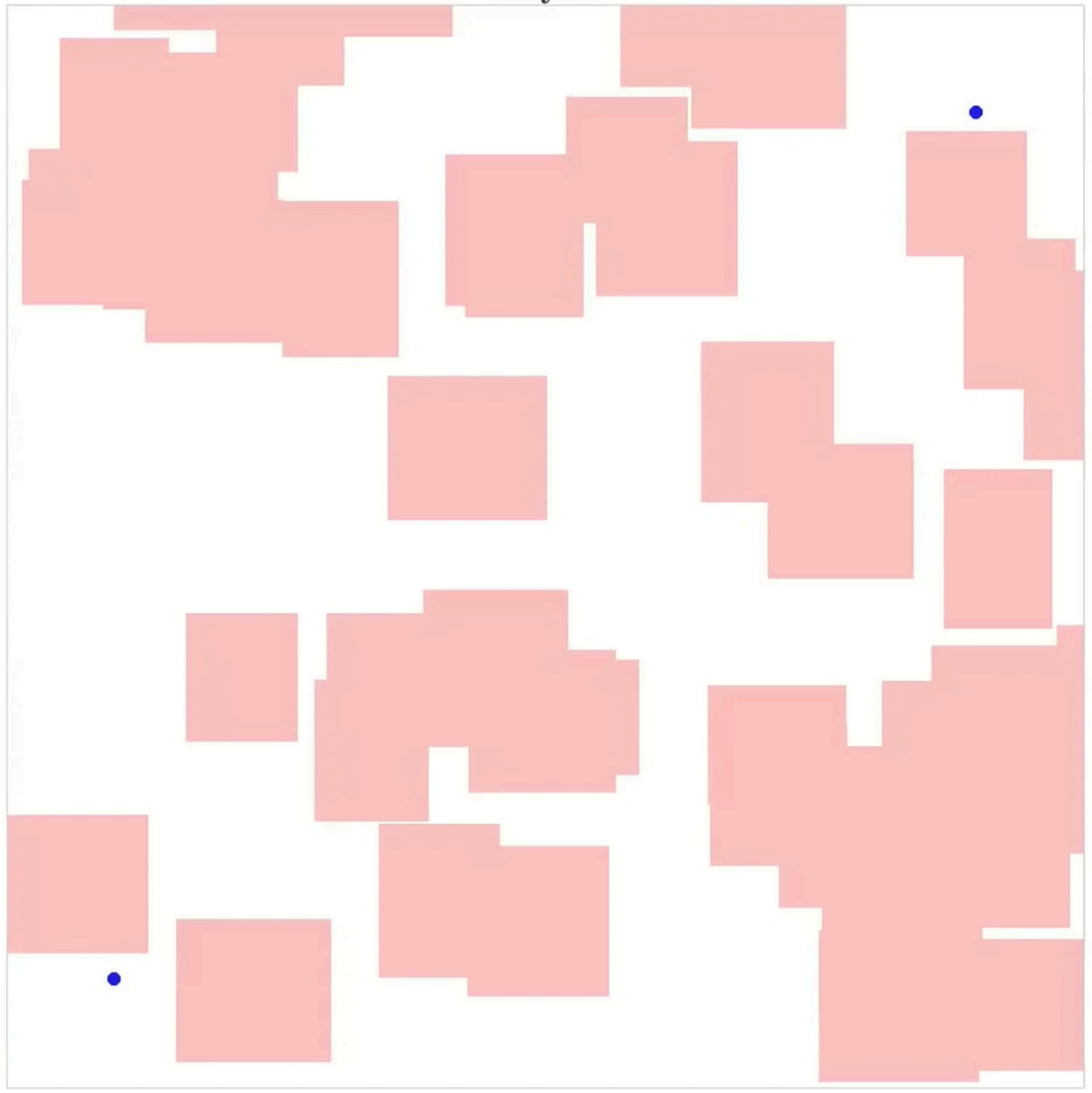
The Provable Virtue of Laziness in Motion Planning, Hagtalab et al., ICAPS 2018.

Number of Edges Evaluated

A*

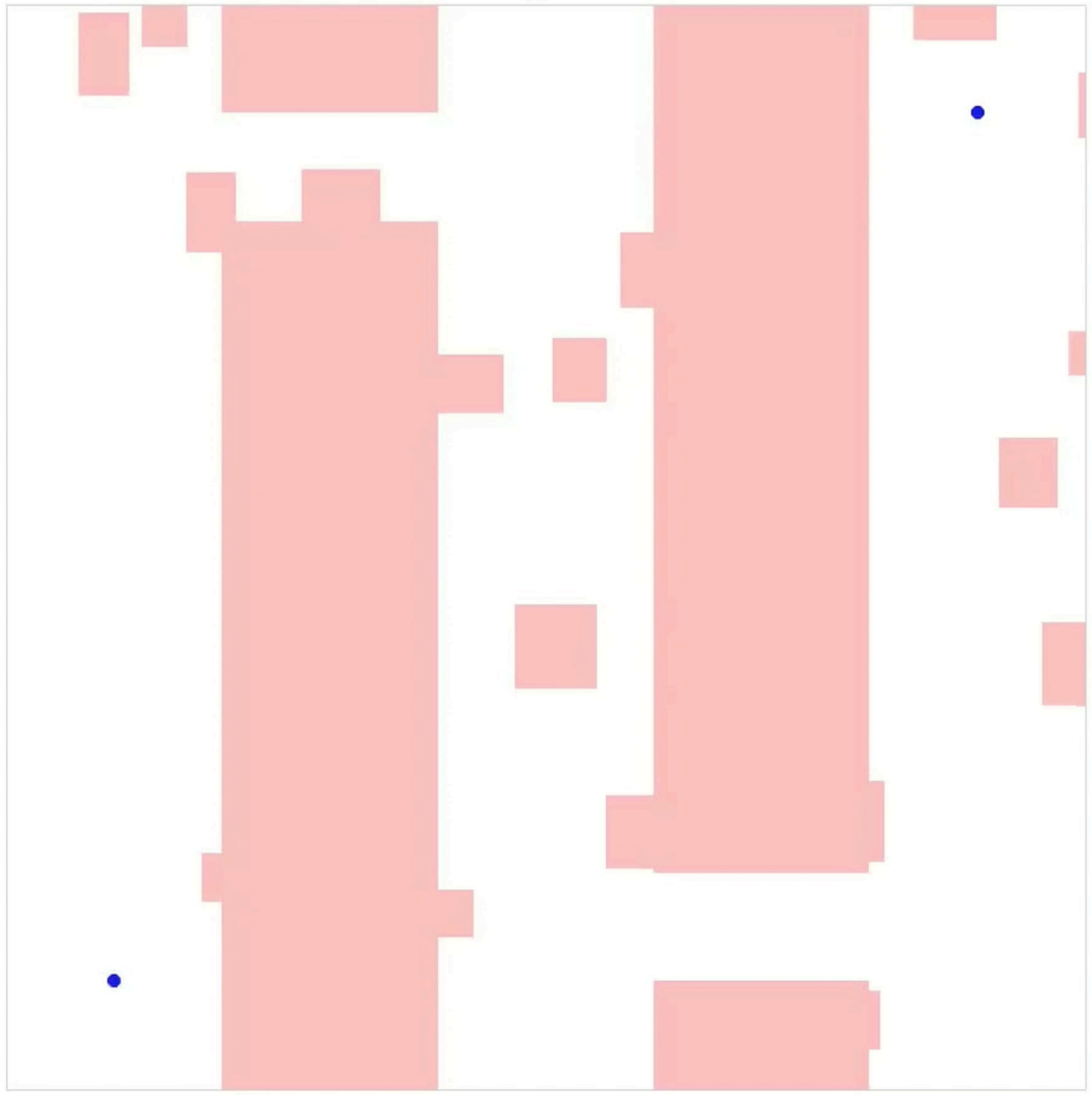


LazySP

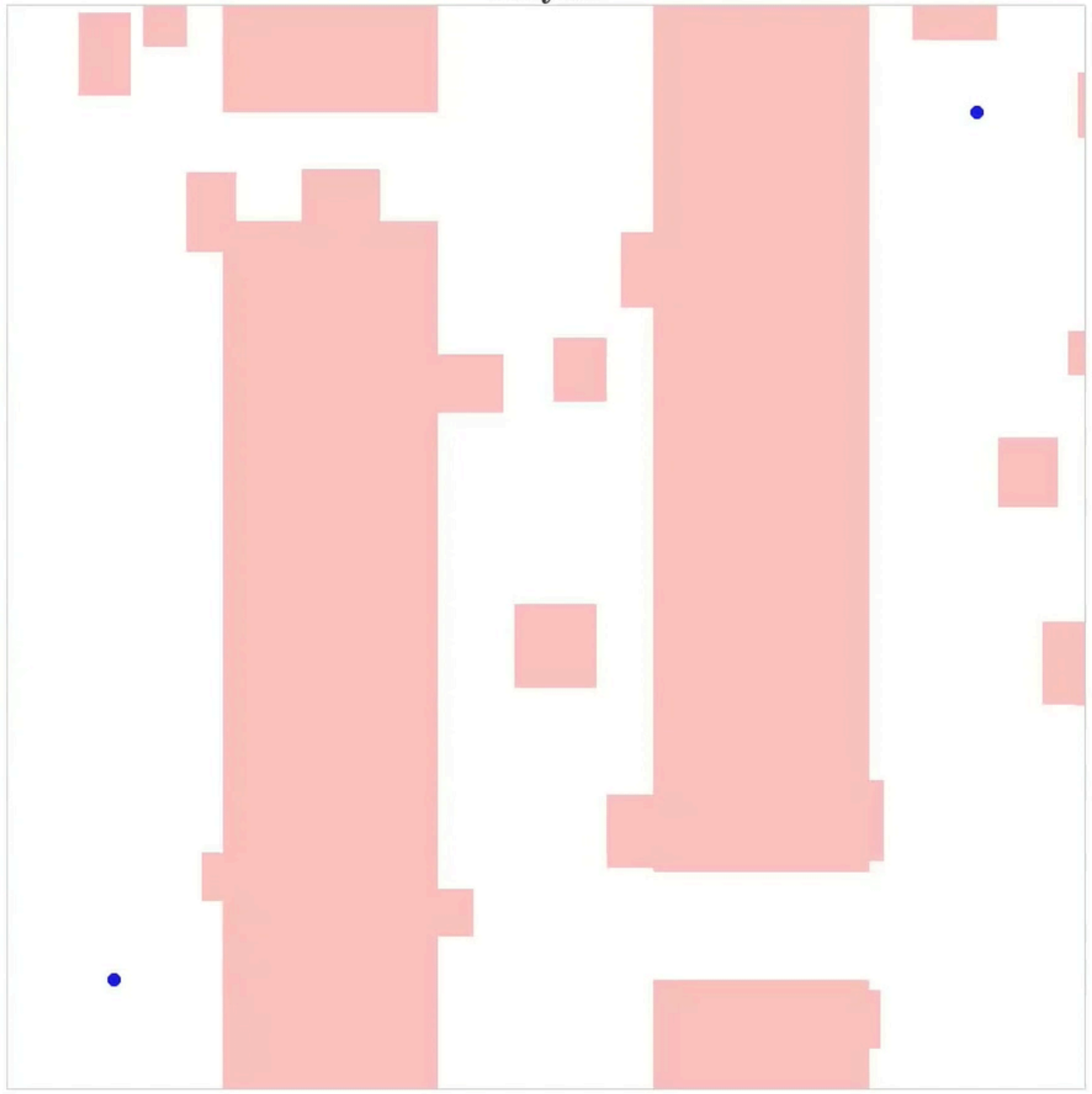


Number of Edges Evaluated

A*

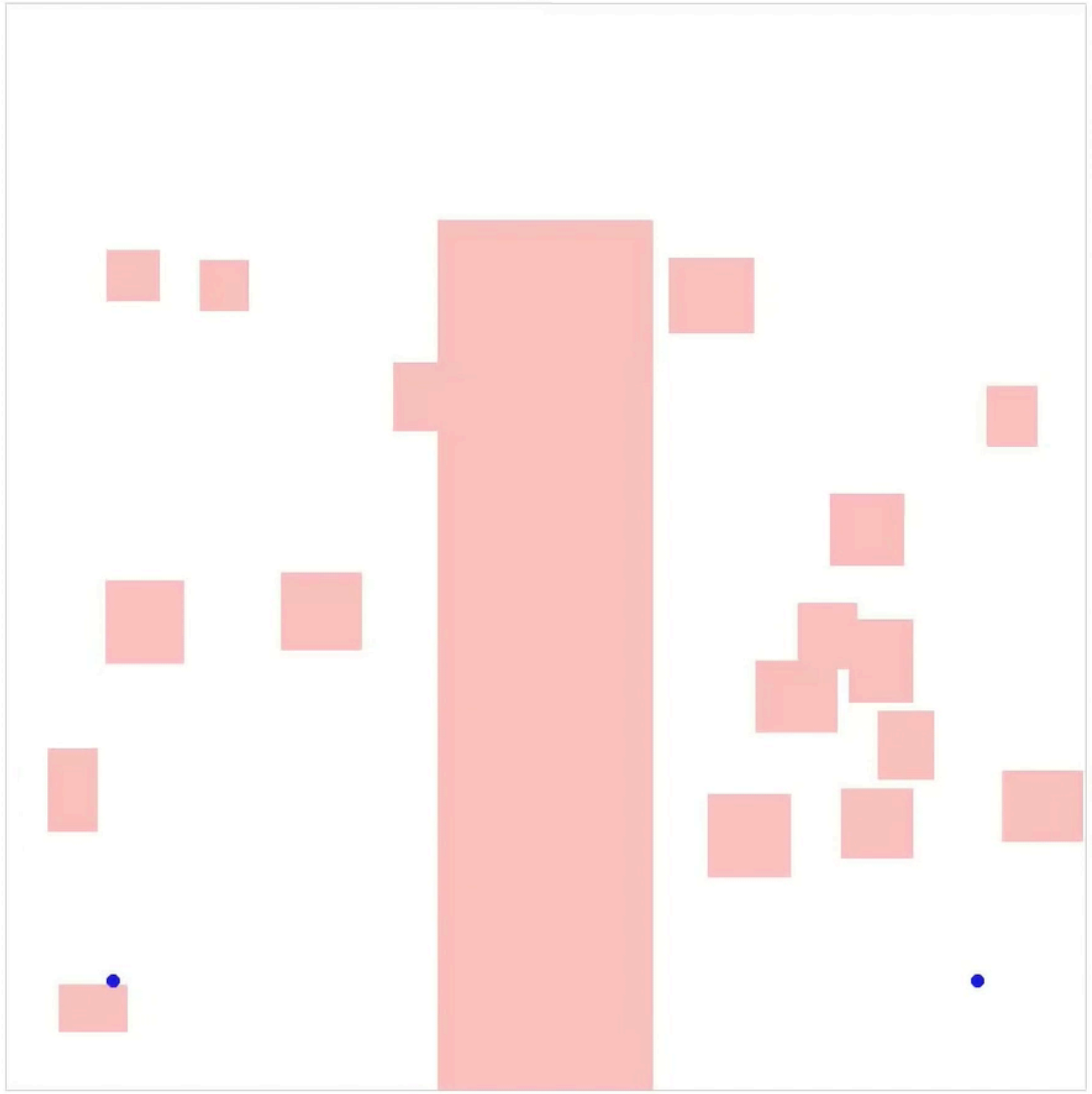


LazySP

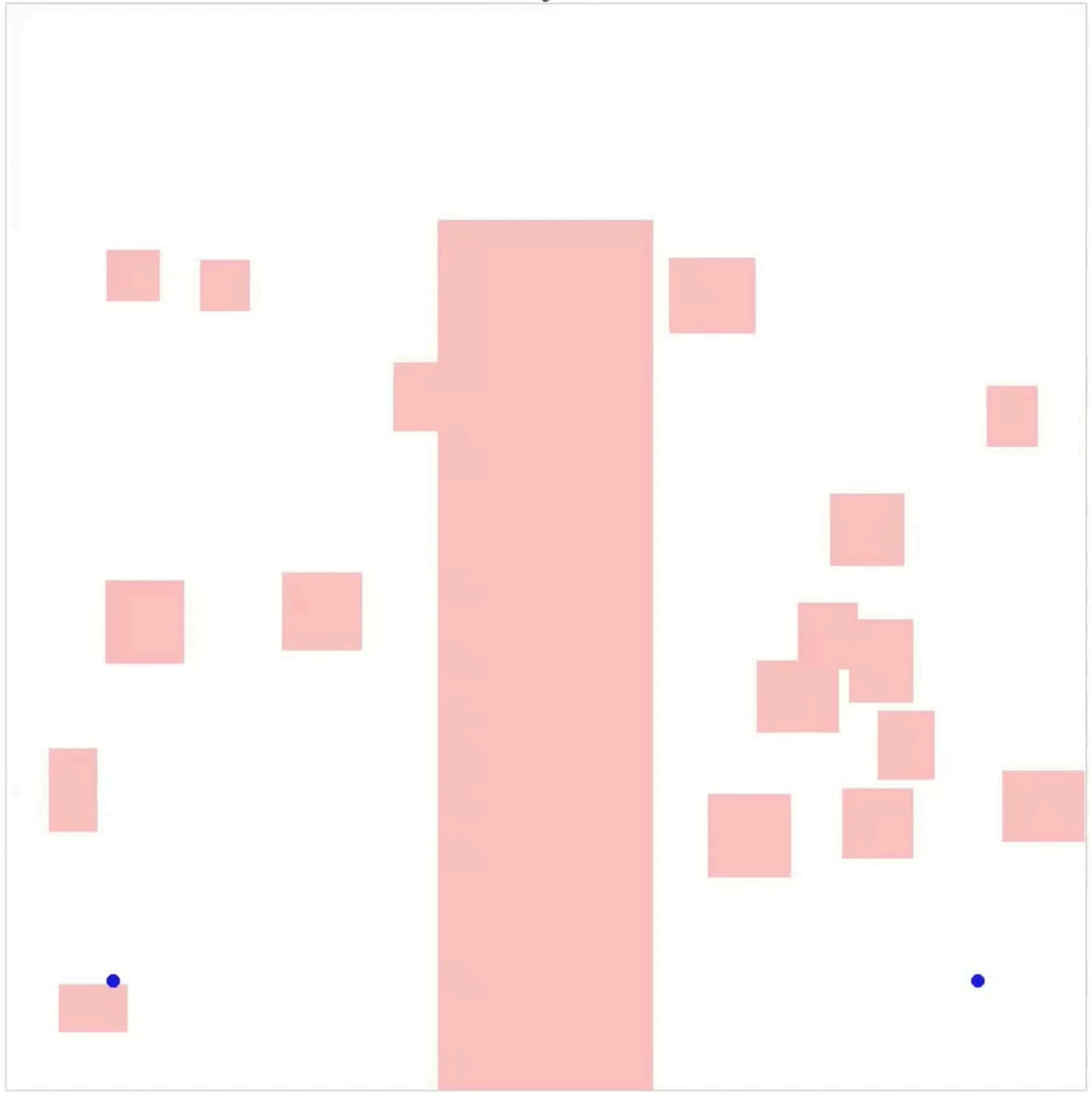


Number of Edges Evaluated

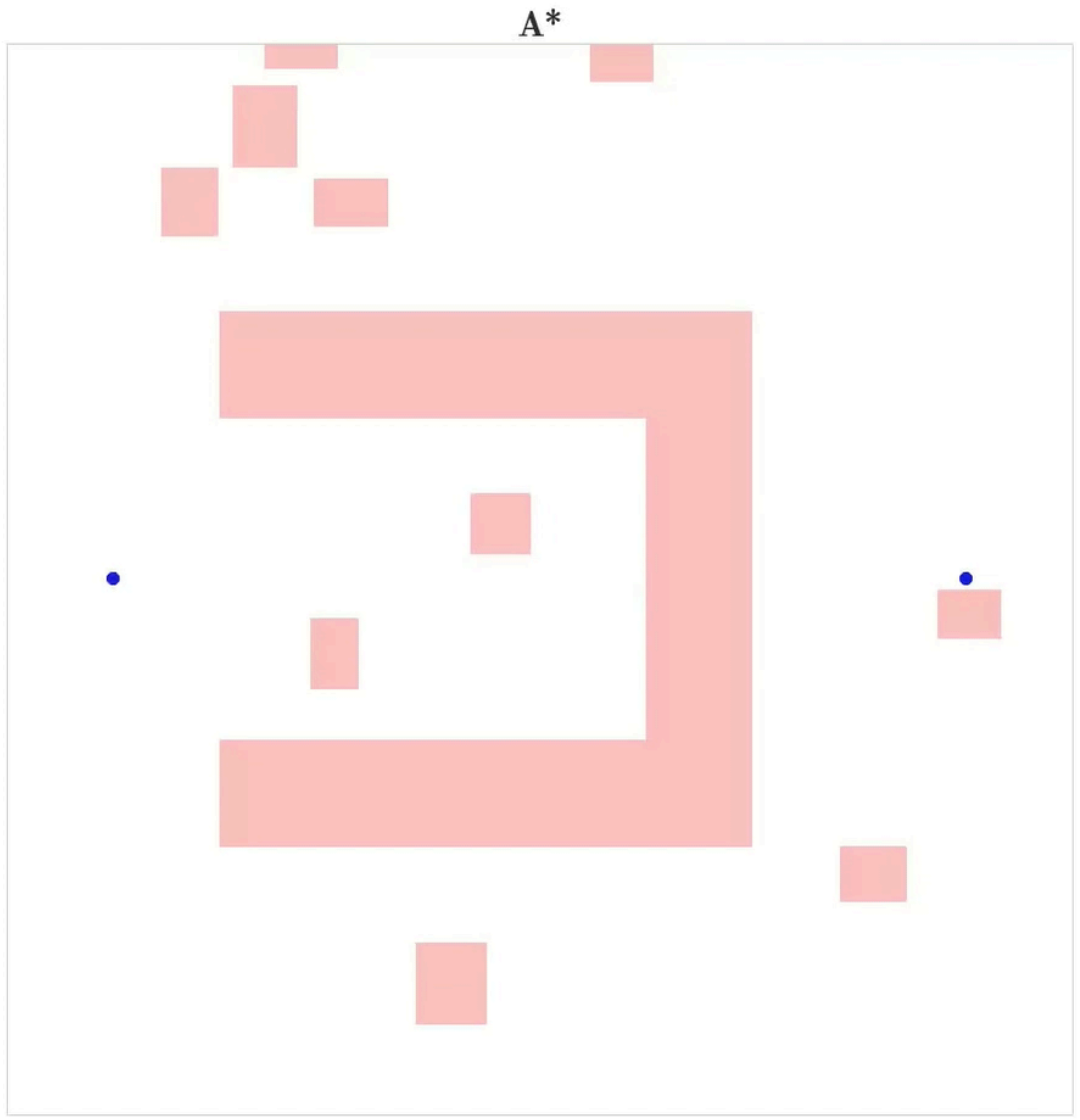
A*

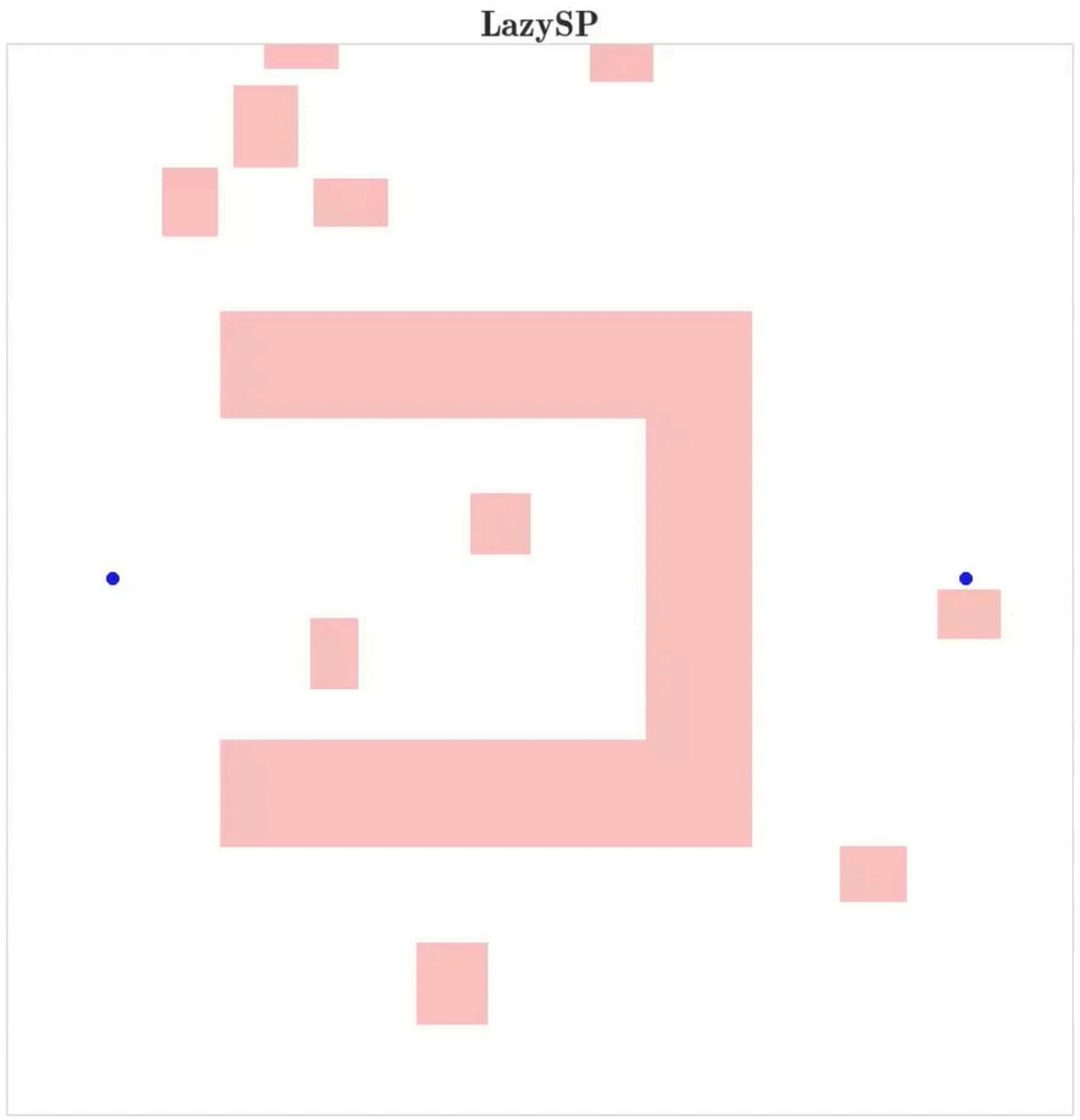


LazySP



Number of Edges Evaluated





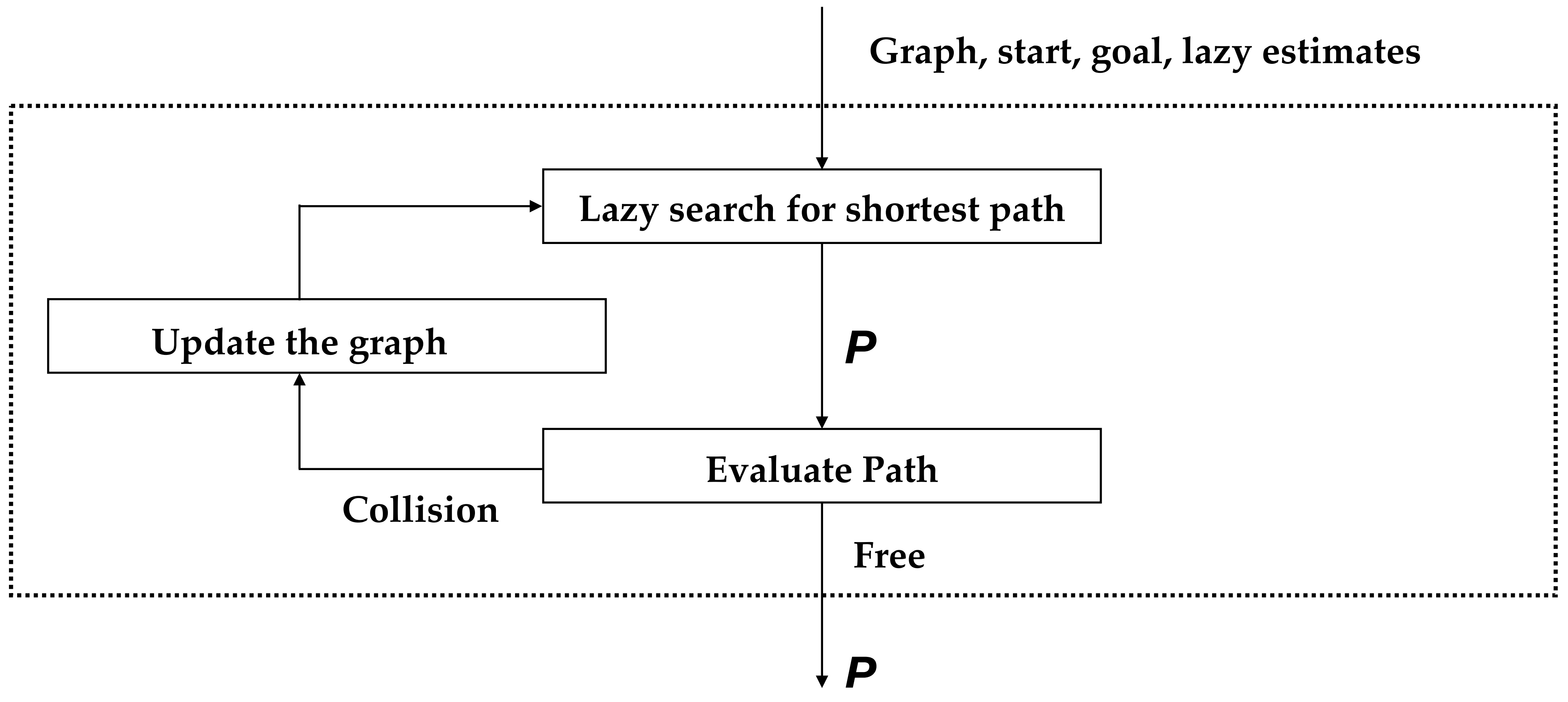
LazySP

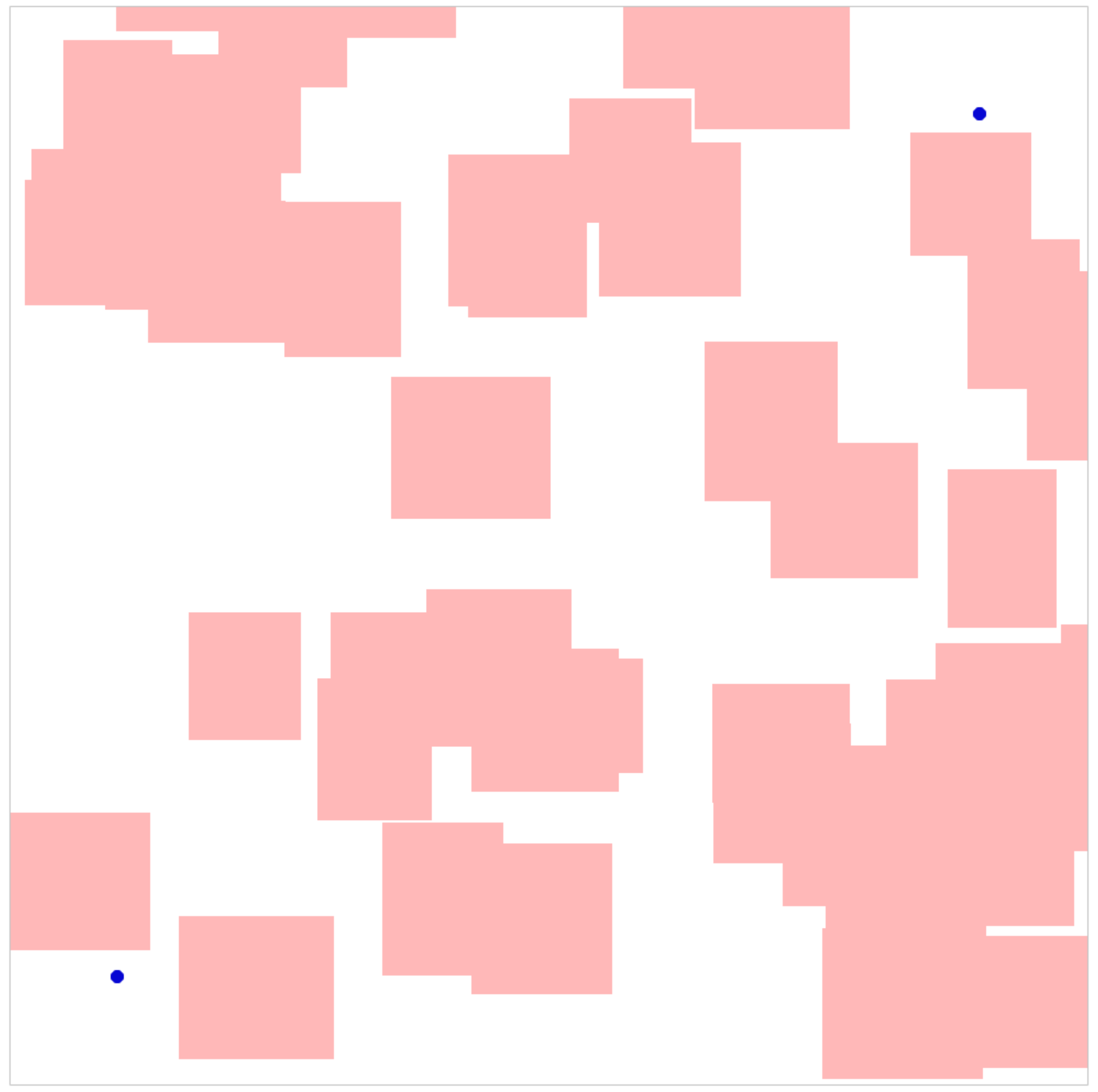
Greedy Best-first Search over Paths

To find the shortest path,
eliminate all shorter paths!

LazySP

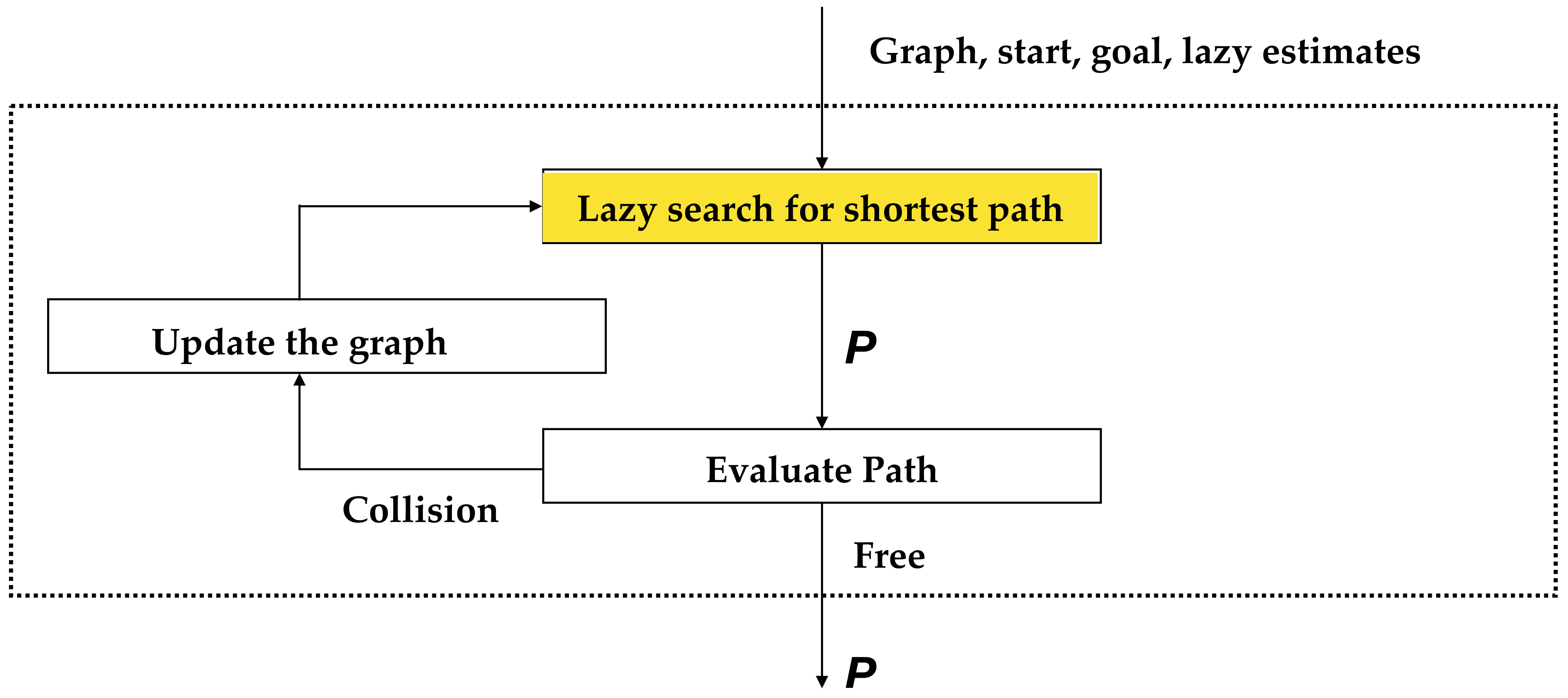
OFU on Steroids!

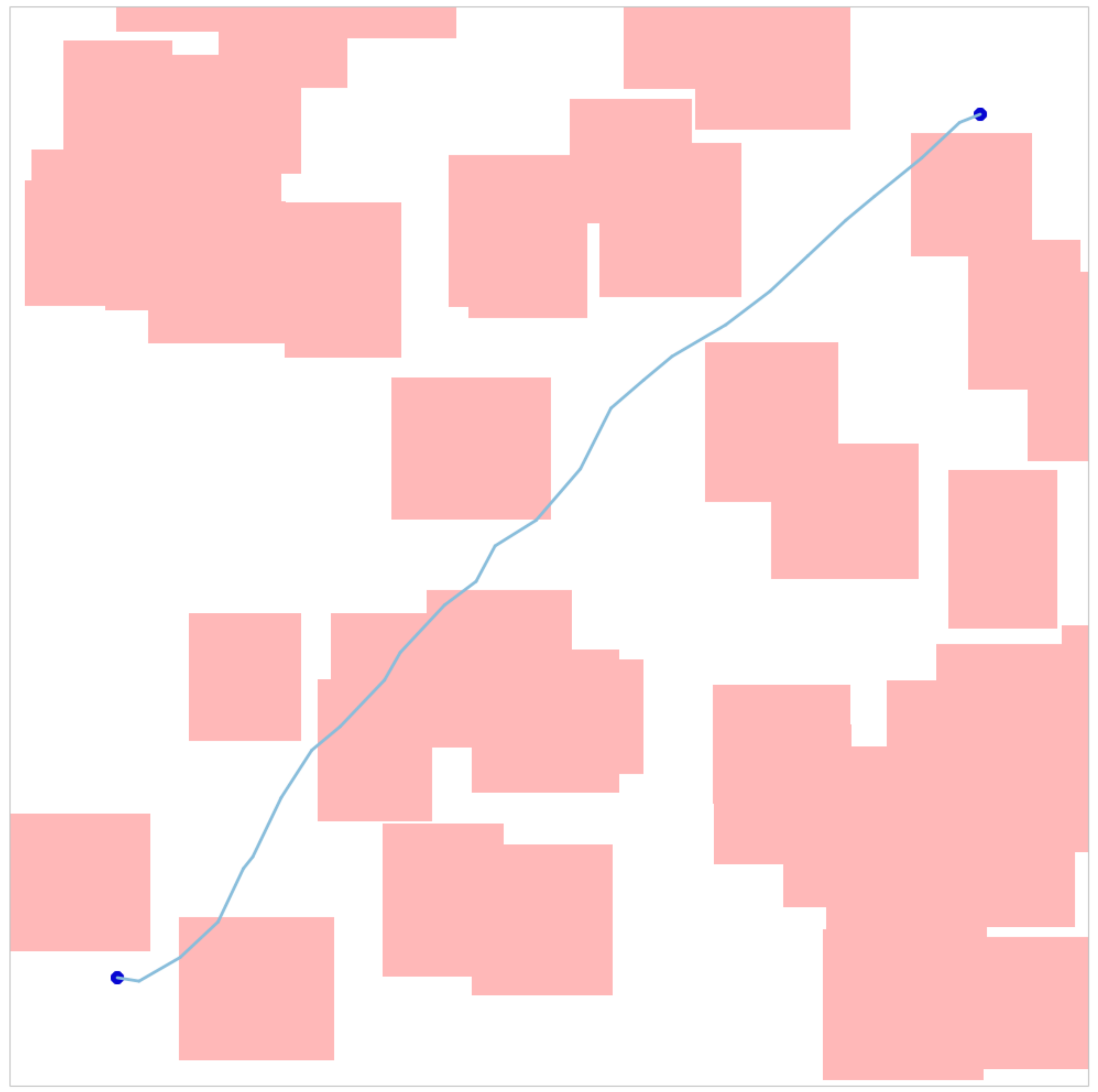

Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

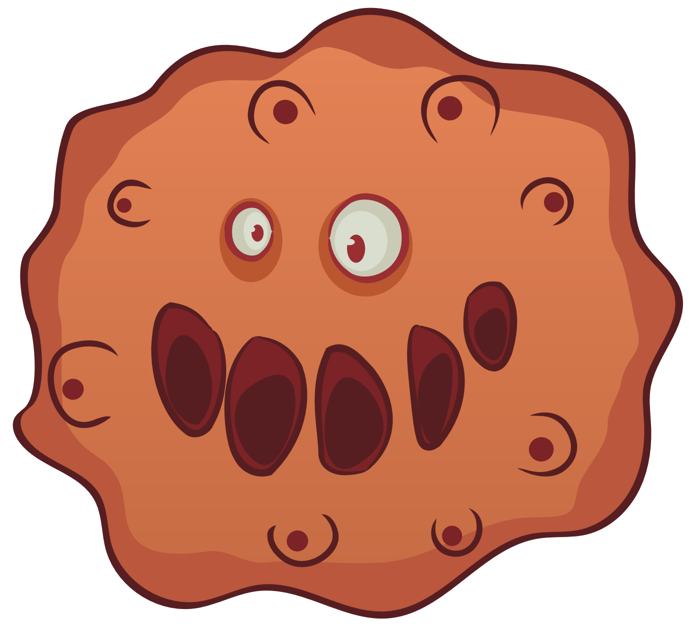


LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

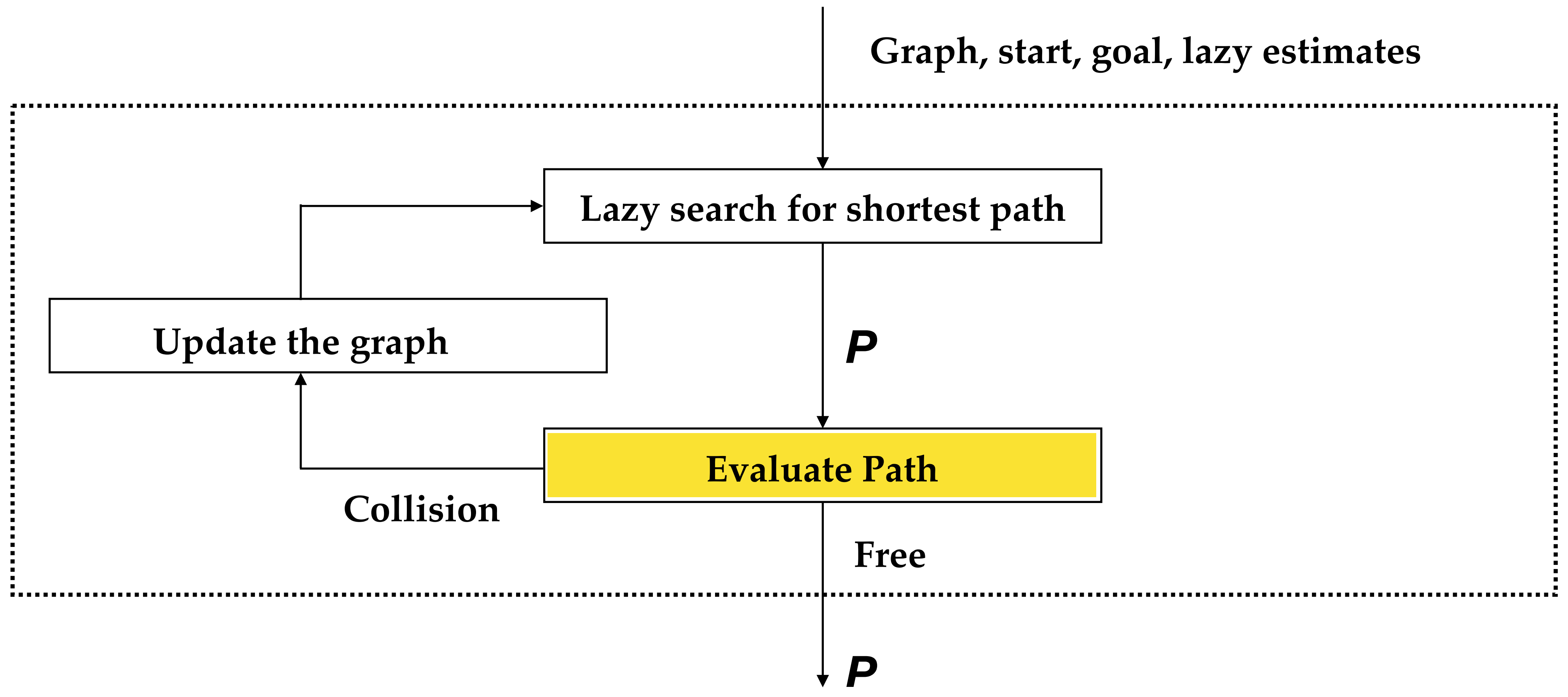


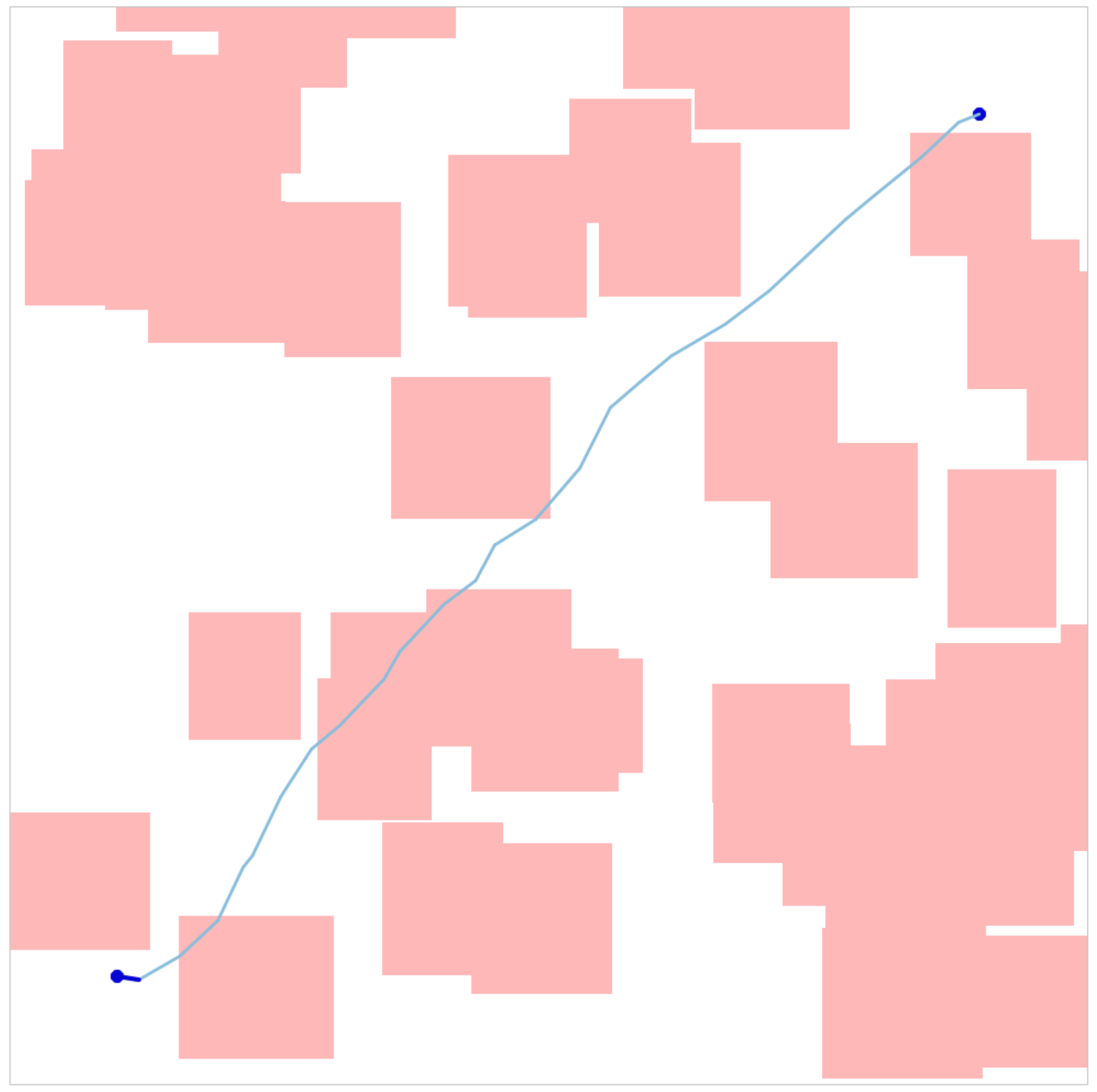


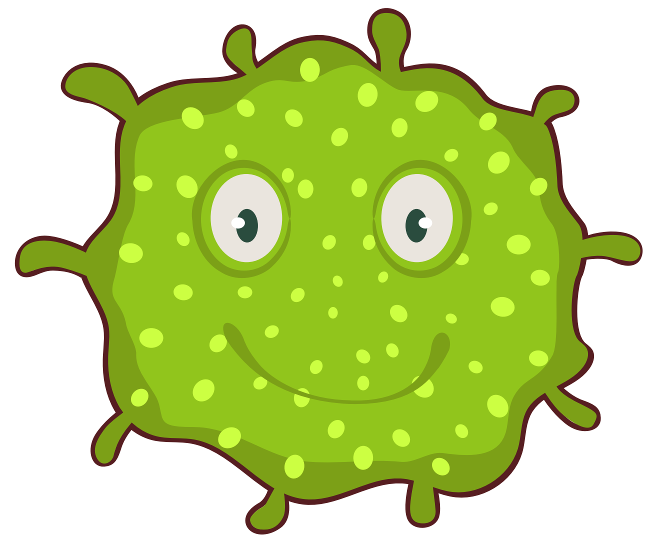
Send out the Ghost Amoebas

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

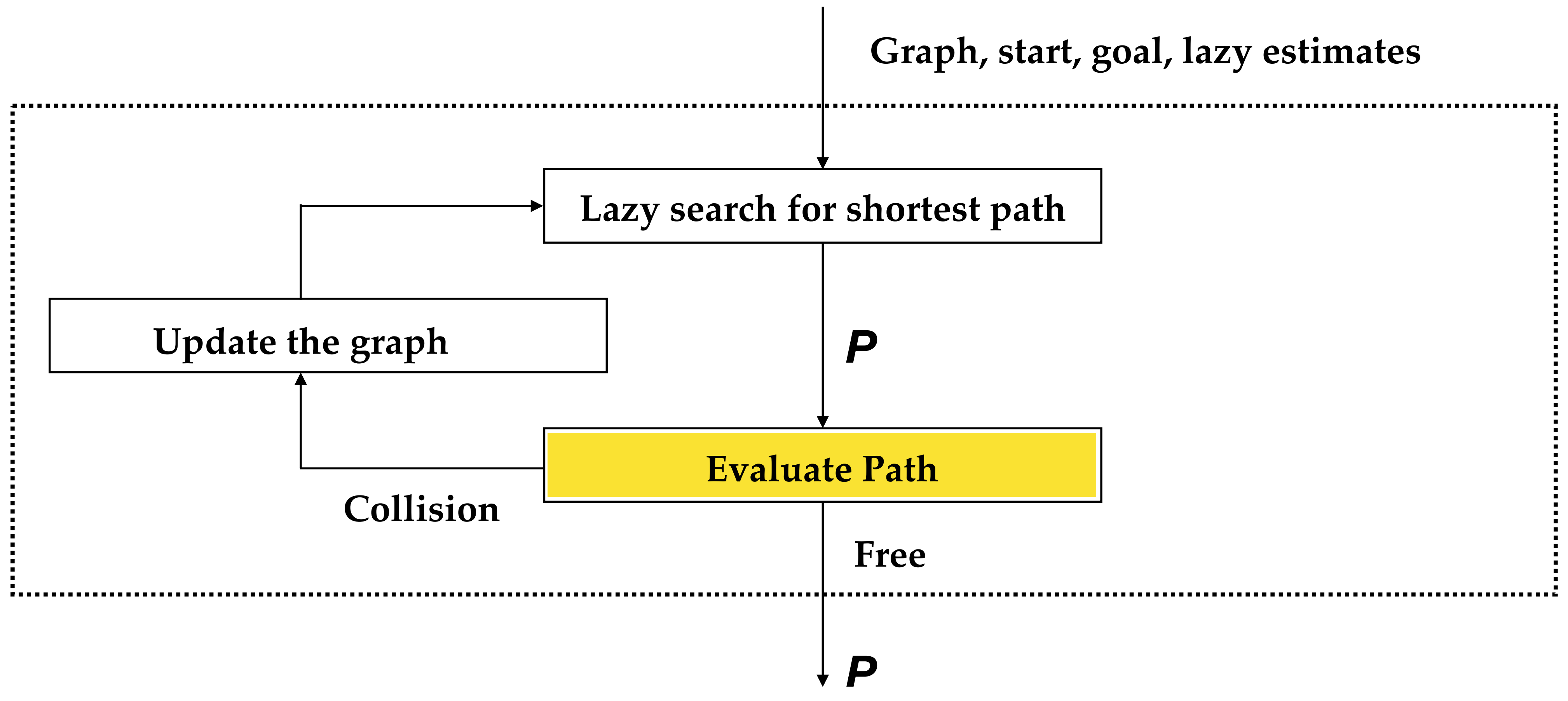


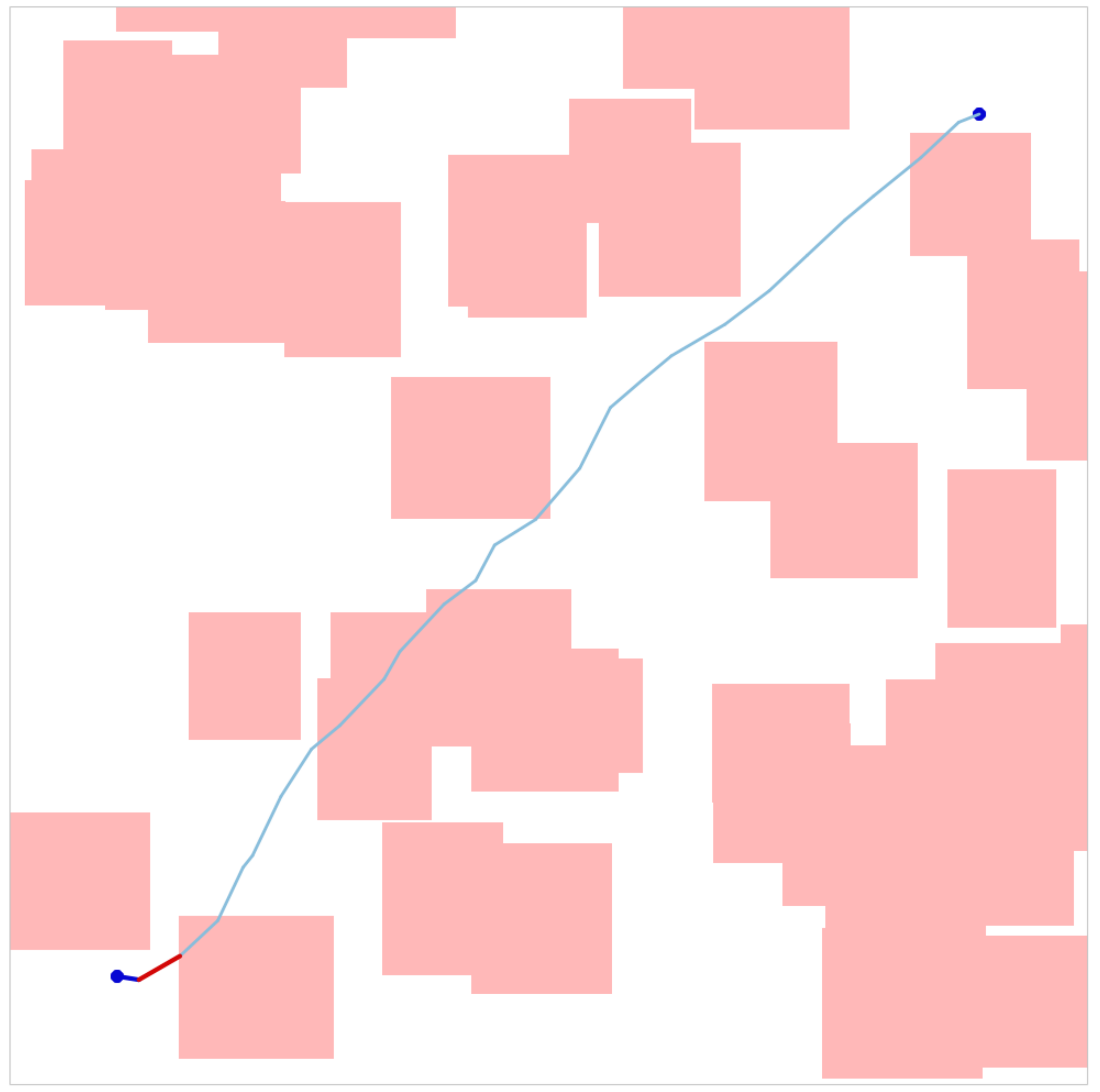


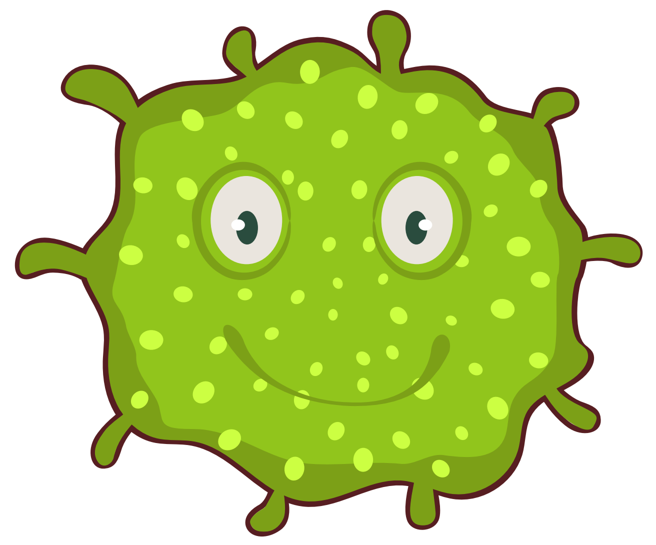
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

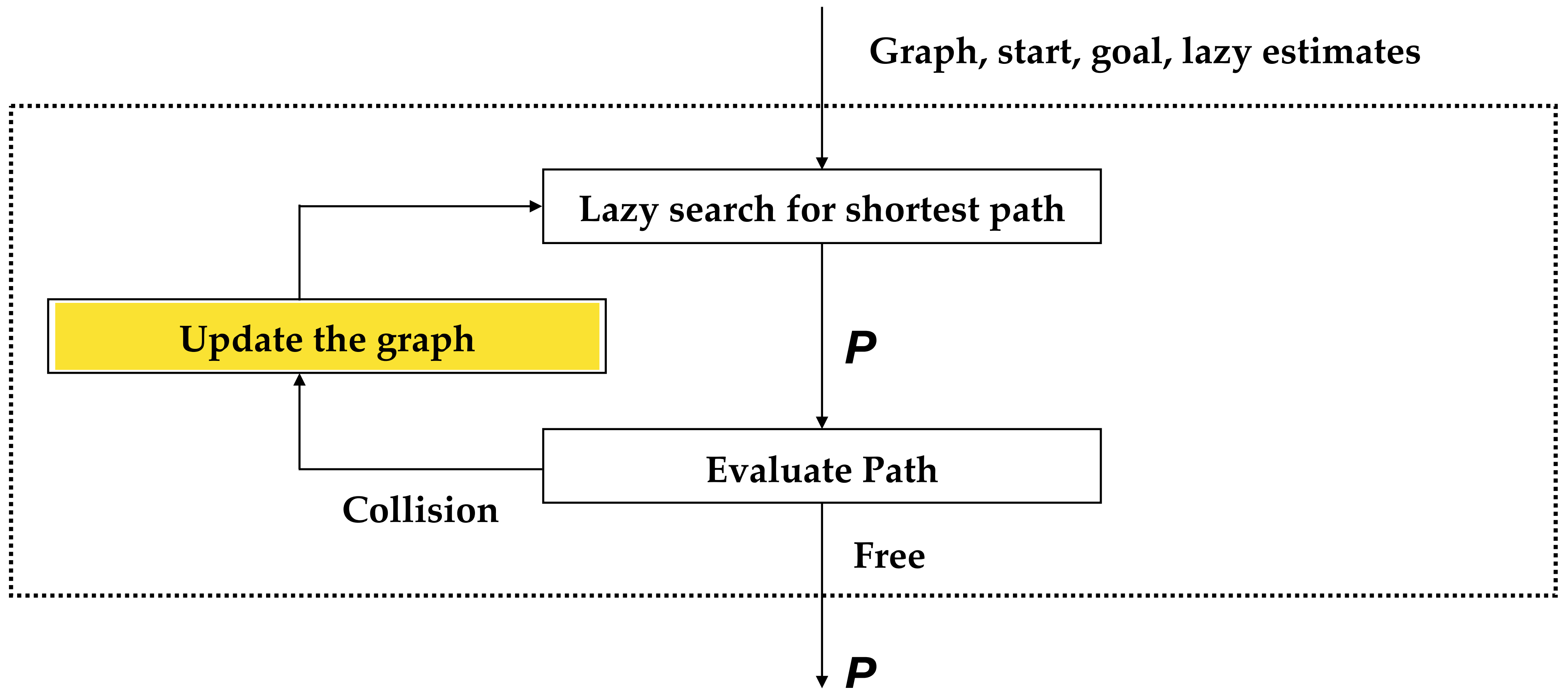


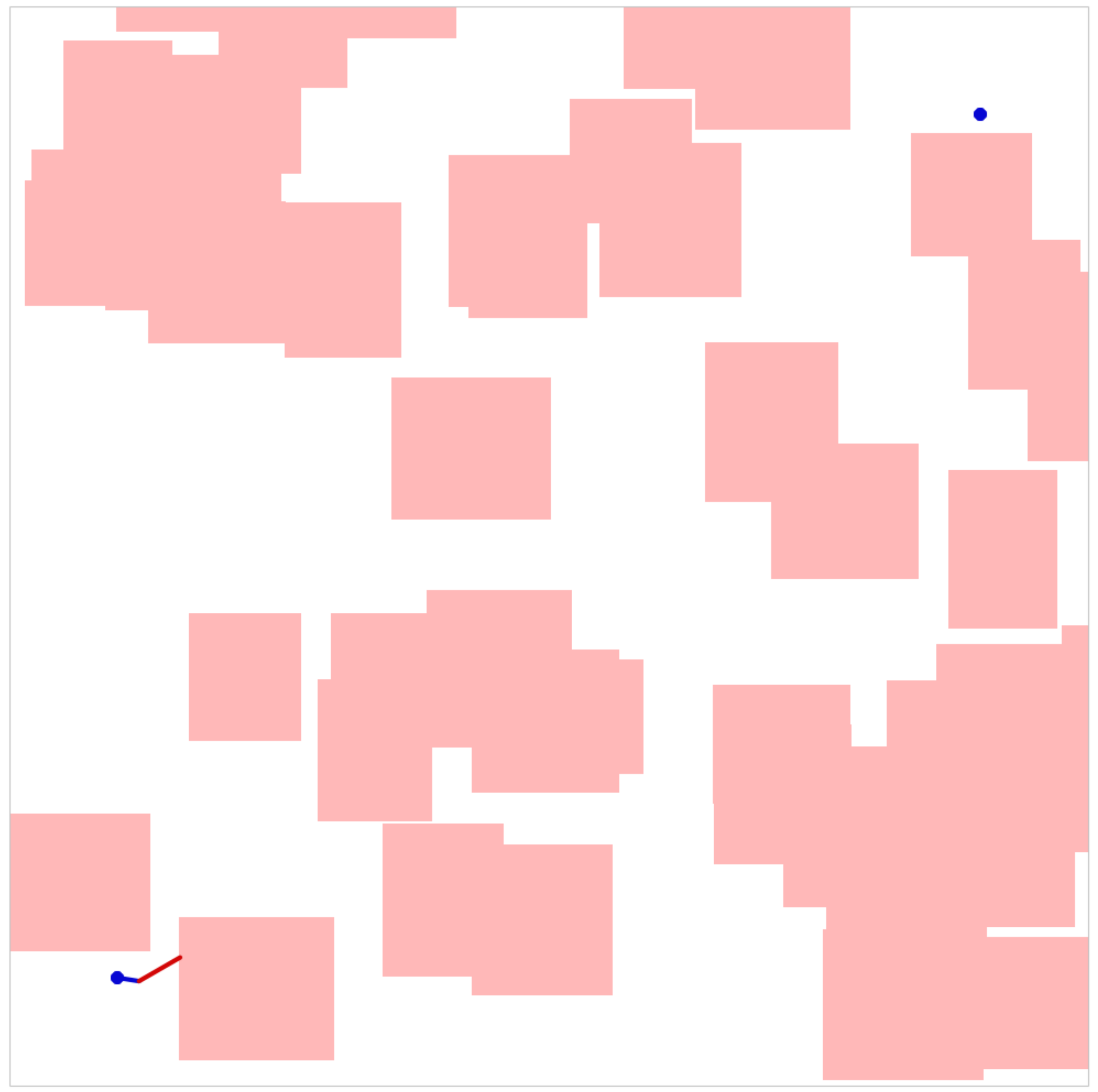


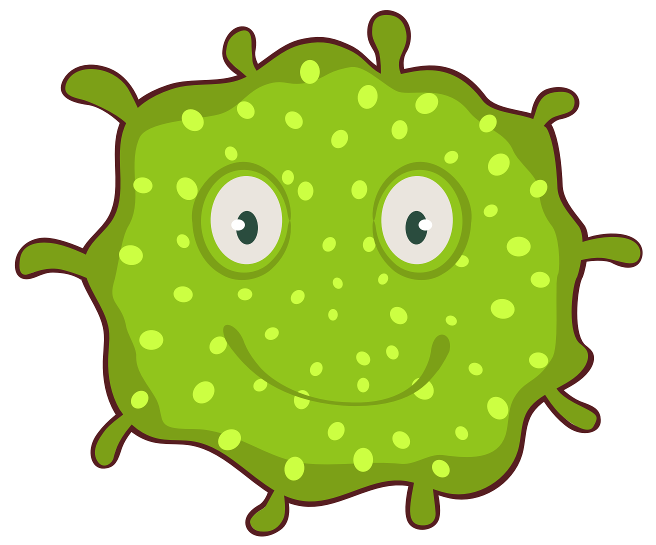
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

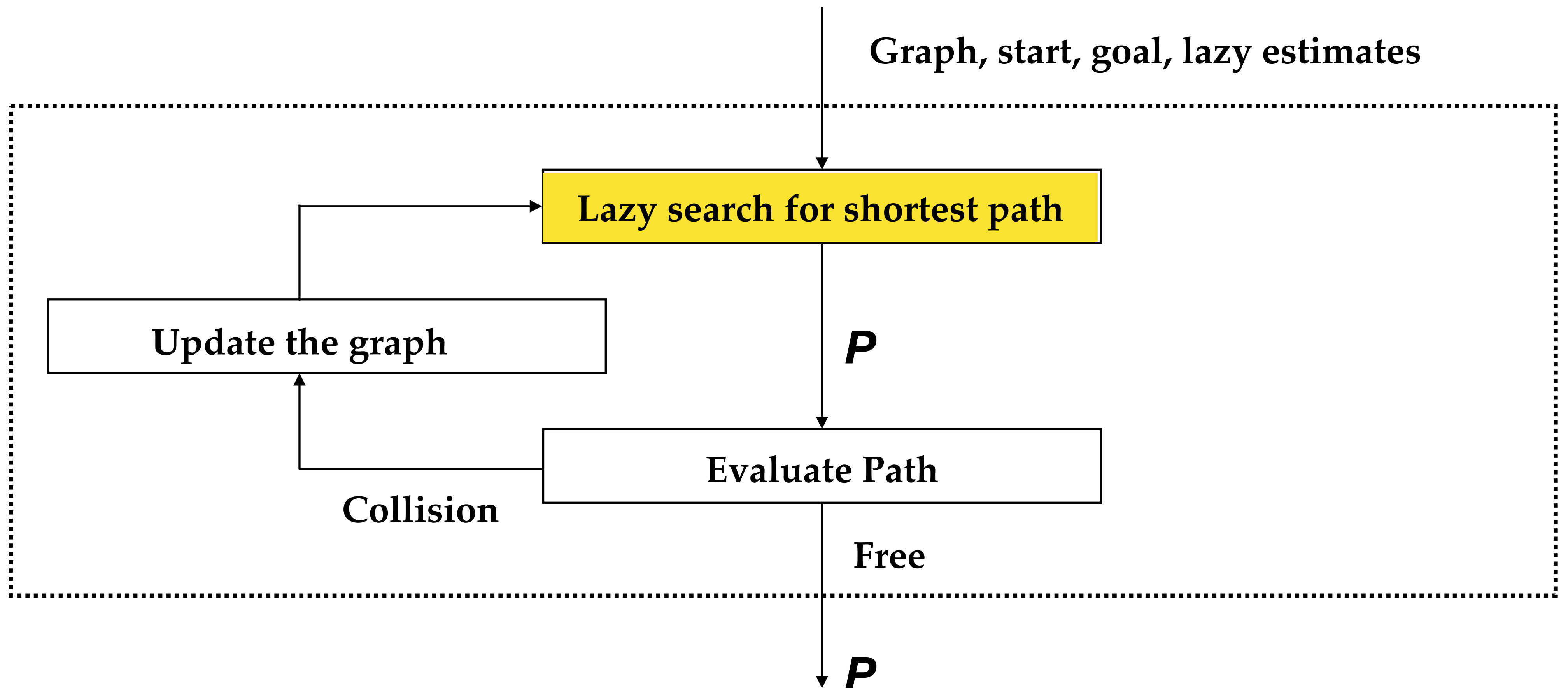


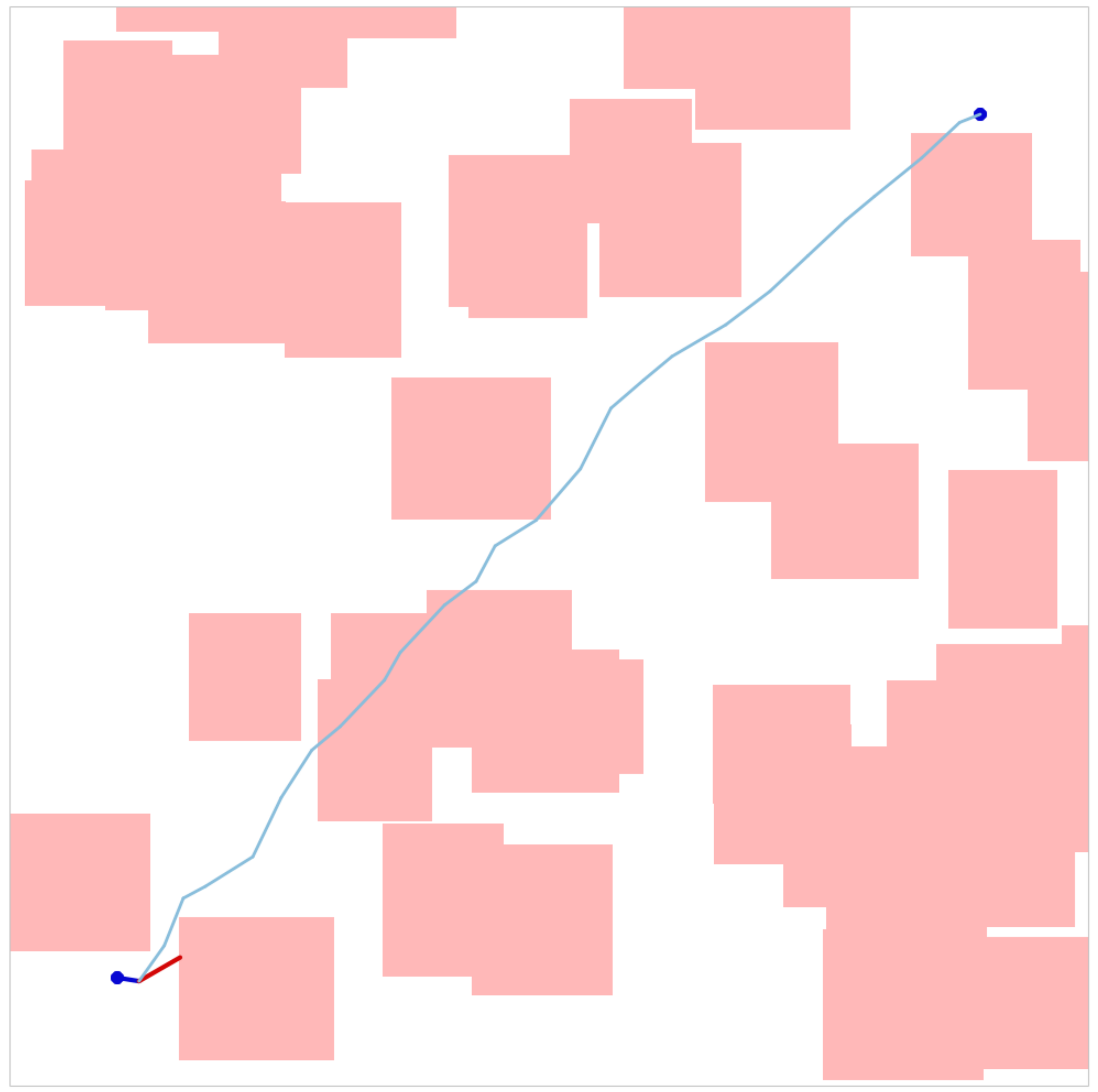


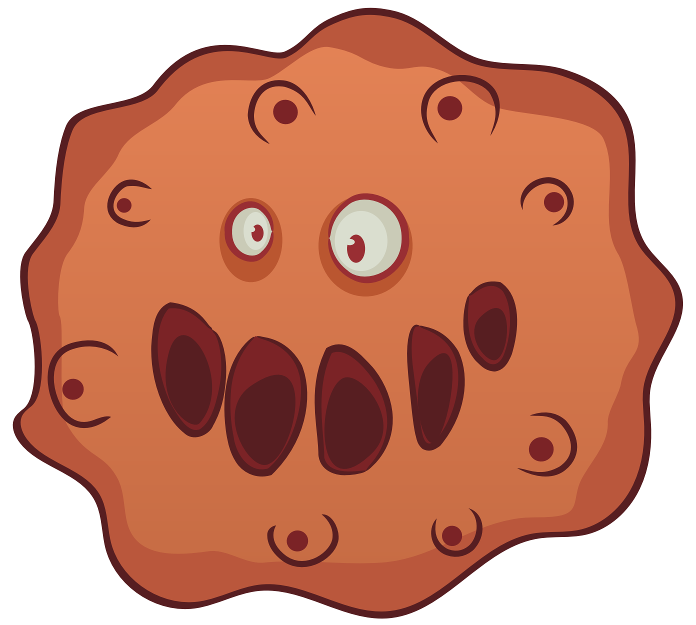
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

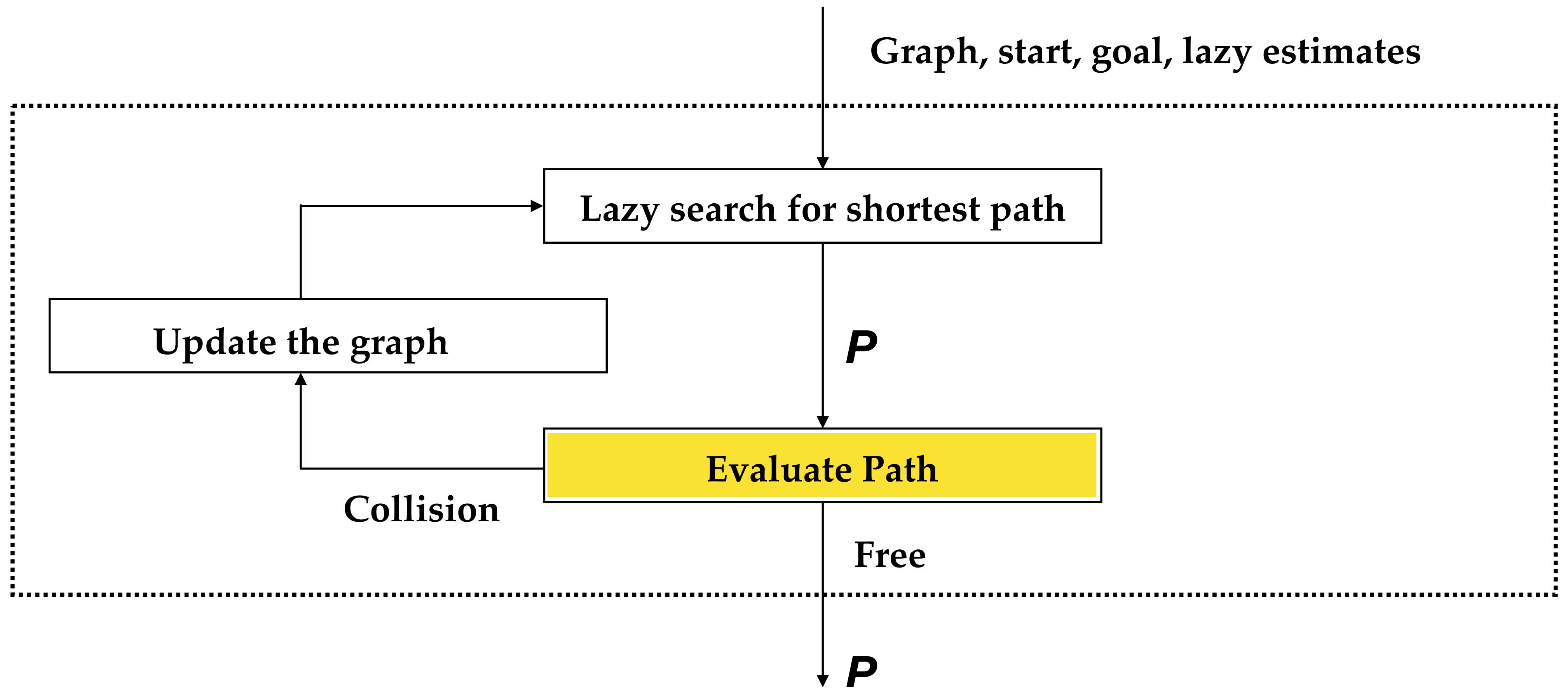


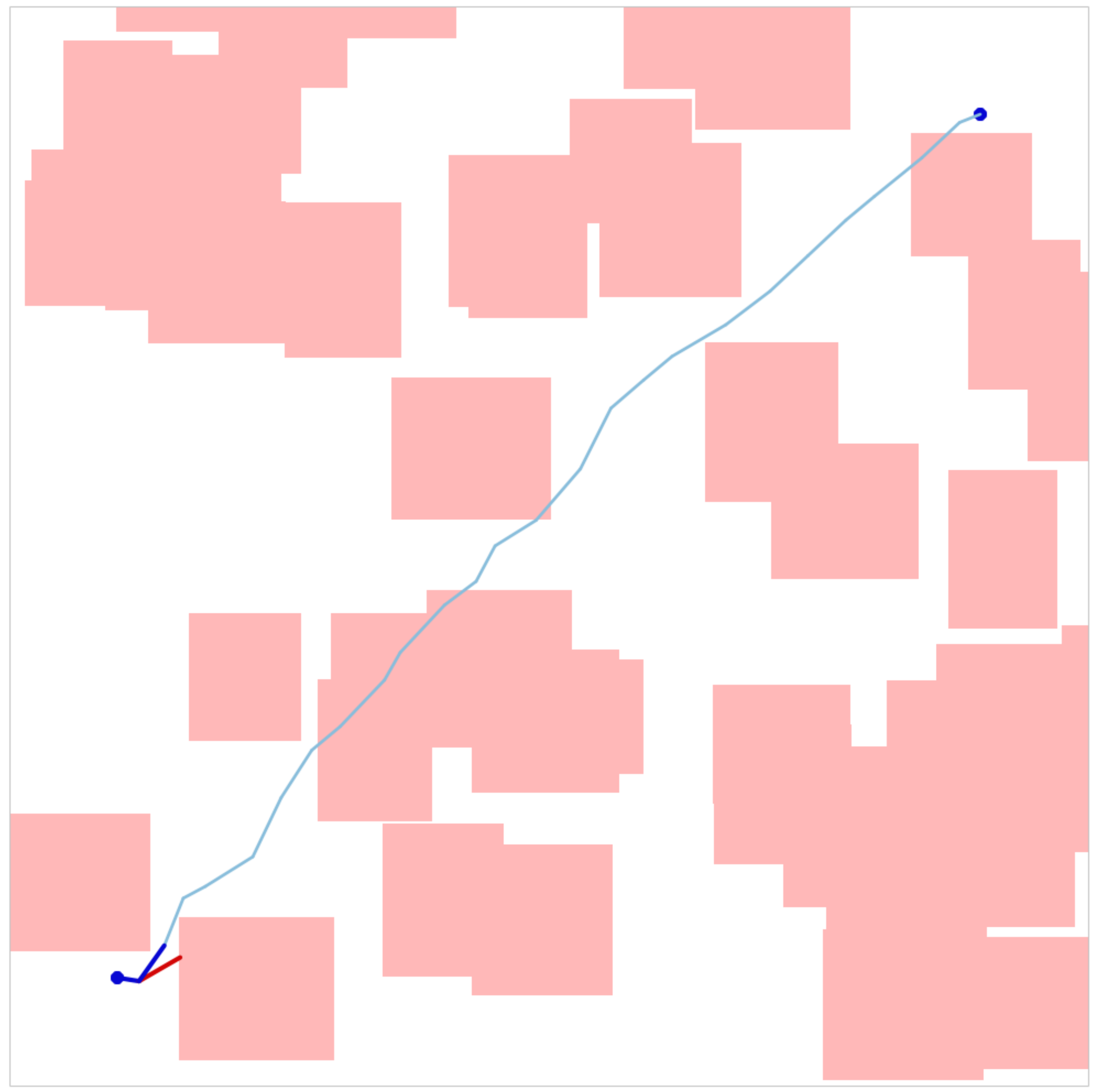


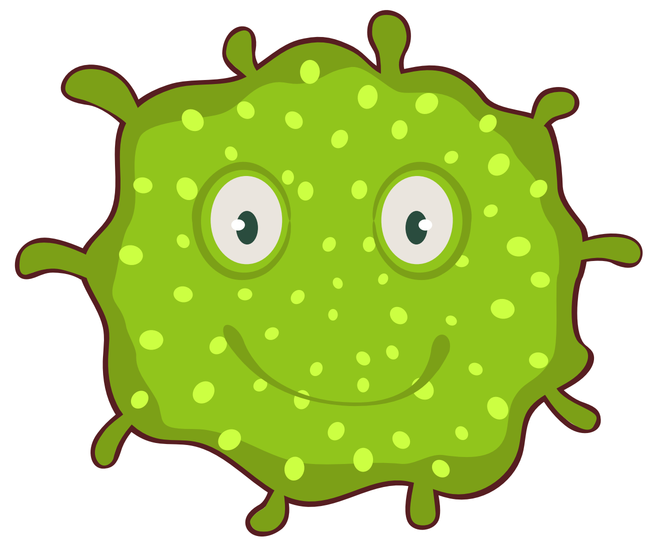
Send out the Ghost Amoebas

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

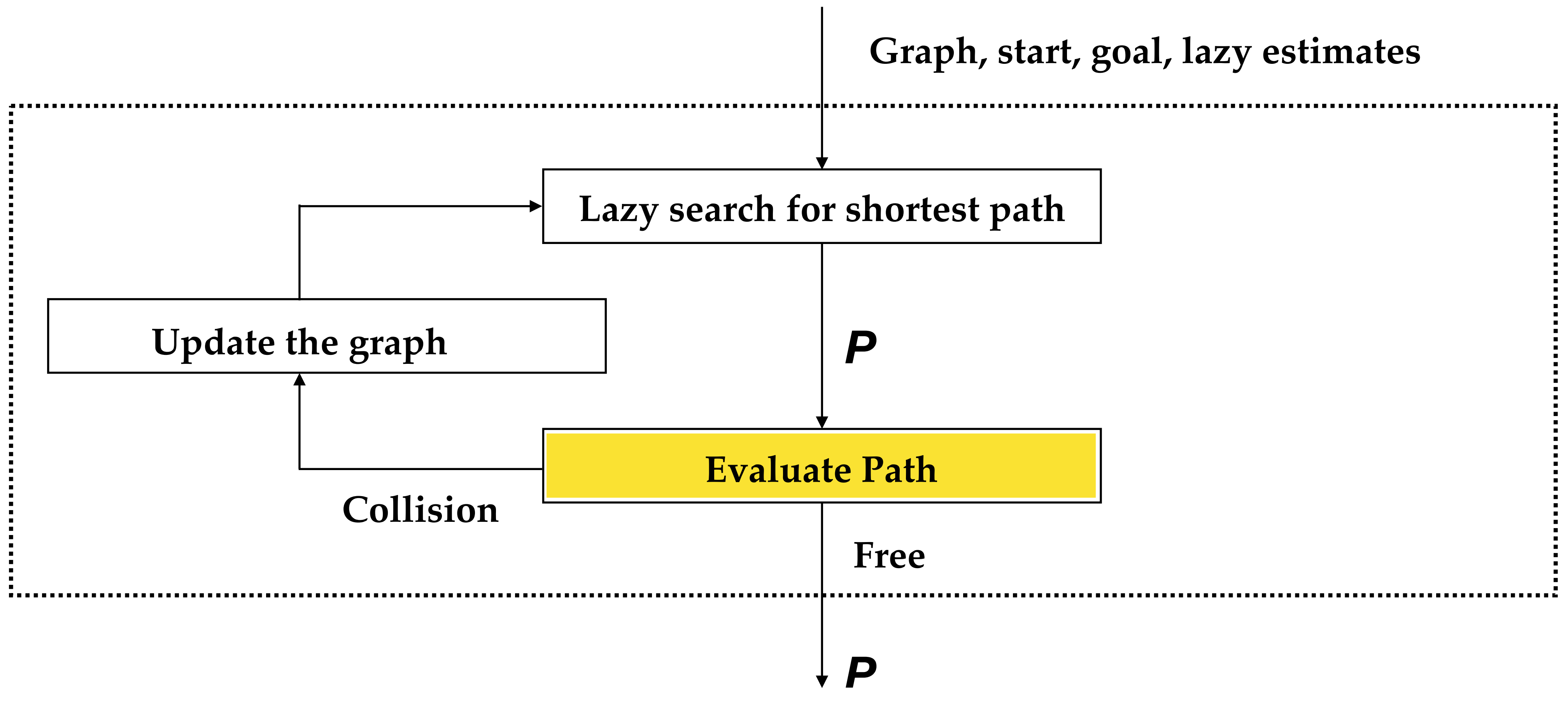


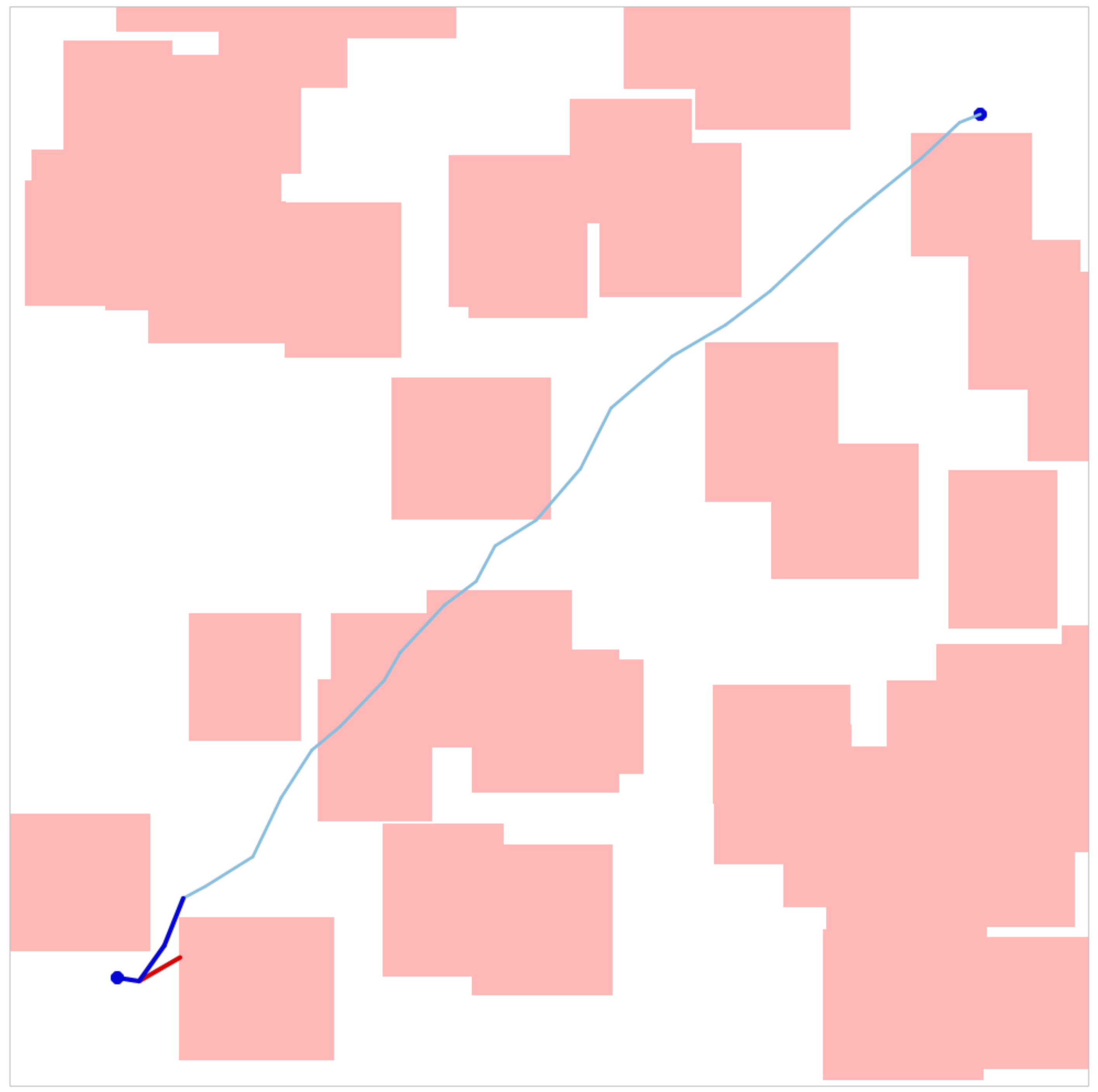


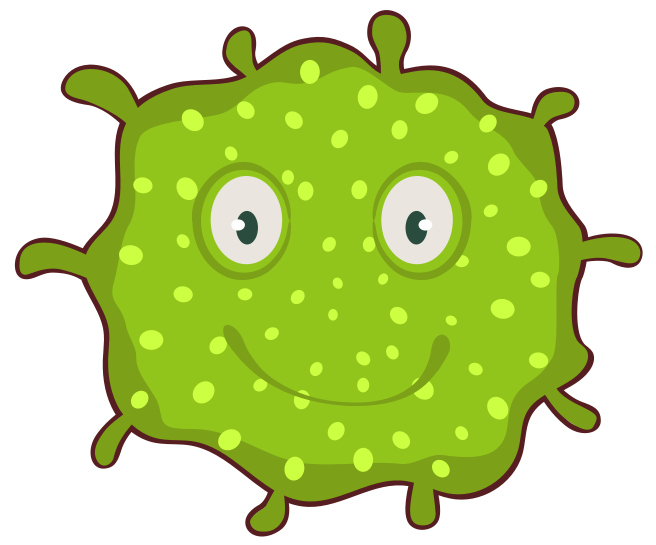
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

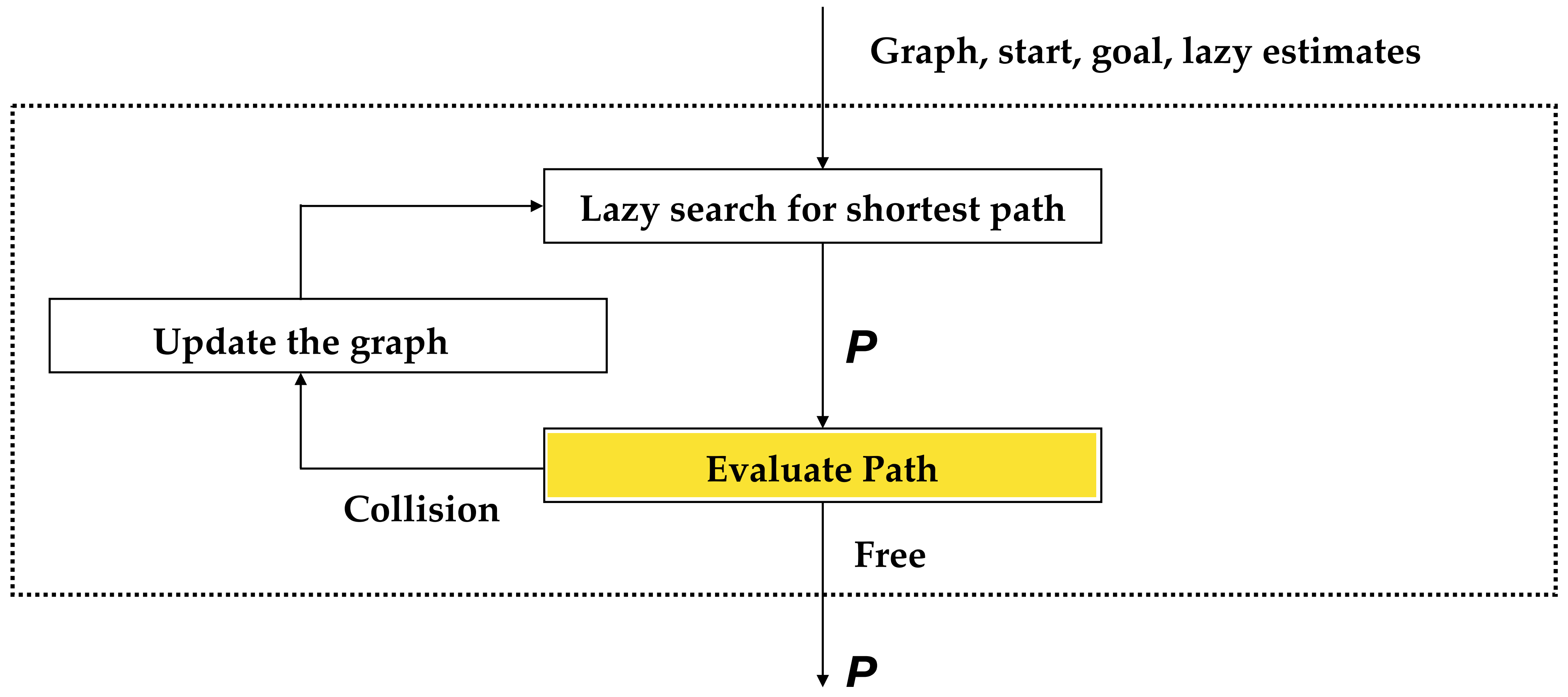


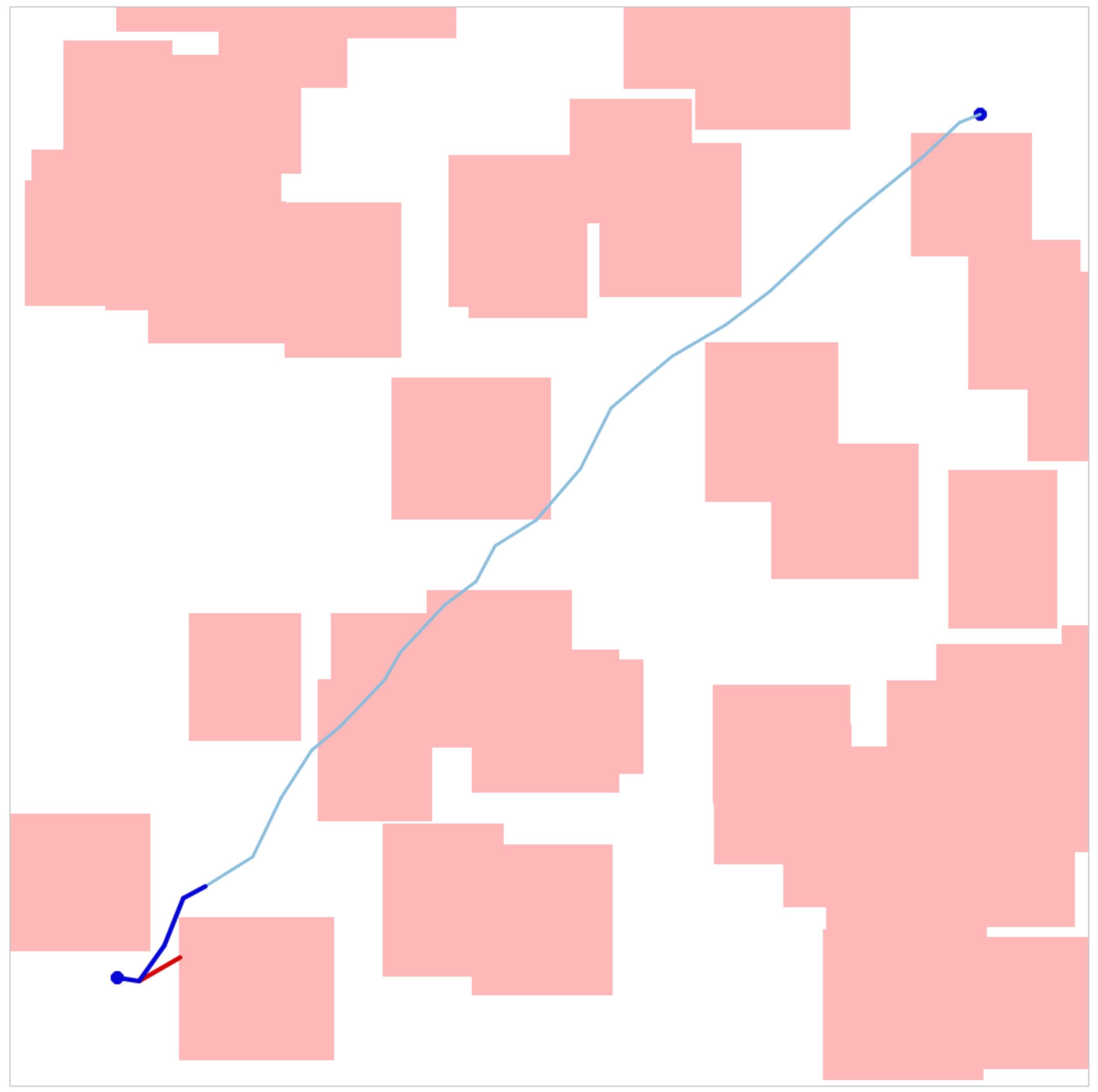


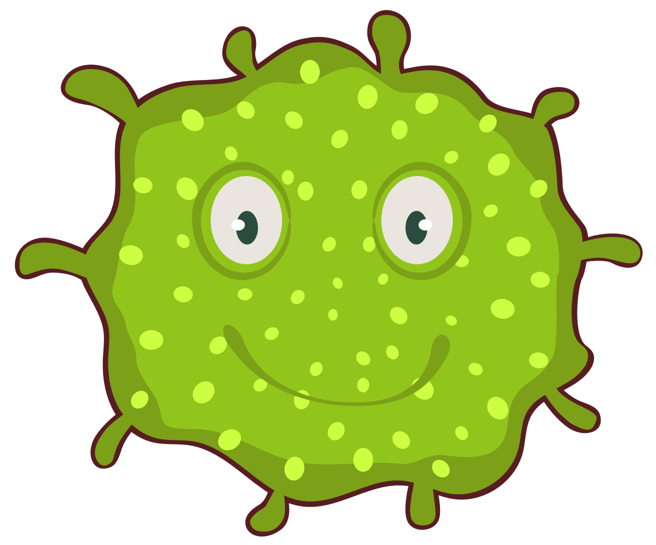
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

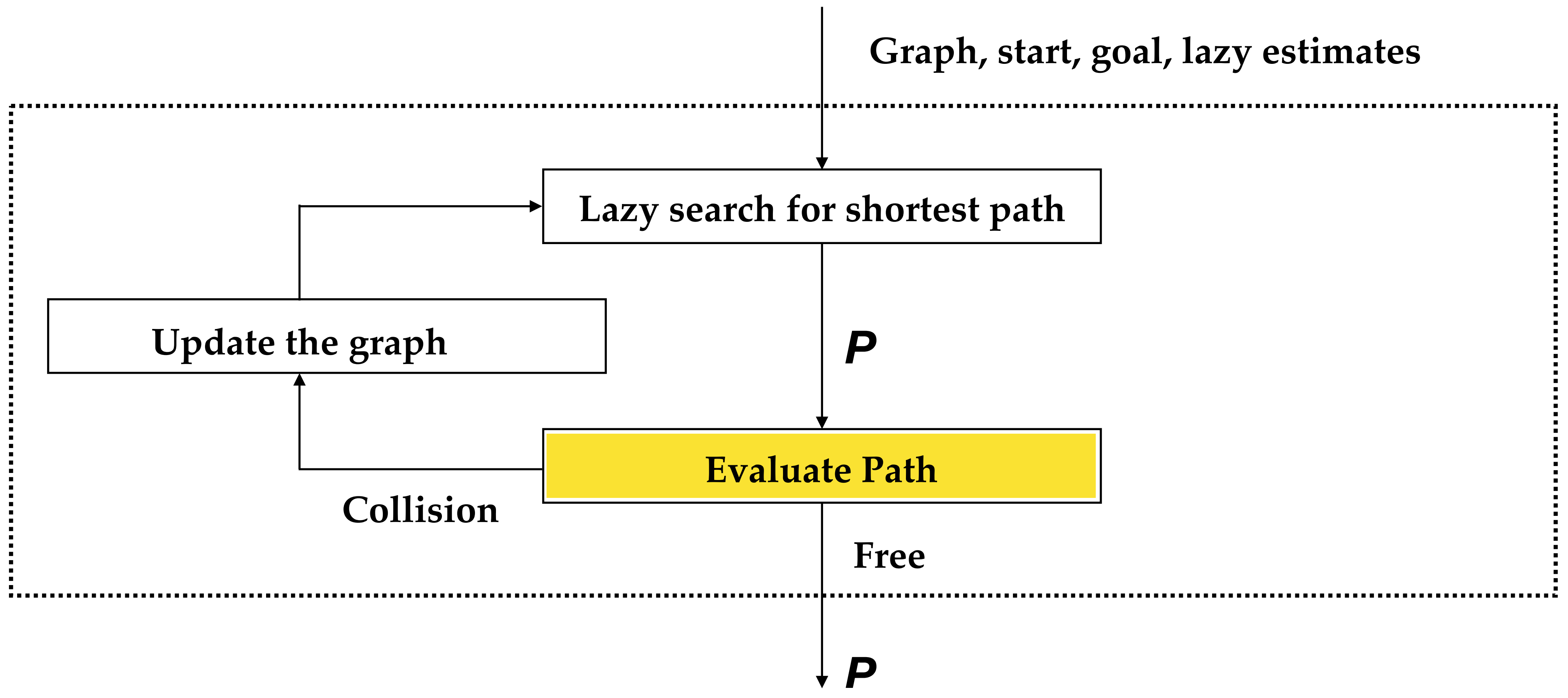




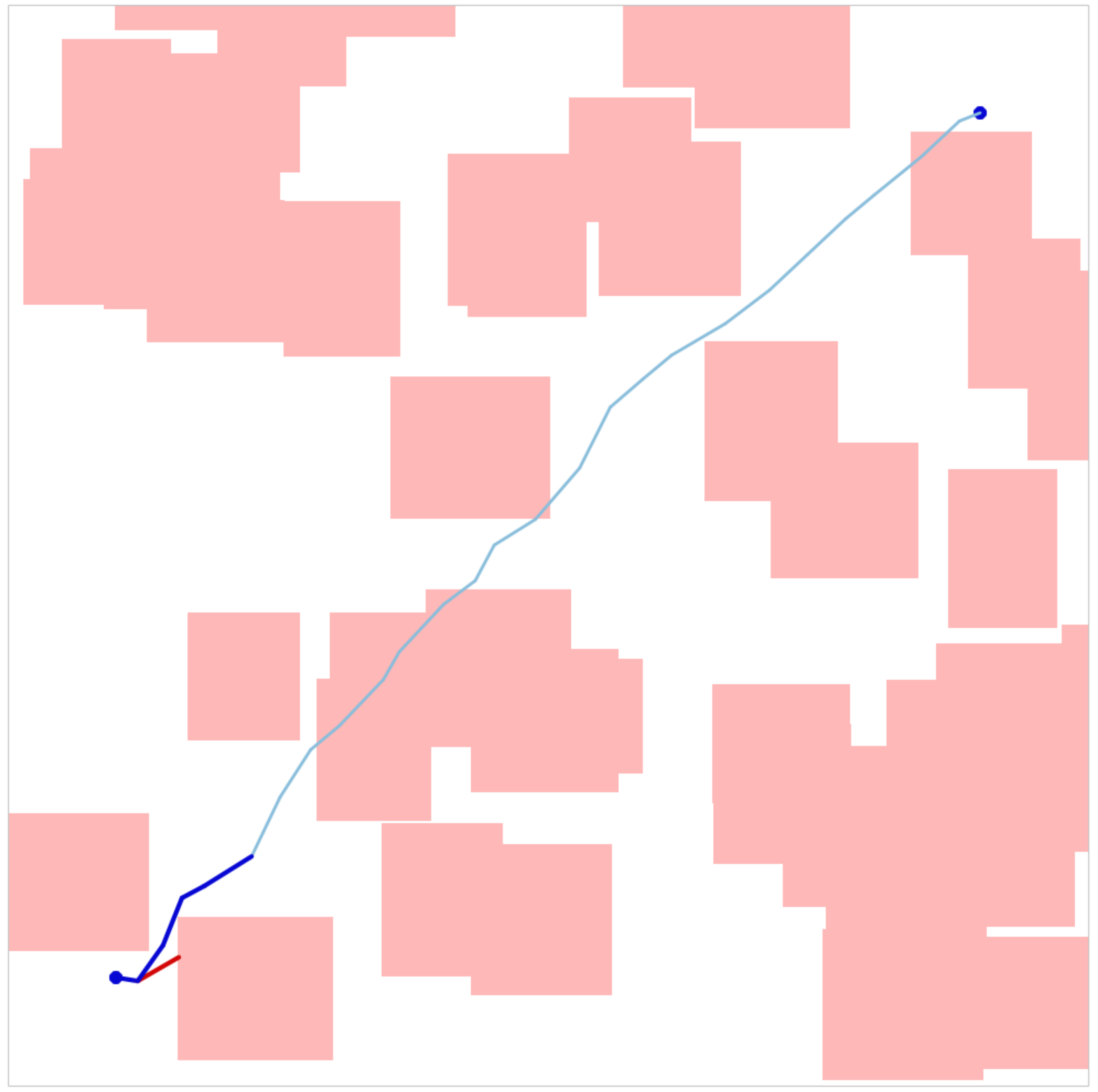
Only Slime Known Shortest Paths

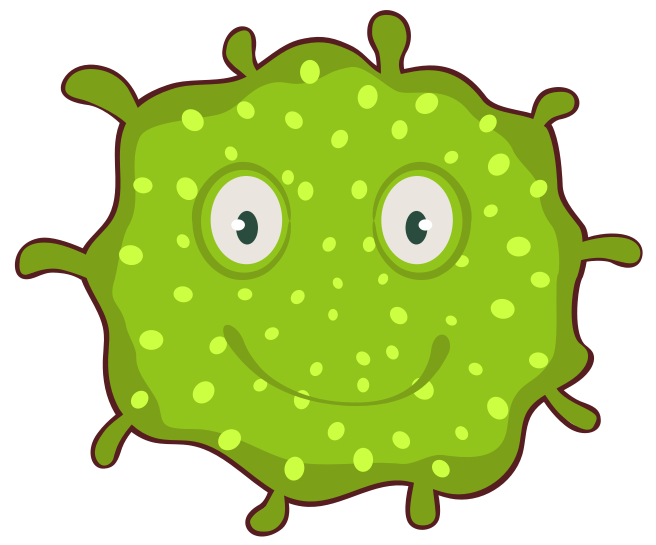
LazySP

OFU on Steroids!



Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

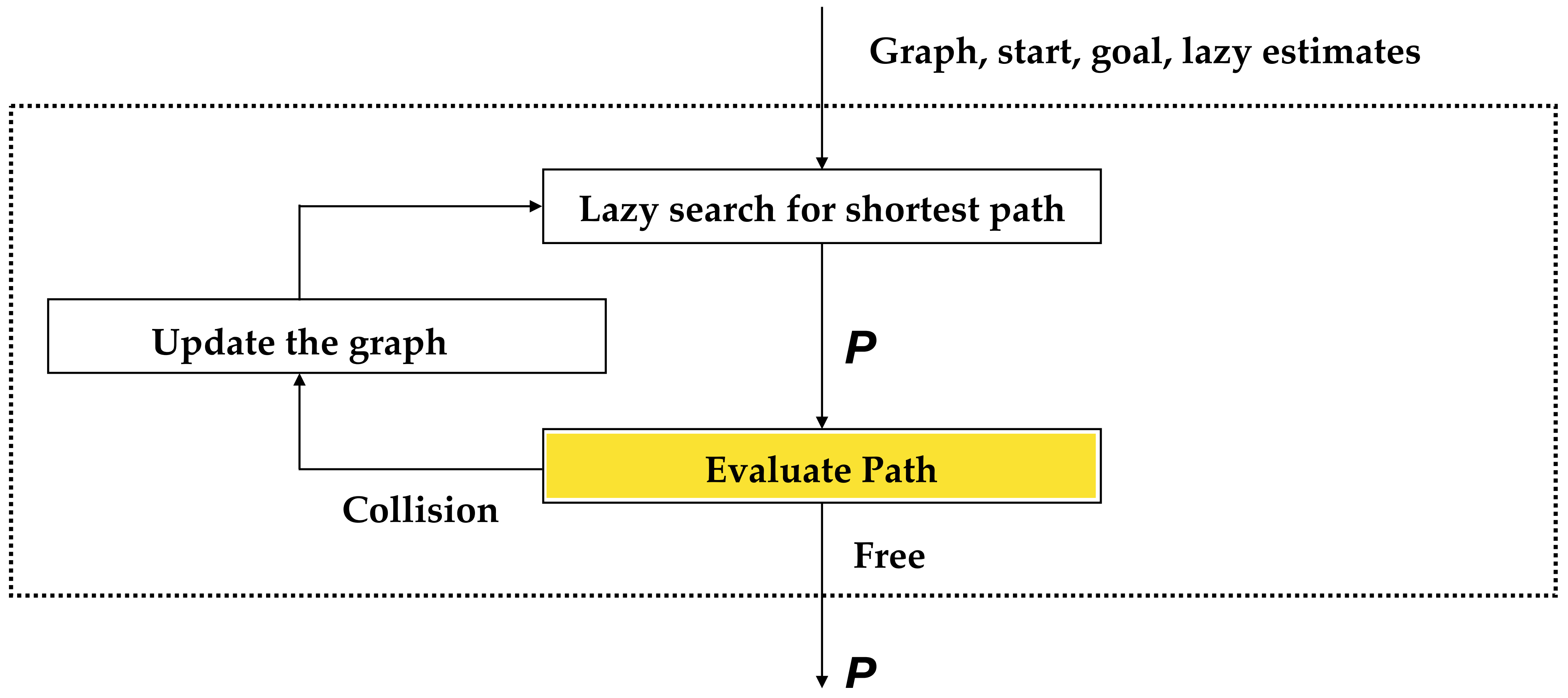


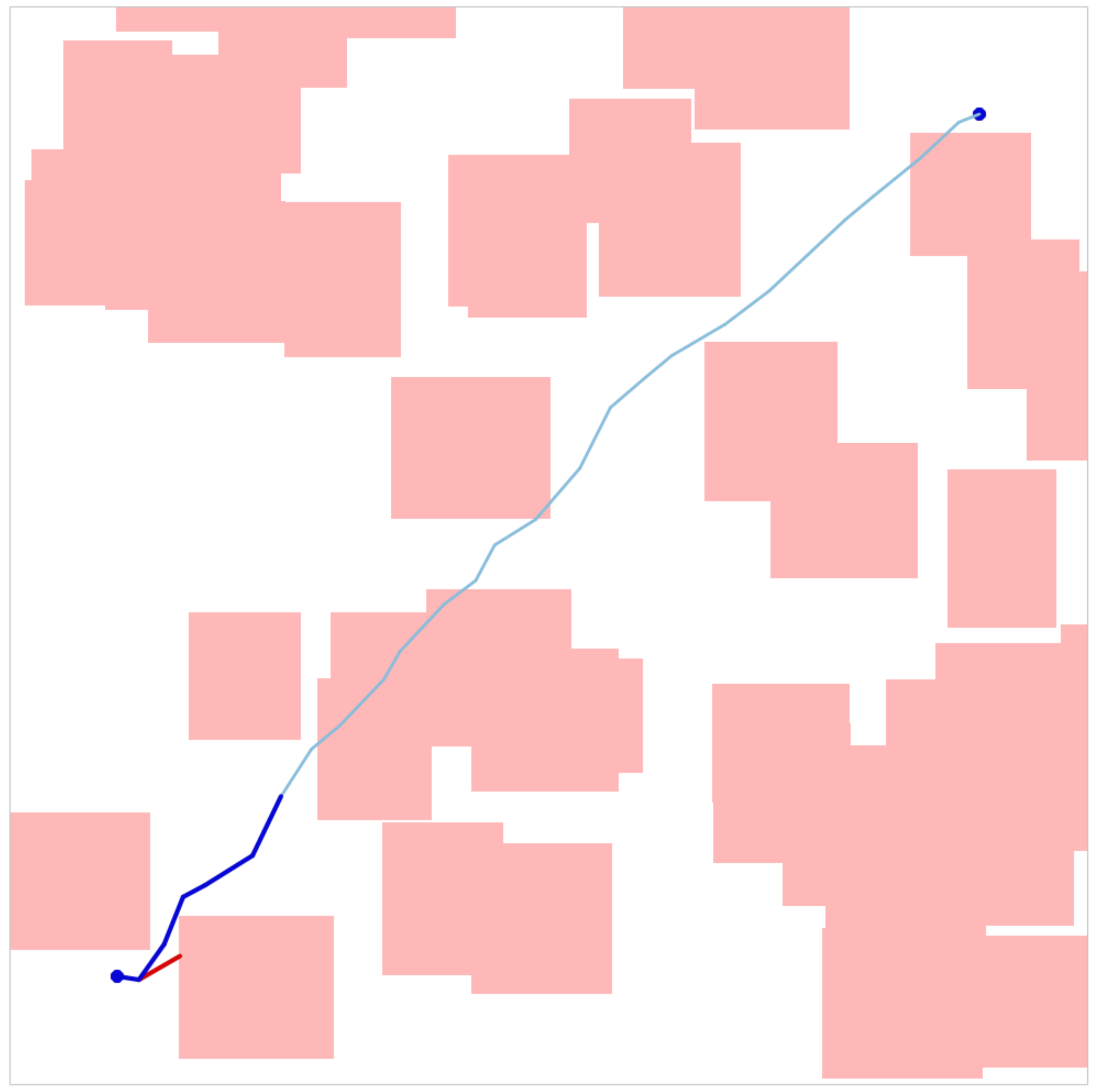


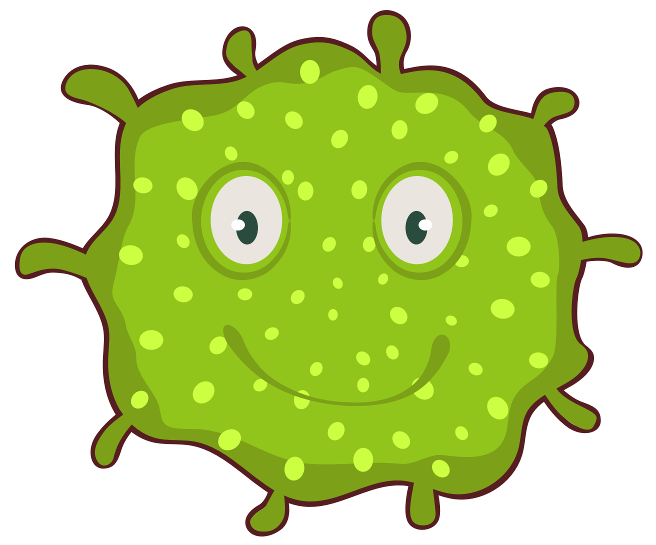
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

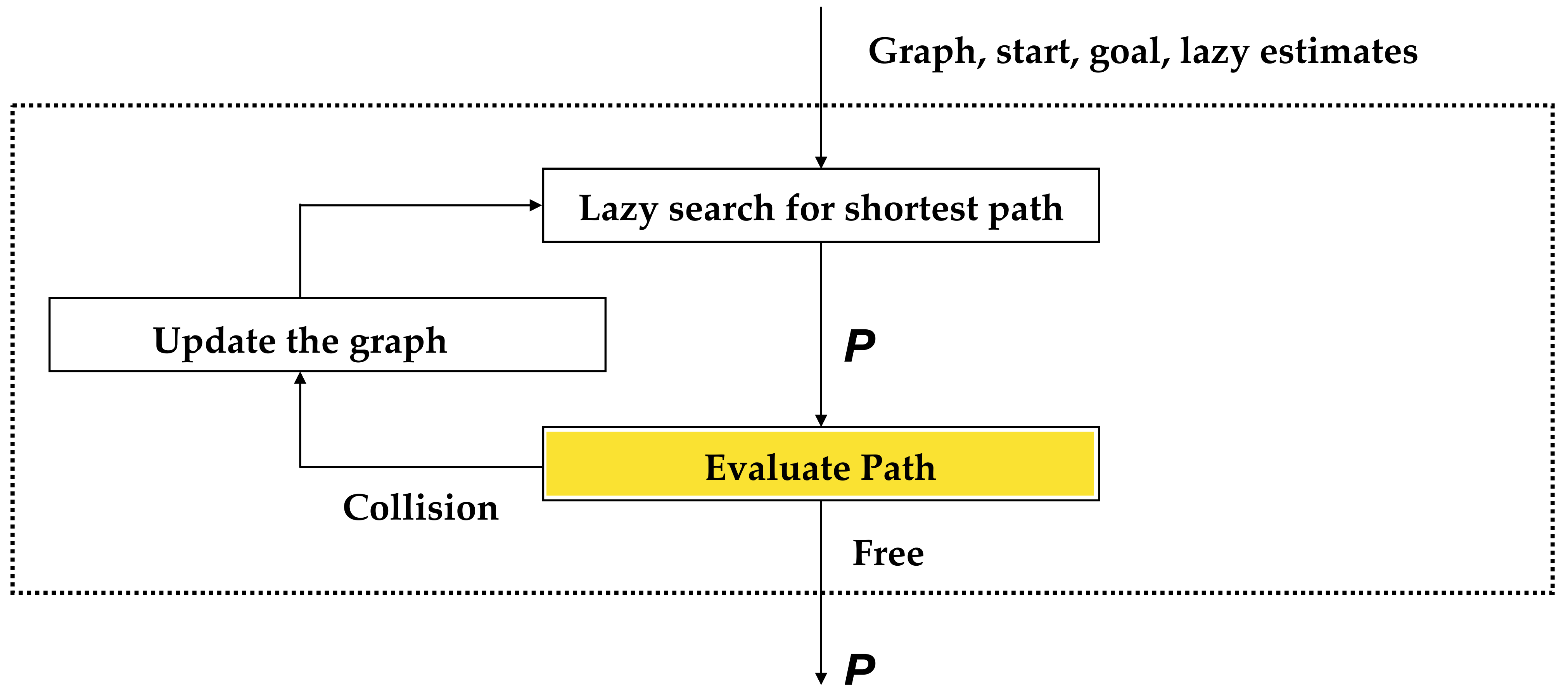


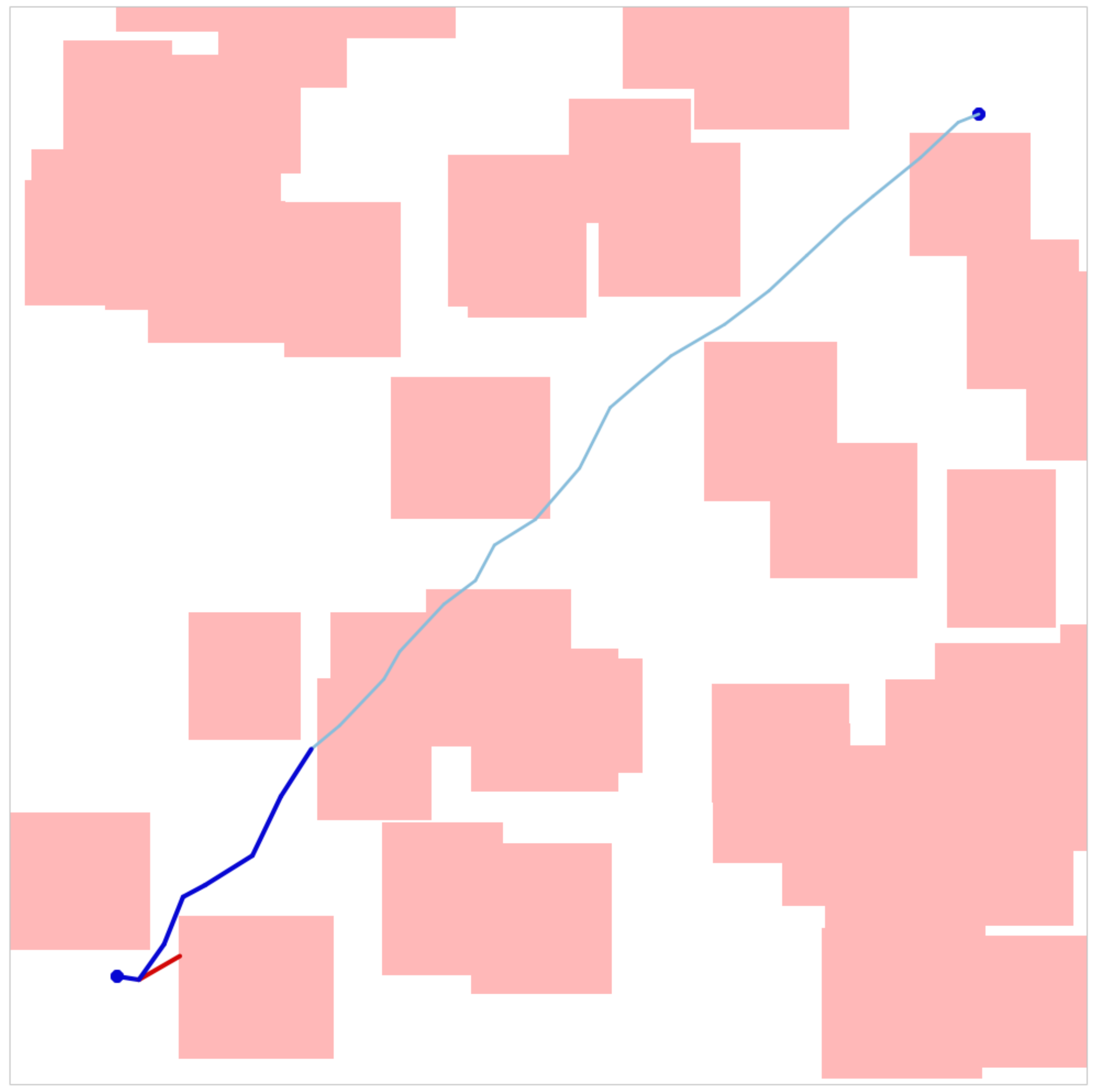


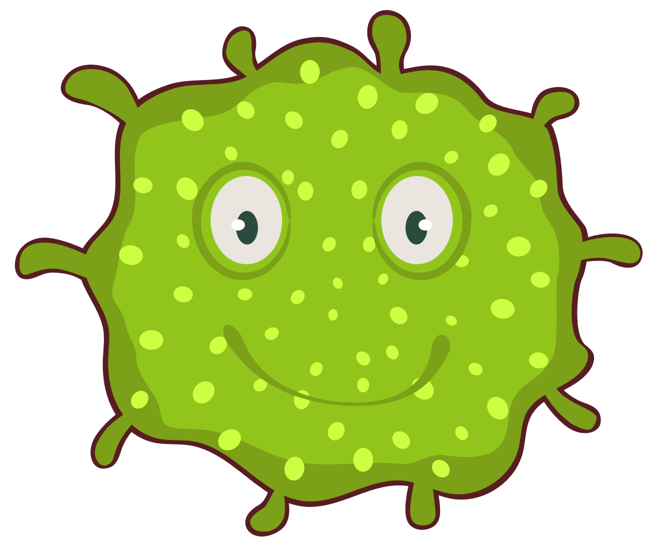
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

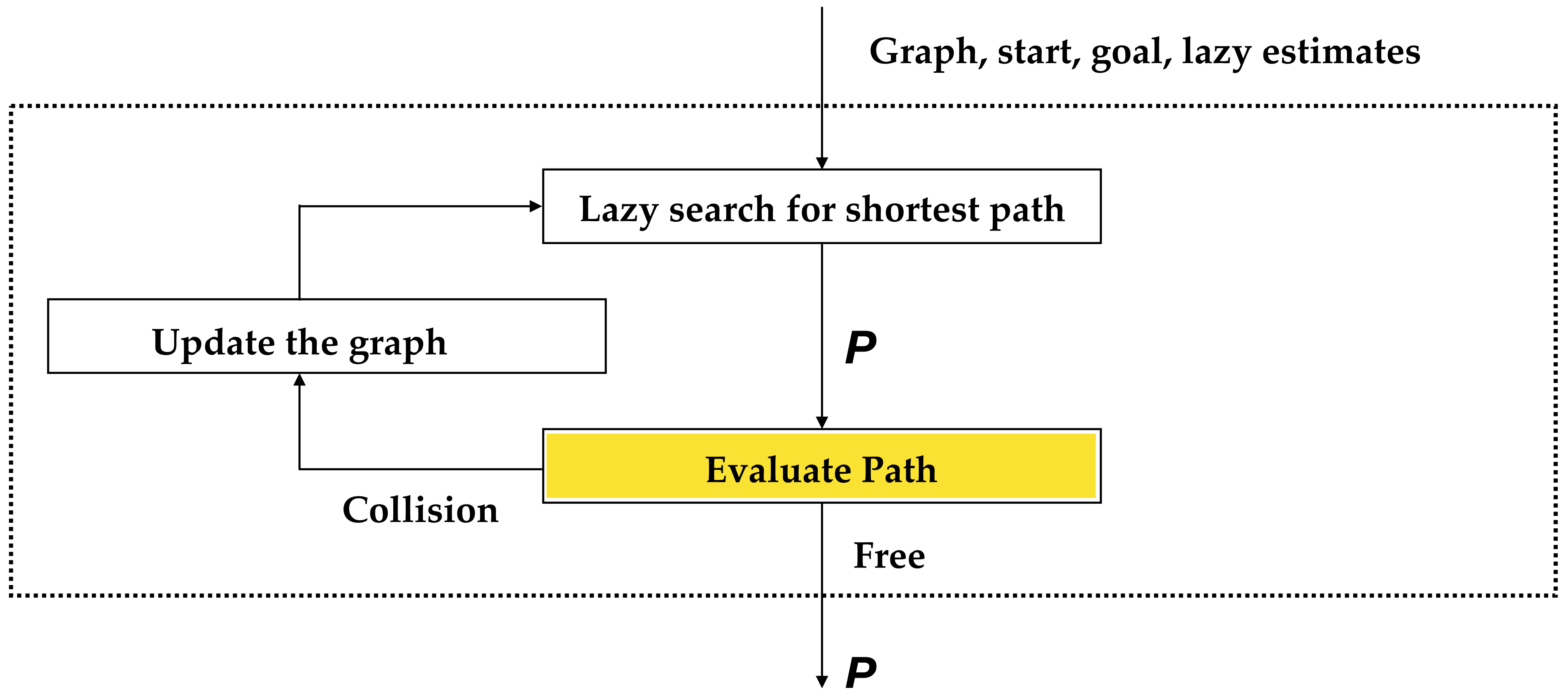


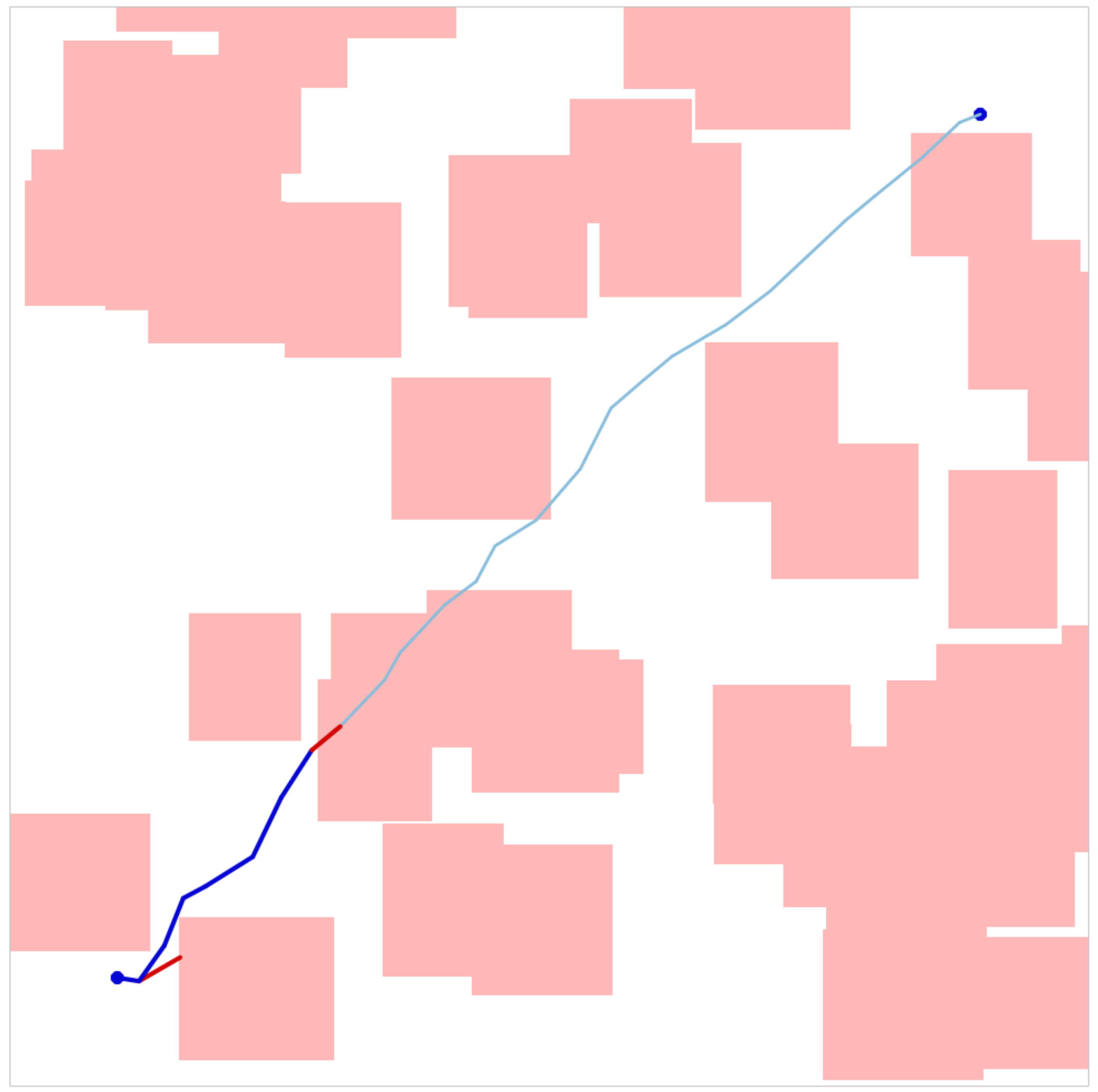


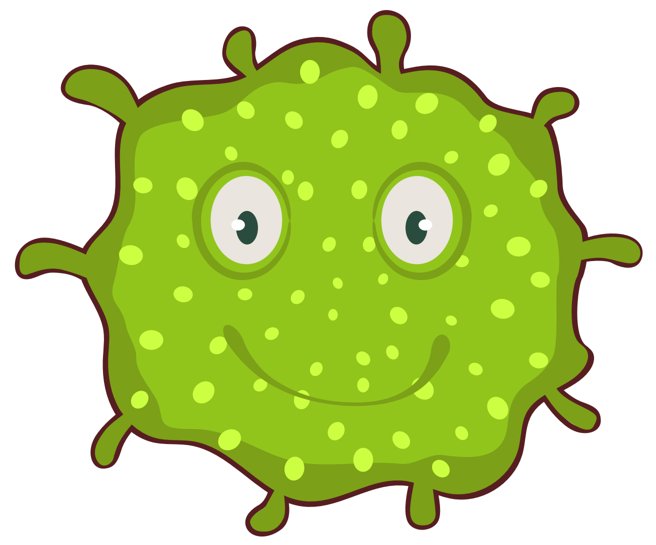
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

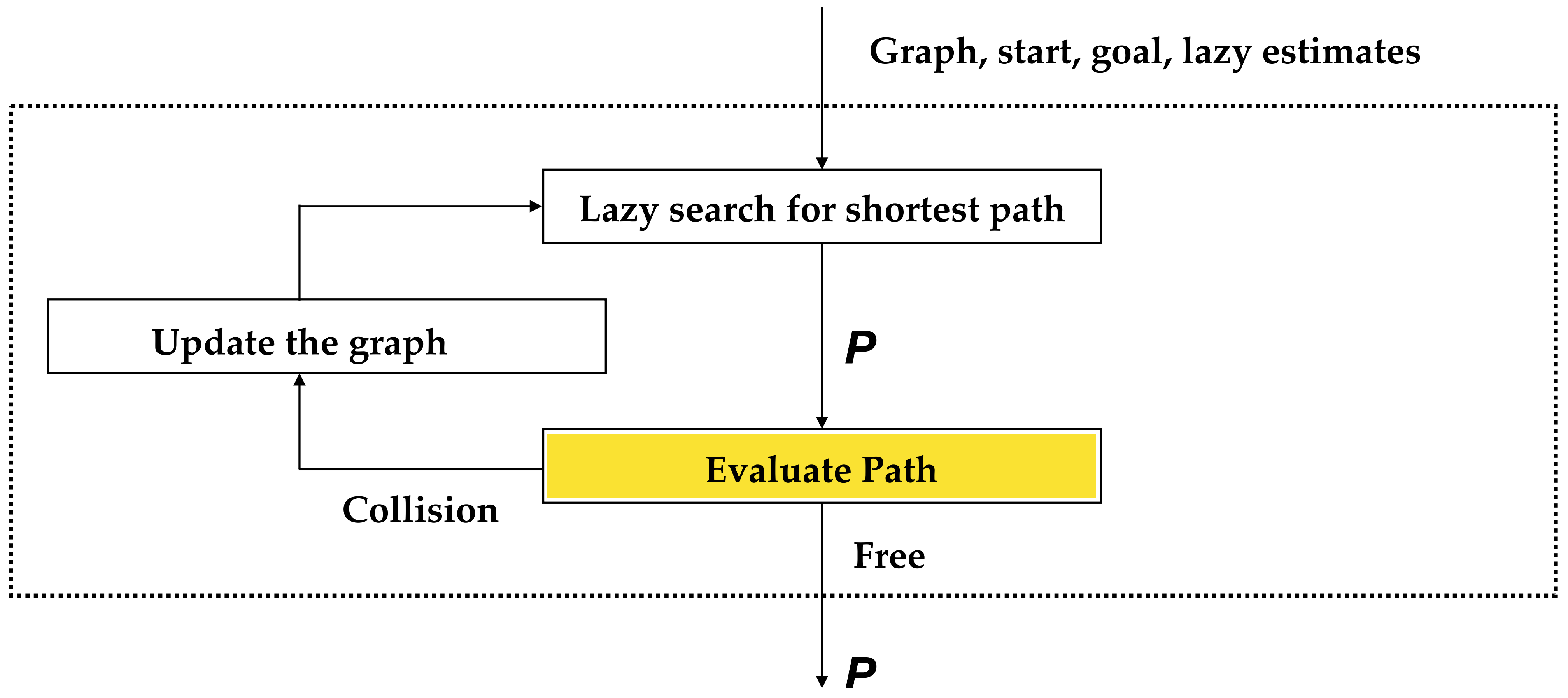


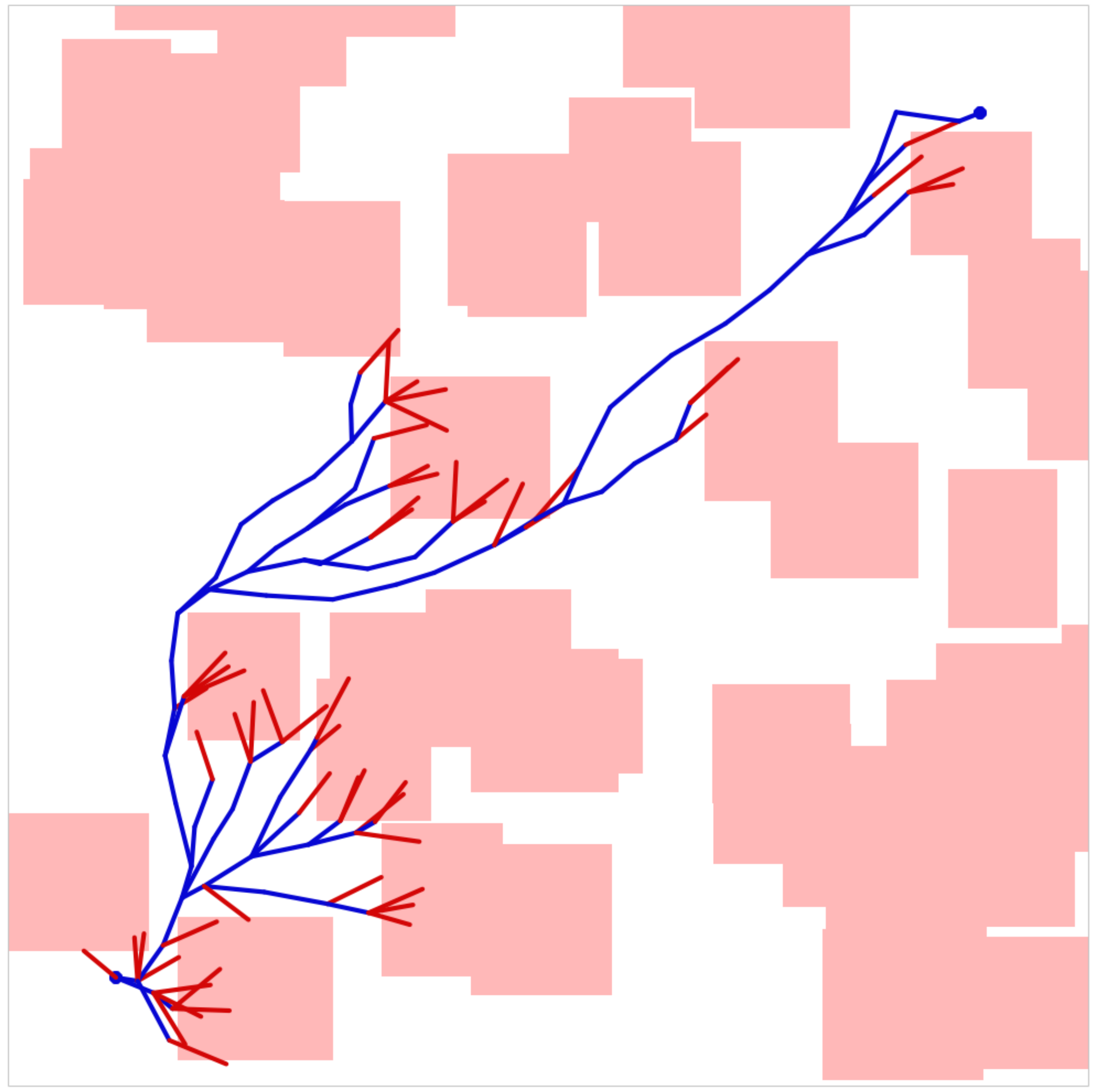


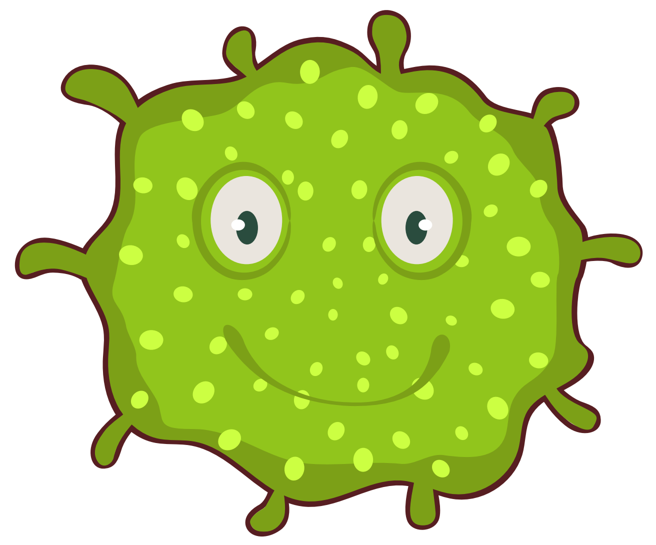
Only Slime Known Shortest Paths

LazySP

OFU on Steroids!


Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P

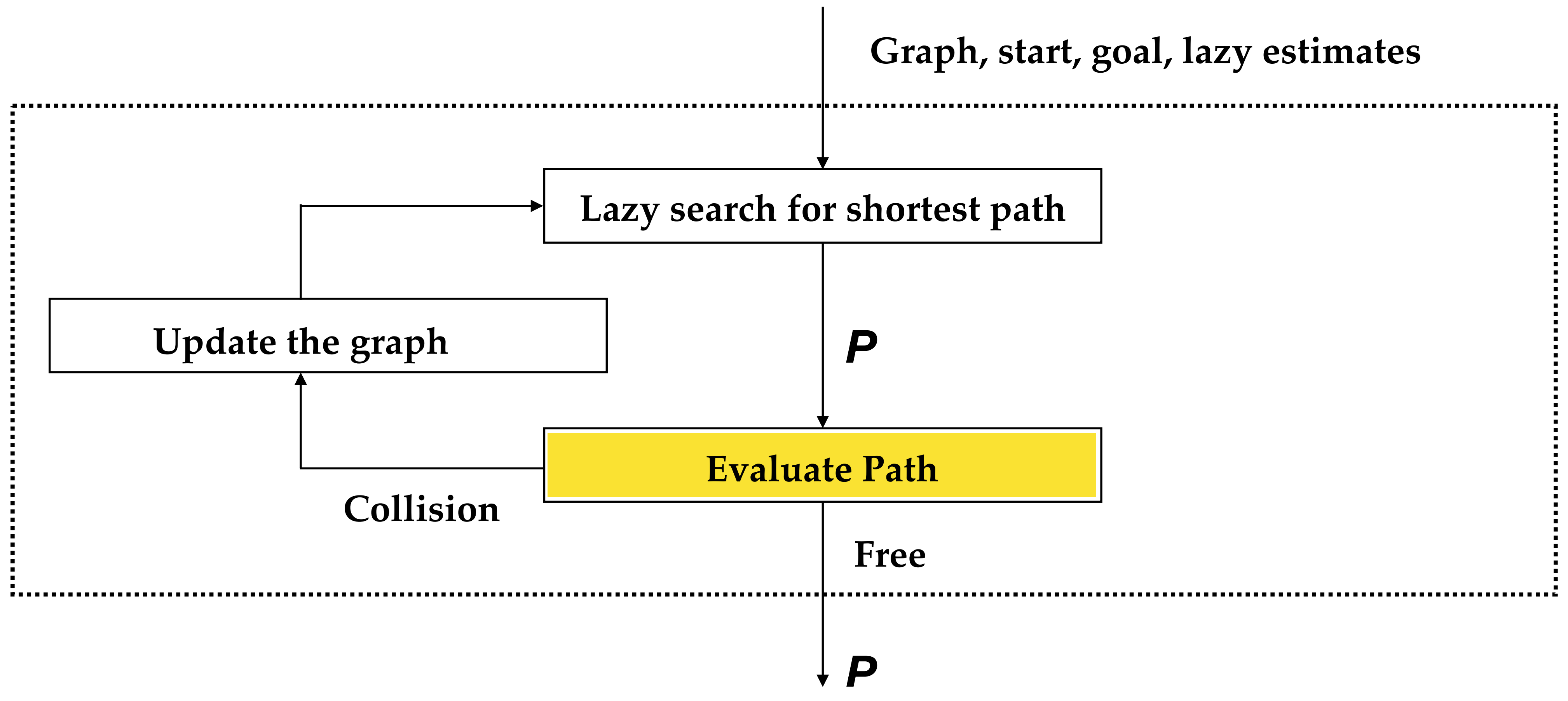


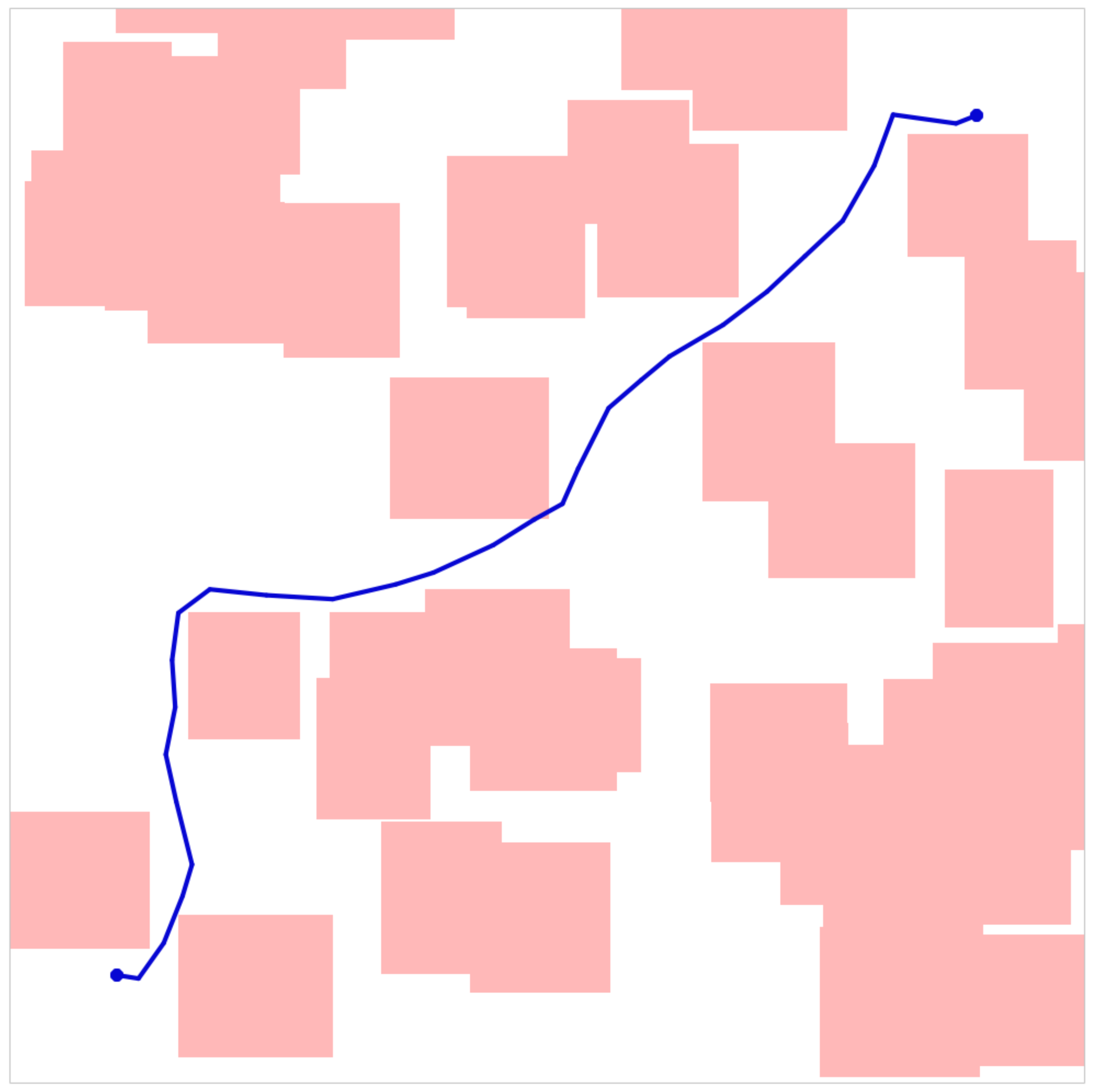


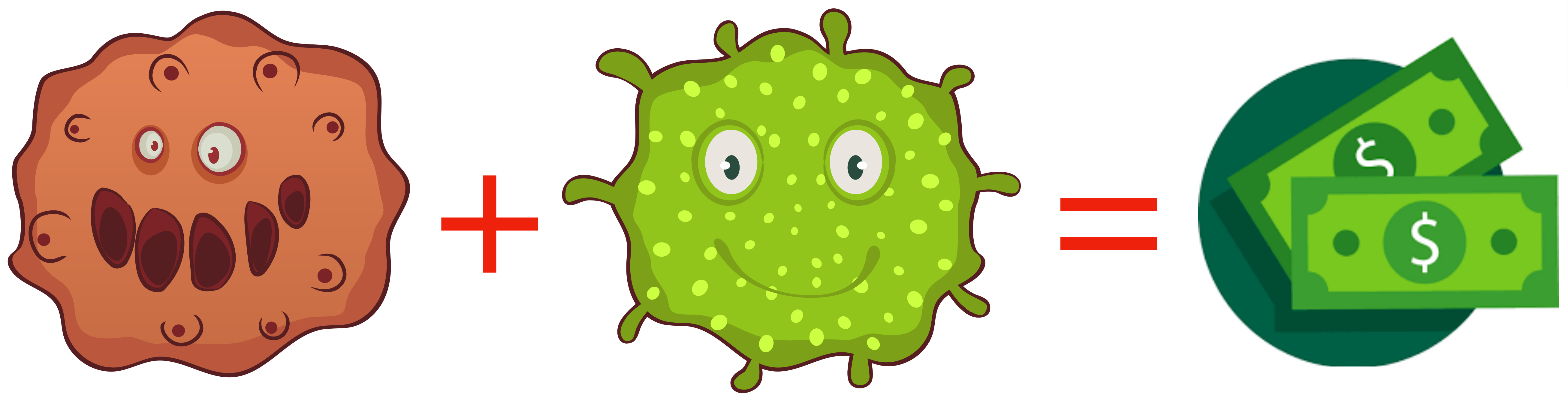
Optimal Slime!

LazySP

OFU on Steroids!

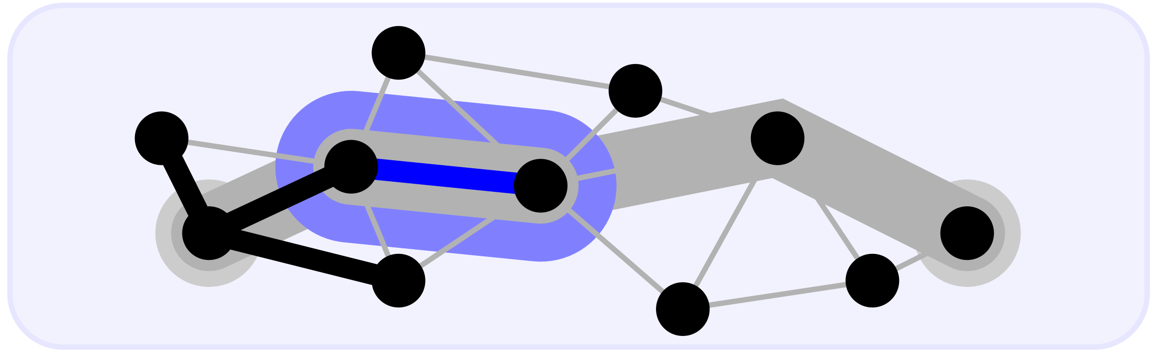

Graph, start, goal, lazy estimates
Lazy search for shortest path
Update the graph
P
Evaluate Path
Collision
Free
P



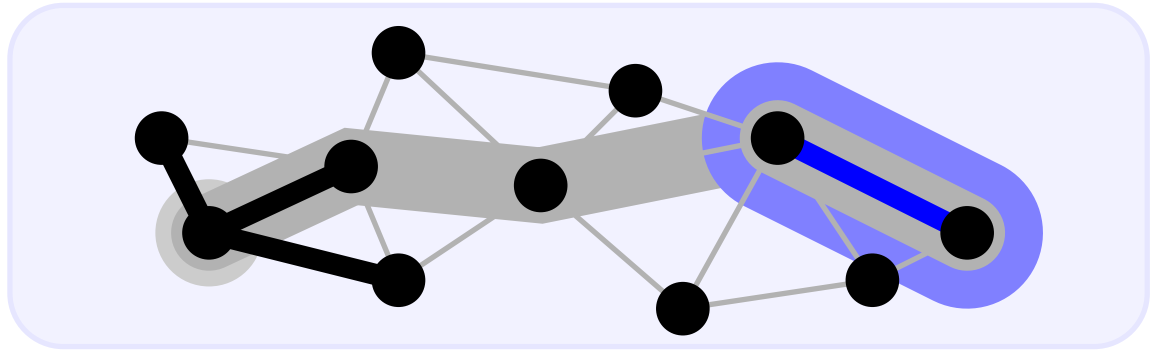


Edge Selectors

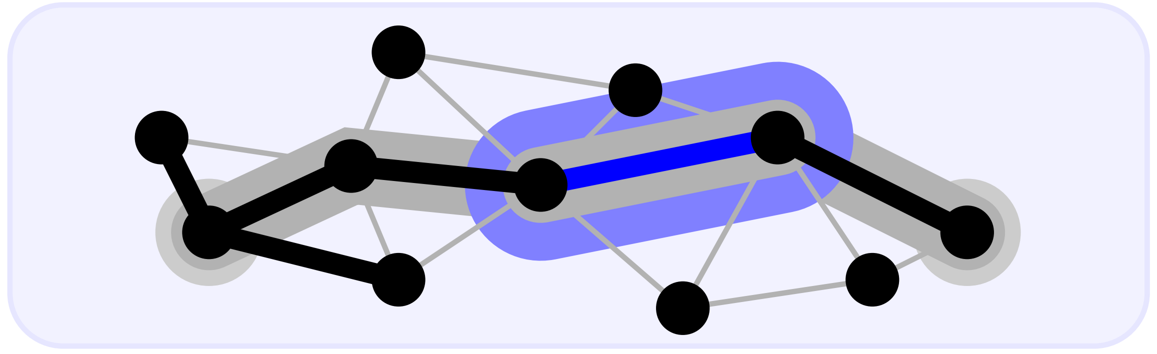
Forward
(first unevaluated edge)



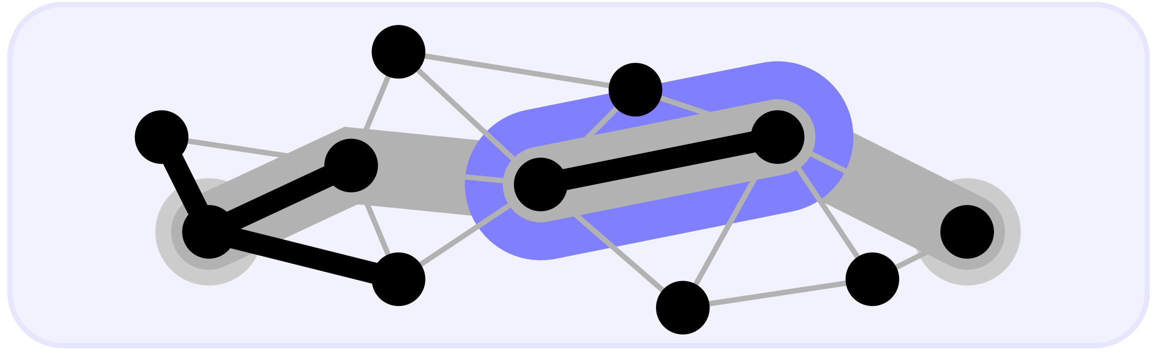
Reverse
(last unevaluated edge)



Alternate
(alternate Forward and Reverse)



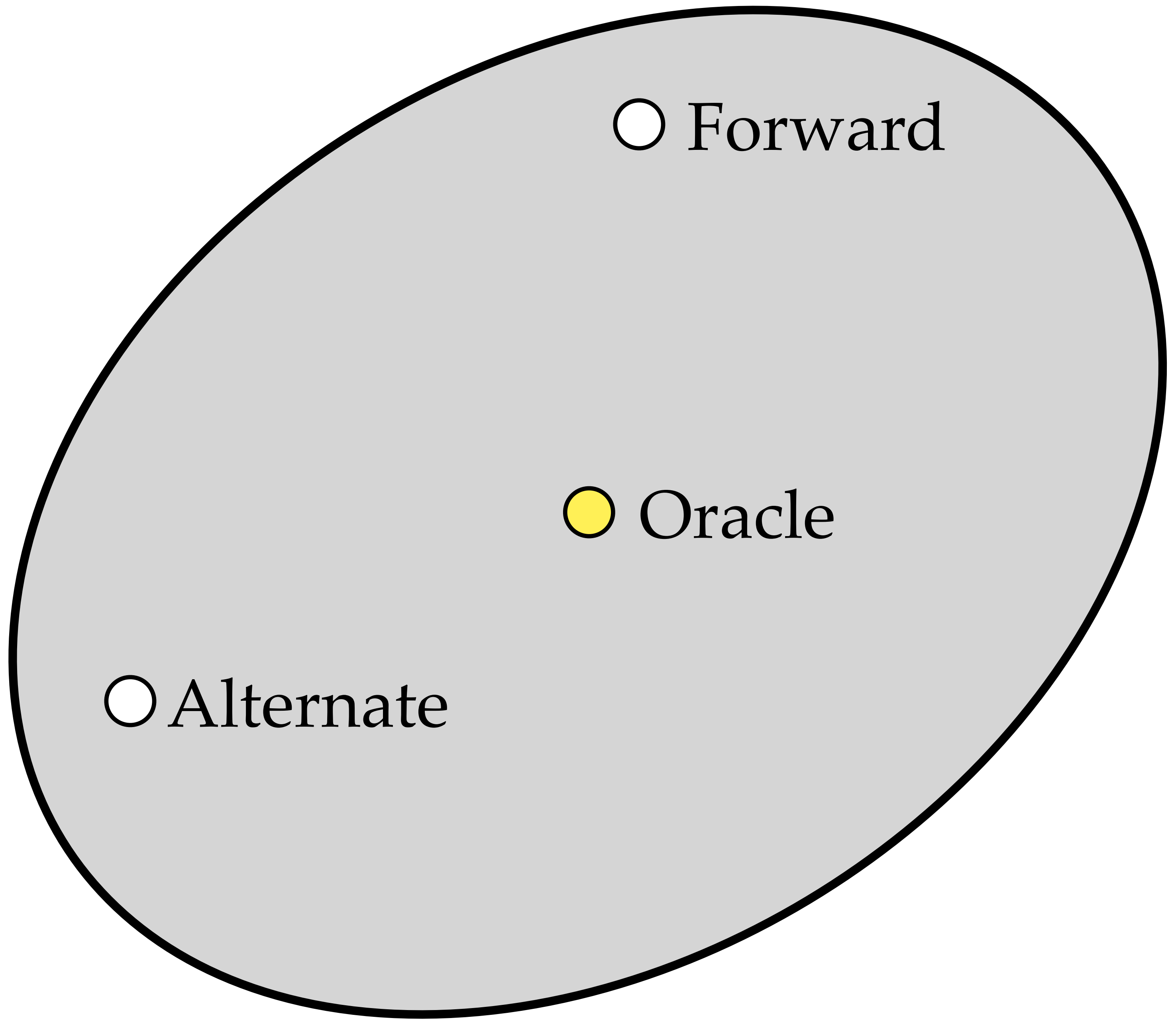
Bisect
(furthest from an unevaluated edge)



The Realizability Assumption

Can we Learn to Imitate the Oracle?

Leveraging experience in lazy search, Bhardwaj et al., RSS 2019.



Hypothesis Class

All LazySP Selectors

The Oracle is a LazySP Selector!

The Provable Virtue of Laziness in Motion Planning, Hagtalab et al., ICAPS 2018.

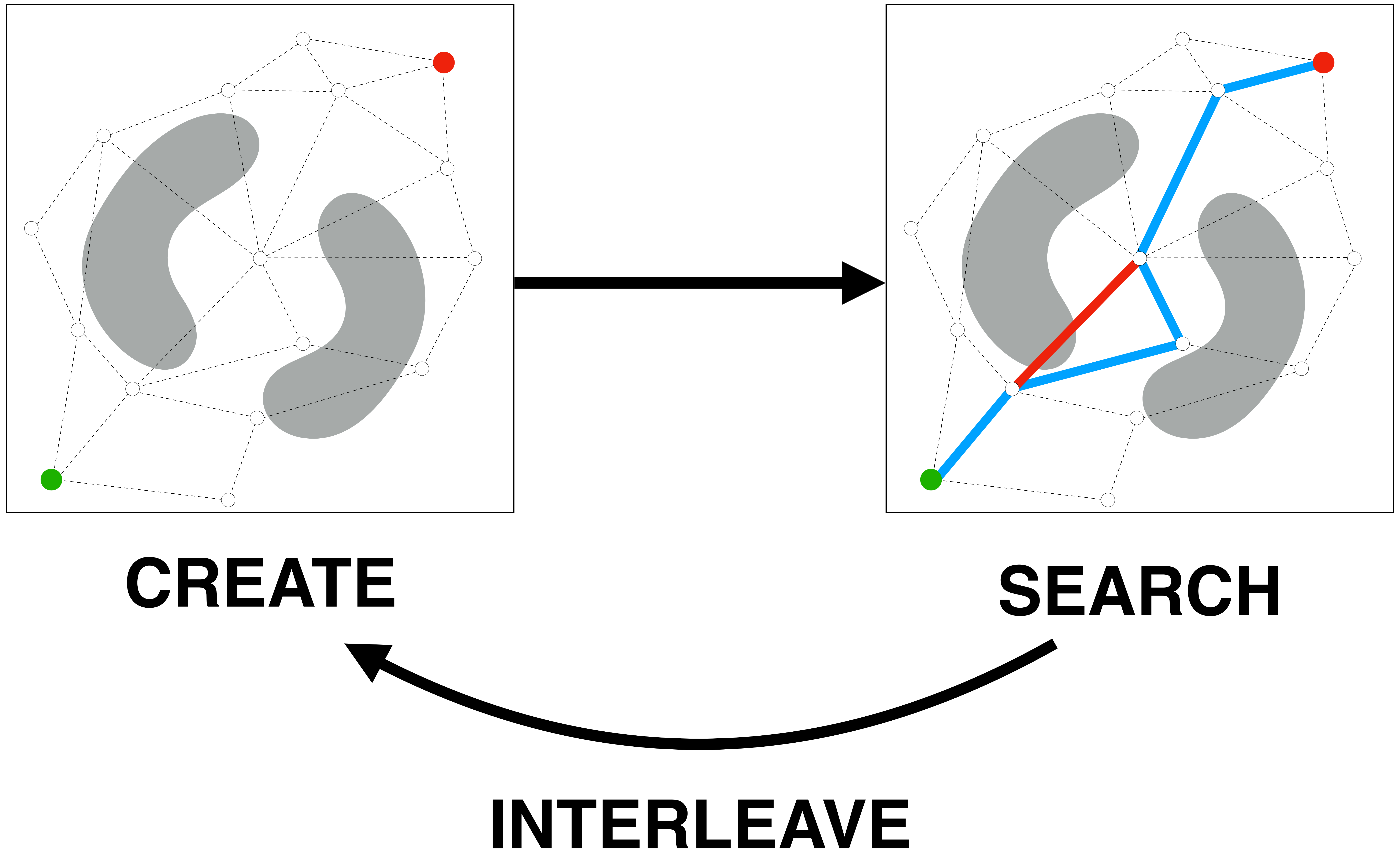
Is there a Search Algorithm
that Minimizes
the Number of Edge Evaluations?

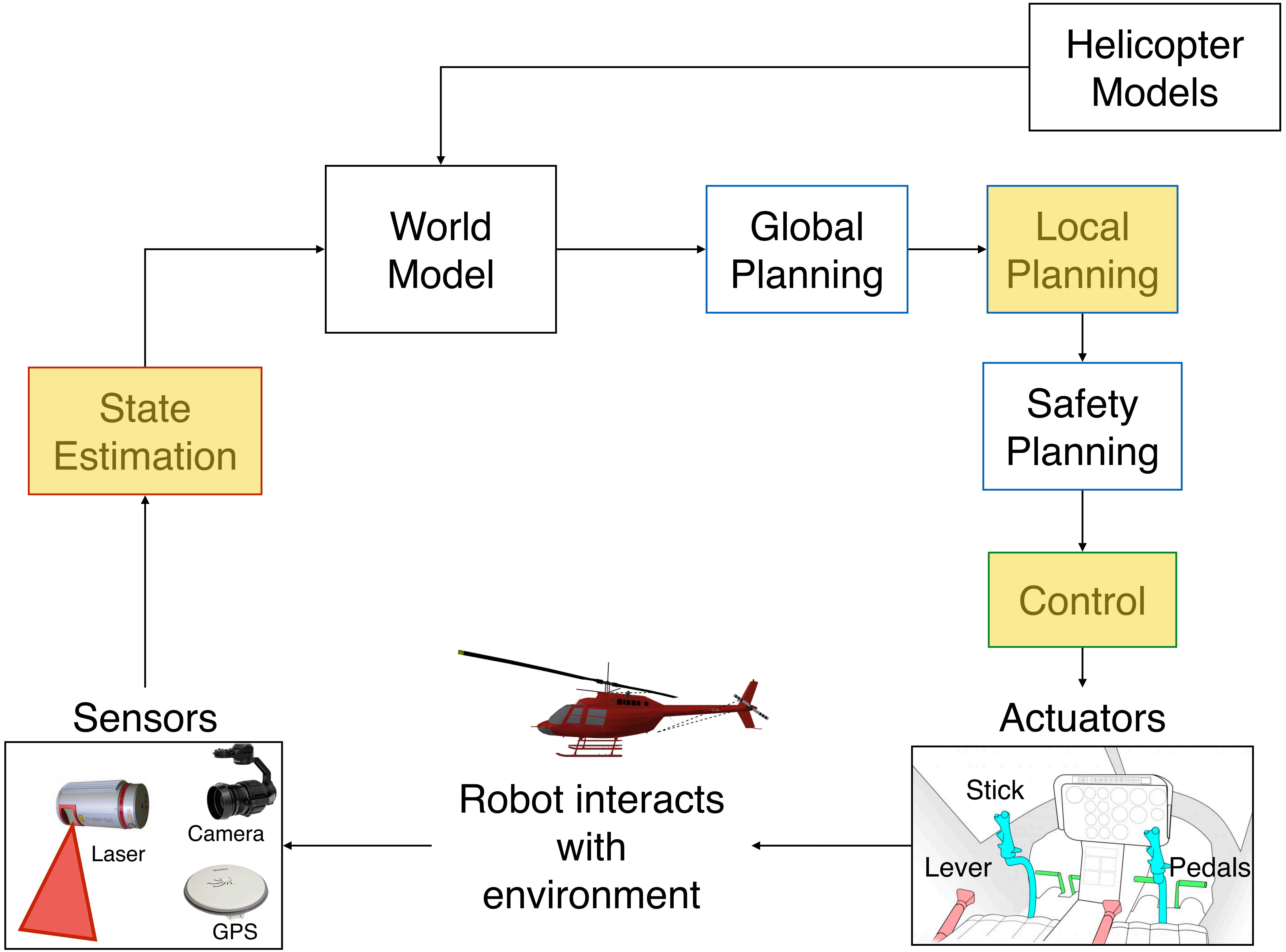
LazySP

ICAPS 2018 [Best Conference Paper Award Winner]

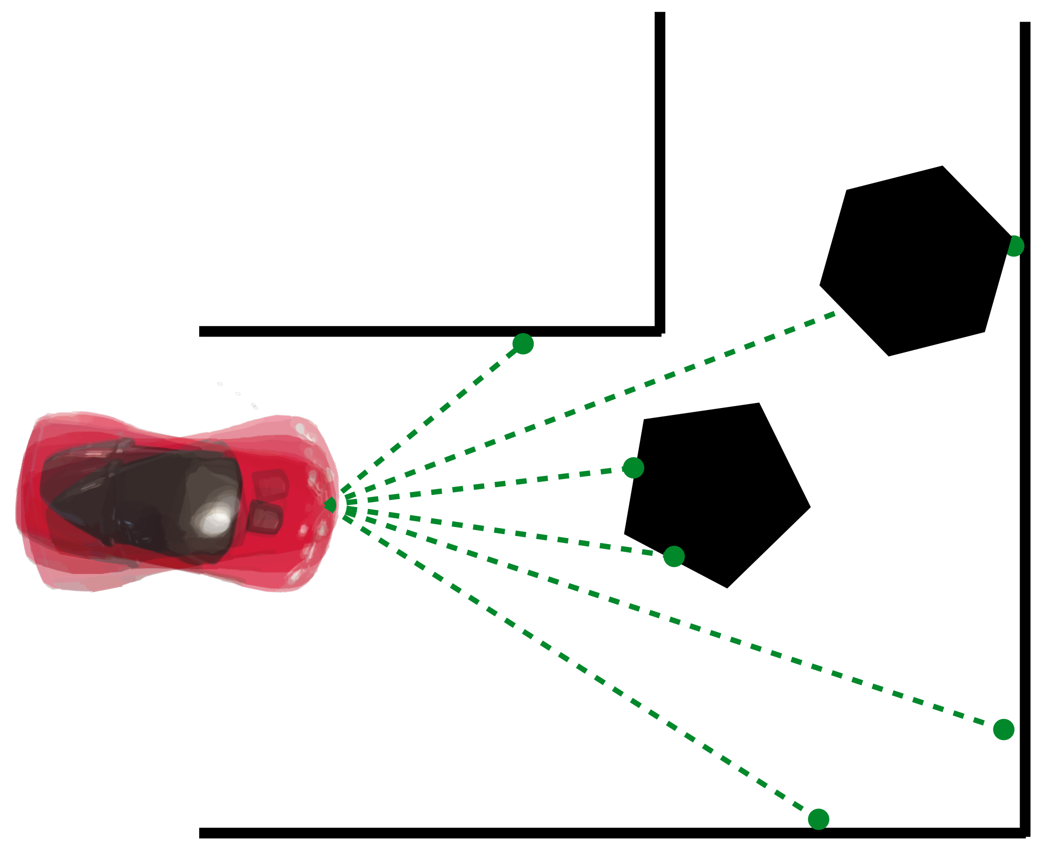
First Provably Edge-Optimal A*-like Search Algorithm

Sampling-Based Motion Planning

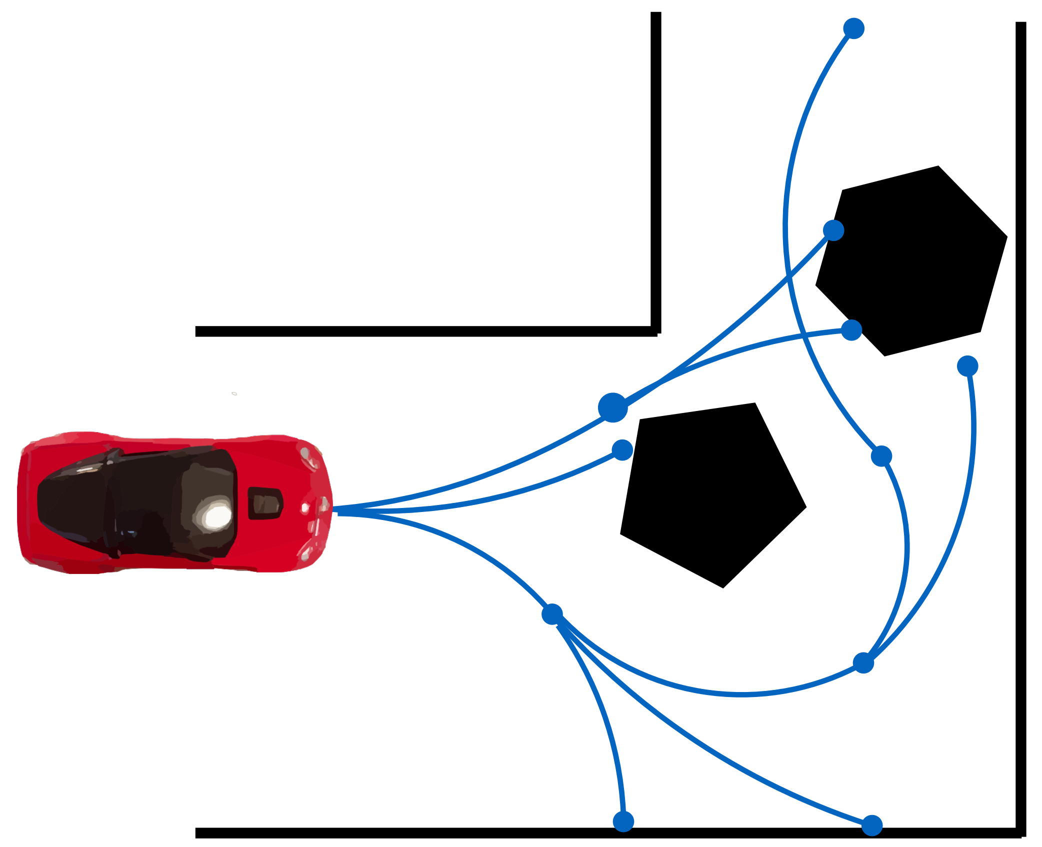



Helicopter Models
World Model
Global Planning
Local Planning
Safety Planning
Control
State Estimation
Sensors
Laser
Camera
GPS
Robot interacts with environment
Actuators
Stick
Lever
Pedals

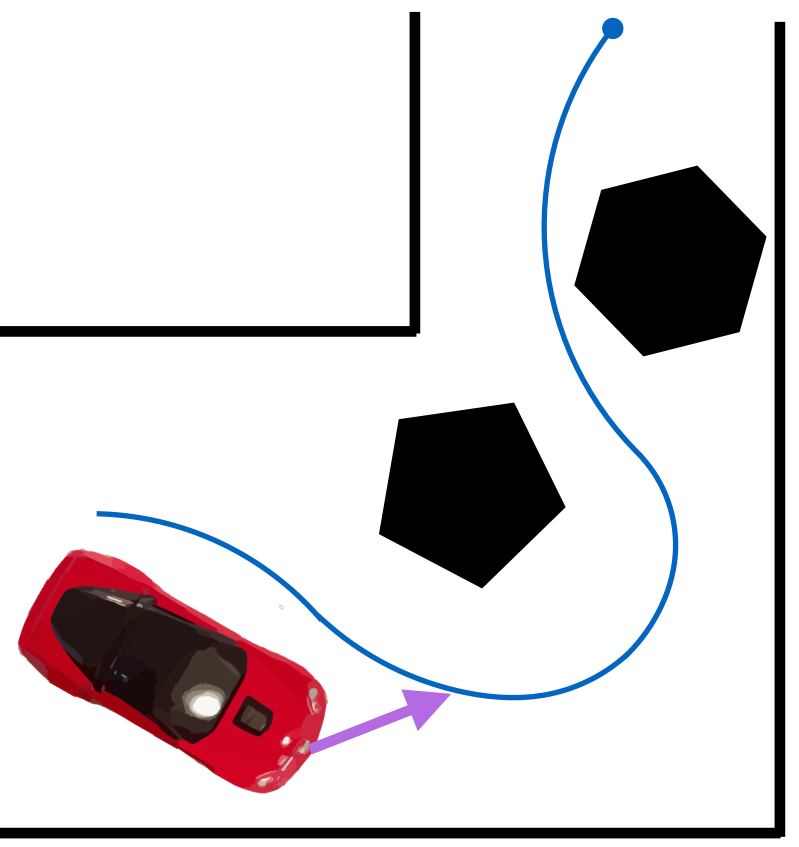
Estimate state



Plan a sequence of motions



Control robot to follow plan



CSE 478 Robot Autonomy
Lazy Search

Abhishek Gupta (abhgupta@cs)
Siddhartha Srinivasa (siddh@cs)

TAs:
Carolina Higuera (chiguera@cs)
Rishabh Jain (jrishabh@cs)
Entong Su (ensu@cs)