Course “Algorithmic Foundations of Sensor Networks” Lecture 8: Geographic routing and obstacle avoidance

Sotiris Nikoletseas

Department of Computer Engineering and Informatics
University of Patras, Greece

Spring Semester 2020-2021

Summary

- GPSR: A geographic routing protocol combining greedy forwarding and a rescue mode (to bypass obstacles).
- GRIC: An improved greedy forwarding component with obstacle avoidance properties.
- TRUST: A "trust" based protocol that gradually "learns" the obstacle presence and converges to optimal routing paths avoiding obstacles.

Geographic Routing

- when location information is available to sensors (if not directly, then through a network localization algorithm)
- to provide location-stamped data
- to satisfy location-based queries

then

making forwarding decisions using geographical information of sensor and destination positions is a natural choice.

Greedy geographic routing/advantages

"make forwarding decisions using only information about the sensors' intermediate neighbors in the network topology"

Advantages:

- keeping state only about the local topology, greedy routing scales better (wrt per intermediate router state eg memory) than shortest path and ad-hoc routing protocols.
- scales better as the number of routing destinations increases
- scales better under frequent topology changes (finds correct new routes quickly)

GPSR's modeling assumptions

- all routers (sensors) know their positions
- packet sources can mark packets they originate with their destination's locations
- bidirectional radio reachability (to allow link-level acknowledgements for packets)
- sensors lie on a plane (2D)

The GPSR algorithm

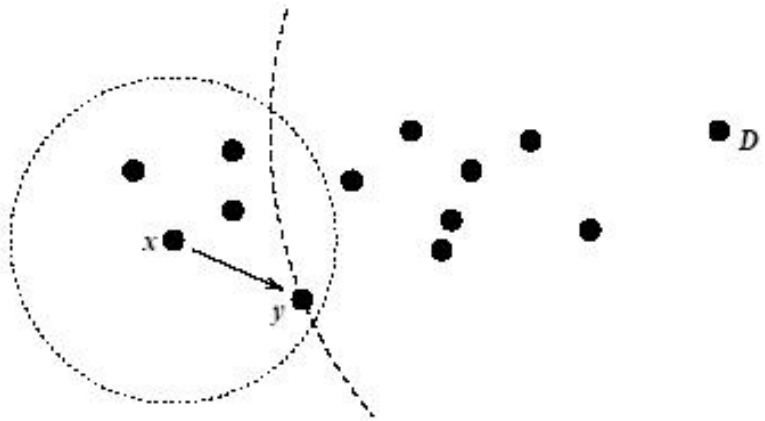
The algorithm consists of two methods for forwarding packets:

- greedy forwarding, which is used whenever possible
- perimeter forwarding, used in regions where greedy forwarding fails

GPSR's greedy forwarding

- a forwarding node makes a locally optimal, greedy choice by selecting the neighbor geographically closest to the packet's destination
- forwarding in this regime follows successively closer geographic hops until the destination is reached

An example of greedy forwarding

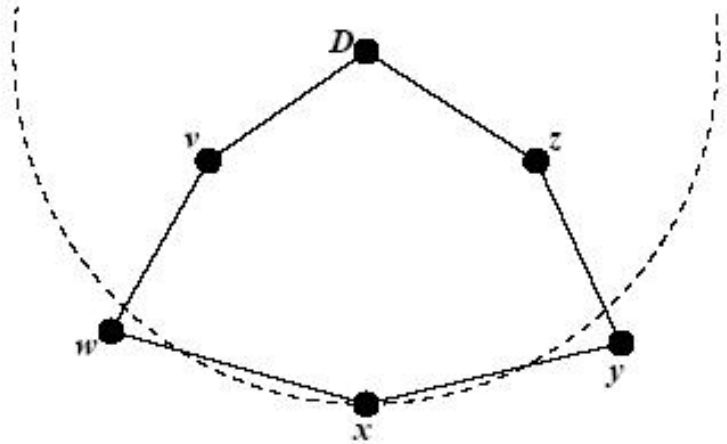


- x received a packet destined for D
- x's radio range is denoted by the dotted circle
- the dashed arc has radius equal to the distance between y and D

$\Rightarrow$ since y is the closest node to D, x forwards the packet to y.

An attendant drawback of greedy forwarding

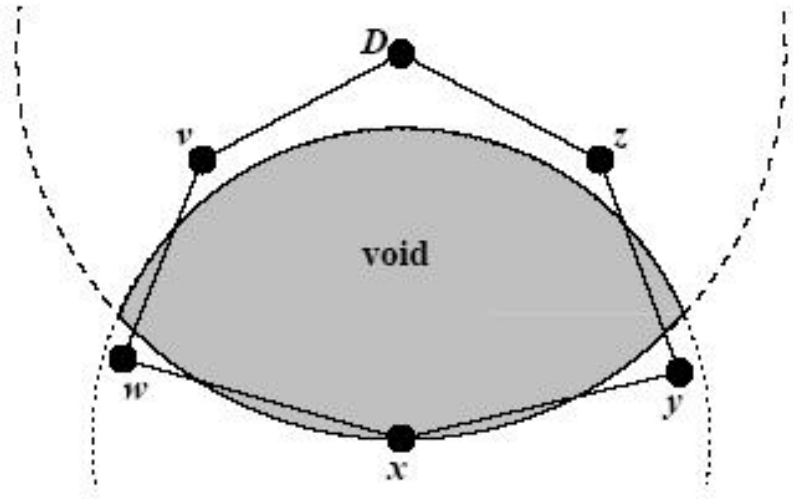
There are topologies where the only route to a destination requires a packet move temporarily farther in geometric distance from the destination.



- x is a local maximum in its proximity to D (w and y are farther from D).
- although 2 paths (xyzD, xwvD) exist to D, x will not use them when running greedy forwarding.

The "void" notion

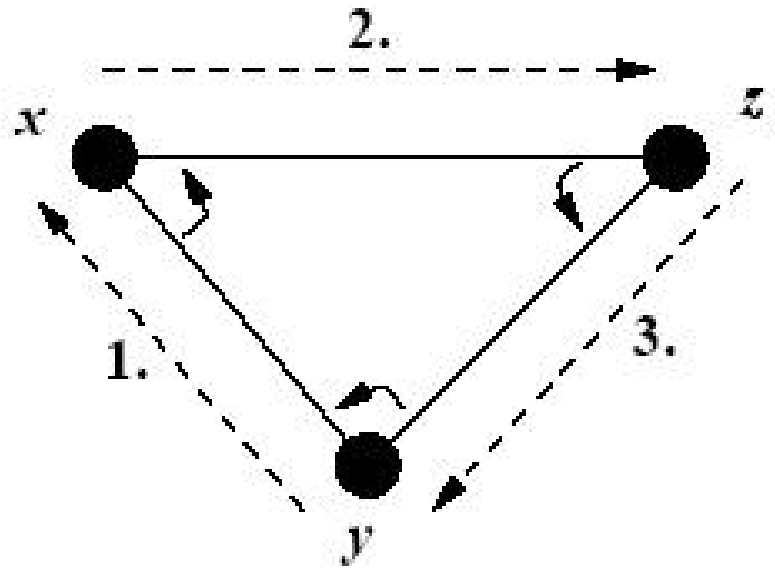
- a void arises when the intersection region of x's radio circle and the circle about D of radius $|xD|$ is empty of sensors:



- x must seek a path around the void

The Right Hand Rule

"when arriving at a node x from node y, the next edge traversed is the next one sequentially counterclockwise around x from edge (x,y)"



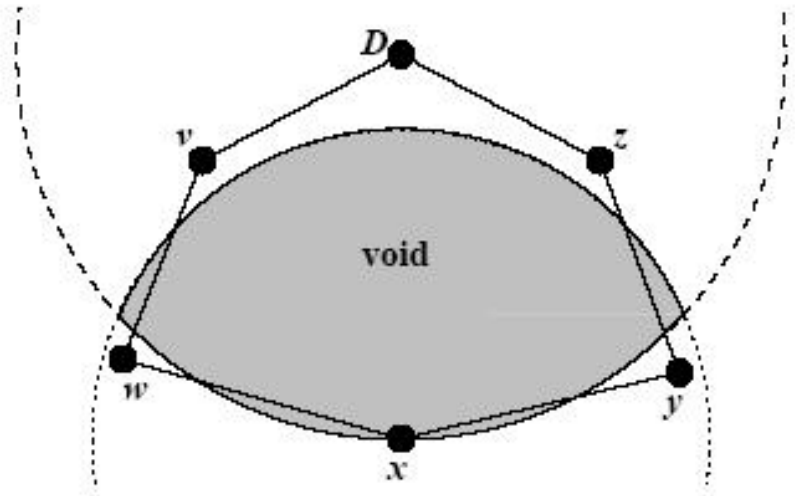
- x receives a packet from y
- then, it forwards it to z
- z then forwards to y

The cycle-traversing property

The right-hand rule is known to traverse the interior of a closed polygonal region (a face) in clockwise edge order (in our example, it traverses the triangle as follows: $y \to x \to z \to y$)

The Perimeter notion

- GPSR exploits the cycle-traversing properties to route around voids



- the rule would traverse as follows:
  $x \rightarrow w \rightarrow v \rightarrow D \rightarrow z \rightarrow y \rightarrow x$
  i.e. it navigates around the void
- We call the sequence of edges traversed by the right-hand rule a perimeter.

The complication of crossing links

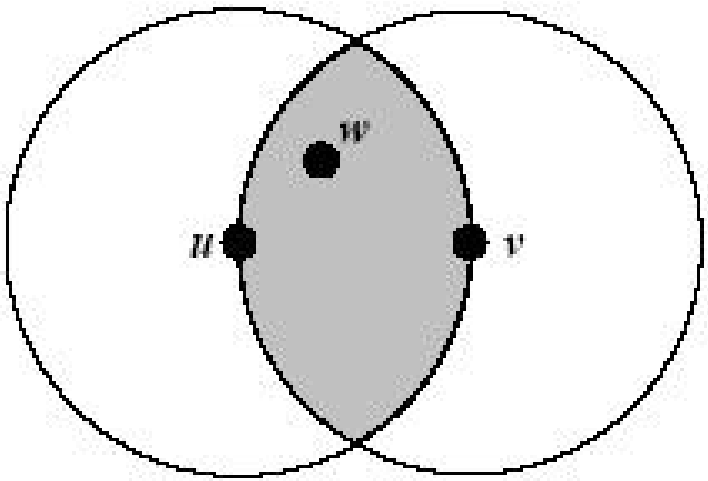
- the application of the right-hand rule obviously requires that the network graph has no crossing edges i.e. it is planar
- there are several methods to "planarize" a graph
- the "no-crossing" heuristic: it blindly removes whichever edge it encounters second in a pair of crossing edges
- serious weakness: the edge it removes may partition the network. If it does, the algorithm will not find routes that cross this partition, i.e. it does not always find routes when they exist.

Planarized Graphs

- A graph representation: A set of nodes with circular radio ranges r can be seen as a graph in which each node is a vertex and an edge (n,m) exists between nodes n and m if their distance is d(n,m)$\leqslant$r (such graphs, whose edges are dictated by a threshold distance between vertices, are termed unit graphs).
- A graph in which no edges cross is called planar.
- Two well-known planar-graphs:
  - the Relative Neighborhood Graph (RNG)
  - the Gabriel Graph (GG)

The Relative Neighborhood Graph (RNG)

- definition: "an edge (u,v) exists between vertices u and v if their distance d(u,v) is less than or equal to the distance between every other vertex w",
  i.e. $\forall w \neq u, v : d(u, v) \leq max[d(u, w), d(v, w)]$



Note: for (u,v) to be included in the RNG, the shaded region (the intersection of the transmission areas) must be empty.

Desired properties for planarization algorithms

- the algorithms should be run in a distributed fashion by each node in the network
- a node should need only local topology information
- removing edges must not disconnect the network

How to get a connected RNG

- we start from a connected unit graph
- we remove edges not part of the RNG

i.e. if u and v are connected by an edge, node u can remove non-RNG links as follows (N is the full list of neighbors of u):

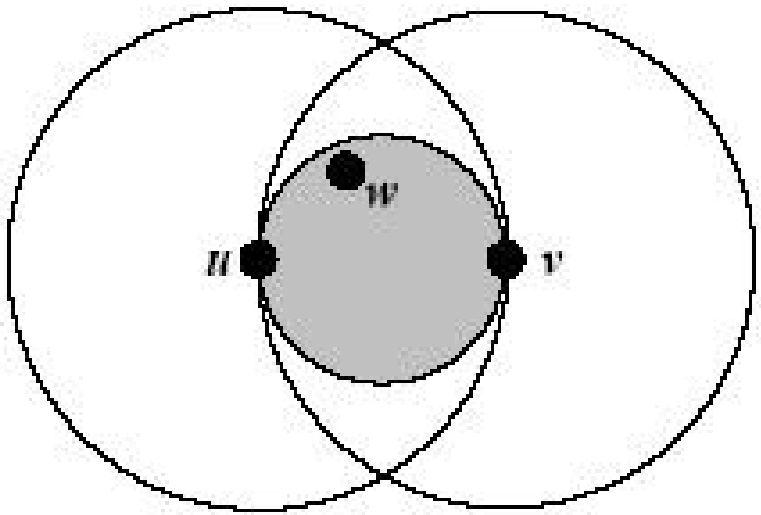
for all $v \in N$ **do**
  for all $w \in N$ **do**
    if w == v **then**
      continue
    else if d(u,v)>max[d(u,w),d(v,w)] **then**
      eliminate edge (u,v)
      break
    end if
  end for
end for

Properties of this procedure

- It is distributed.
- It uses only local information (knowledge of immediate neighbors).
- When removing non-RNG edges
  we cannot disconnect the graph since an edge (u,v) is eliminated from the graph only when there exists another vertex w within range of both u and v, thus an alternate path through a witness exists.

The Gabriel Graph (GG)

- definition: "an edge (u,v) exists between vertices u and v if no other vertex w is present within the circle whose diameter is $\overline{uv}$", i.e.
  $\forall w \neq u, v : d^2(u, v) < d^2(u, w) + d^2(v, w)$



Note: for edge (u,v) to be included in the GG, the shaded region must be empty

How to get a connected GG graph

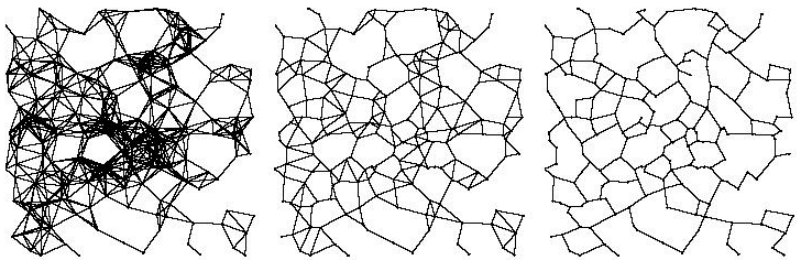
Since the midpoint of $\overline{uv}$ is the center of the circle with the diameter $\overline{uv}$, a node u can remove its non-GG links from a full neighbor list N as follows:

for all $v \in N$ **do**
    m = midpoint of $\overline{uv}$
    for all $w \in N$ **do**
        if w == v **then**
            continue
        else if d(w,m)<d(v,m) **then**
            eliminate edge (u,v)
            break
        end if
    end for
end for

Note: Similarly to the RNG case, this procedure is distributed, local and cannot disconnect the graph.

The complexity of the planarization algorithms

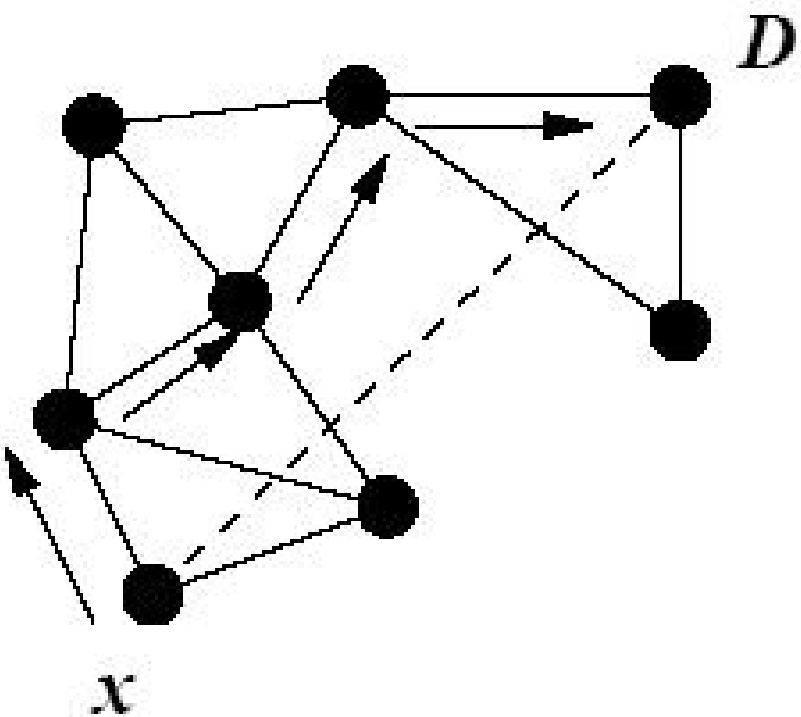
- Clearly, both algorithms for rendering the graph planar take $O(d^2)$ time at each node, where d is the node's degree.
- It can be shown that RNG is a subset of the GG. This is intuitively consistent with the smaller shaded region (for removing links) in the GG graph compared to the RNG graph. An example follows (left:full graph, center:GG subgraph, right:RNG subgraph)



- using fewer links may improve efficiency (wrt MAC considerations) through spatial diversity.

The full GPSR algorithm (I)

- GPSR combines greedy forwarding on the full network graph with perimeter forwarding on the planarized network graph (when greedy forwarding is not possible).
- When a packet enters perimeter mode at node x bound for destination D, GPSR forwards it on progressively closer faces of the planar graph, each of which is crossed by line $\overline{xD}$.



The full GPSR algorithm (II)

- On each face, the traversal uses the right-hand rule to reach an edge that crosses line $\overline{xD}$.
- At that edge, the traversal moves to the adjacent face crossed by $\overline{xD}$.
- when a next hop sensor is found lying closer to D than the current one, GPSR resumes to the greedy forwarding mode (this is only one of the possible variations).

Evaluation of GPSR

The original paper compares GPSR to DSR, using 3 metrics: packet delivery success rate, protocol overhead and path optimality.

- more than 97% of data packets are successfully delivered (slightly greater success ratio than DSR).
- GPSR achieves threefold (and even fourfold) overhead reduction, especially when mobility increases.
- they measure the percentage of delivered packets in terms of the number of hops beyond the ideal shortest path length: In a dense network, GPSR delivers 97% of its packets along optimal-length paths (vs 85% for DSR). GPSR delivers 3% of packets using one hop more than the optimal length (vs 10% for DSR) while DSR delivers the rest 5% of packets using two hops more than optimality.

The GRIC algorithm - Compared Protocols

Comparison of three algorithms

1. Greedy algorithms
2. FACE algorithms (like GPSR)
3. The GRIC algorithm
   GRIC: geographic routing with contour and inertia

Greedy

This algorithm is very simple:

- Always send a message to the neighbour which is the closest to the destination.

- Unless the network is *very* dense, messages get trapped inside of *routing holes*.

Routing holes

Routing holes follow from the *local minimum* phenomenon, and occur in regions of the network with *low density*.

- Even in dense nets of uniformly distributed sensors, there is high probability for routing holes to appear.

Remark

Greedy also fails when there are obstacles

FACE

The FACE family of algorithms (like GPSR) *guarantee delivery.*

Idea

- Extract a planar subgraph of the communication graph.
- Using the *right-hand rule*, route messages to the destination.

Weaknesses

- The need to run on a planar graph $\rightsquigarrow$ extra topology maintenance.

Comparison

	Greedy	Face	GRIC
lightweight	yes	extra topology maintenance	yes
success rate	low	Guaranteed	high
path length	good	many hops	good
robustness	yes	no [1]	yes

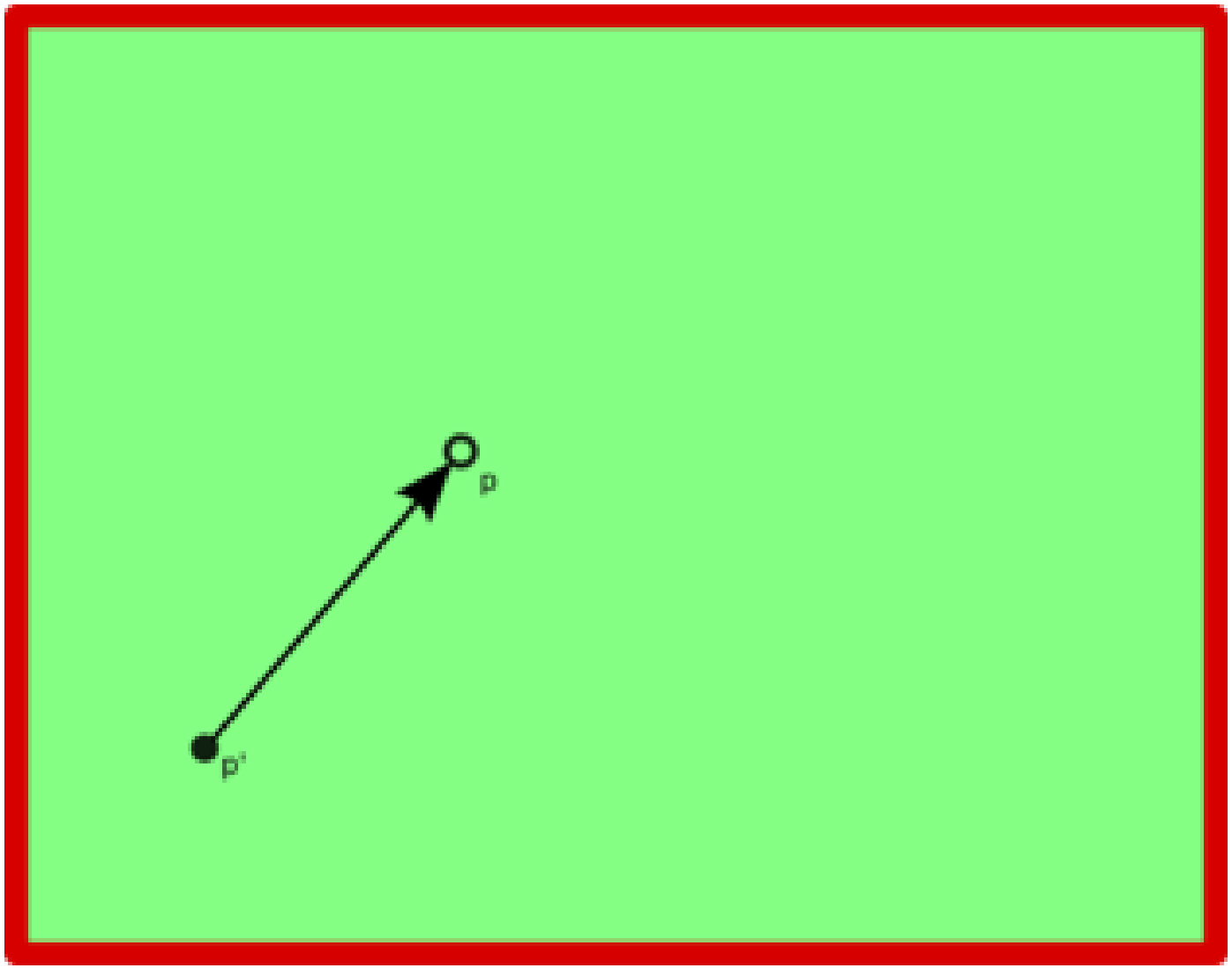
[1] because extracting planar subgraphs is not robust

Two modes

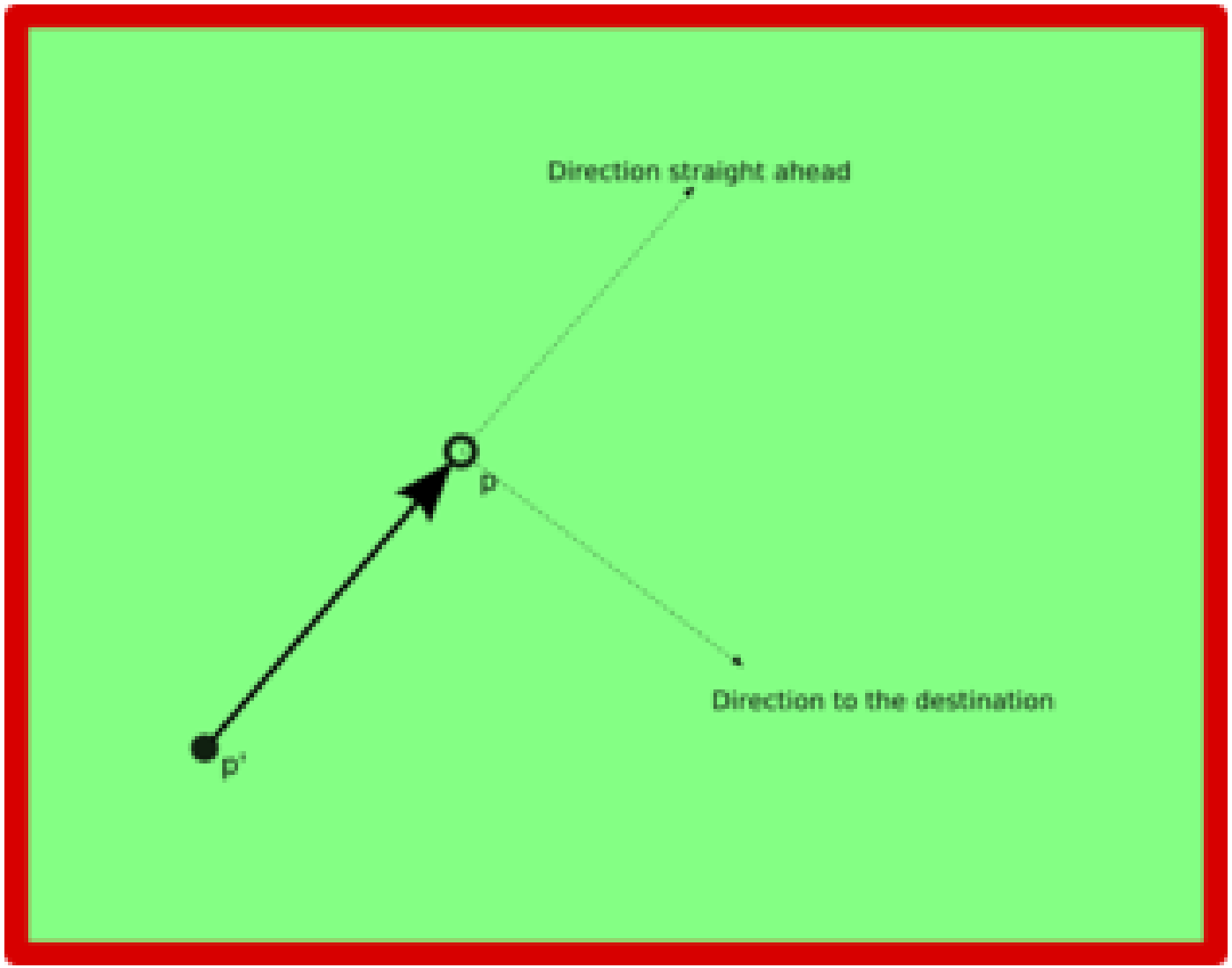
- Normal mode.
- Recovery mode.
- Randomness can be added to improve performance.

- Normal mode is a bit like GREEDY, but inertia is introduced.
- Recovery mode is a bit like FACE, but it runs on the *complete communication graph*.

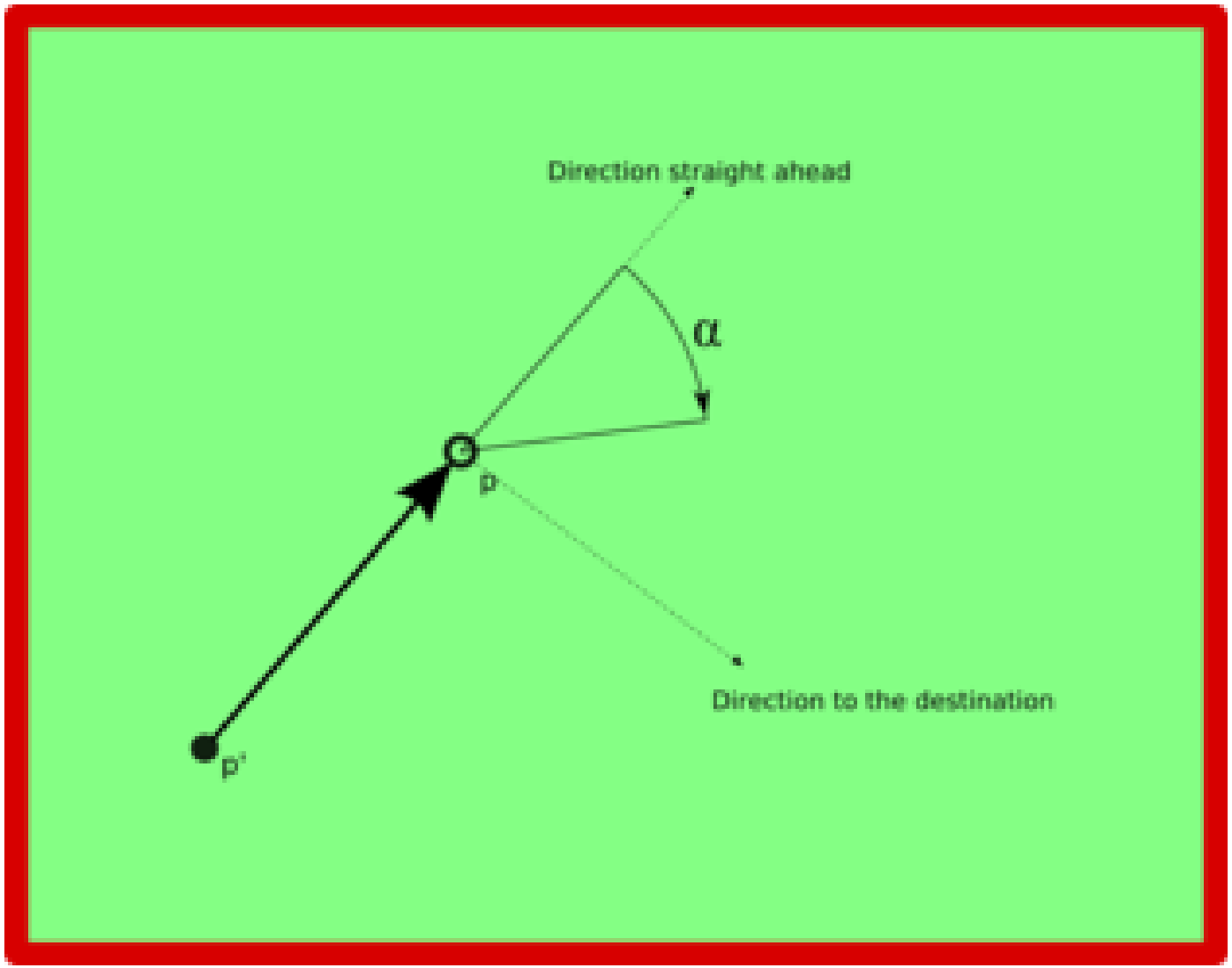
Inertia



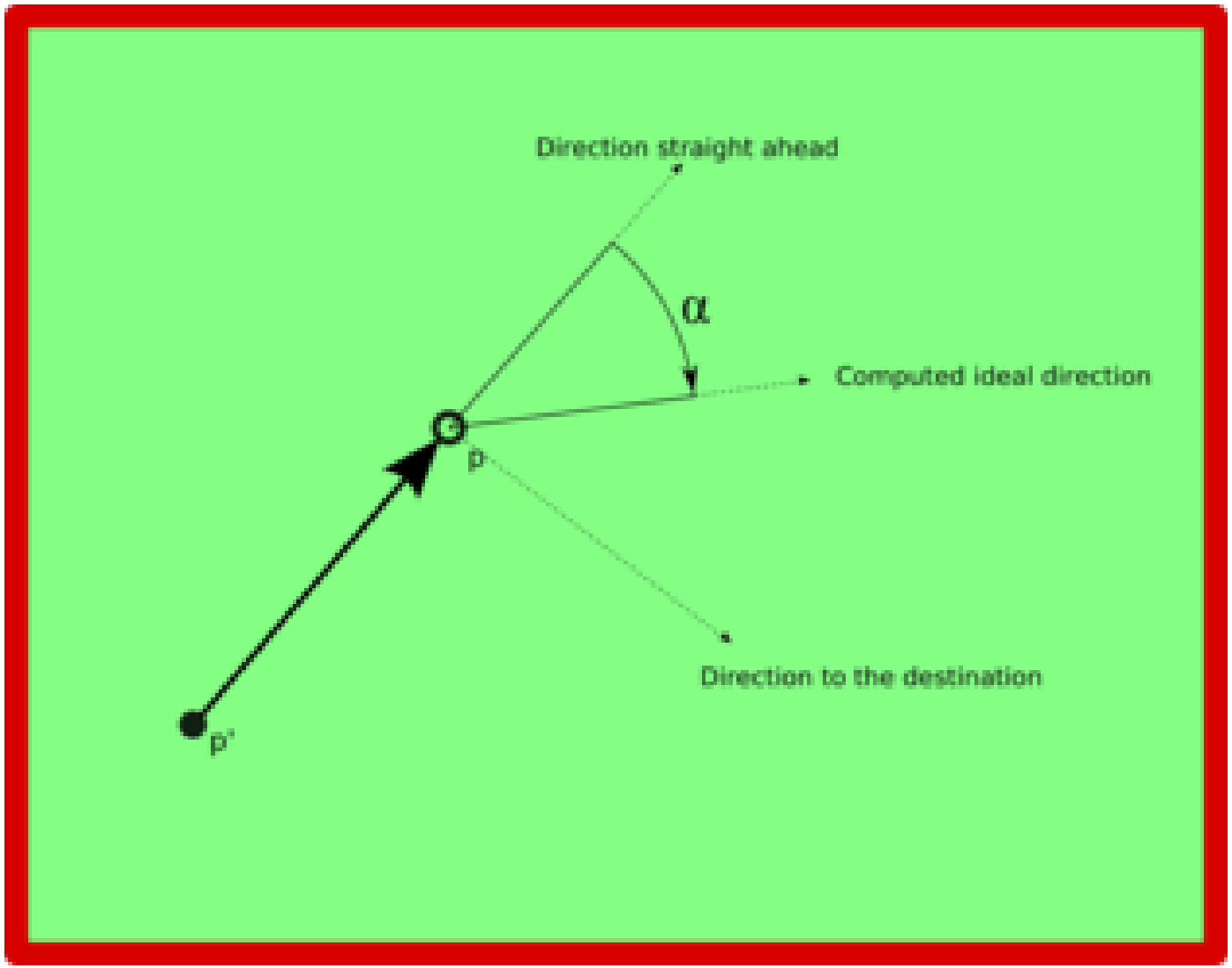
Inertia



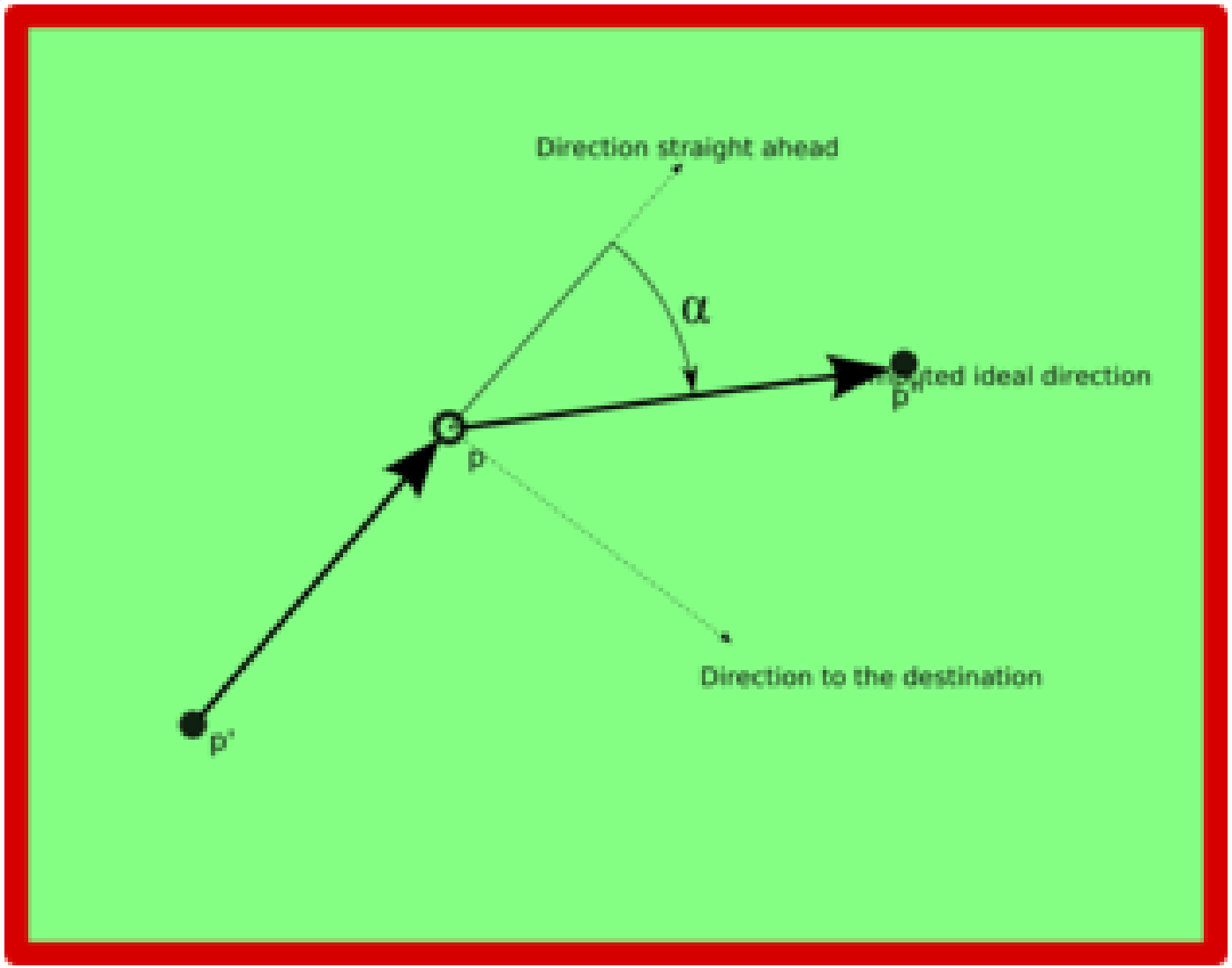
Inertia


Direction straight ahead
α
p
Direction to the destination
p'

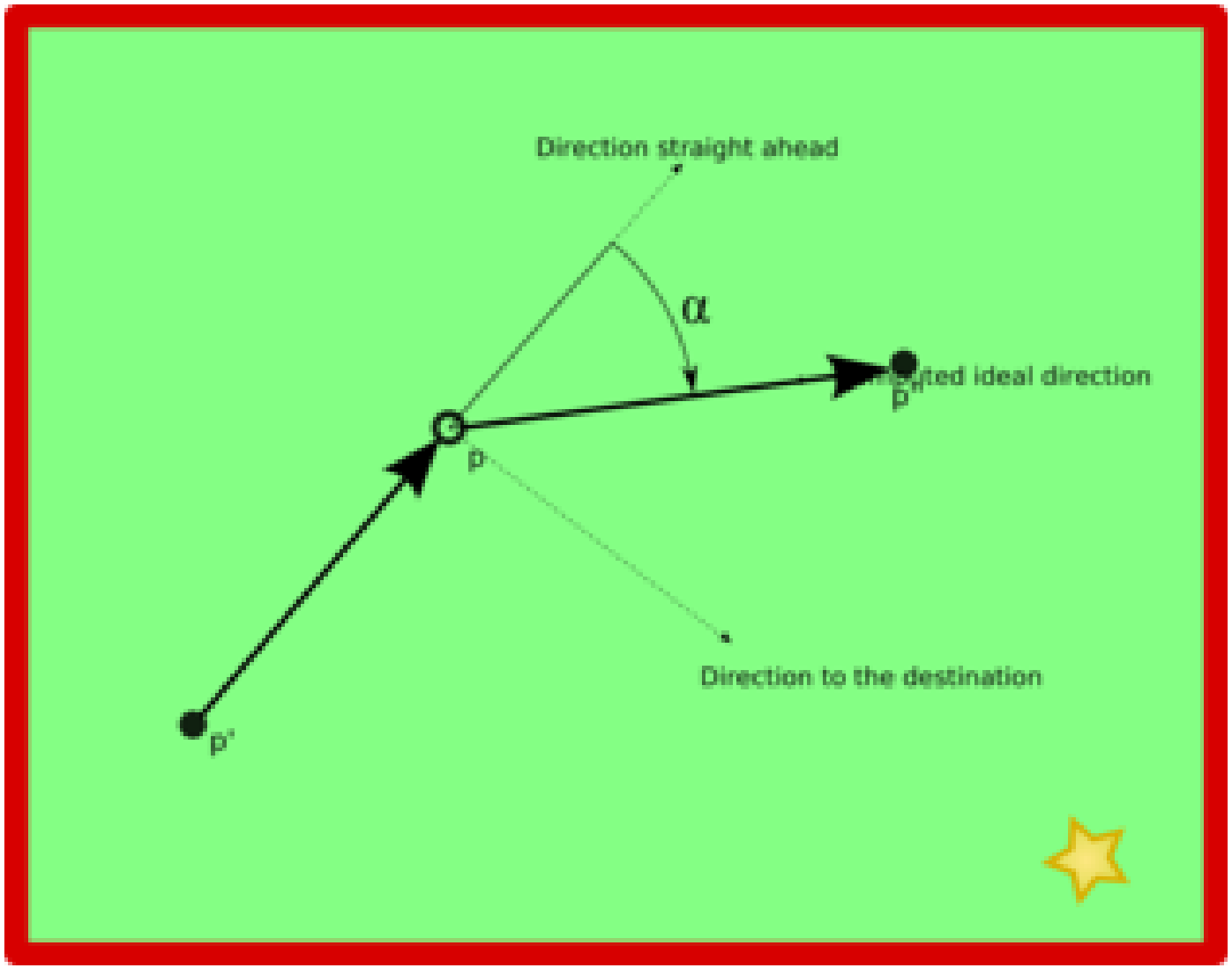
Inertia


Direction straight ahead
α
Computed ideal direction
p
Direction to the destination
p'

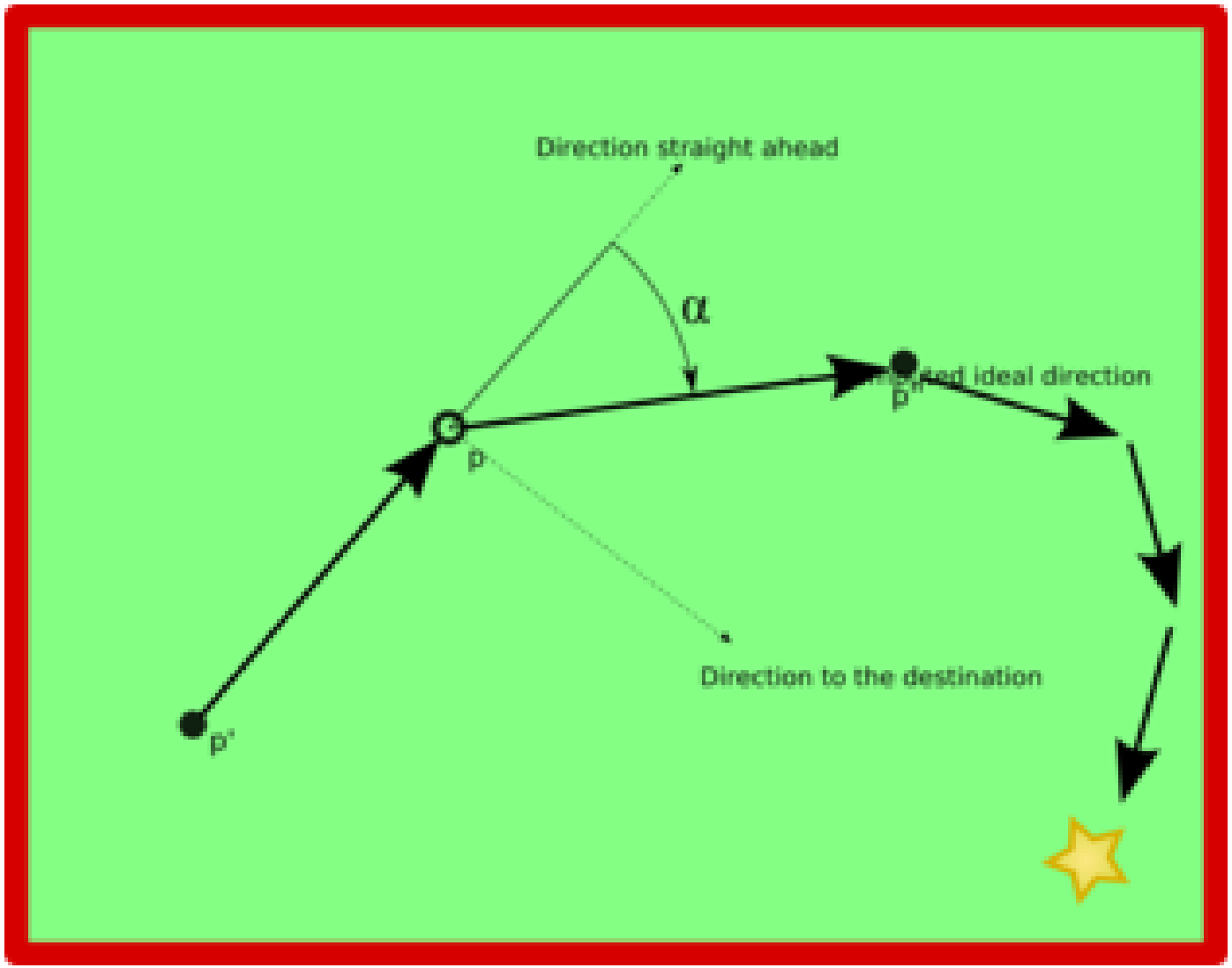
Inertia


Direction straight ahead
α
ted ideal direction
p'
p
Direction to the destination
p*

Inertia


Direction straight ahead
α
ted ideal direction
p'
p
Direction to the destination
p*

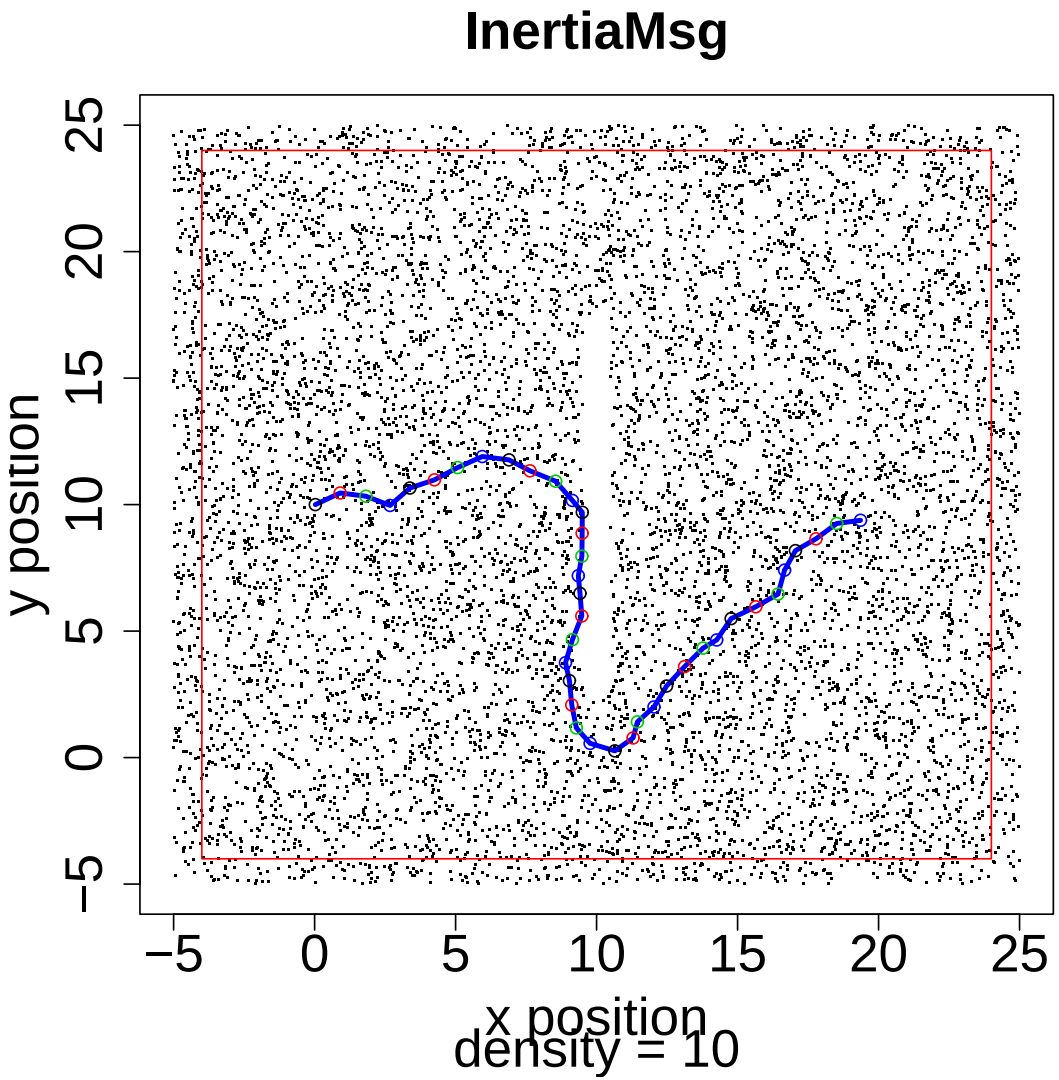
Inertia


Direction straight ahead
α
ed ideal direction
p'
p
Direction to the destination
p''

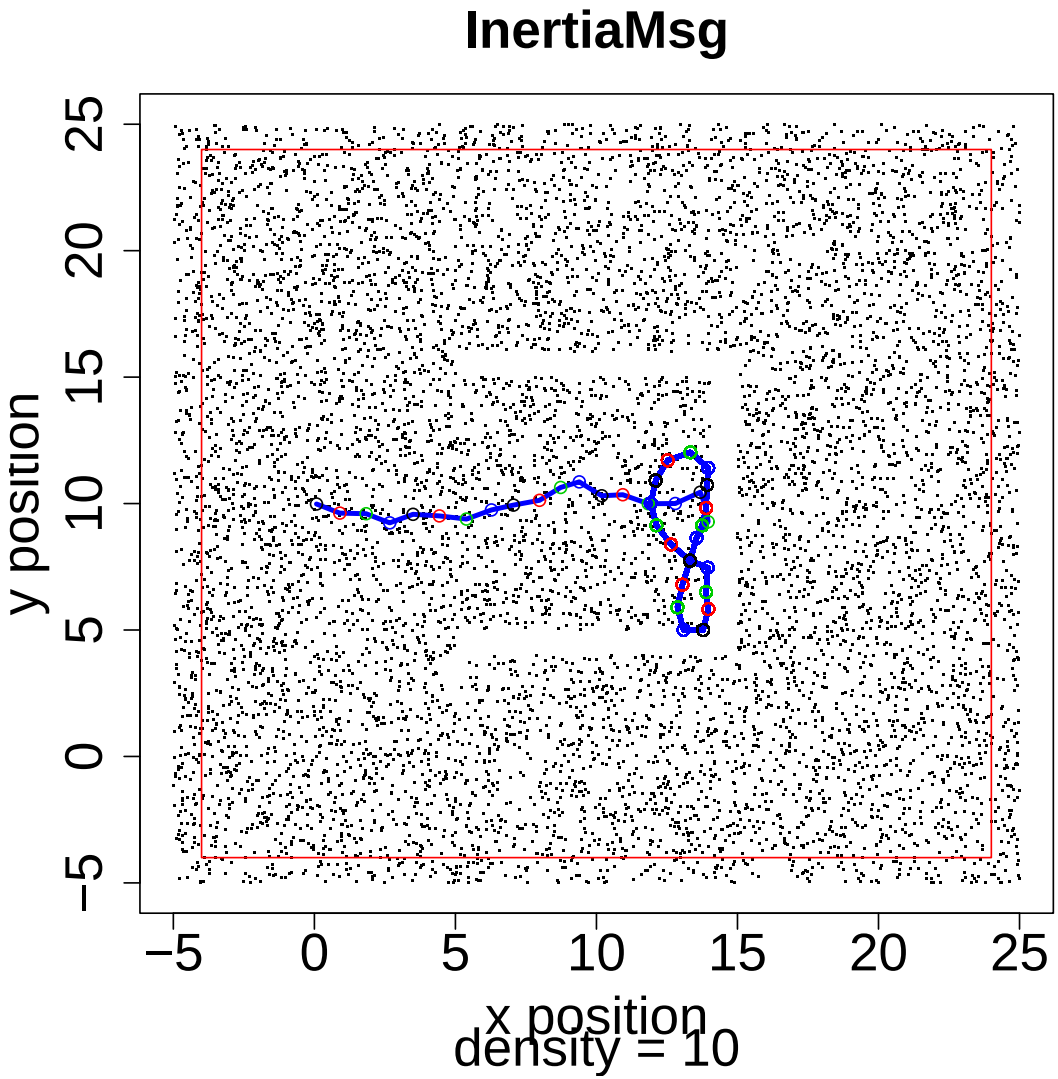
Experimental finding

- Inertia is good at getting out of routing holes.
- It can even route messages around obstacles.

An example where inertia succeeds



An example where inertia fails



Recovery (contour) mode

In order to manage to route messages around obstacles, the recovery mode, or *contour mode*, is introduced.

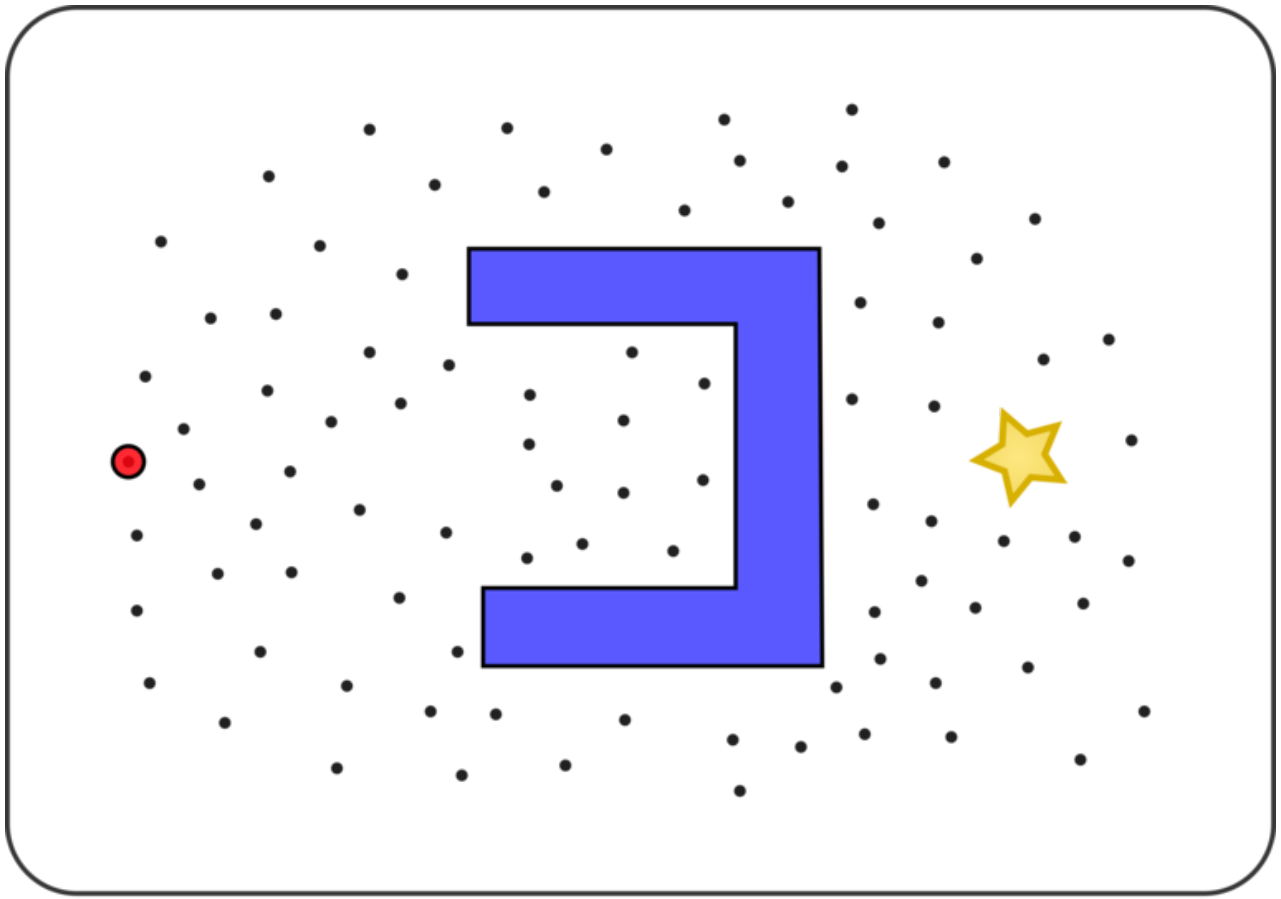
When to use it

- When a message goes *backward*, switch to the recovery mode.
- When a message goes *towards destination*, switch back to normal mode.

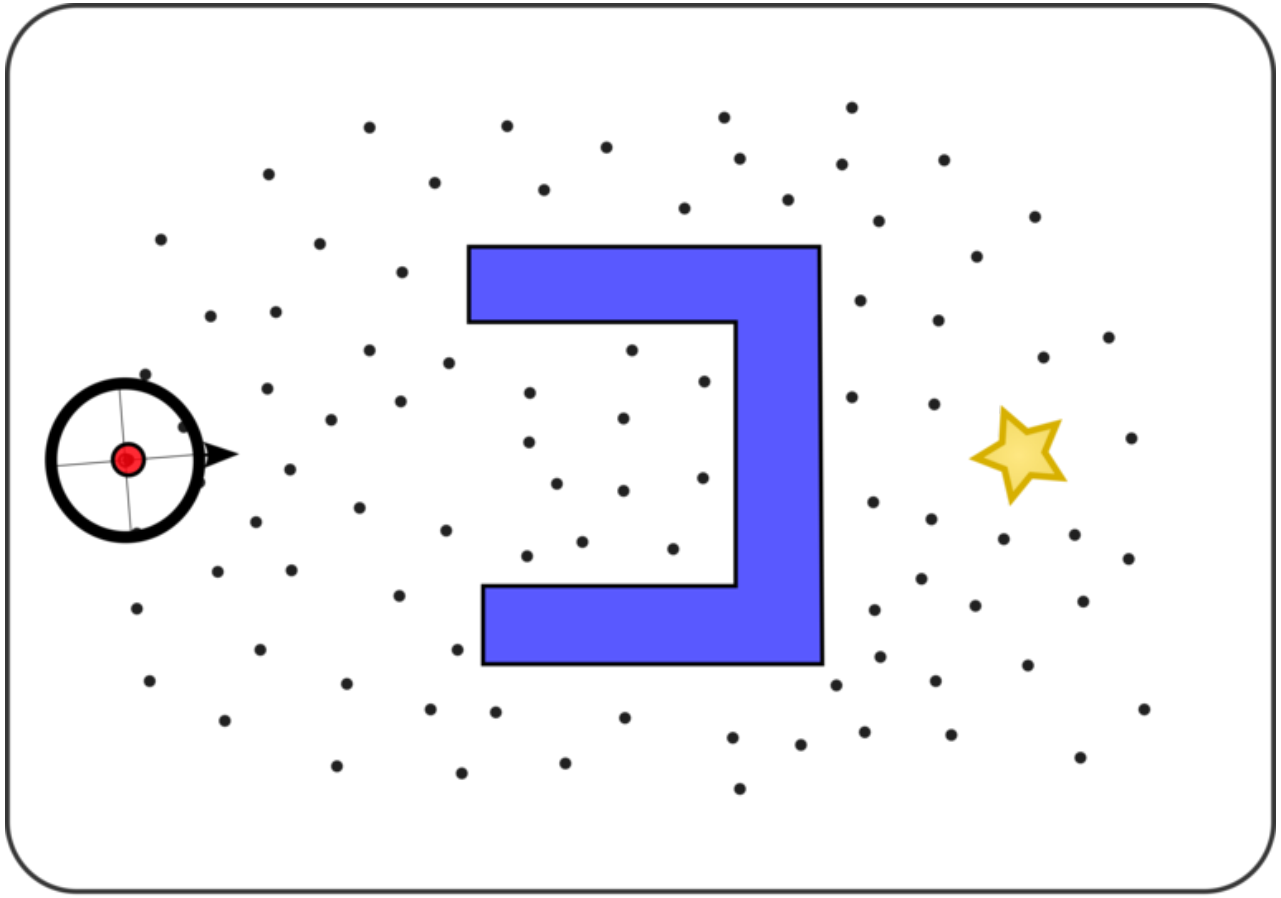
What is does

- Force turning left (or right) to imitate the *lefthand rule*.
- Keep the idea of inertia routing at the same time.

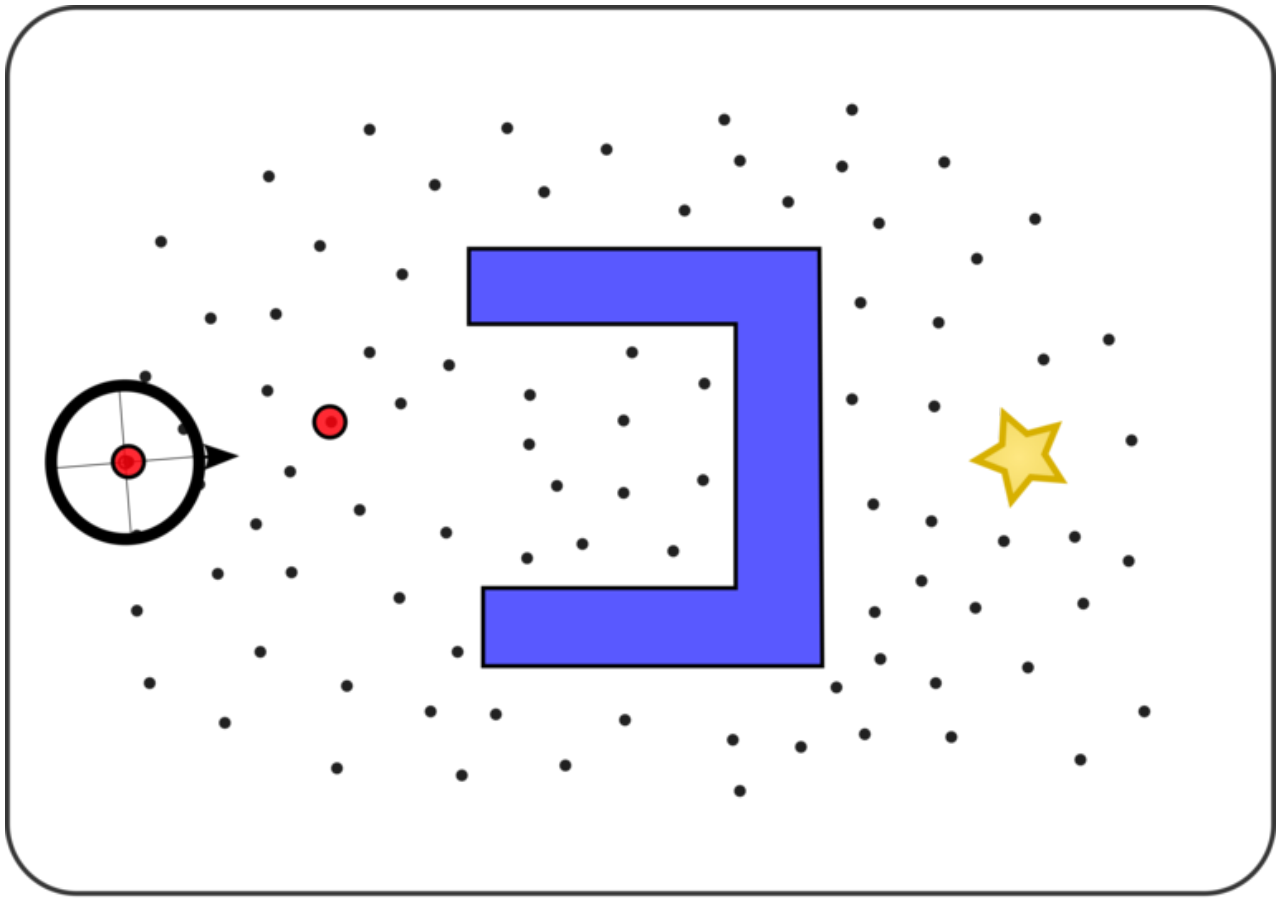
Graphical explanation



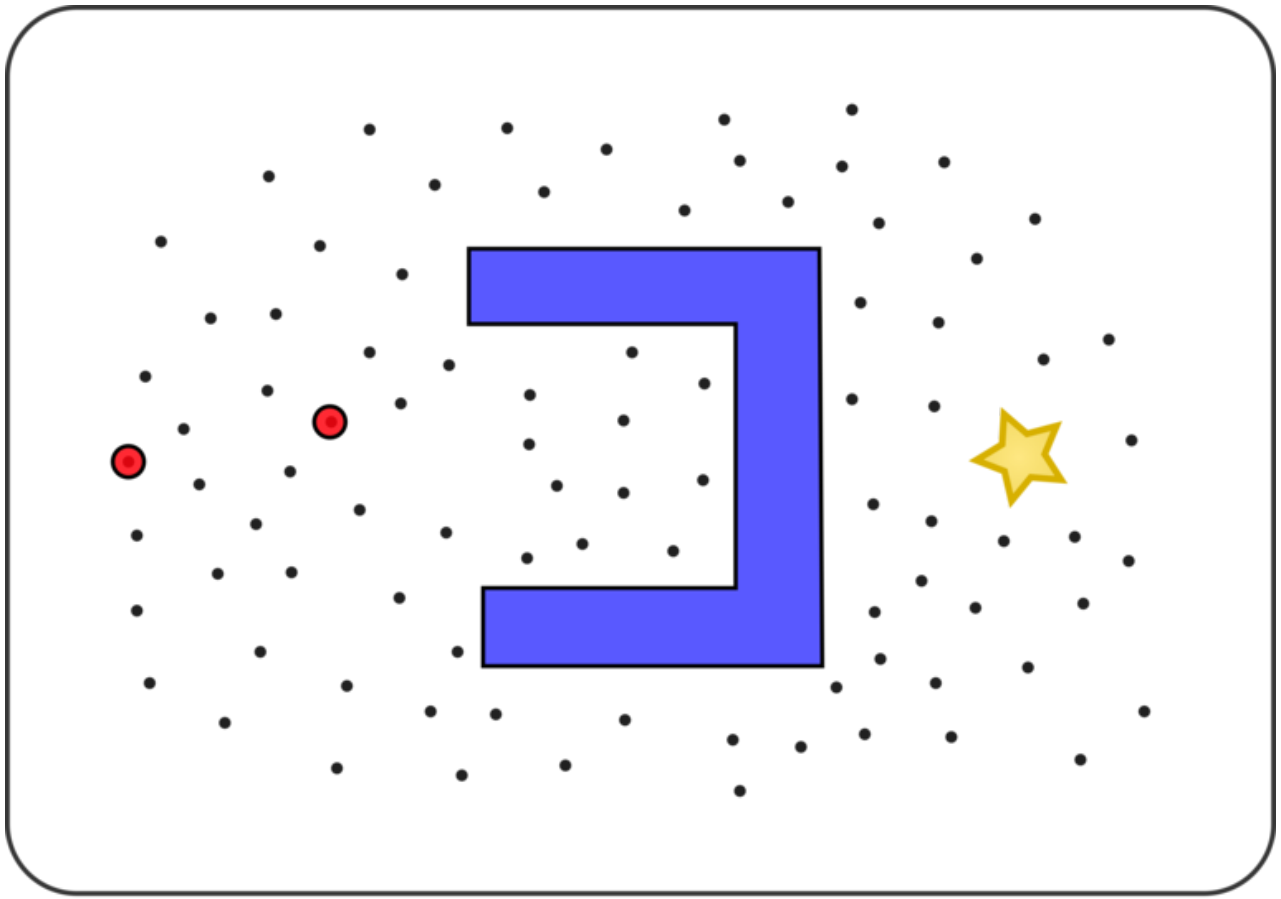
Graphical explanation



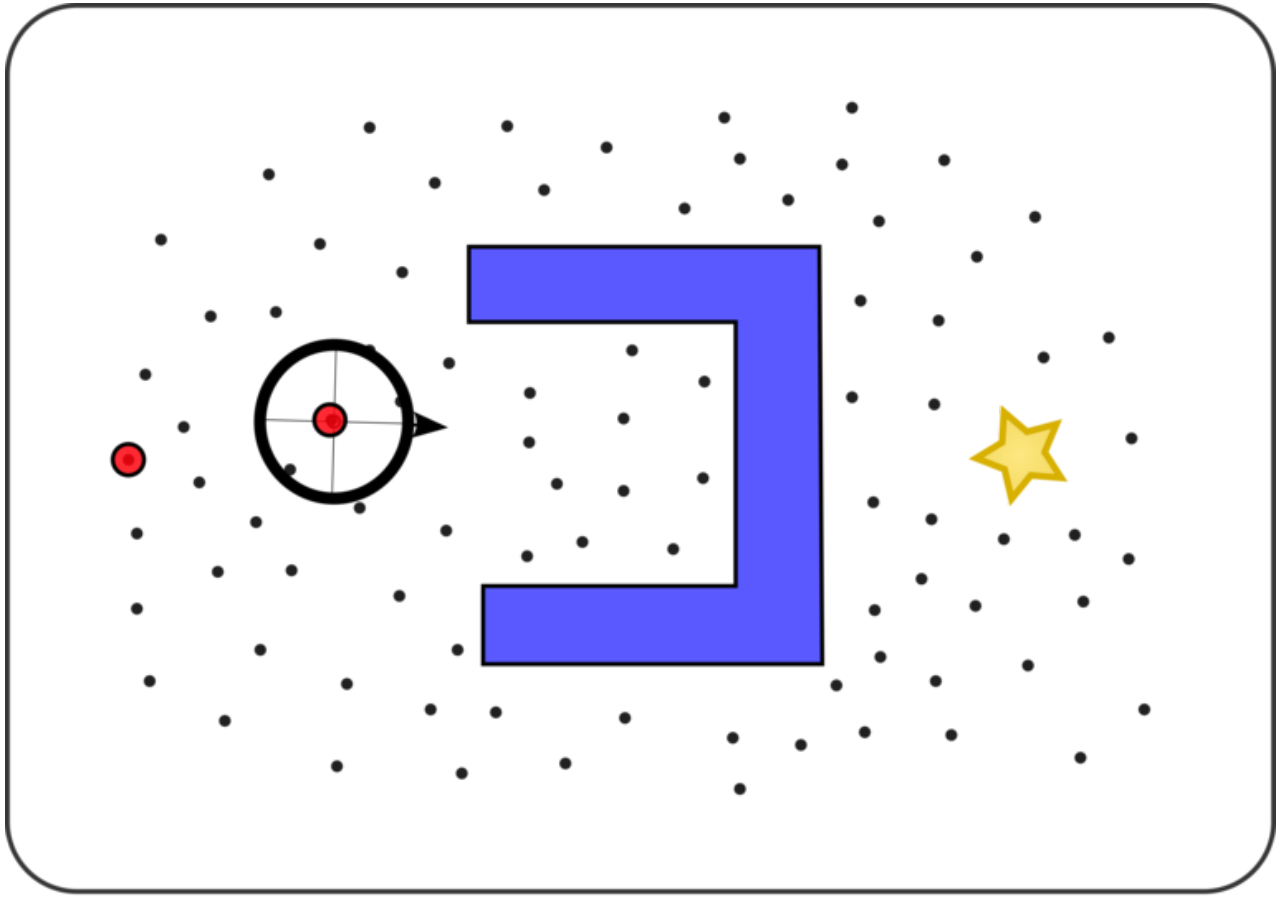
Graphical explanation



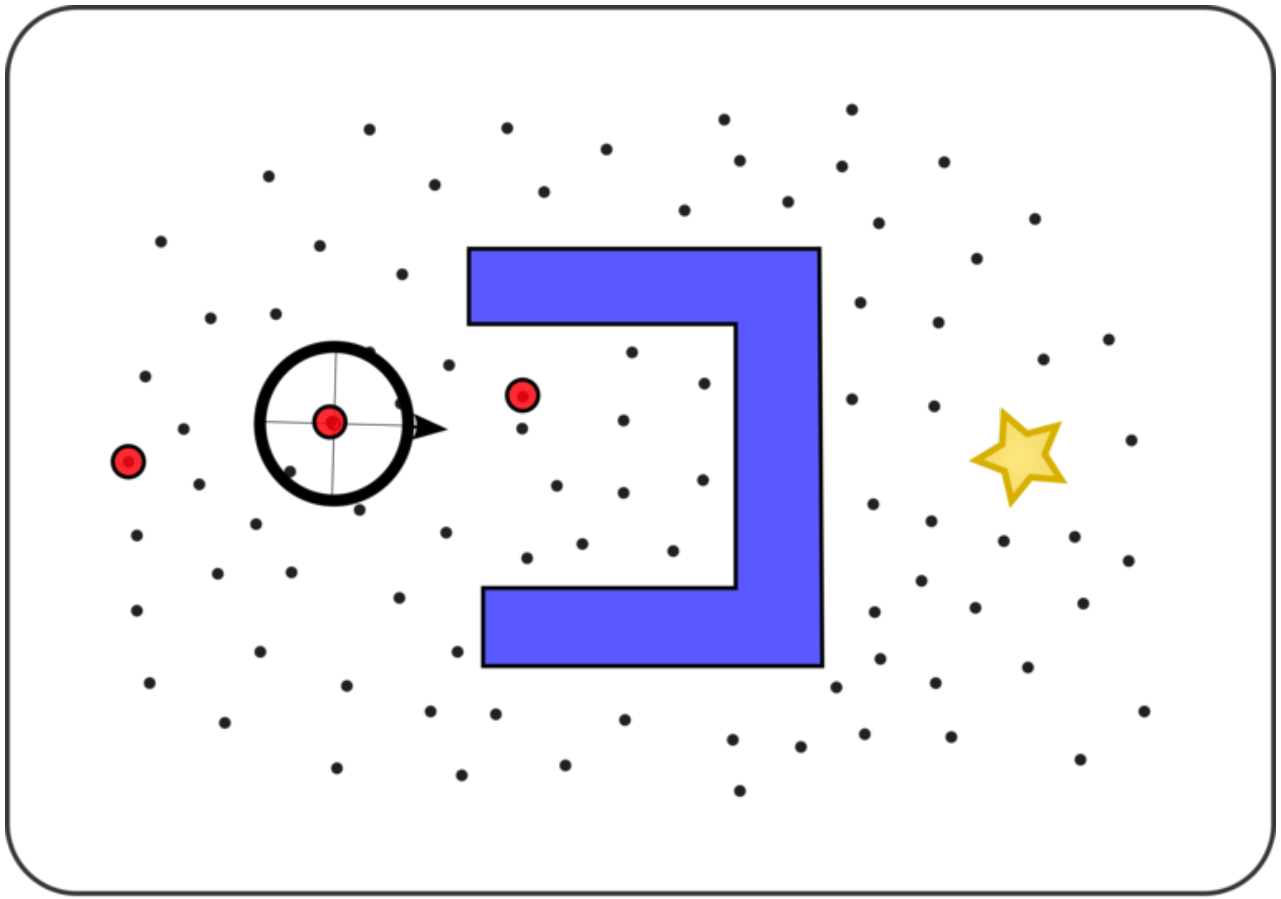
Graphical explanation



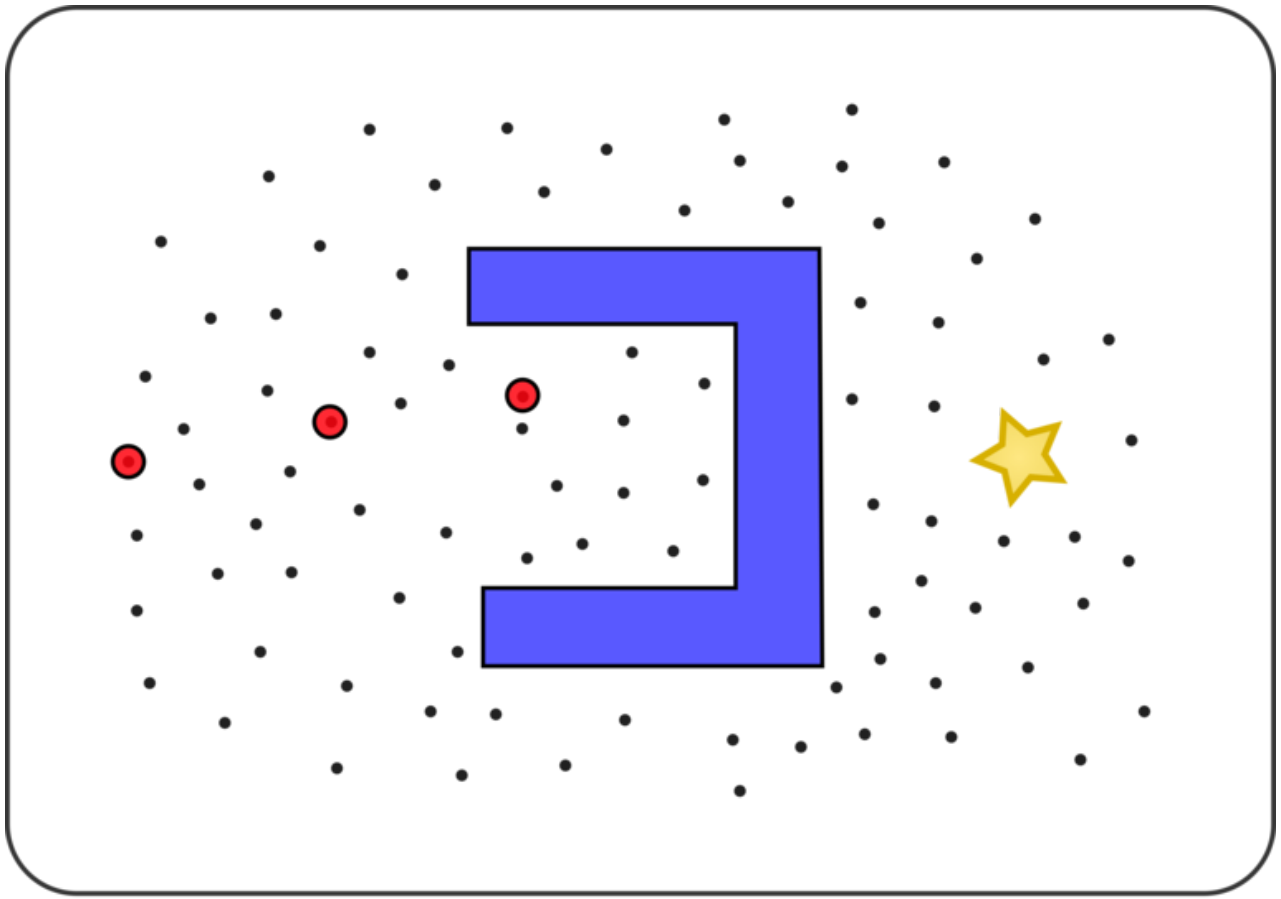
Graphical explanation



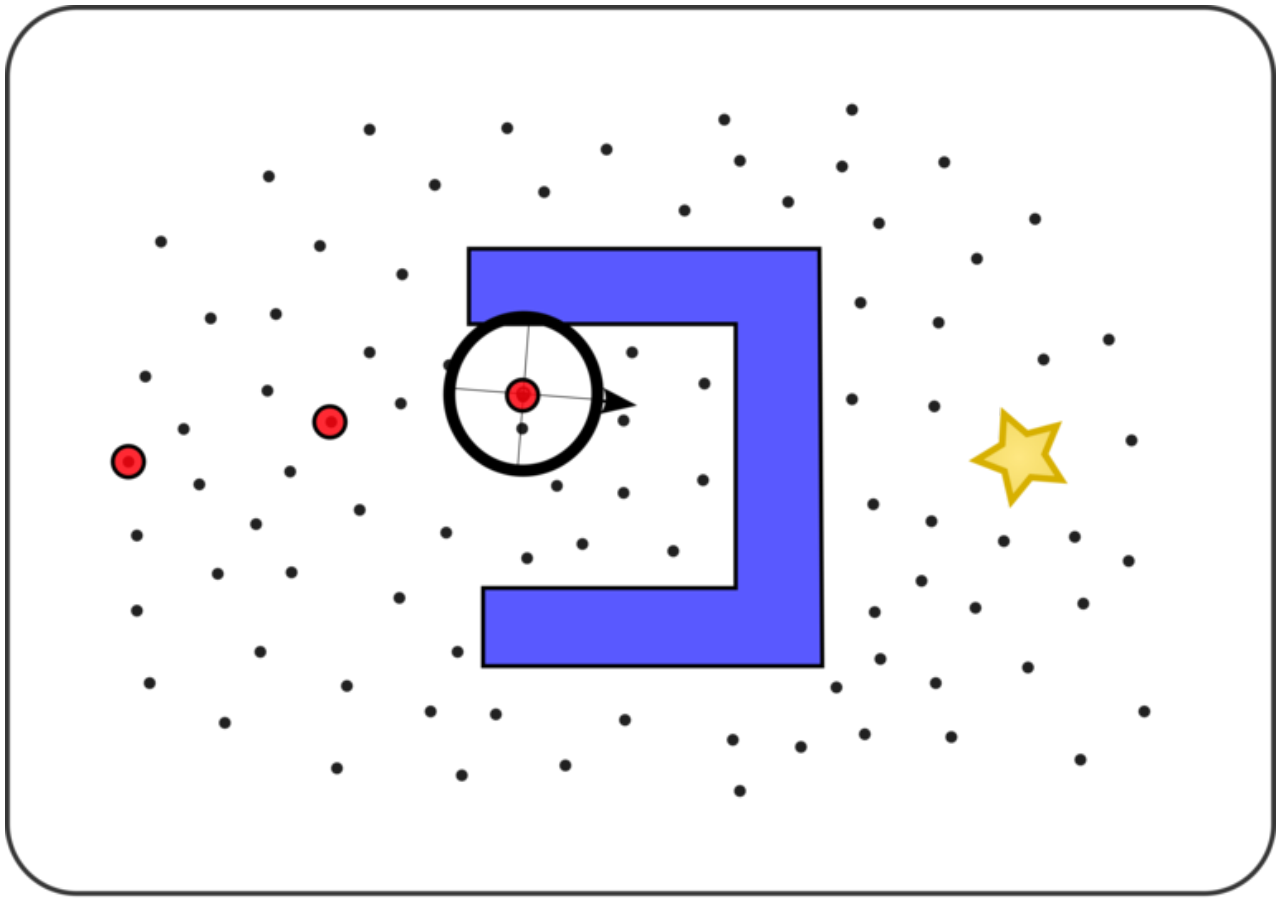
Graphical explanation



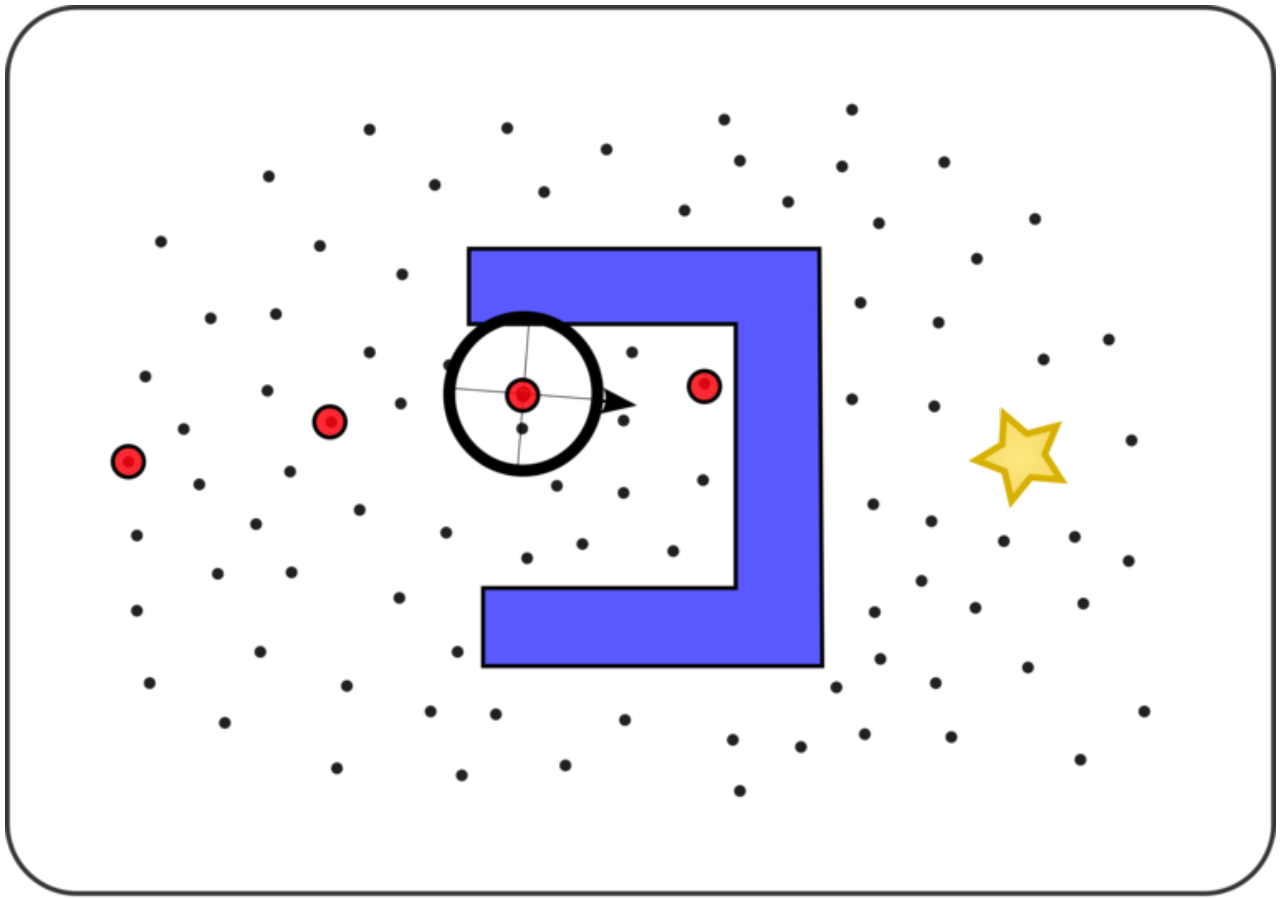
Graphical explanation



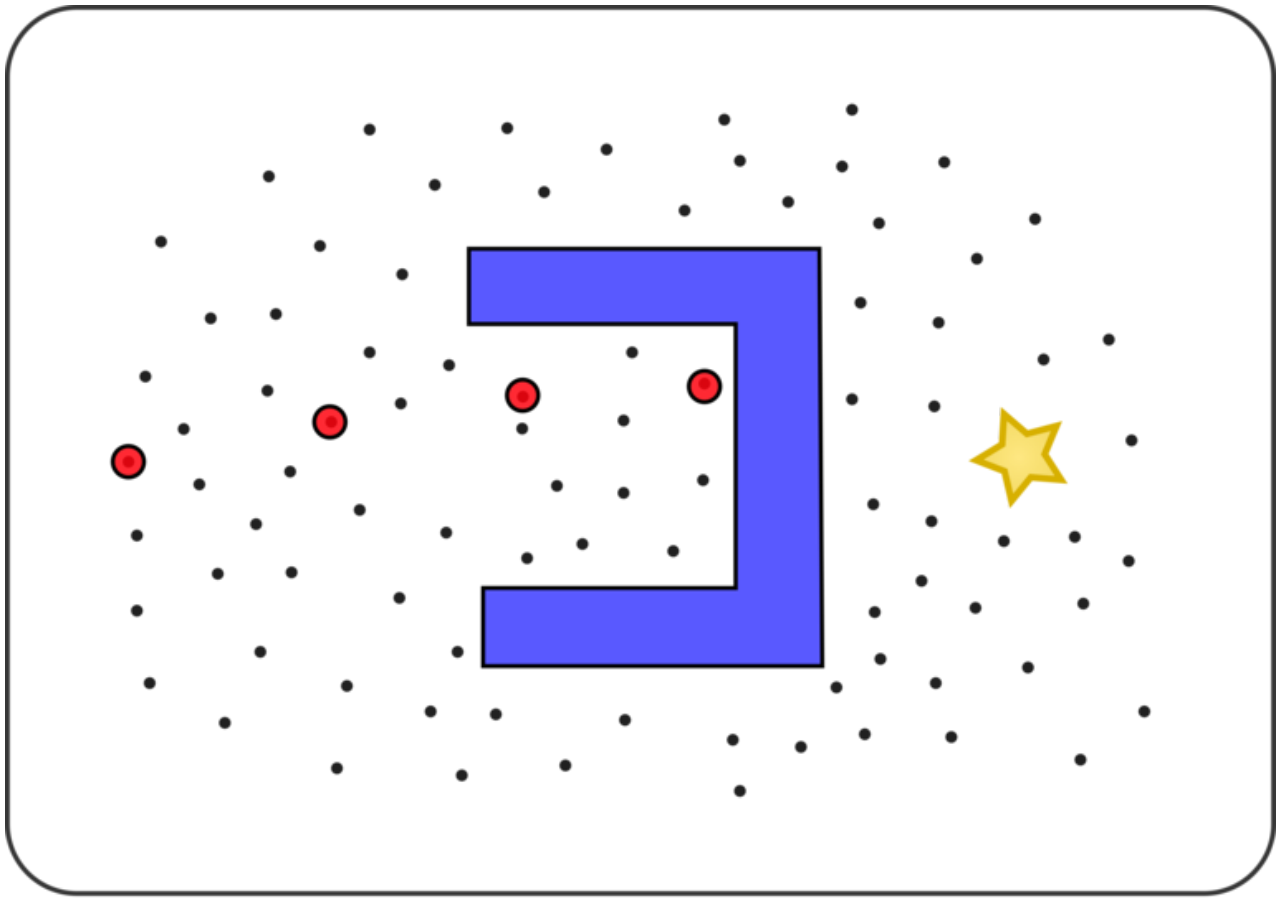
Graphical explanation



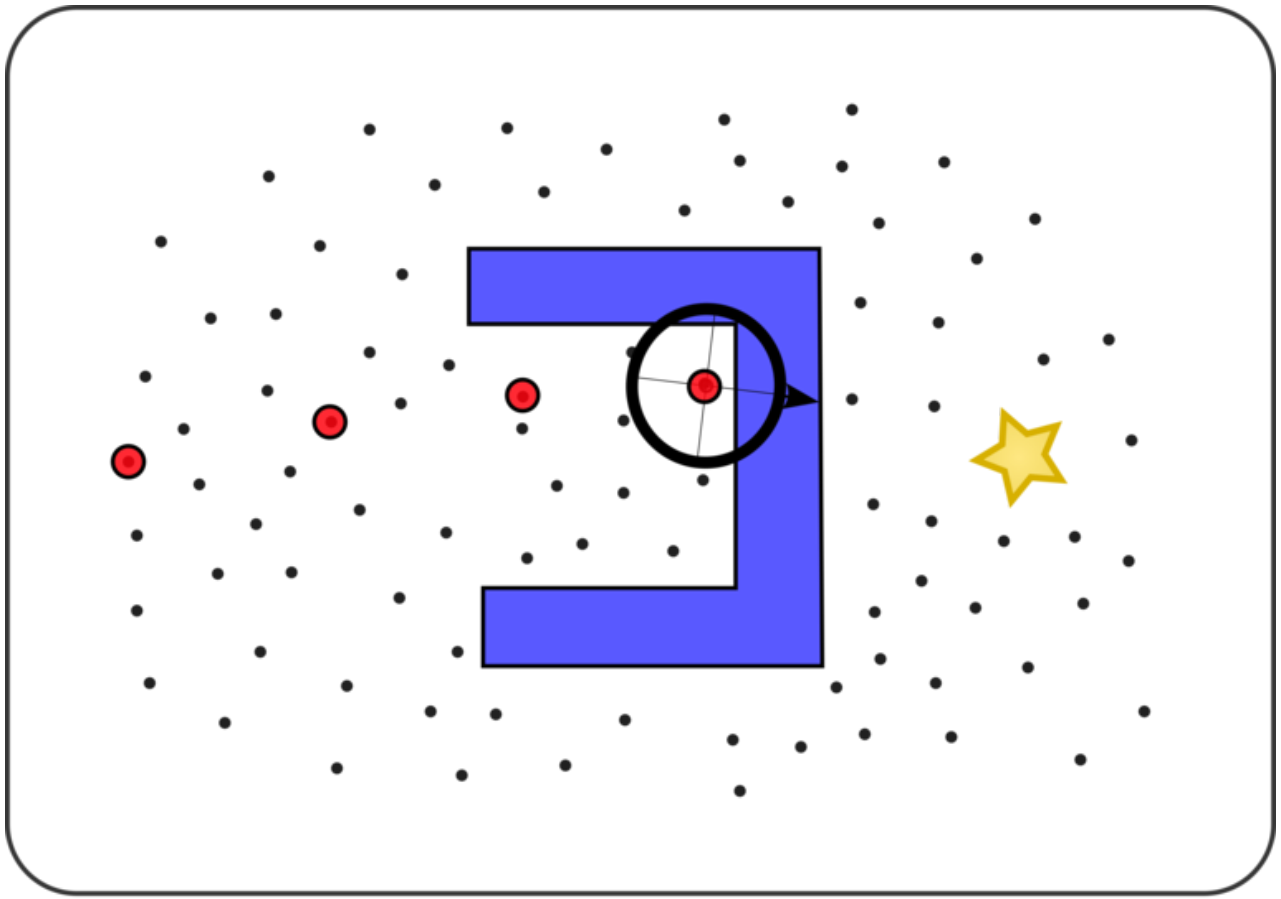
Graphical explanation



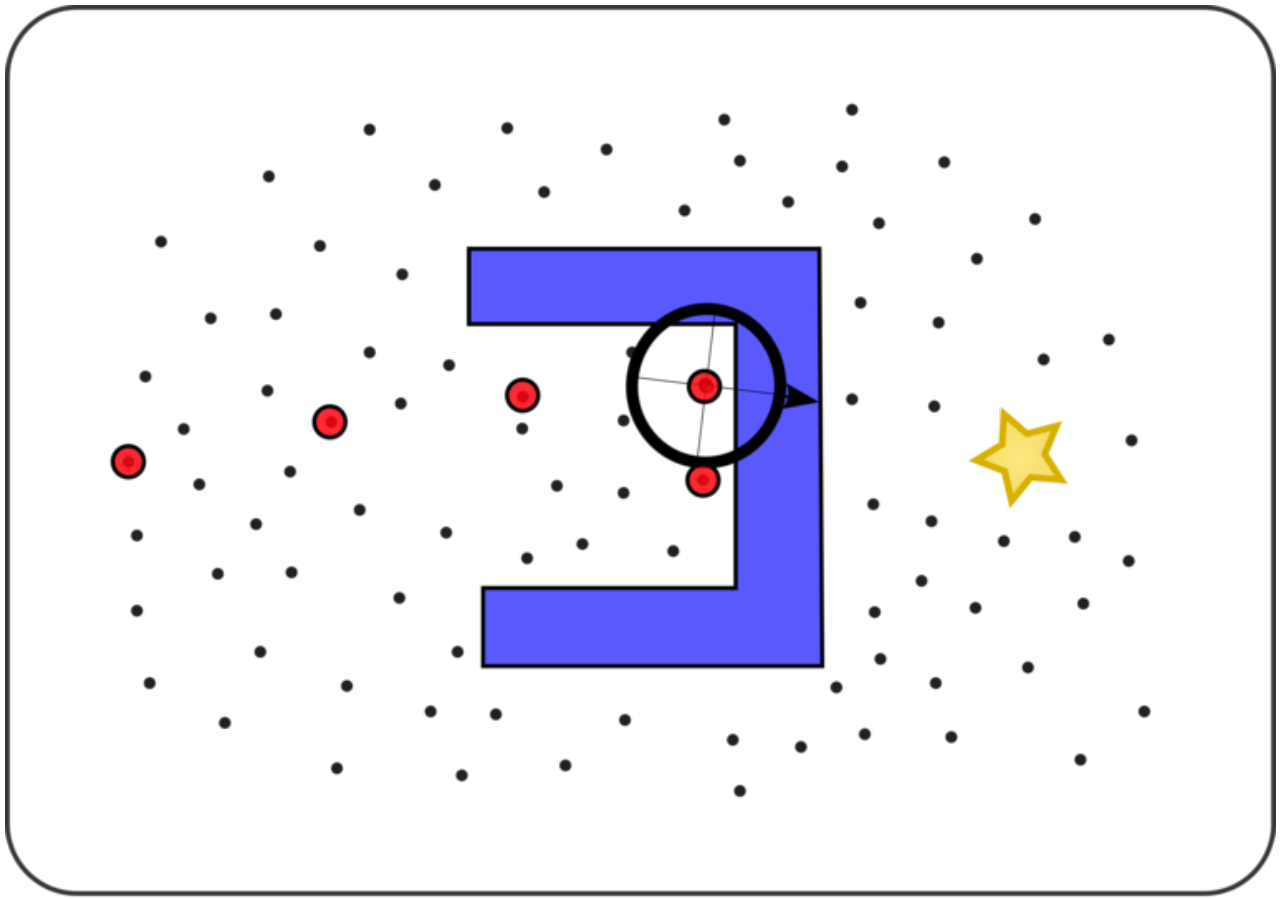
Graphical explanation



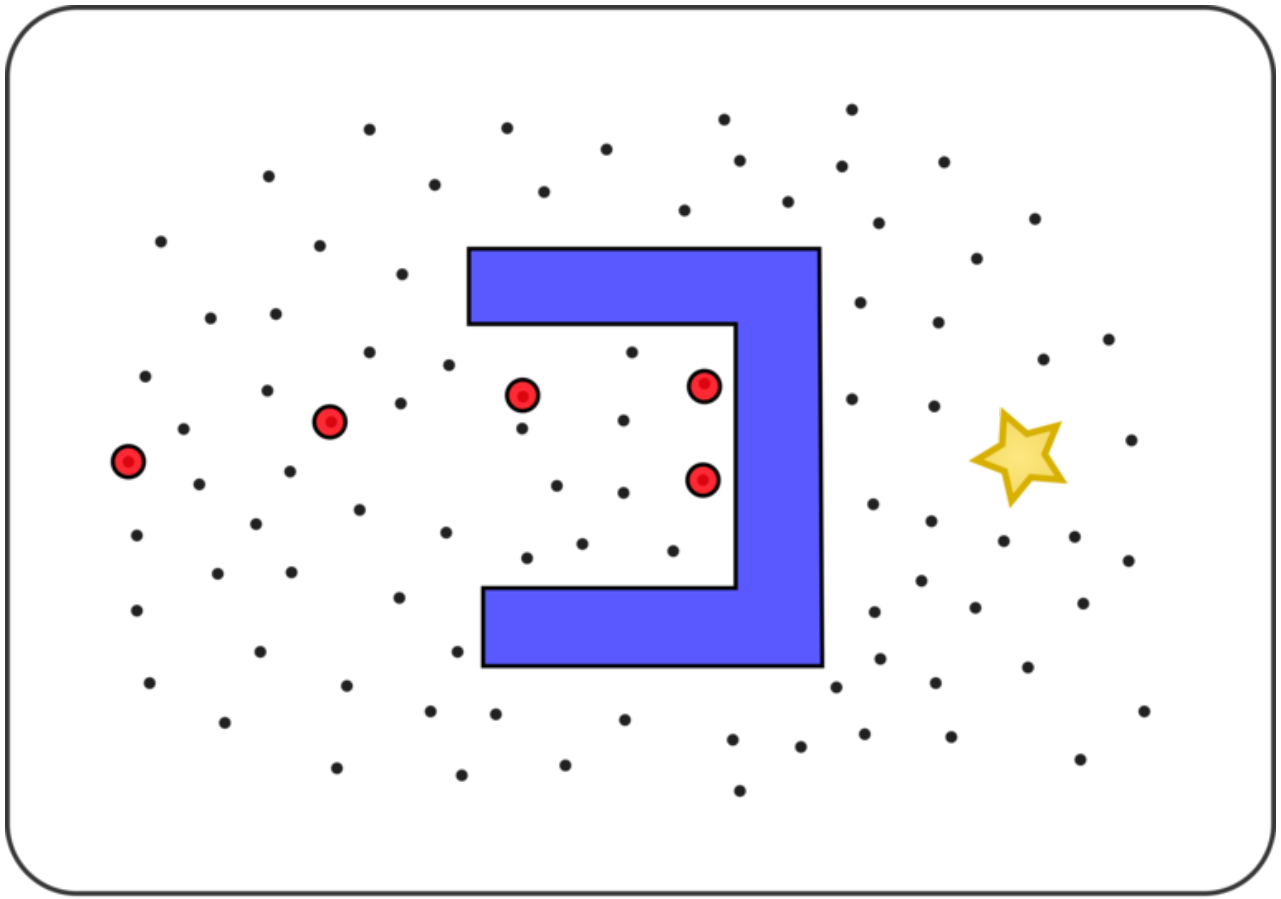
Graphical explanation



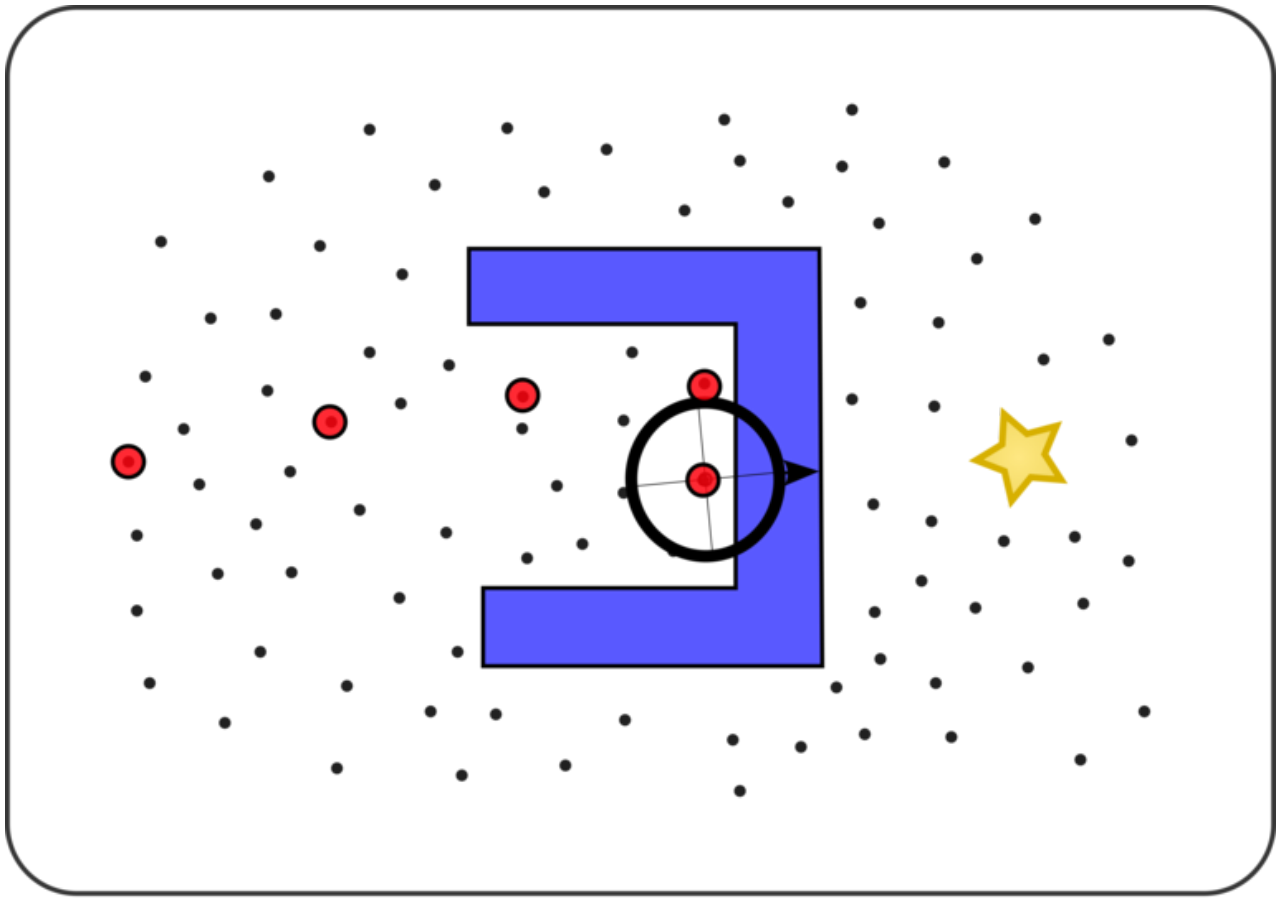
Graphical explanation



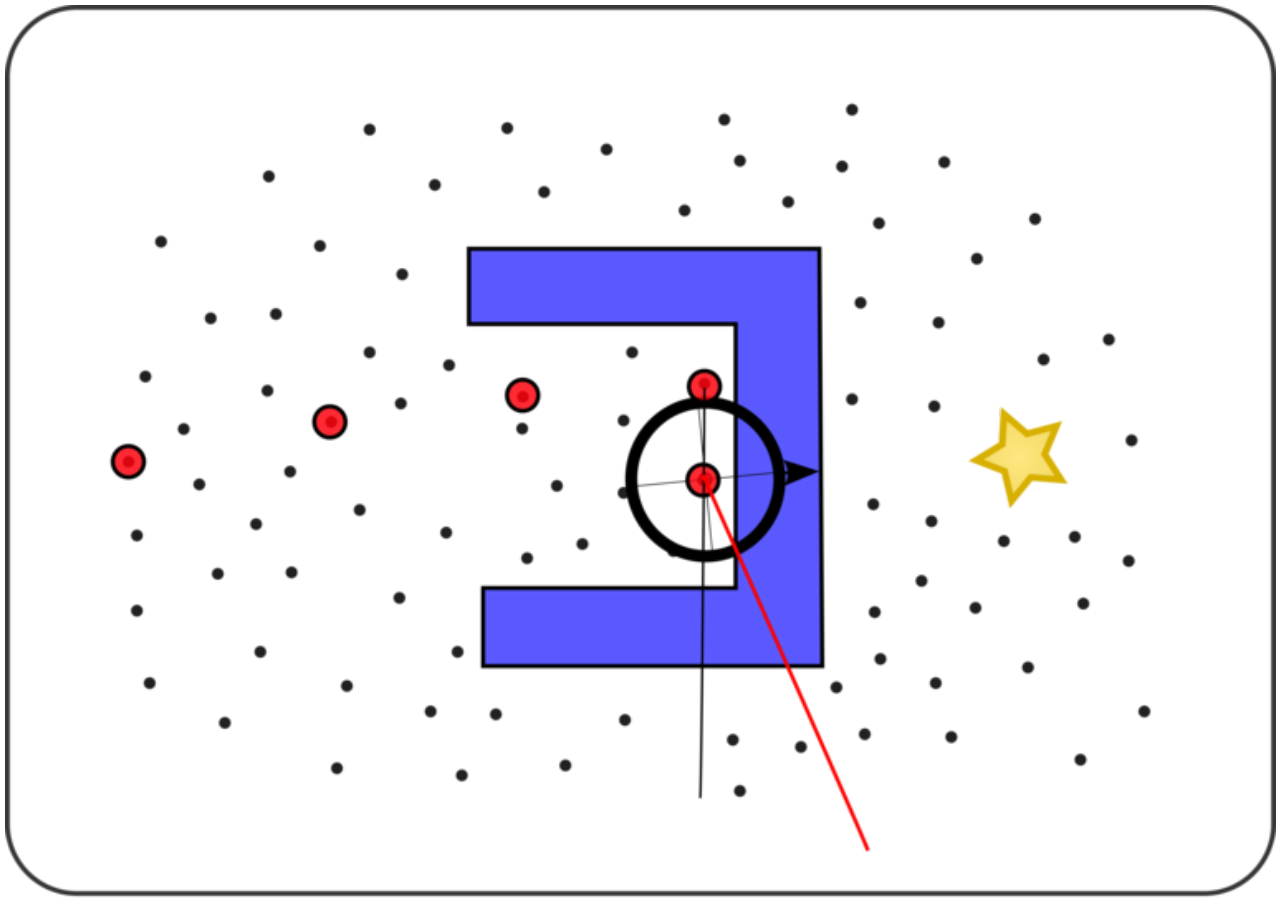
Graphical explanation



Graphical explanation

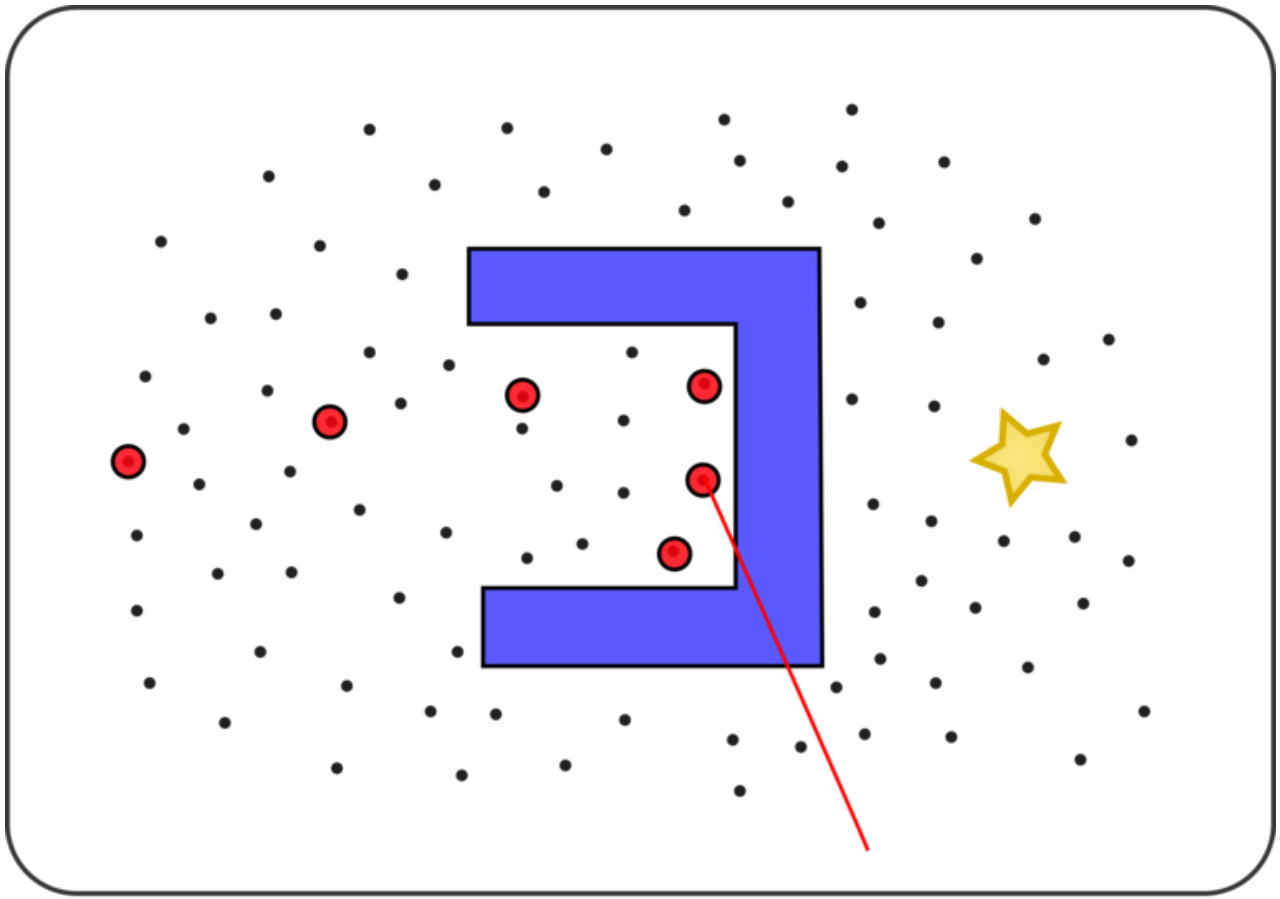


Graphical explanation



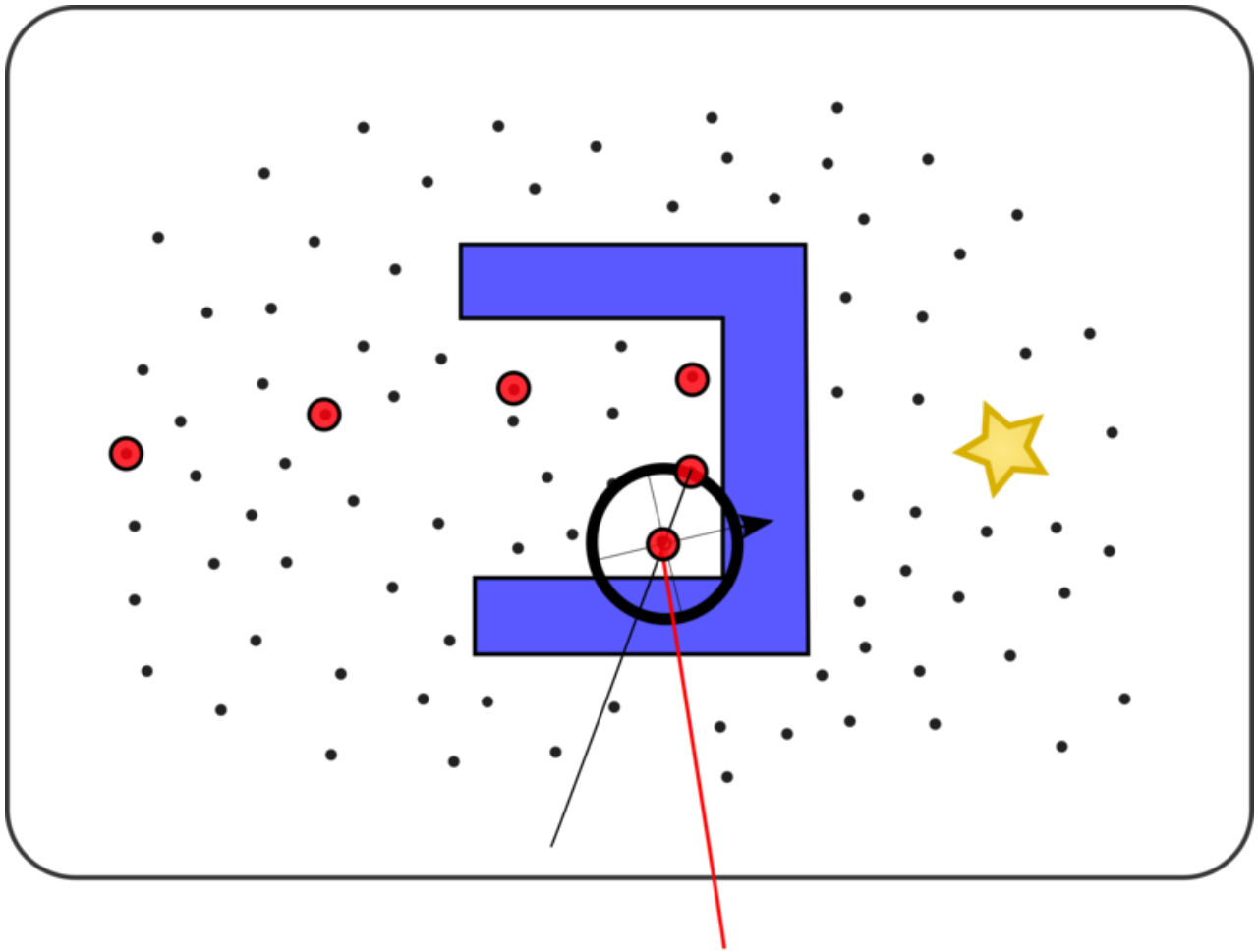
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



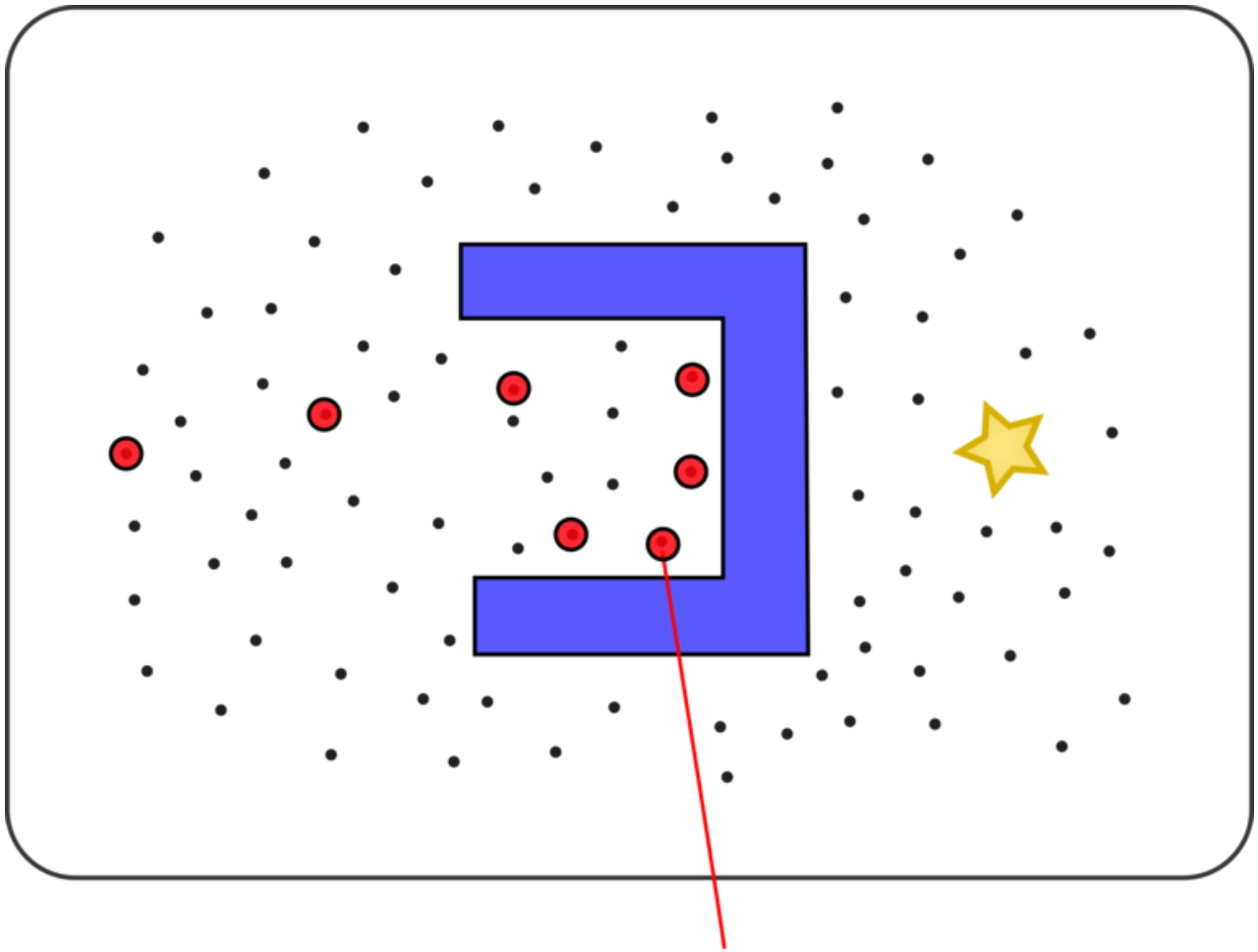
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



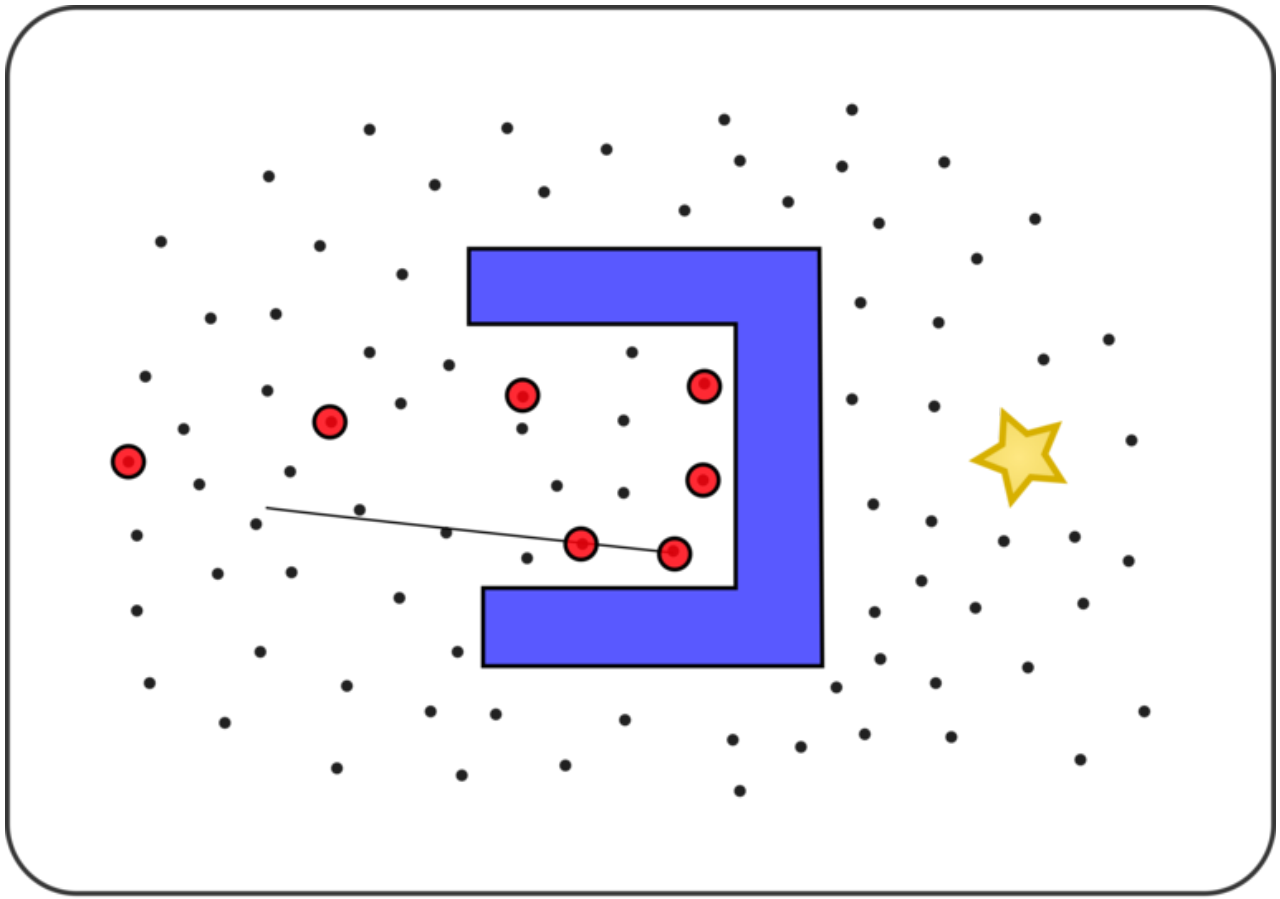
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



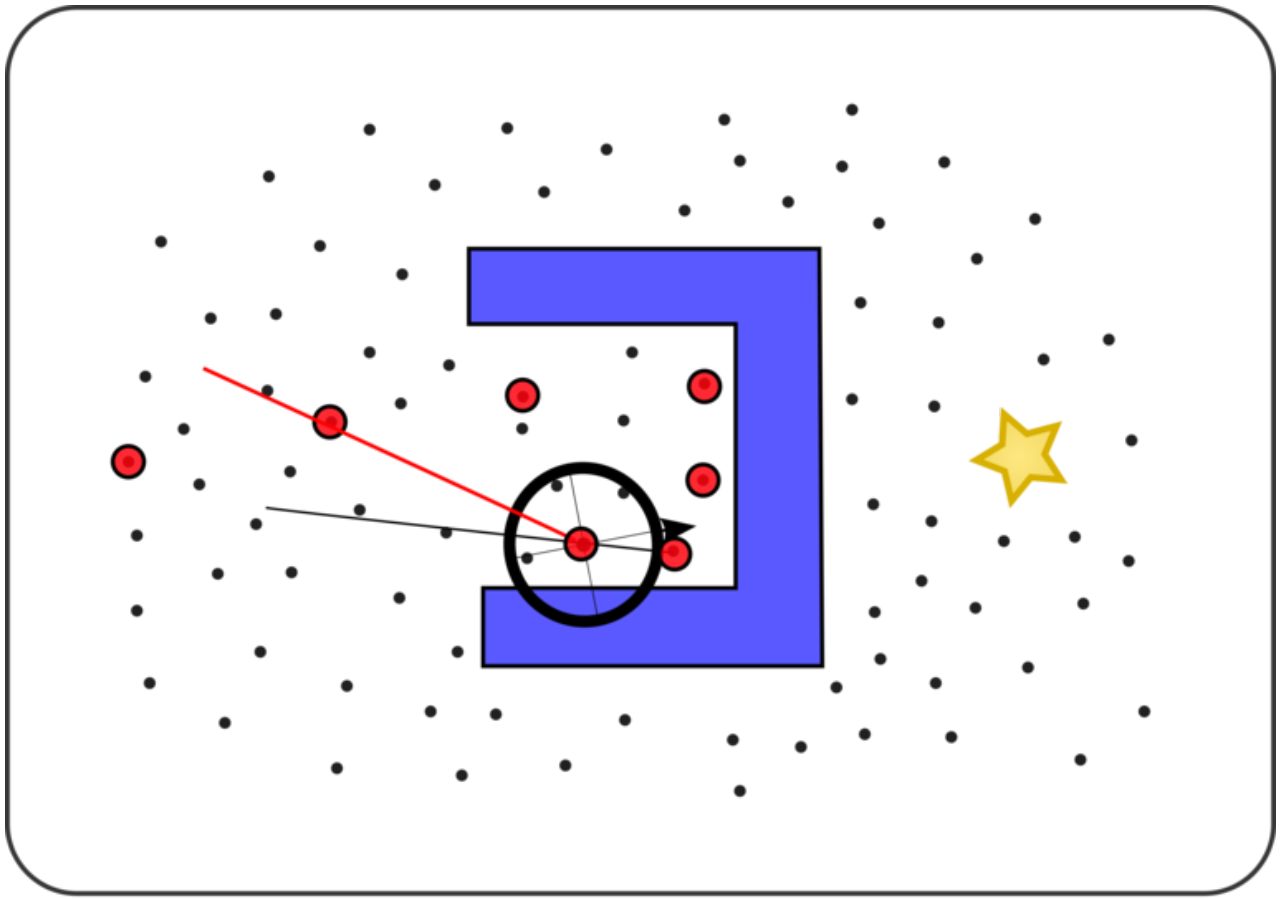
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



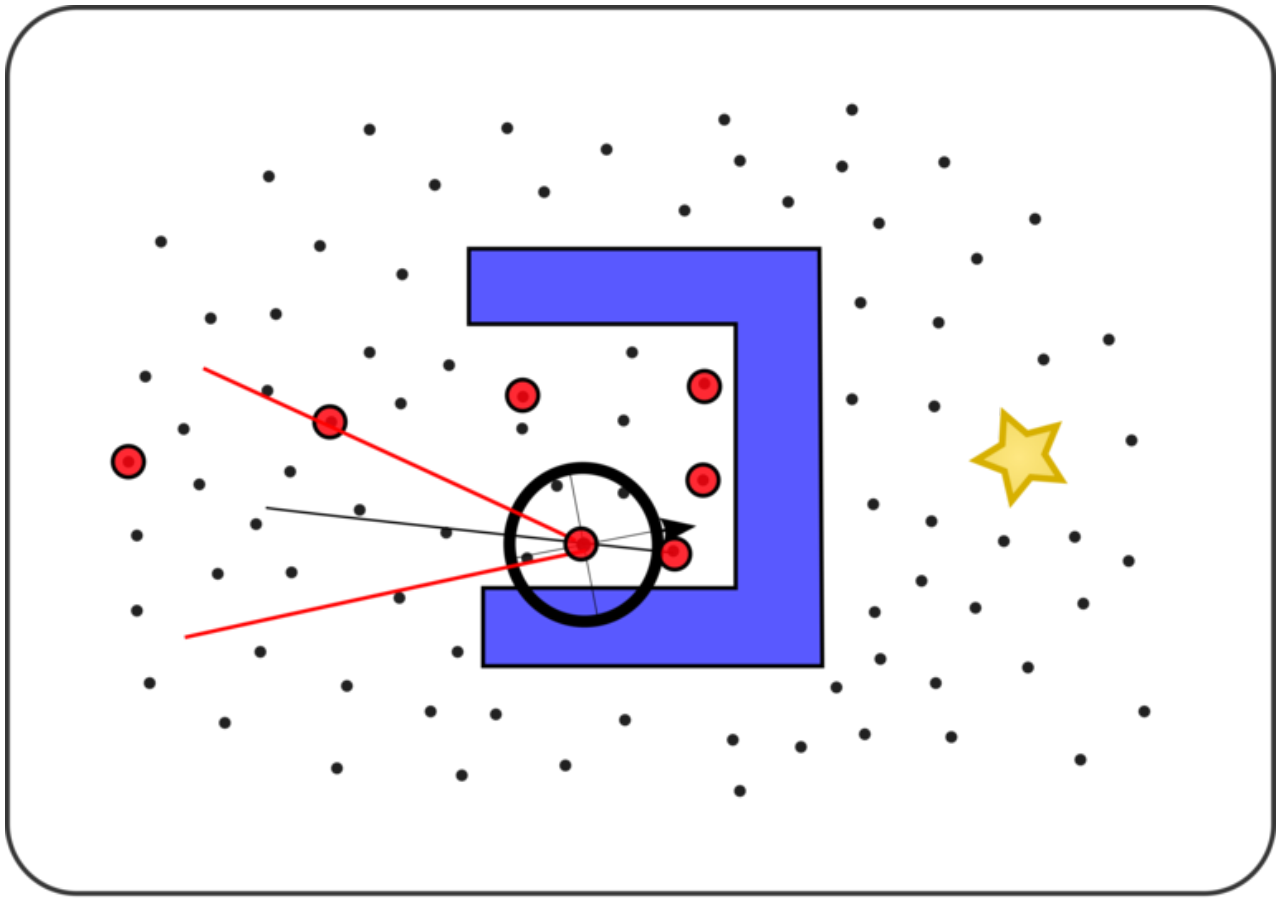
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



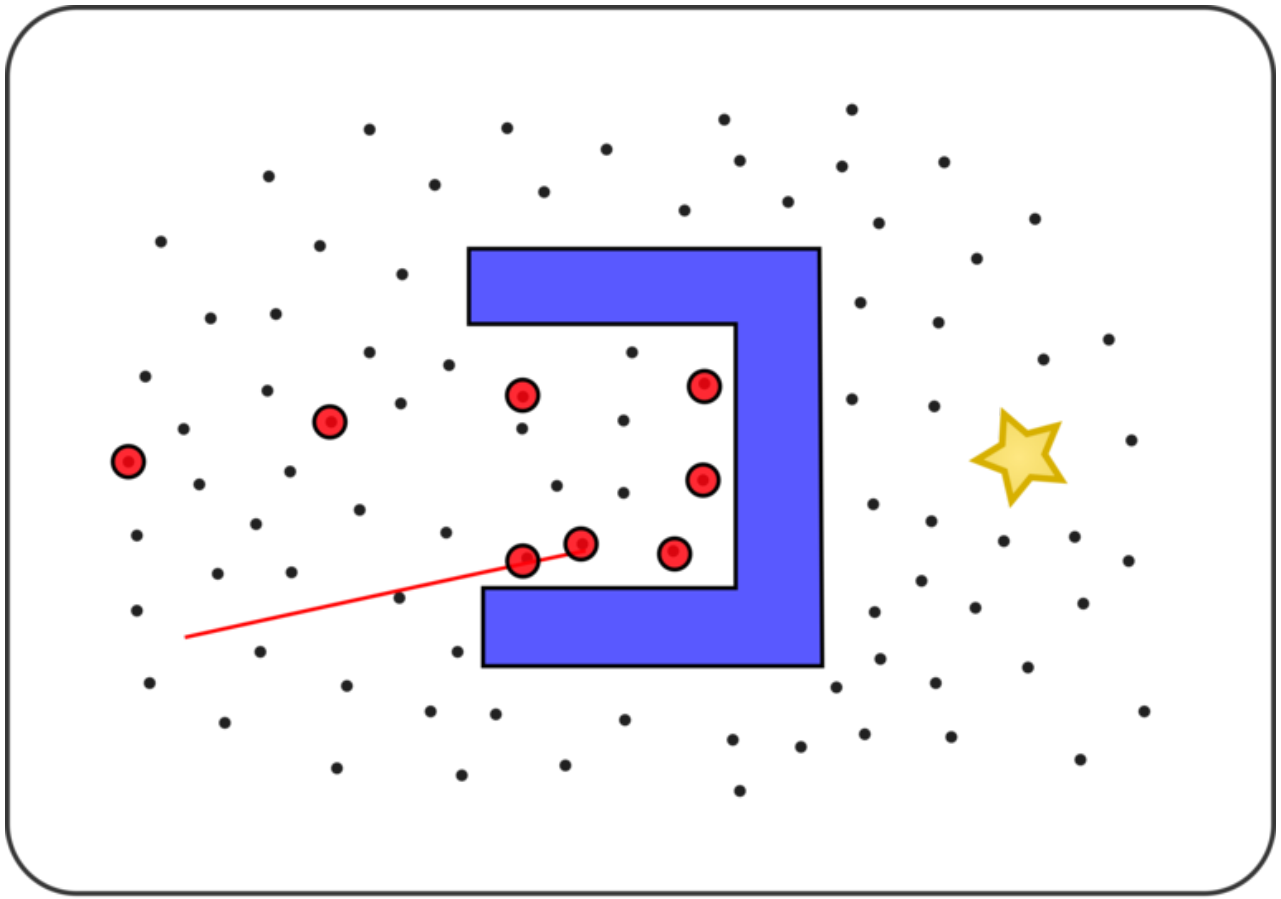
- Raise the flag $\rightsquigarrow$ use recovery mode.

Graphical explanation



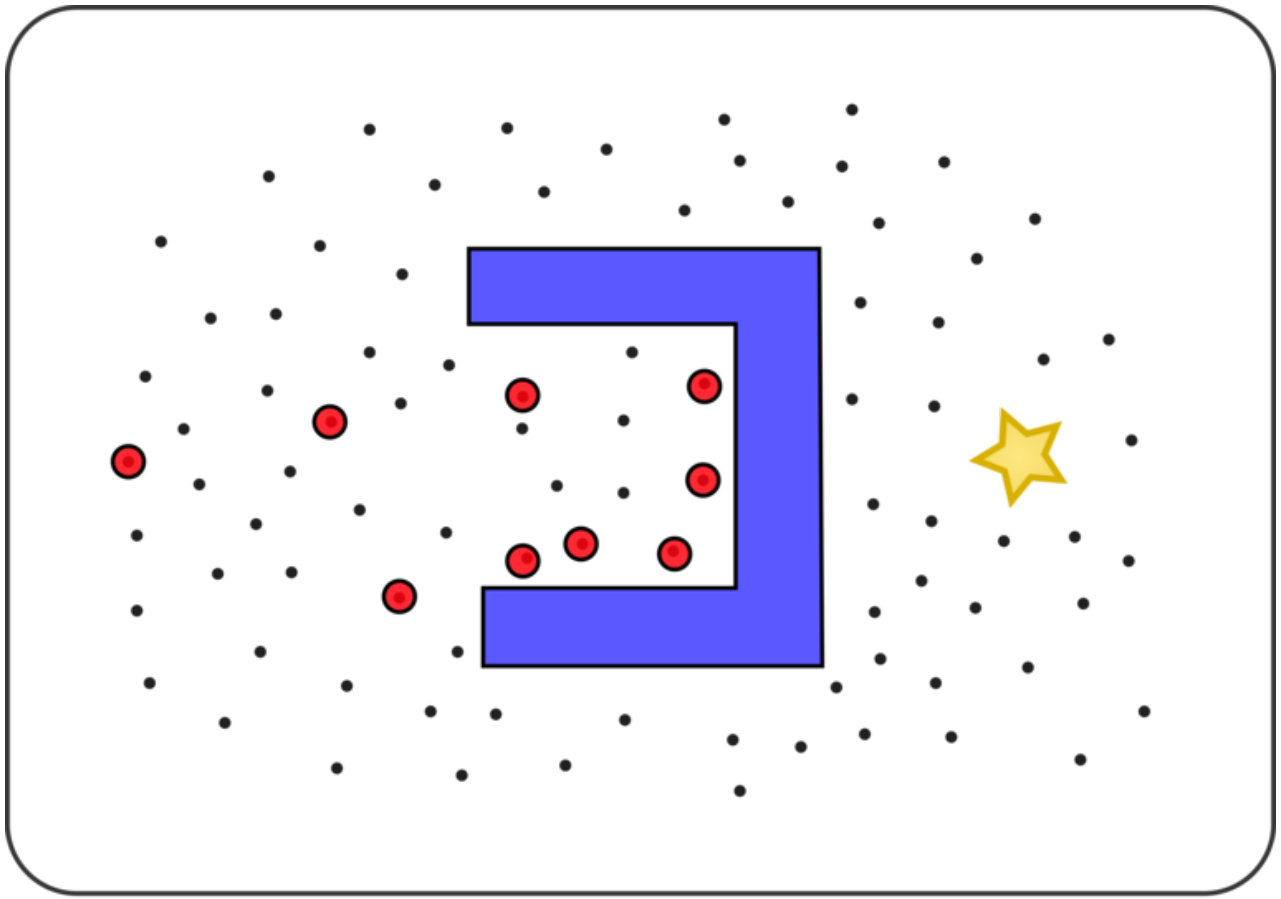
- Force turning left.

Graphical explanation



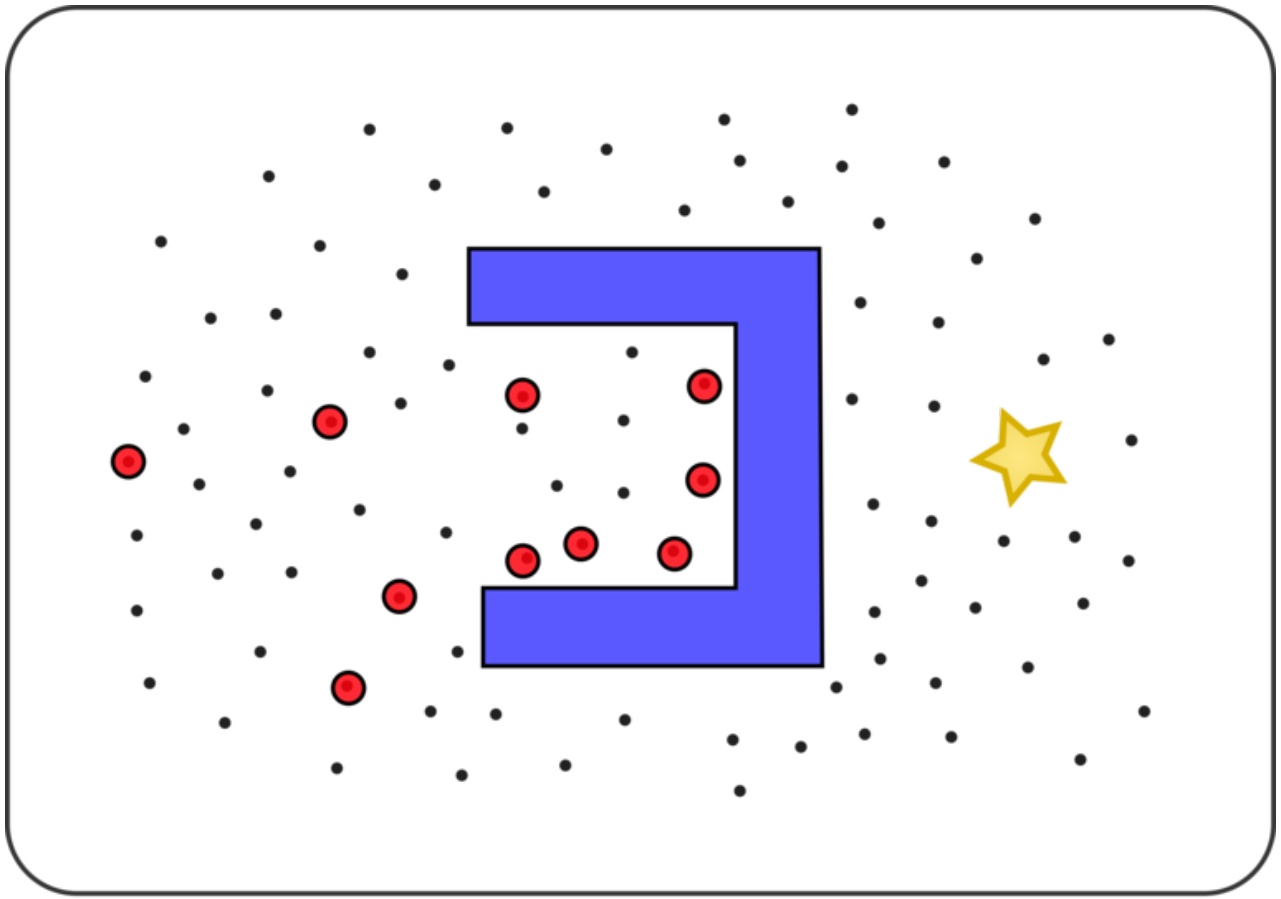
- The flag stays up.

Graphical explanation



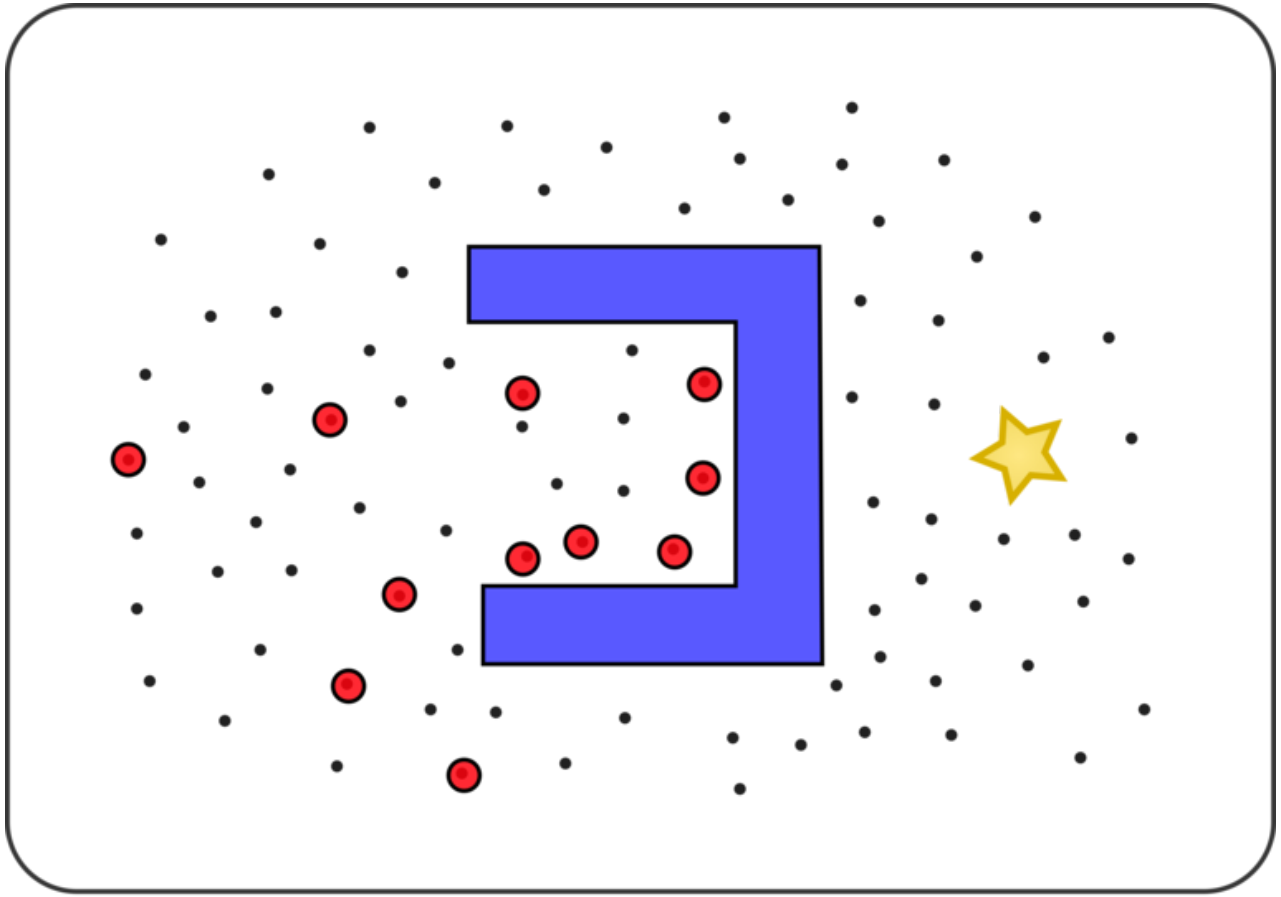
- The flag stays up.

Graphical explanation



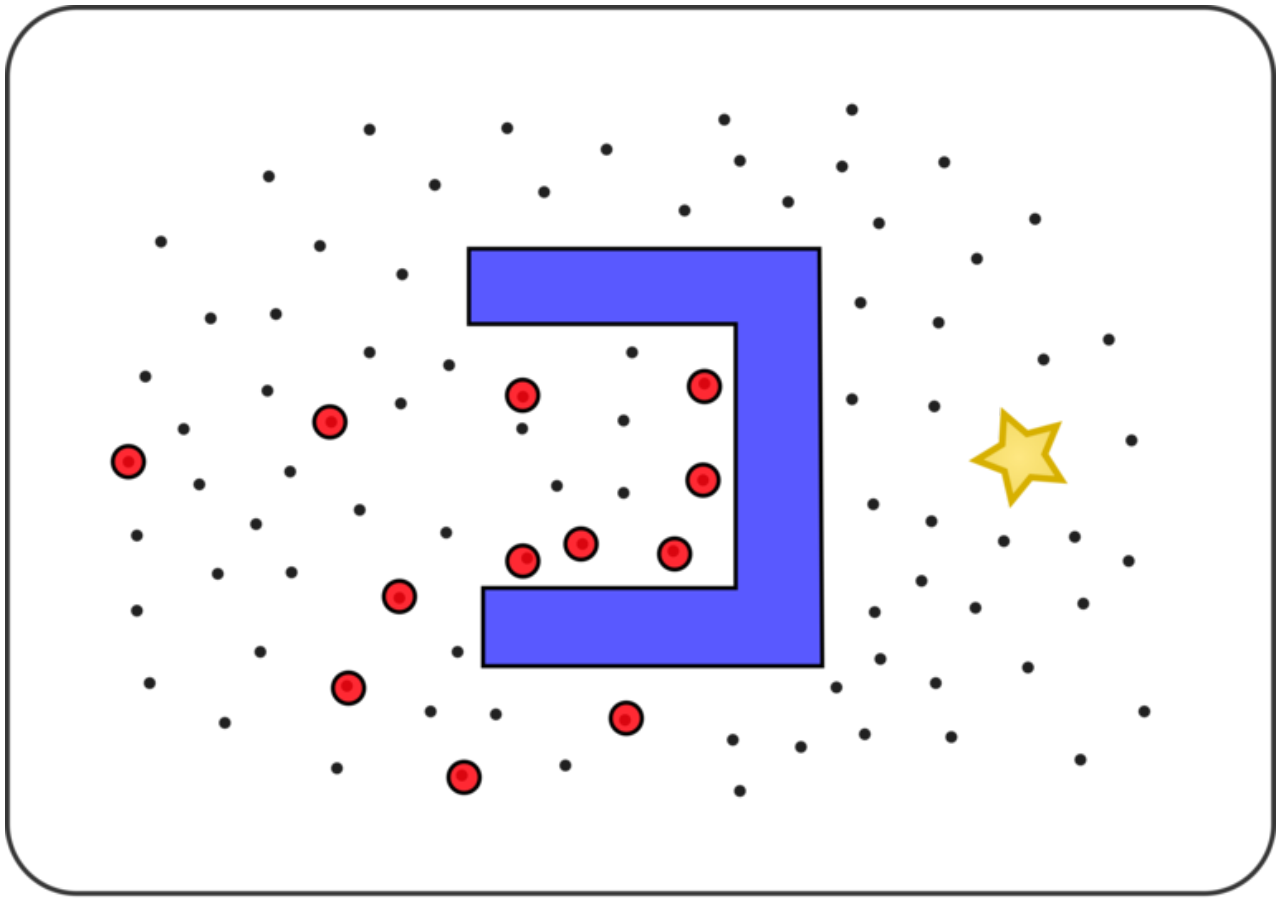
- The flag stays up.

Graphical explanation

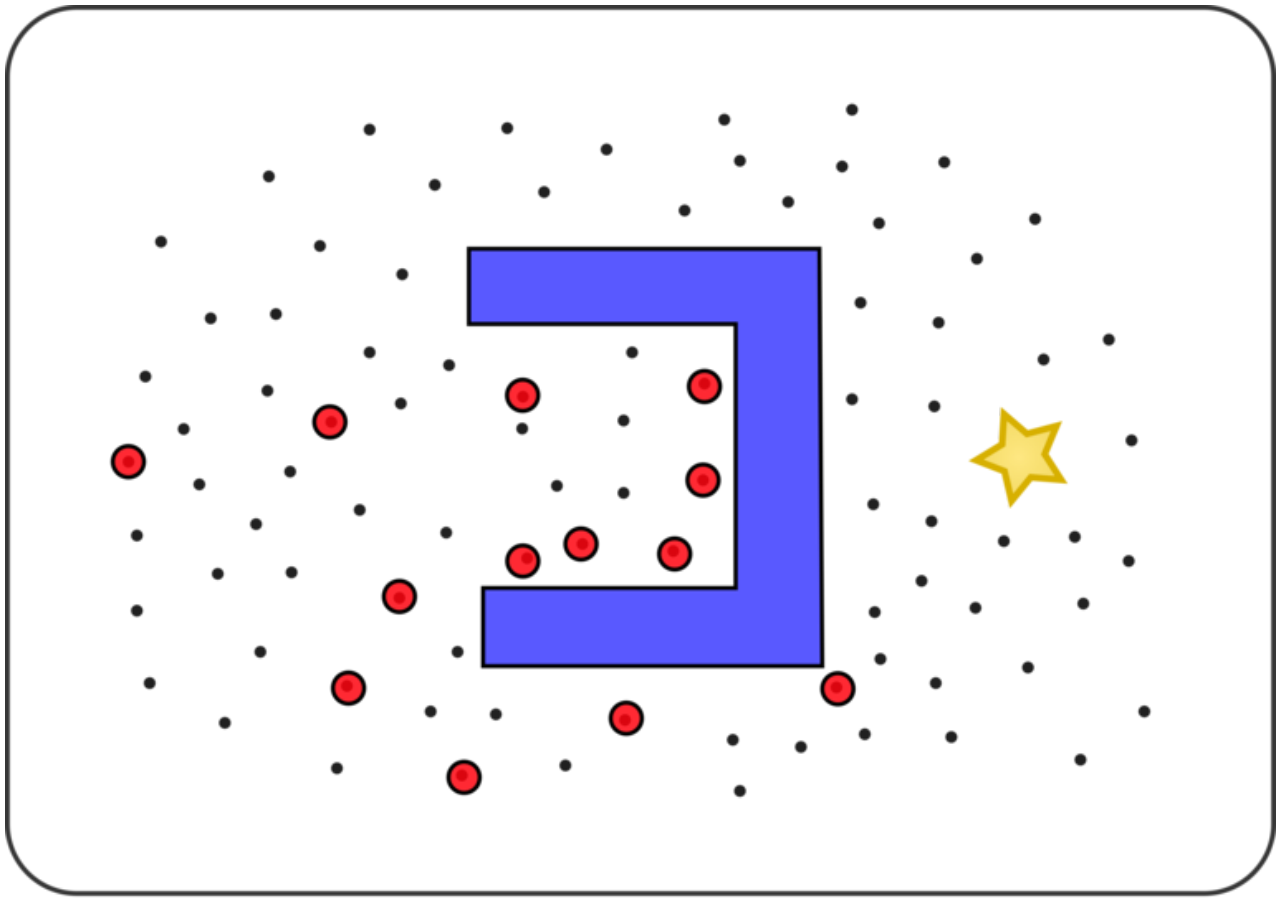


- Put the flag down.

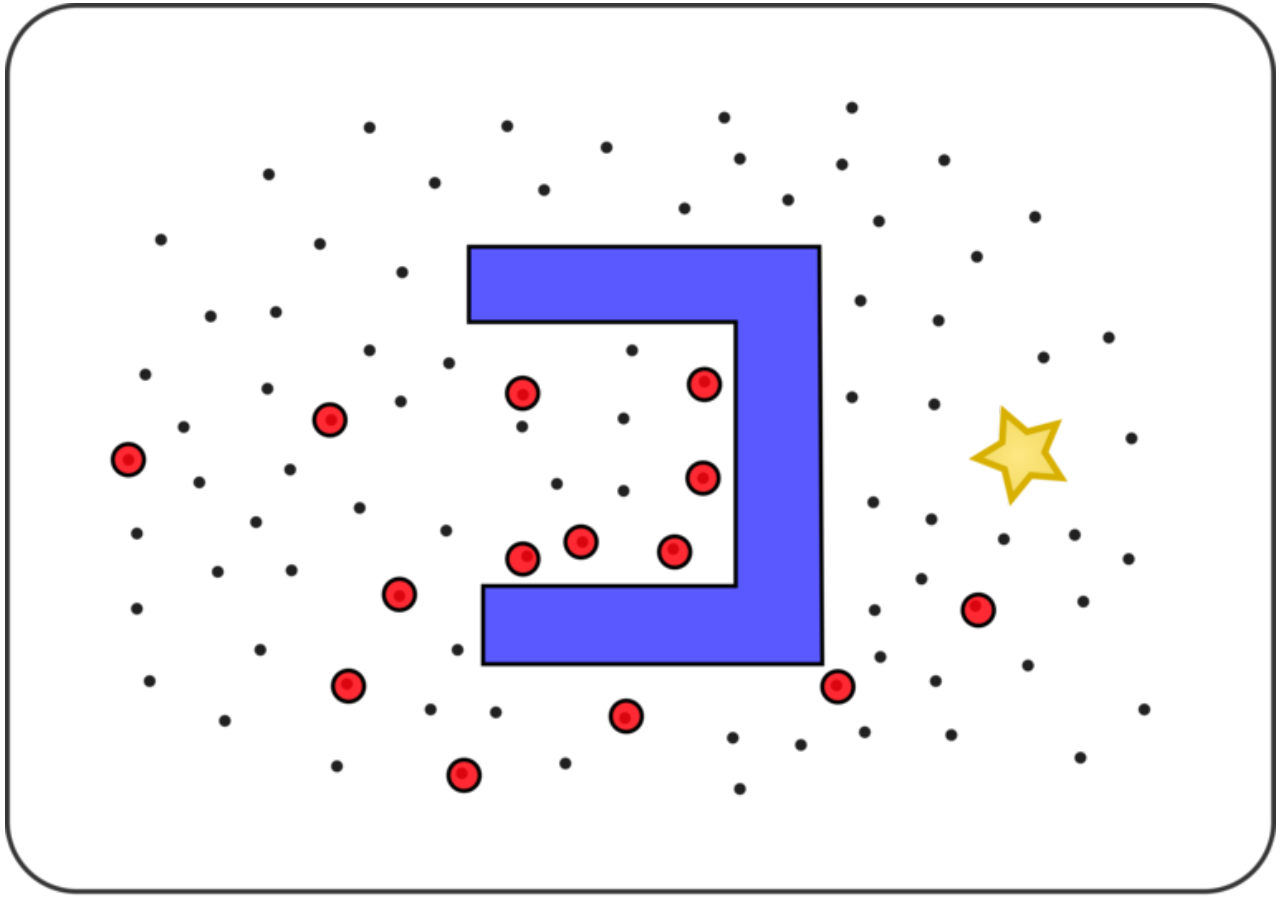
Graphical explanation



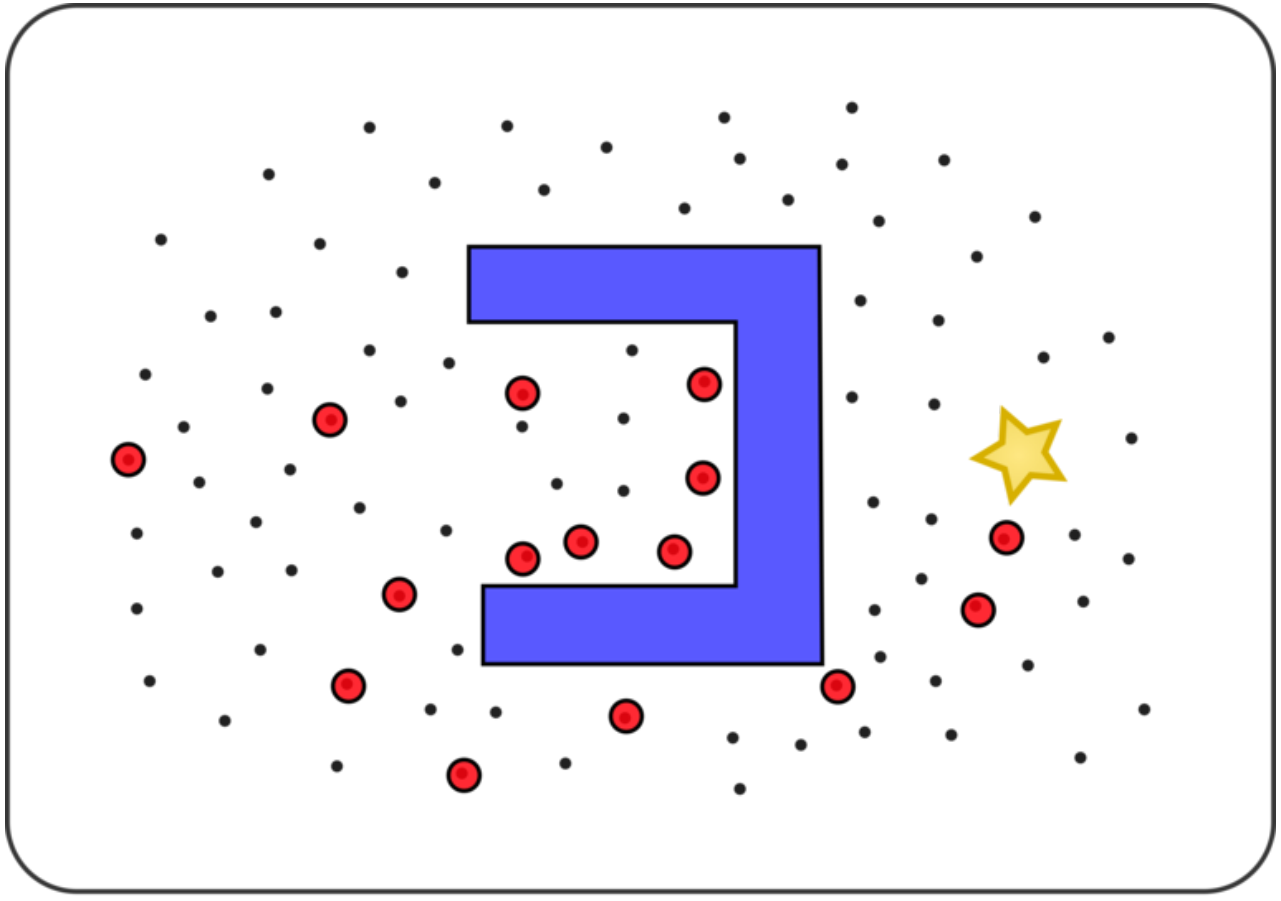
Graphical explanation



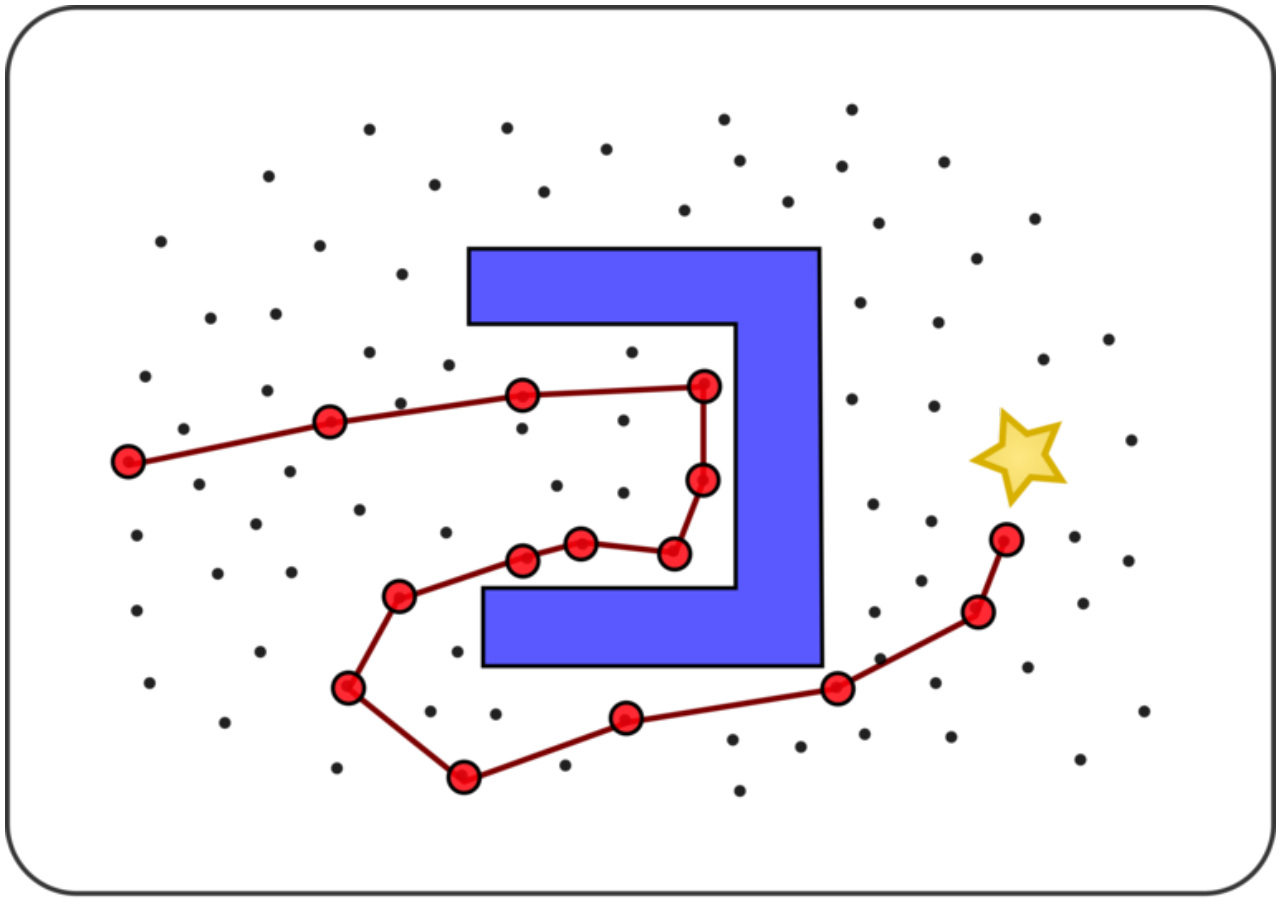
Graphical explanation



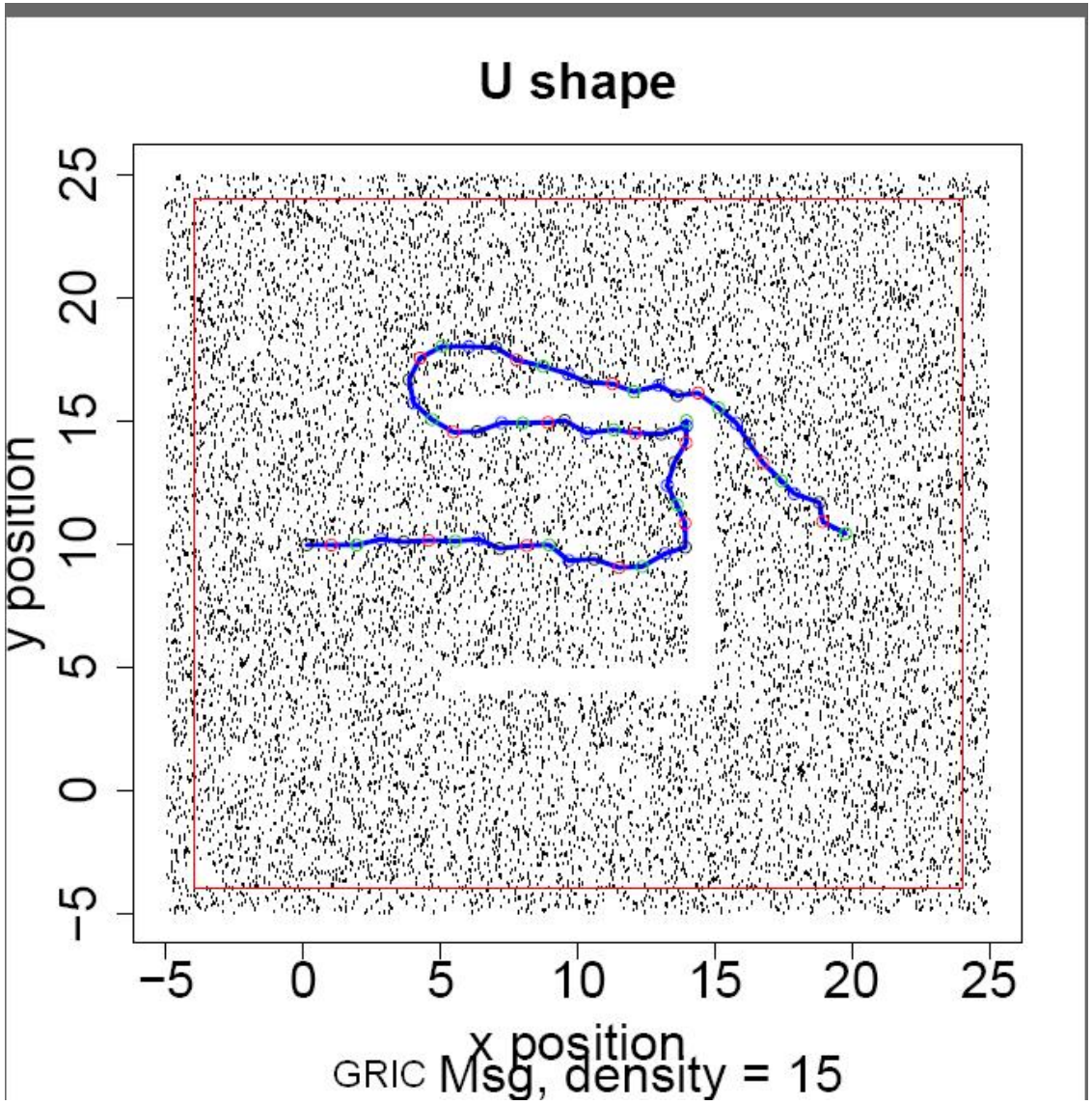
Graphical explanation



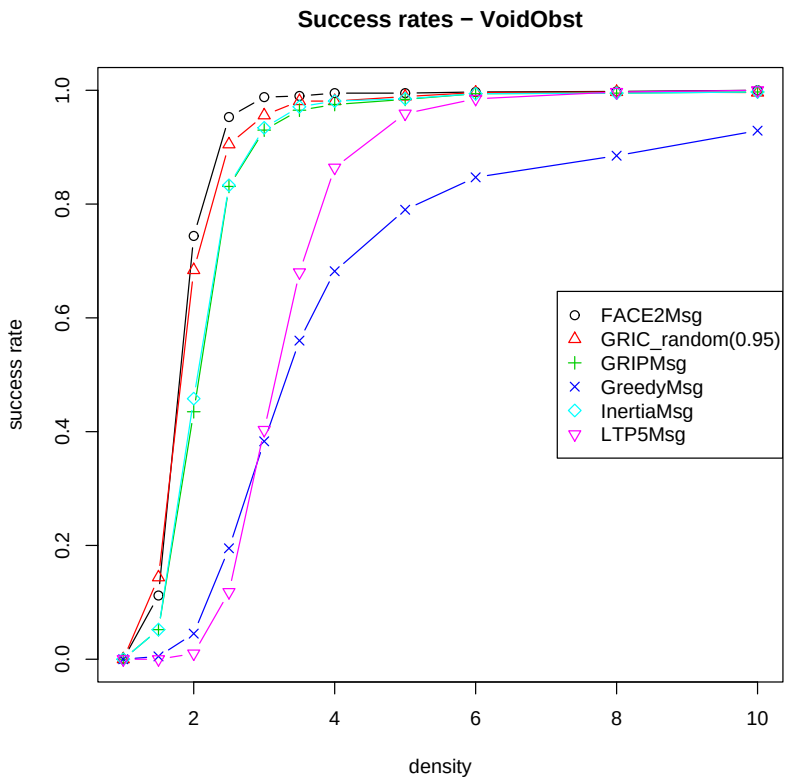
Graphical explanation

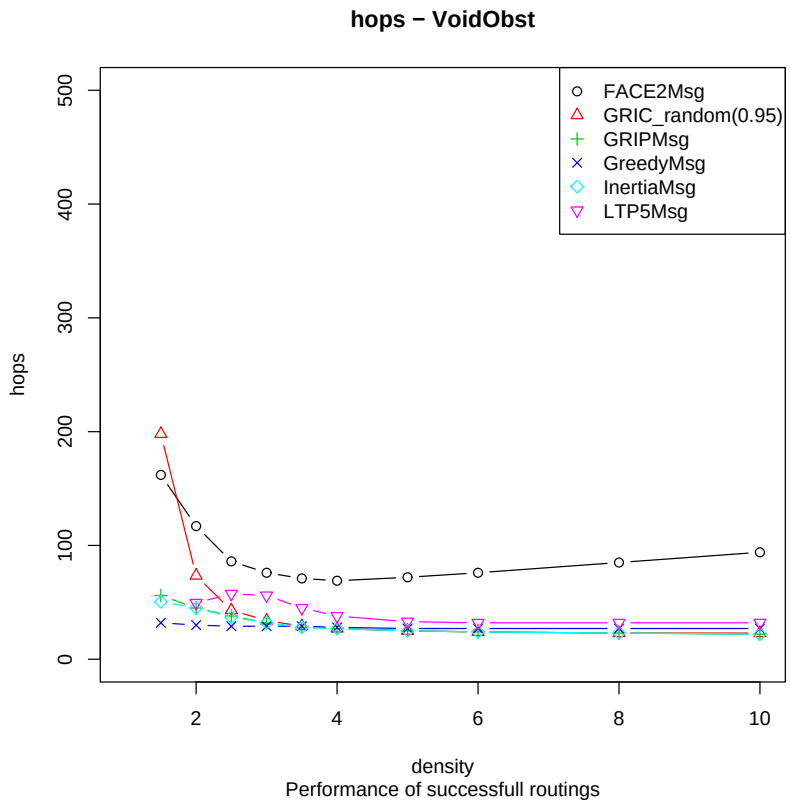


An example where GRIC succeeds

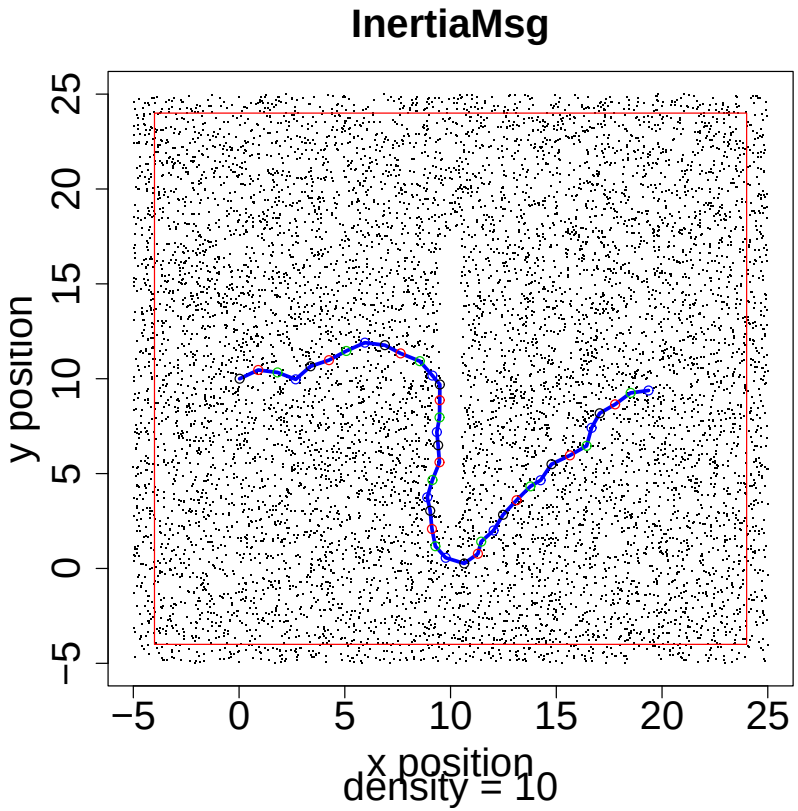


Statistical evaluation of the protocol:Void obstacle

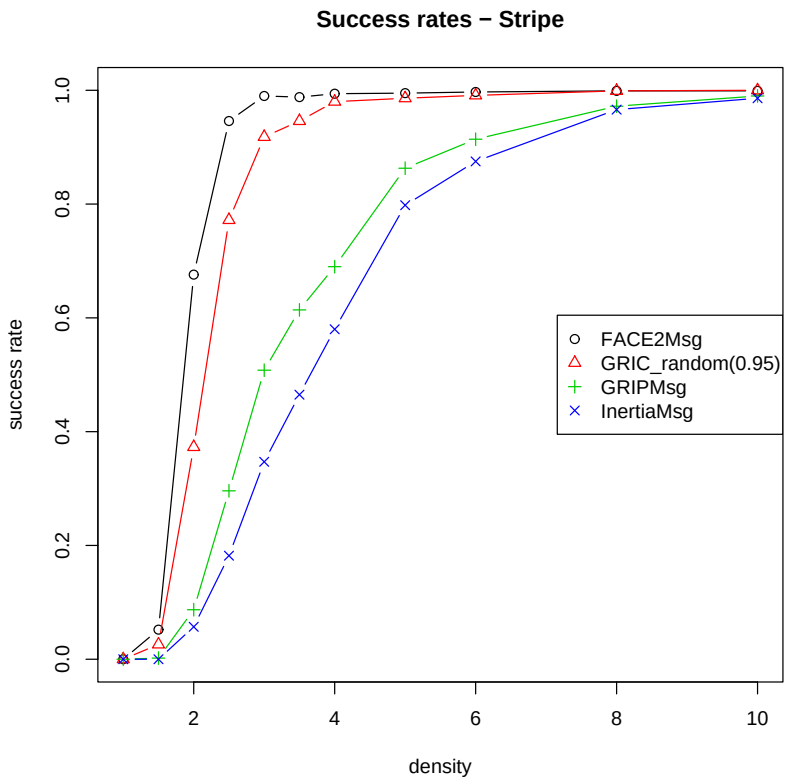


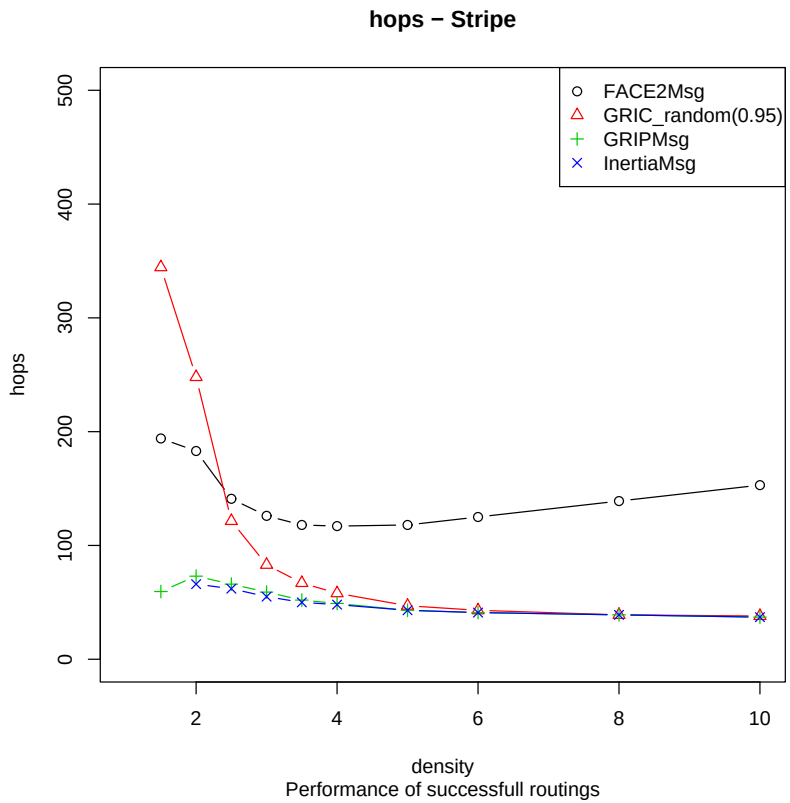


Statistical evaluation of the protocol:Stripe

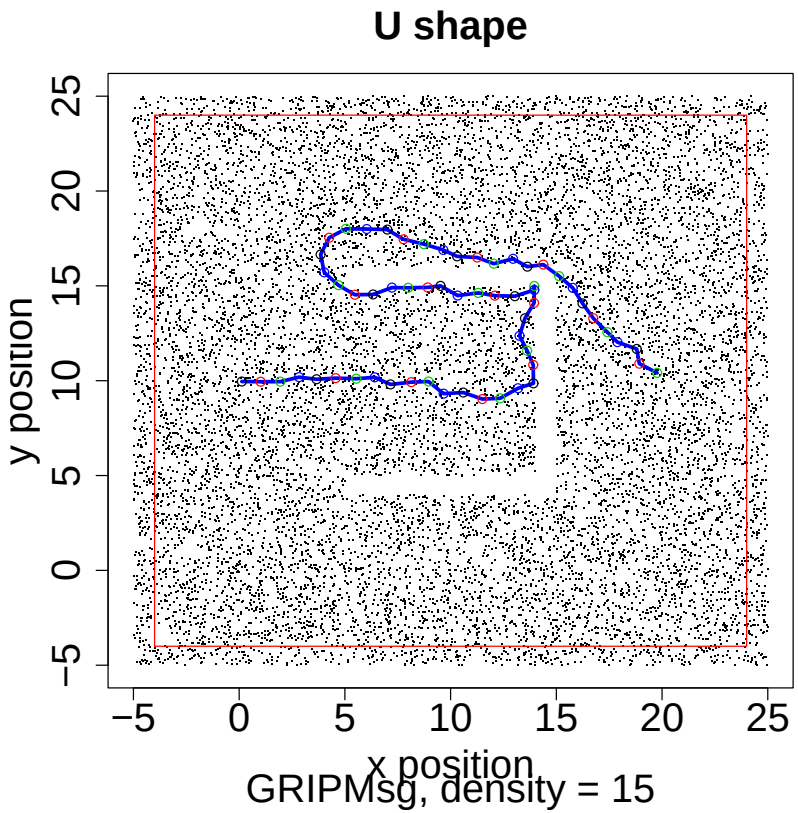


Statistical evaluation of the protocol:Stripe

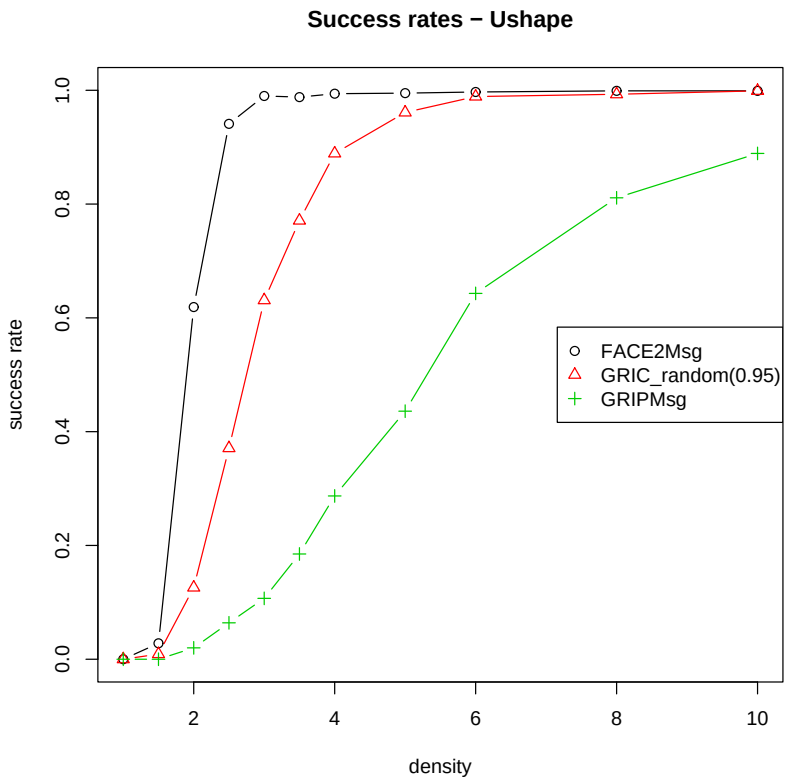


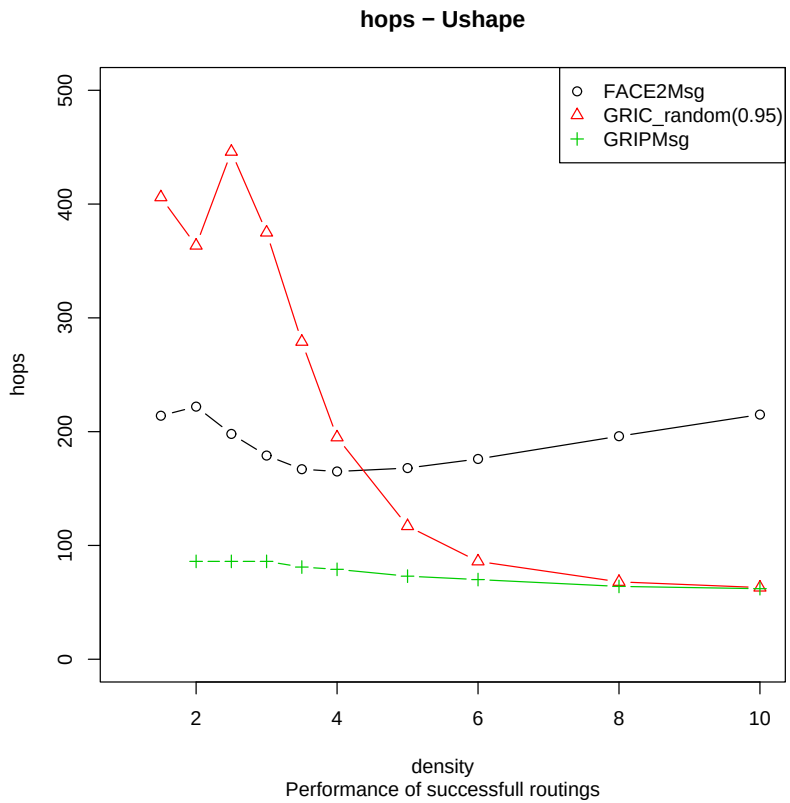


Statistical evaluation of the protocol:Ushape



Statistical evaluation of the protocol:Ushape





Conclusions

1. GRIC is lightweight (no extra topology maintenance).
2. GRIC has a high success rate:
   1. almost as good as FACE for routing holes.
   2. routes around hard obstacles.
   3. however, it will *not* get a message out of a maze!!!
3. GRIC is robust, because not relying on the UDG.

The trust based algorithm - Void avoidance

Perimeter routing

- Uses greedy routing
- Uses a planar graph traversal algorithm when greedy fails
- Turns back to greedy when the void is bypassed

Nodes' trust evaluation

Bayesian interference

- outcome observation (p, n) (p:positive, n:negative)
- trust evaluation
  - $t = \frac{p}{p+n}$
- decision making when $t > threshold$

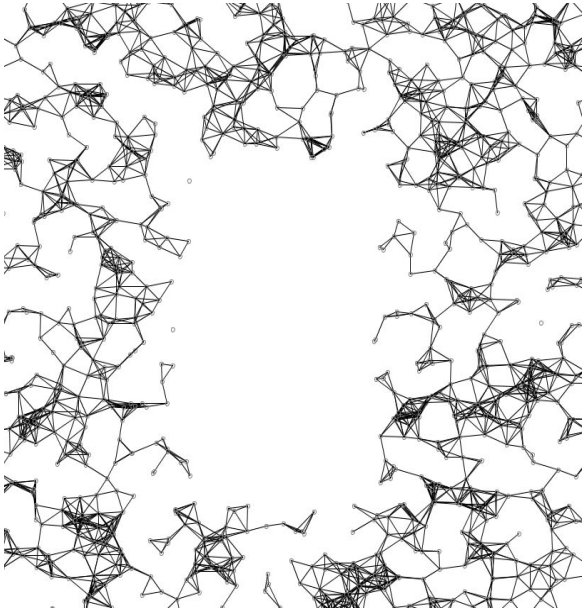
Interaction evaluation

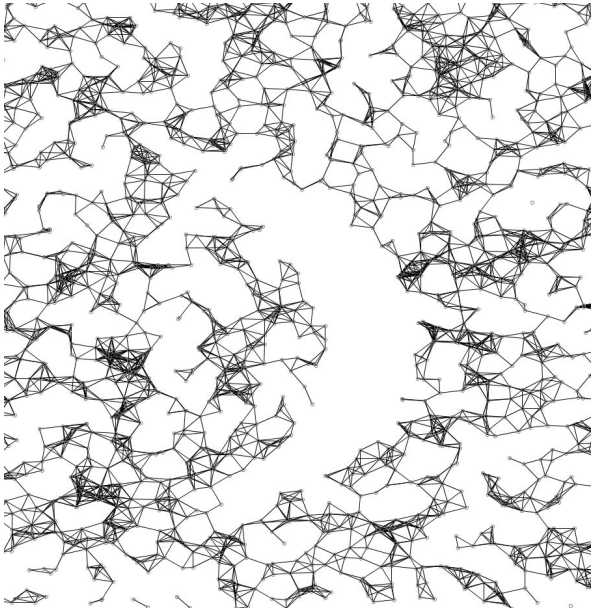
- greedy routing $->p++$
- perimeter routing $->n++$
- $t > threshold ->$ optimal path

Perimeter routing with interaction evaluation

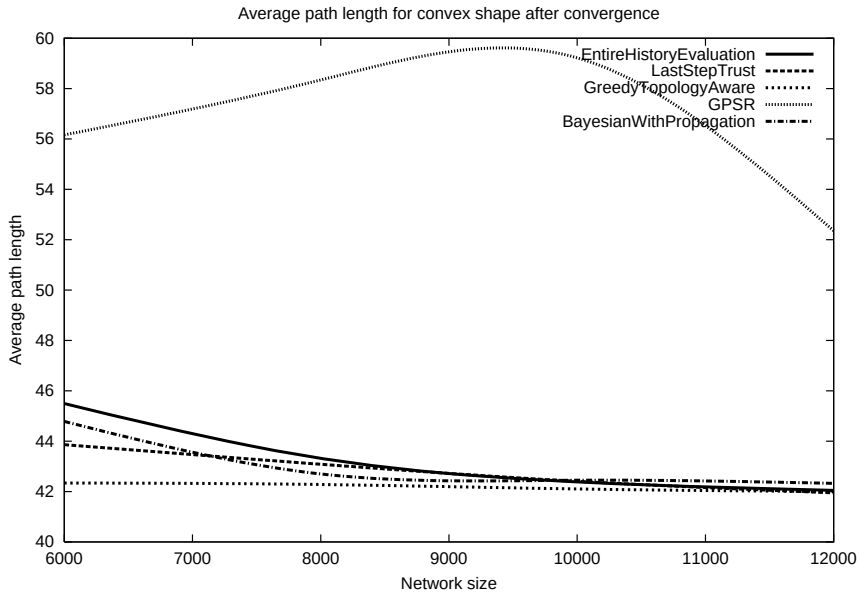
- Uses greedy routing on selected neighbors
  - Neighbors are filtered by optimal path
- Uses a planar graph traversal algorithm when greedy fails
- Turns back to greedy when the void is bypassed

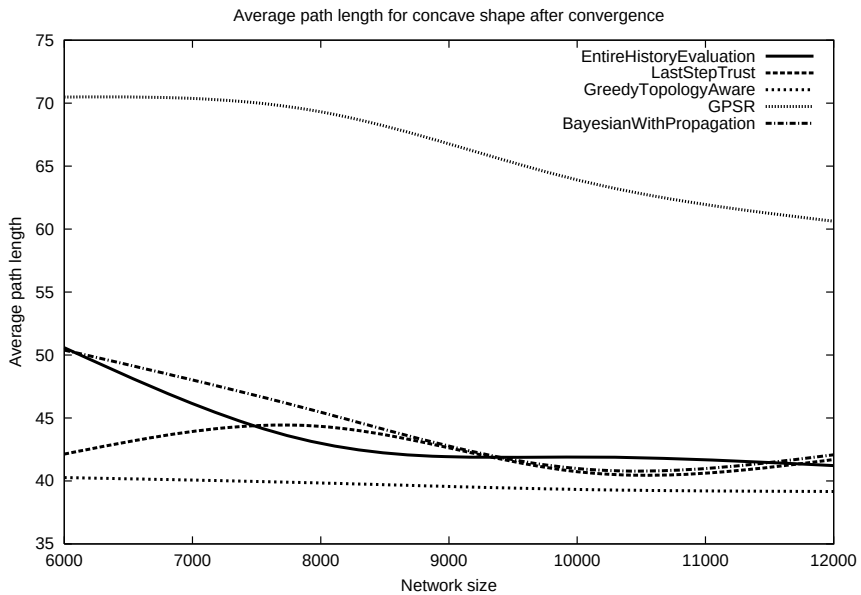
Object shapes





Path length





Comment: significant reduction in path length, which is near

Conclusions

Optimization

- Path length is close to optimal
- Little or no overhead

Possible extensions

- Mobile and multiple base station
- Mobile obstacles

Back to greedy

- We note that the likelihood of local maxima/dead-ends decreases with density.
- It has been shown (Guibas et al) that if the graph is dense enough that each interior node has a neighbor in every $2\pi/3$ angular sector, then greedy forwarding will always succeed.