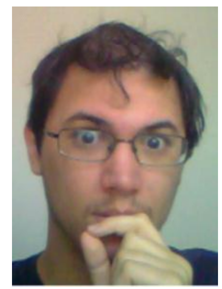
Incremental ARA*: An Anytime Incremental Search Algorithm For Moving Target Search





Xiaoxun Sun William Yeoh
Tansel Uras Sven Koenig

University of Southern California
Singapore Management University

Moving Target Search

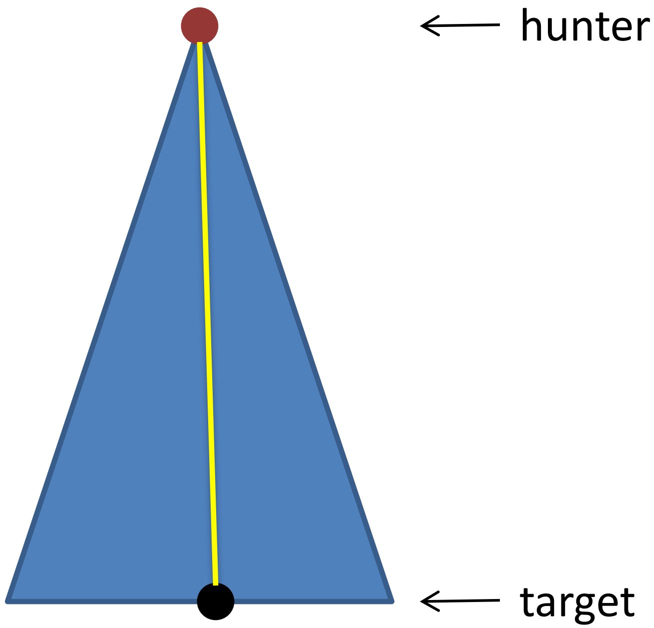
- Assumptions
  - The hunter knows the terrain.
  - The hunter knows its own cell.
  - The hunter knows the cell of the target.

Moving Target Search

- Offline search
  - e.g. minimax search (Reverse Minimax A*)
- Online search
  - e.g. repeated deterministic searches
    - The hunter finds a short path to the target and moves along the path.
    - Whenever the target moves off the path, the hunter repeats the process.

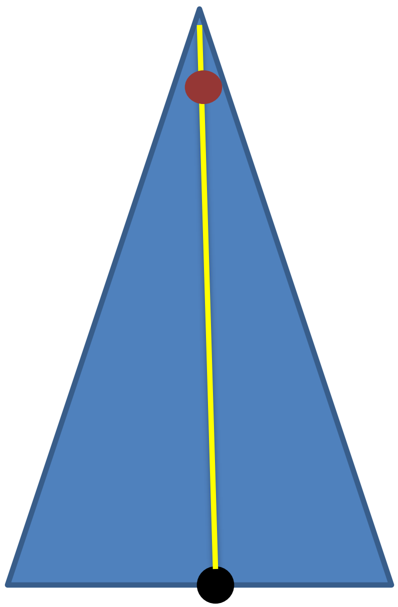
A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A* (with consistent h-values).



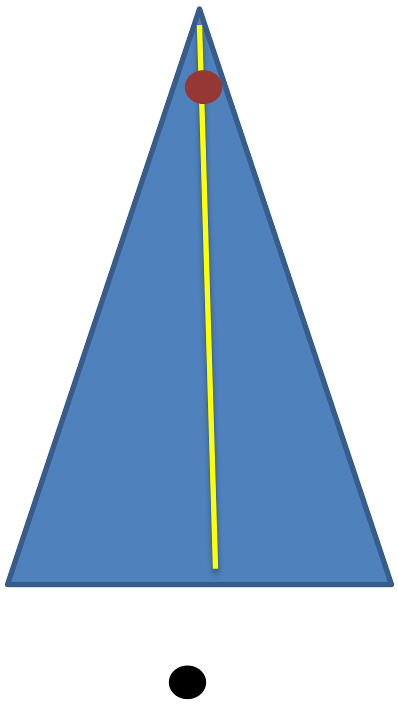
A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A*.



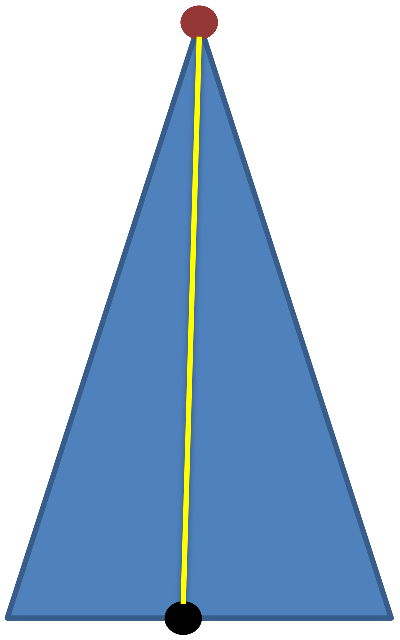
A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A*.



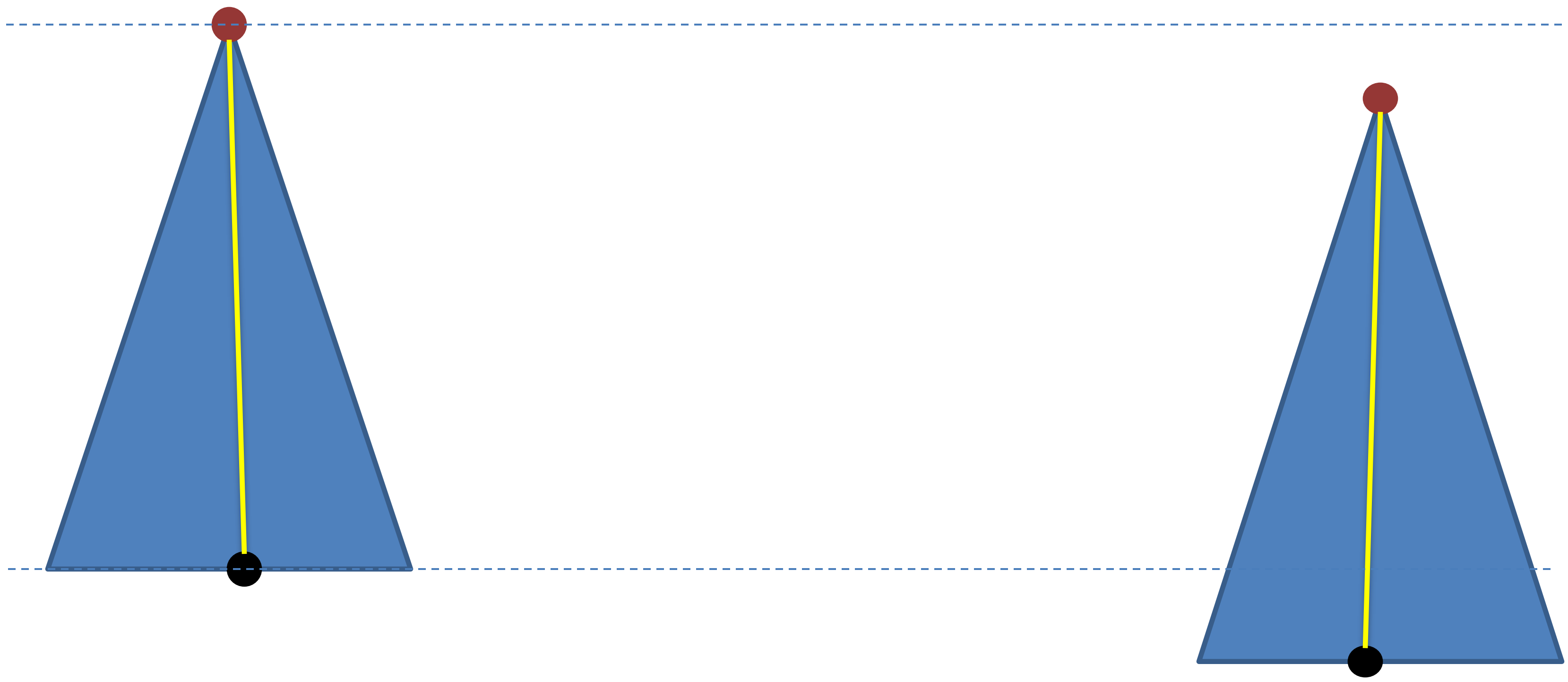
A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A*.

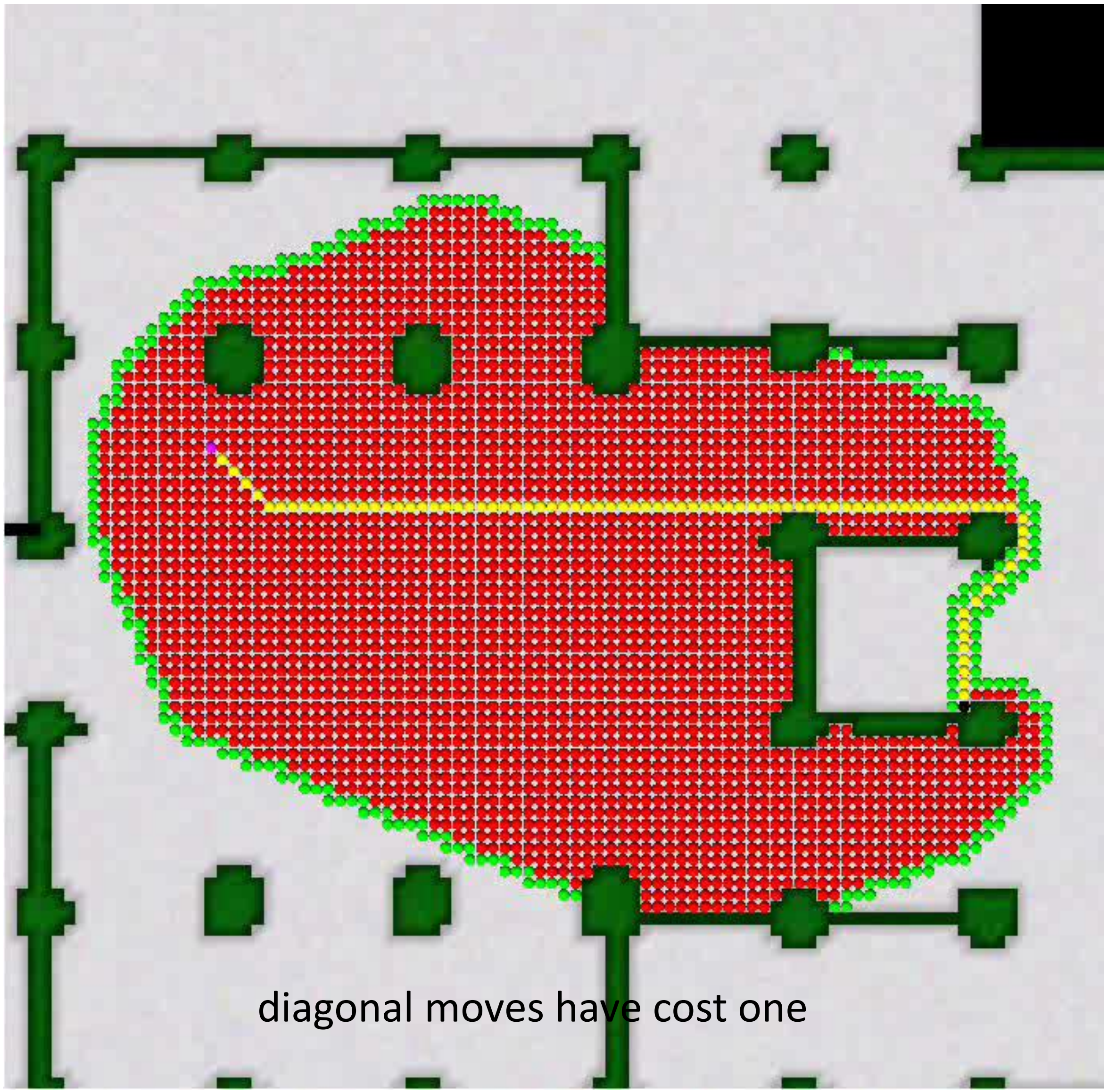


A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A*.



A* [Hart, Nilsson, Raphael, 1968]



diagonal moves have cost one

A* [Hart, Nilsson, Raphael, 1968]

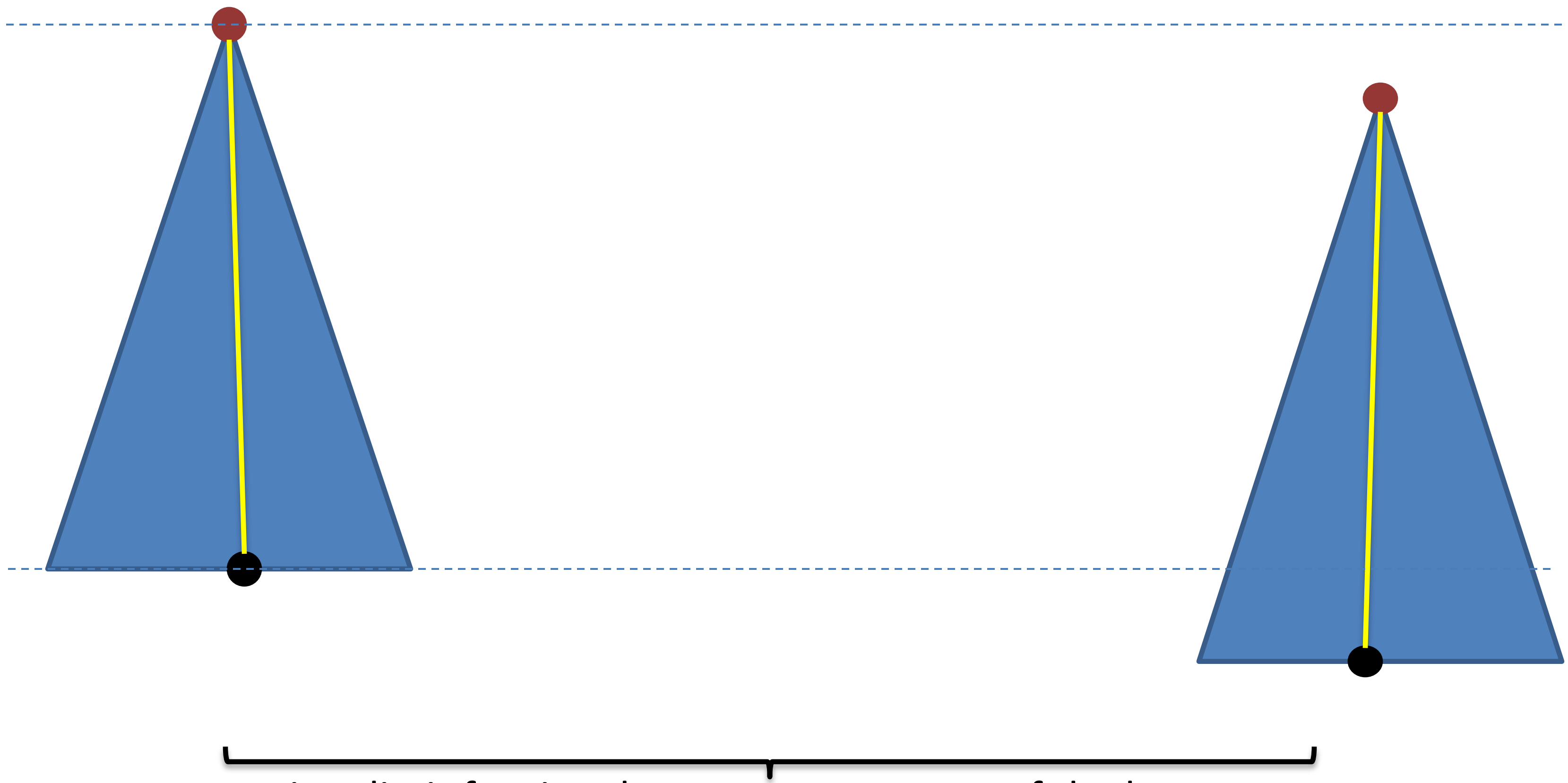
A* [Hart, Nilsson, Raphael, 1968]

$$\underbrace{\text{1 move}}$$

Small (but soft) time limit for time between two moves of the hunter
1-3 ms per search for Bioware [Bulitko et al, 2007]

A* [Hart, Nilsson, Raphael, 1968]

- The hunter uses A*.



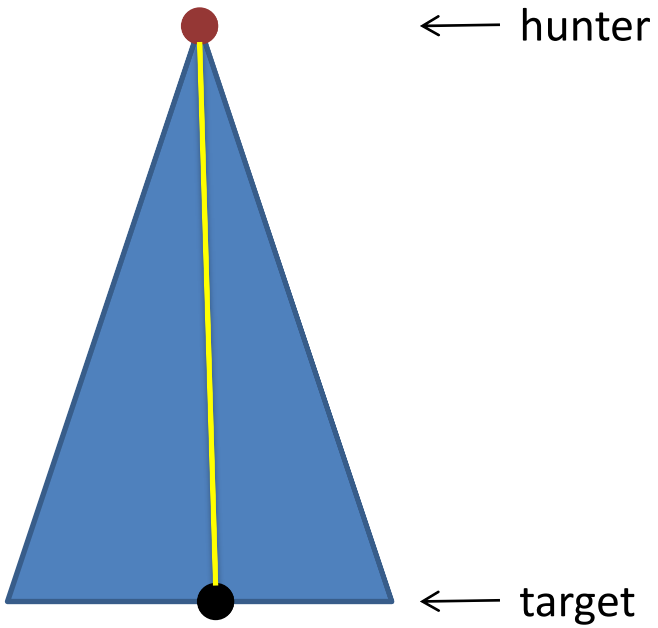
time limit for time between two moves of the hunter
1-3 ms per search for Bioware [Bulitko et al, 2007]

FRA* [Sun, Yeoh, Koenig, 2009]

- Idea 1:
  - Reduce the runtime of the A* search by using incremental A* search

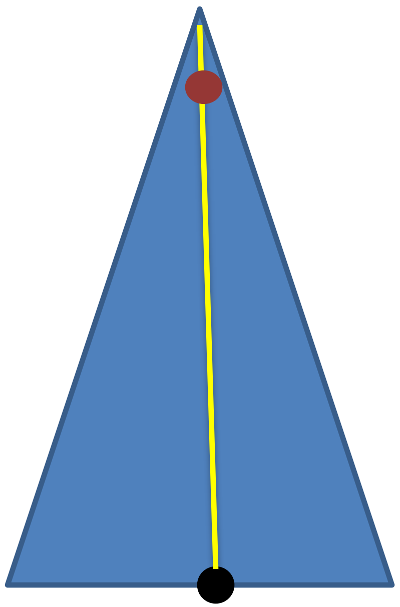
FRA*

- The hunter uses A*.



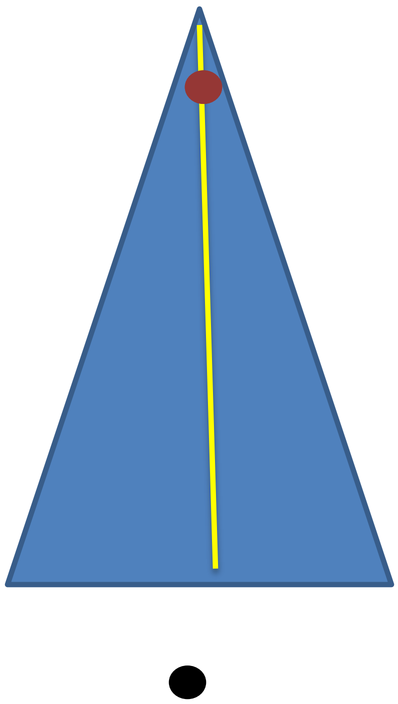
FRA*

- The hunter uses A*.



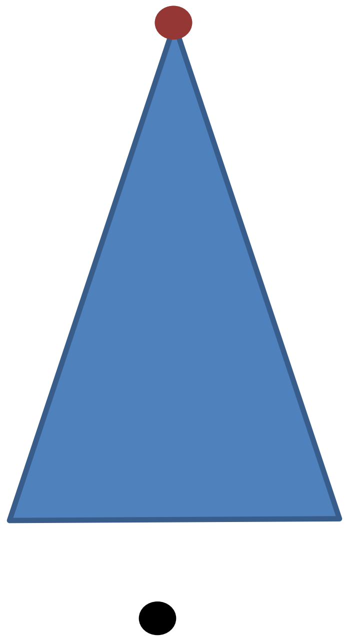
FRA*

- The hunter uses A*.



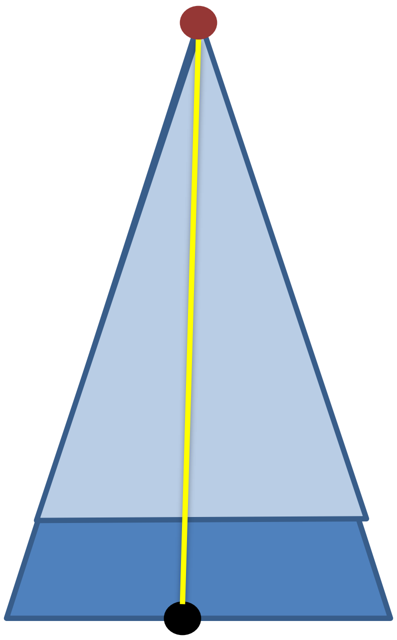
FRA*

- The hunter uses A*.

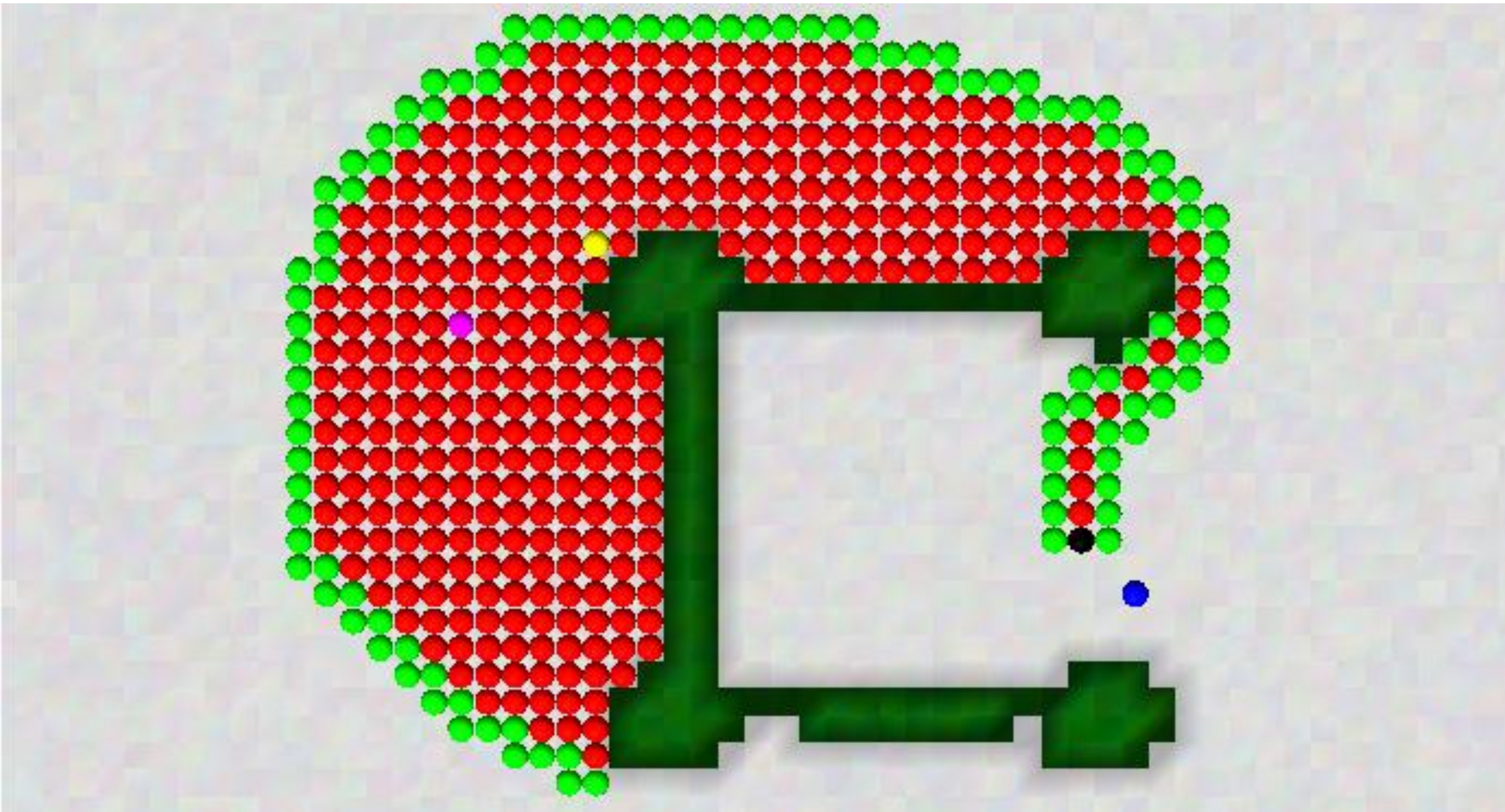


FRA*

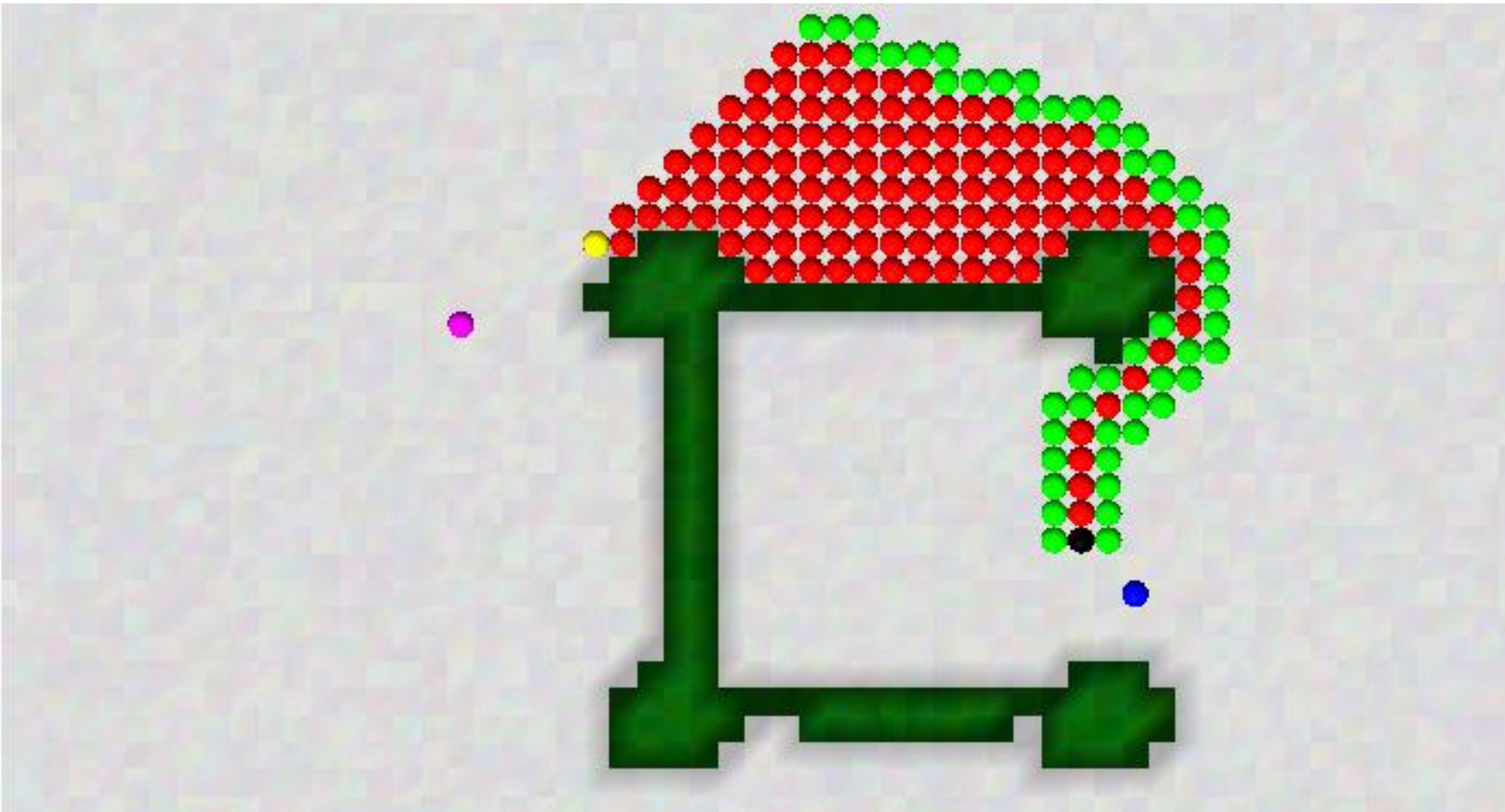
- The hunter uses A*.



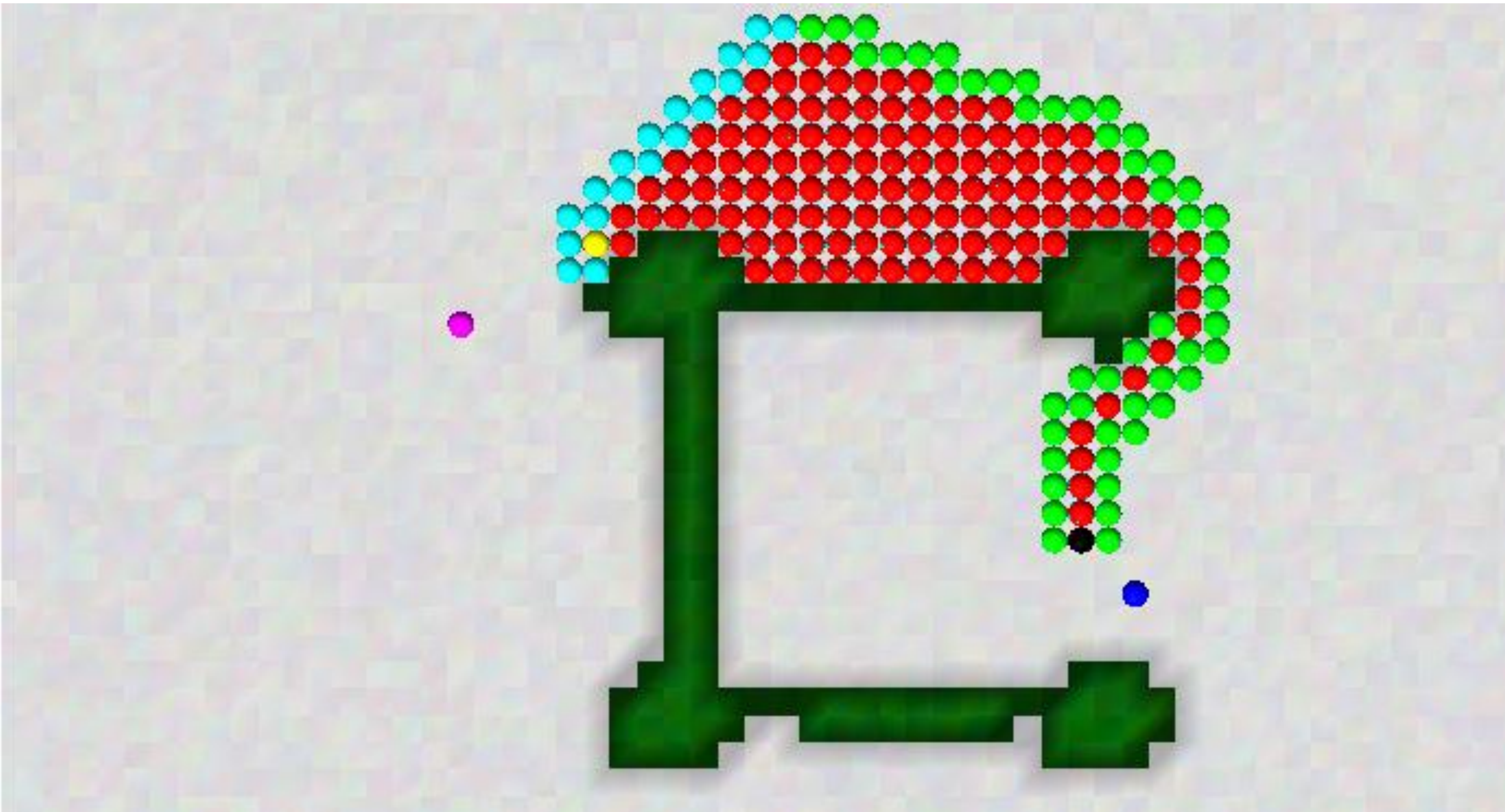
FRA*



FRA*

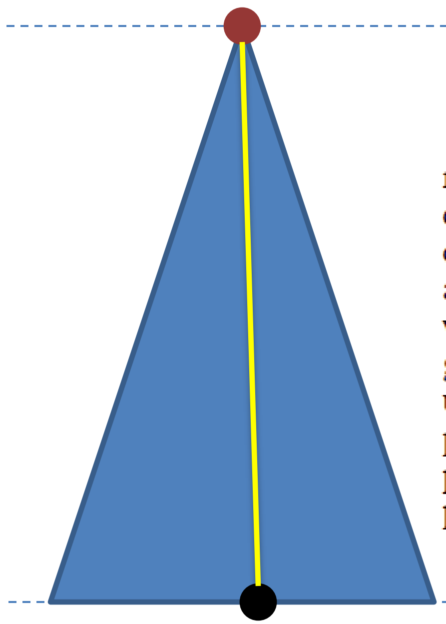


FRA*



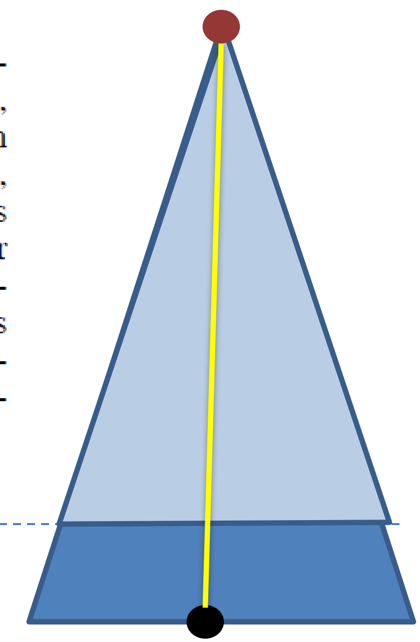
FRA*

- The hunter uses incremental A*.



DISCLAIMER

It is important to realize that experimental results, such as the runtimes of the search algorithms, depend on a variety of factors, including implementation details (such as the data structures, tie-breaking strategies, and coding tricks used) and experimental setups (such as whether the gridworlds are four-neighbor or eight-neighbor gridworlds). We do not know of any better method for evaluating search algorithms than to implement them as best as possible, publish their runtimes, and let other researchers experiment with their own and thus potentially different implementations and experimental setups.



time limit for time between two moves of the hunter

WA* [Pohl, 1970]

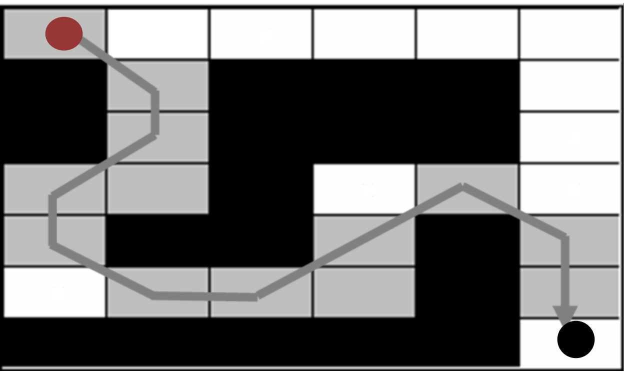
- Idea 2:
  - Reduce the runtime of the A* search with weighted A* search

WA*

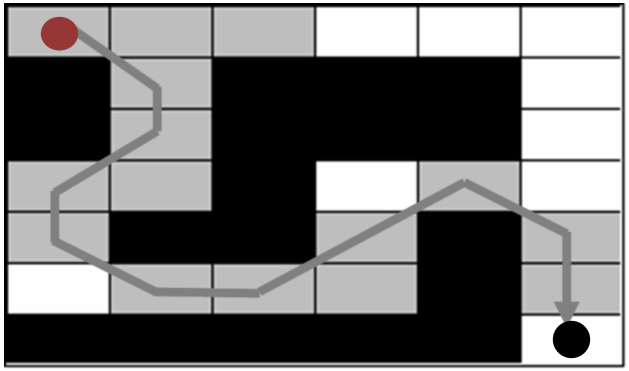
- The hunter uses weighted A*.

  The smaller the weight w, the slower the search but the shorter the path.
  Weighted A* with weight one is identical to A*.

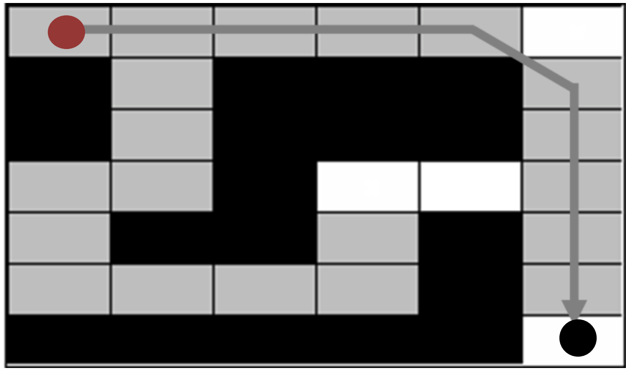
$$f(s) = g(s) + w\ h(s)$$



w=2.5
13 expansions
11 movements



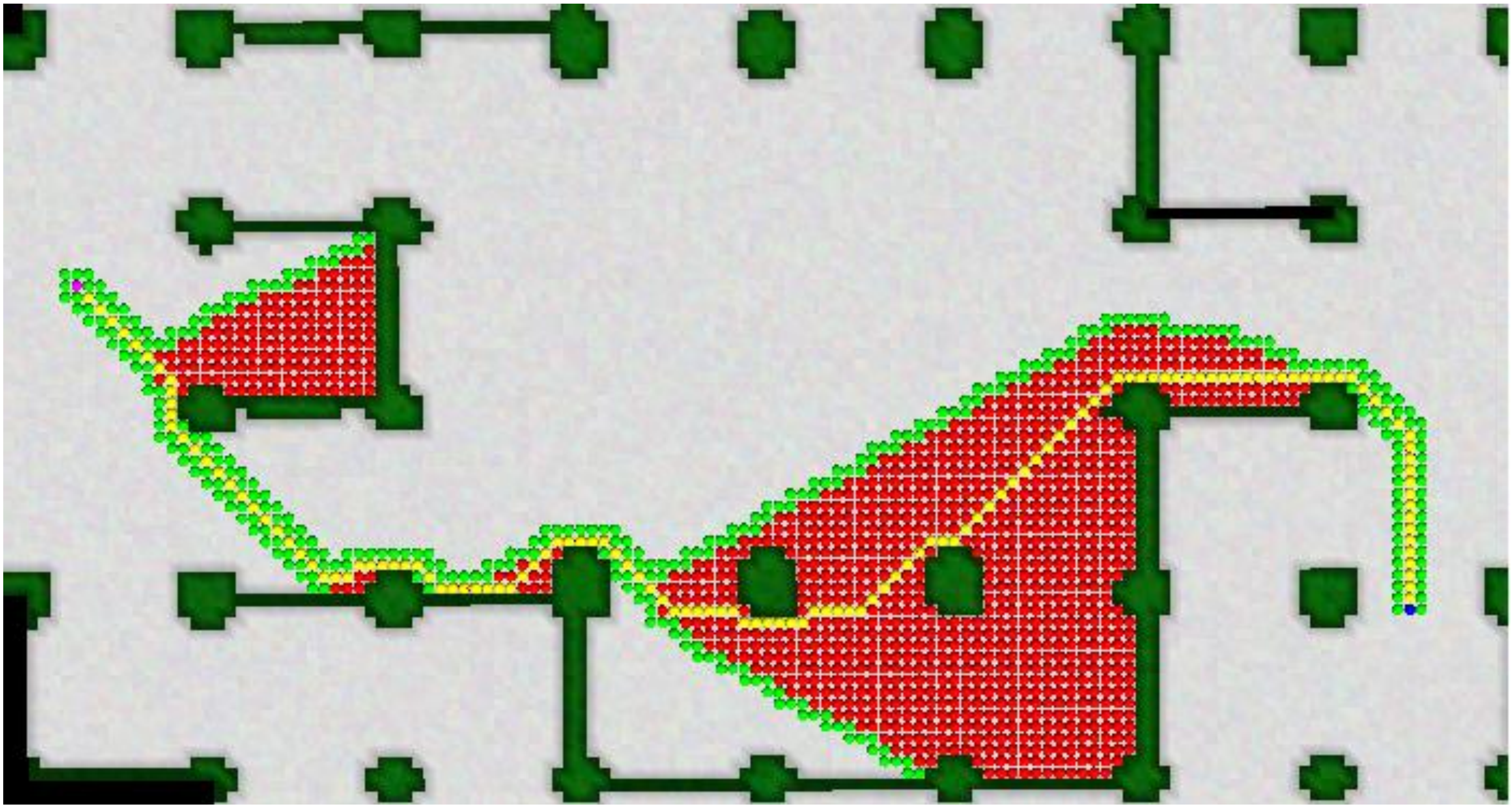
w=1.5
15 expansions
11 movements



w=1.0
20 expansions
10 movements

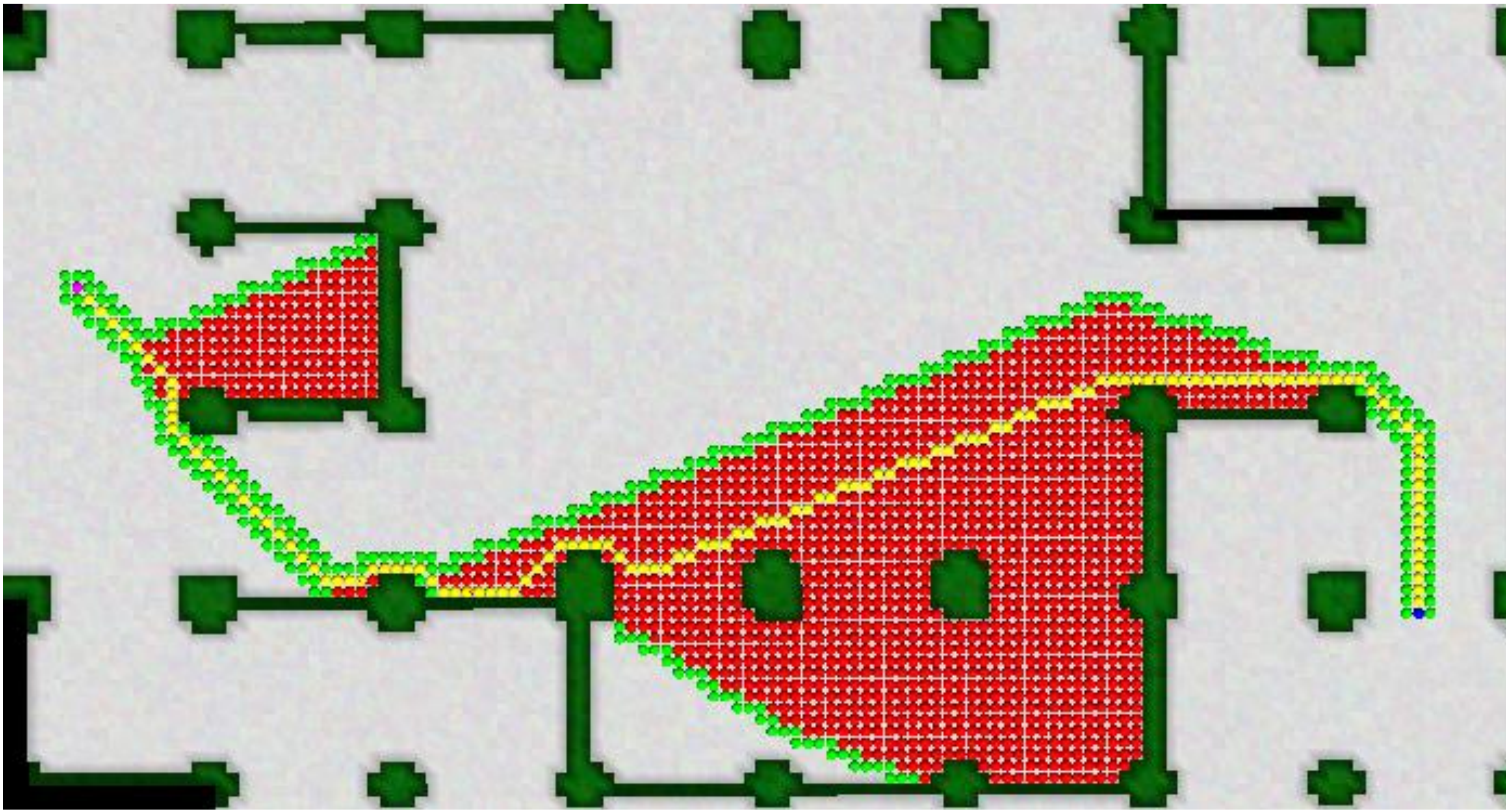
Courtesy of Maxim Likhachev

WA*



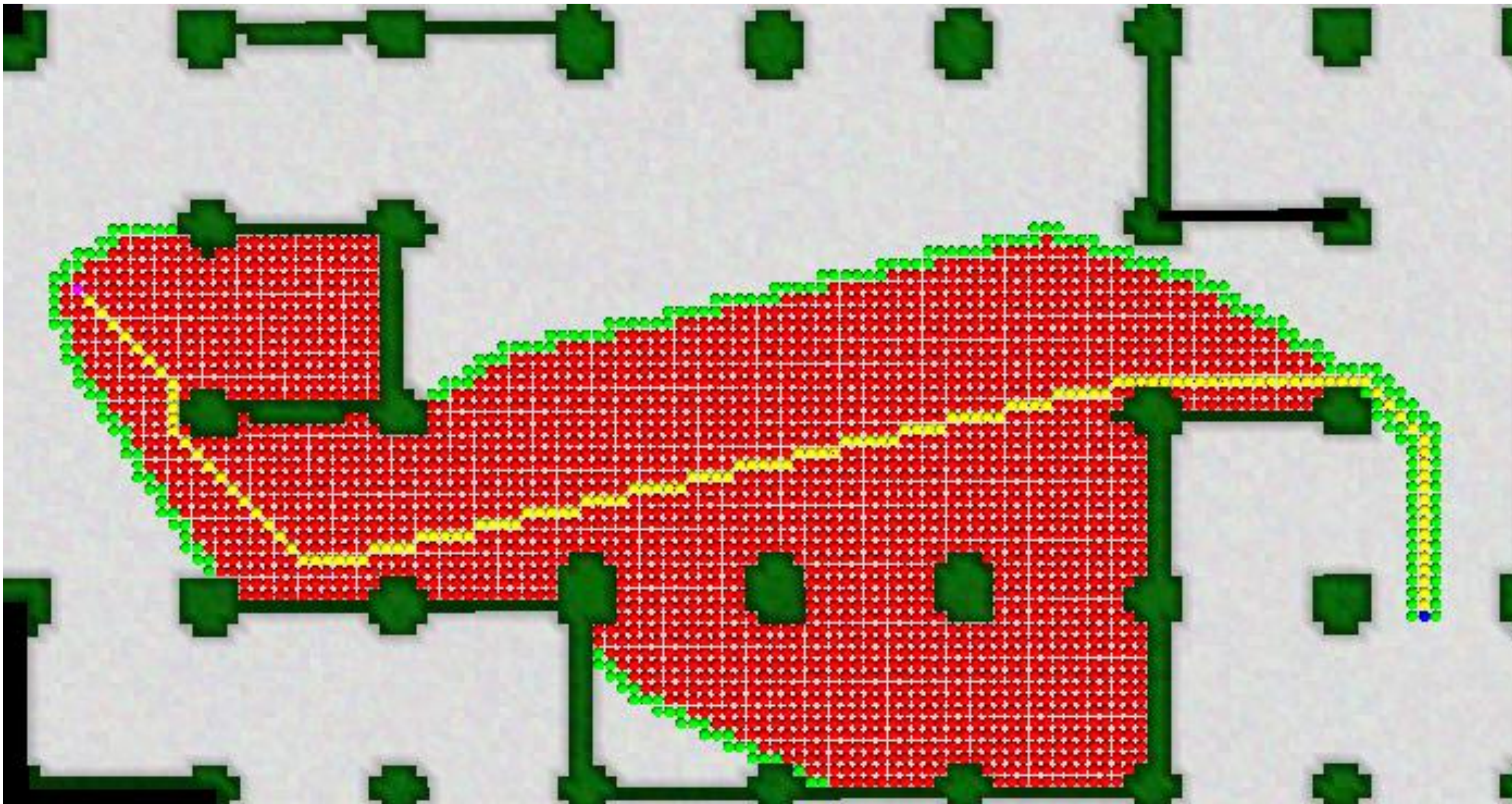
w=1.6

WA*



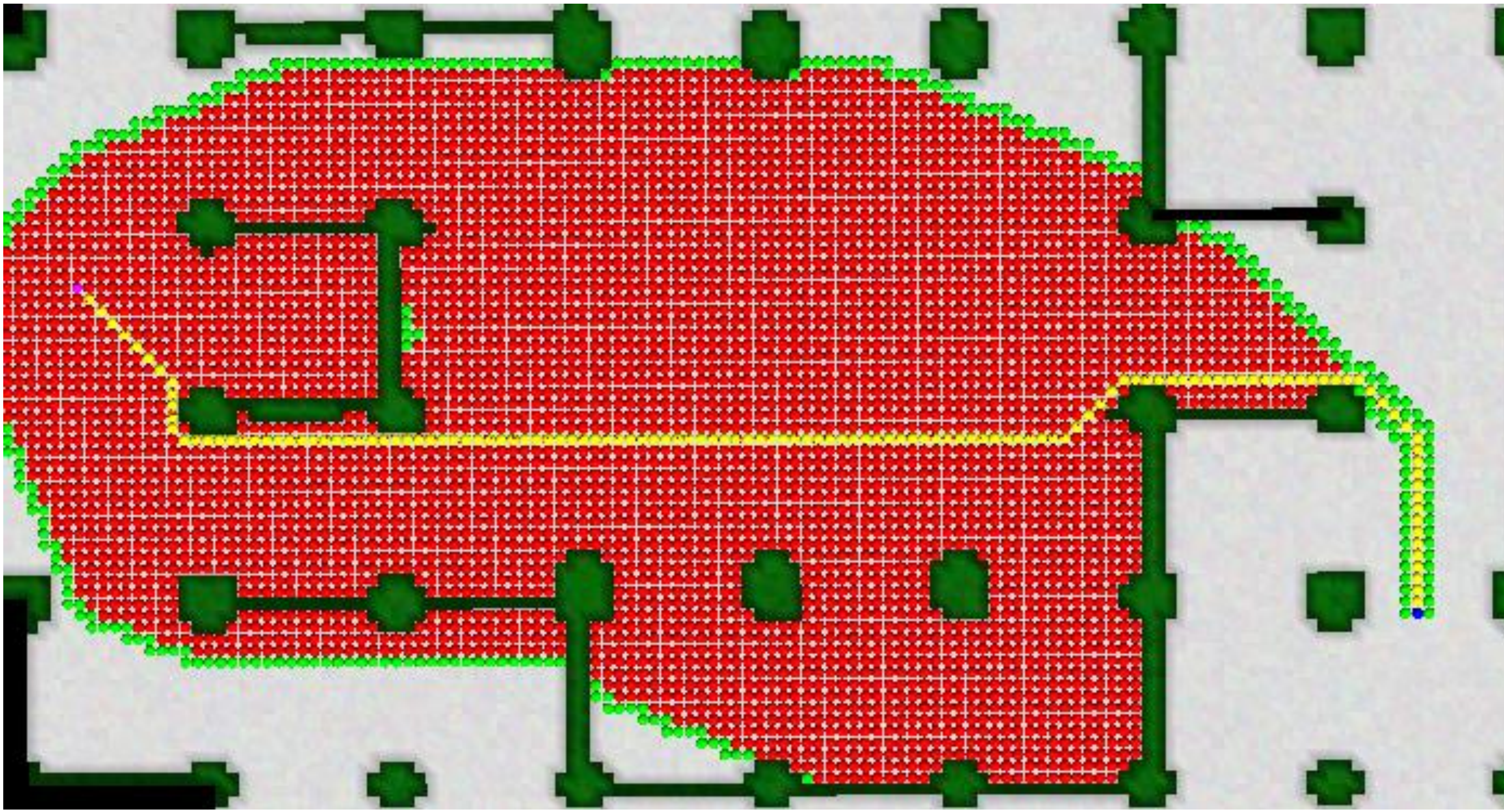
w=1.4

WA*



w=1.2

WA*

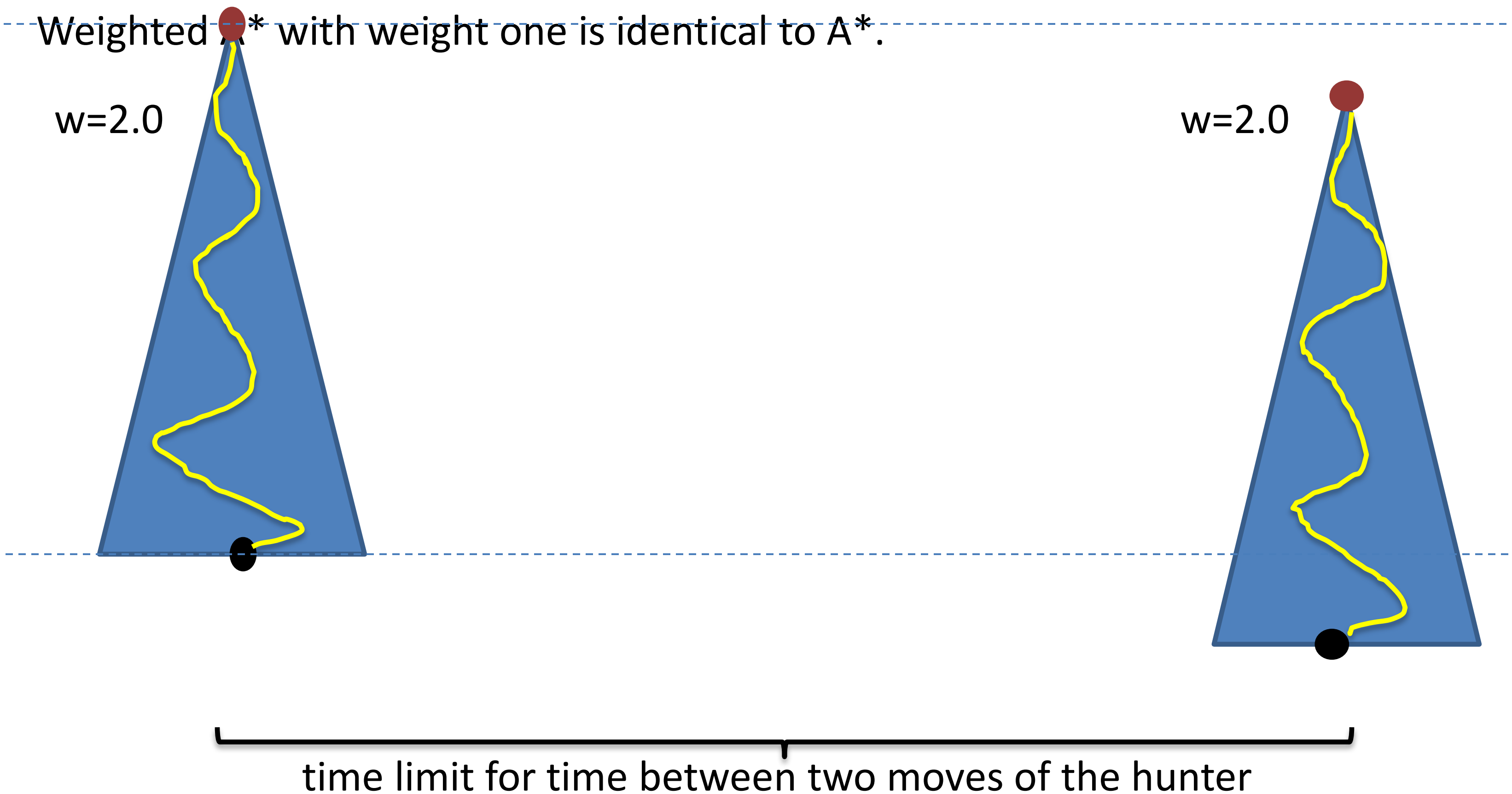


w=1.0

WA*

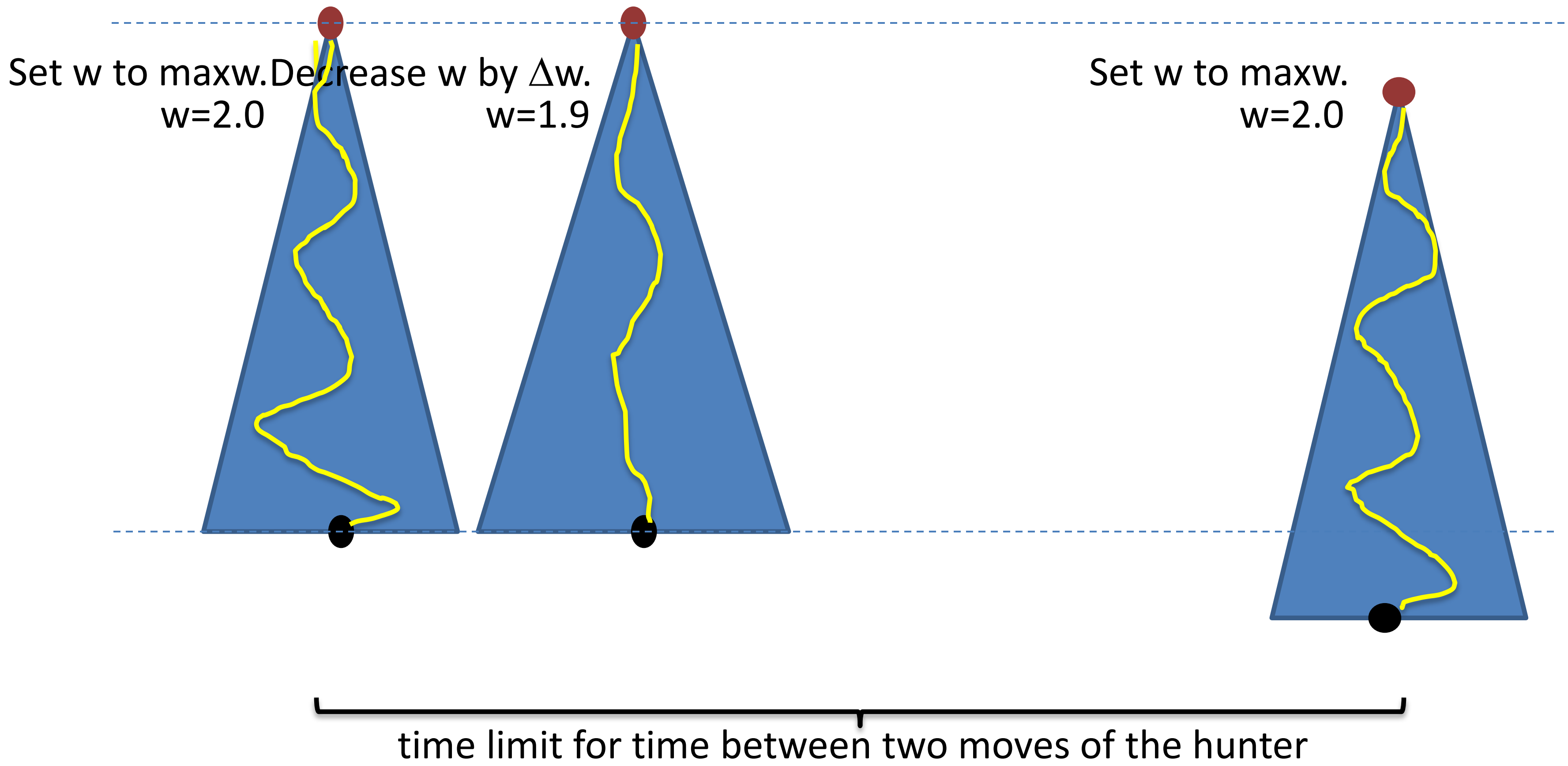
- The hunter uses weighted A*.

The smaller the weight w, the slower the search but the shorter the path. Weighted A* with weight one is identical to A*.



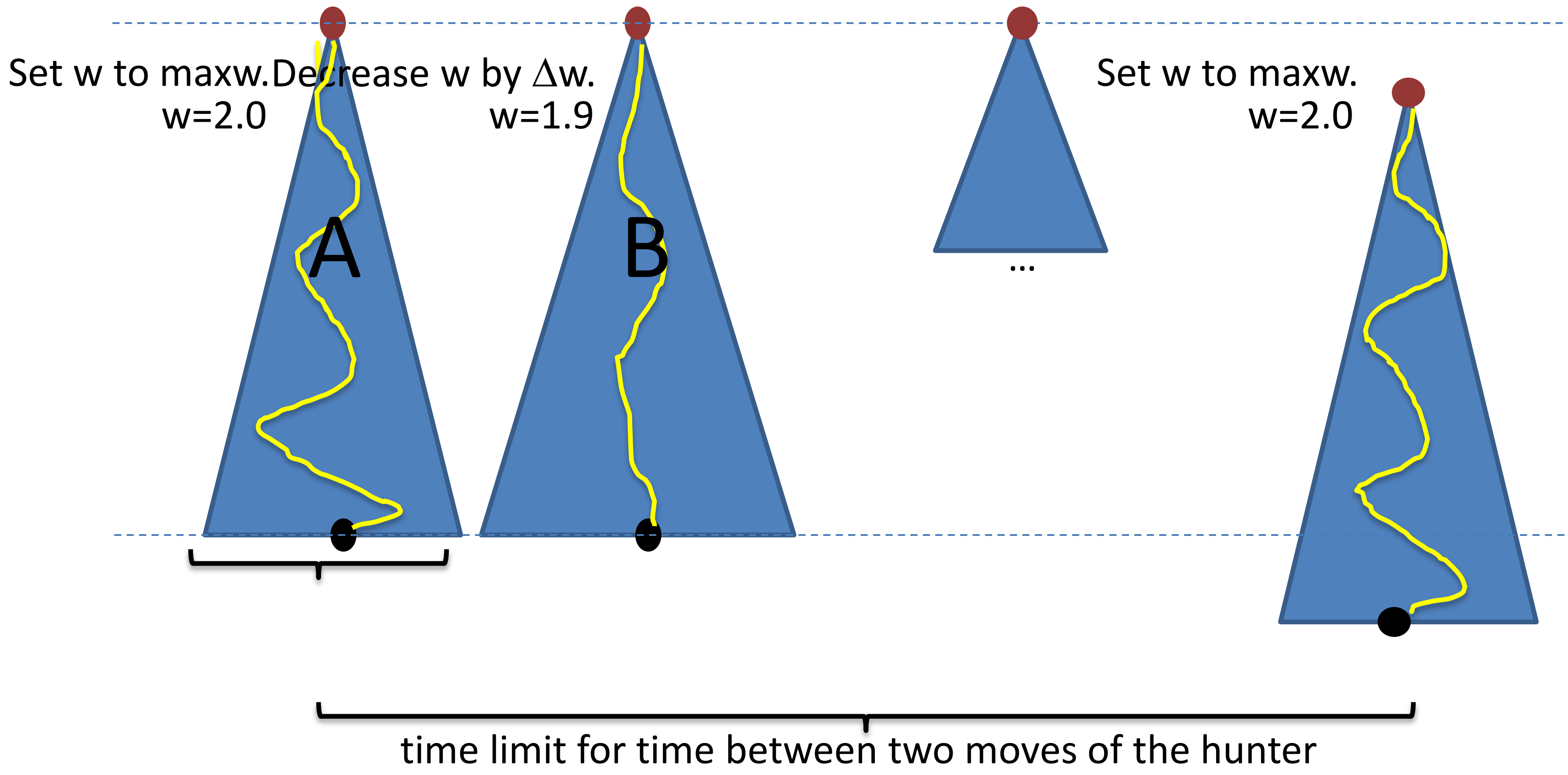
Repeated WA*

- The hunter uses weighted A* repeatedly,
  where the weight decreases over time until it is one.



Repeated WA*

- The hunter uses weighted A* repeatedly,
  where the weight decreases over time until it is one.

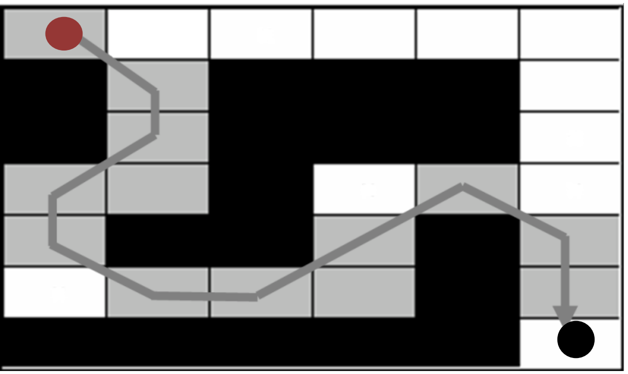


ARA* [Likhachev, Gordon, Thrun, 2003]

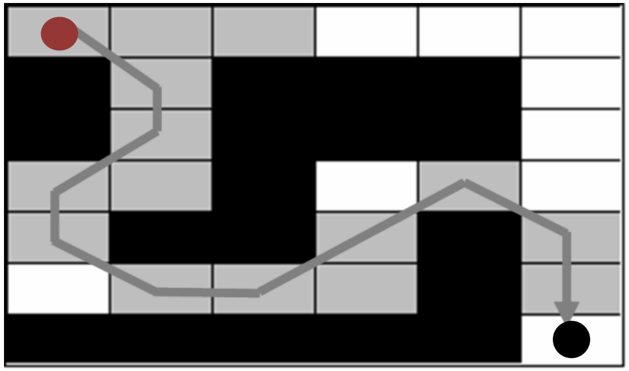
- The hunter uses weighted A*.

  The smaller the weight w, the slower the search but the shorter the path.
  Weighted A* with weight one is identical to A*.

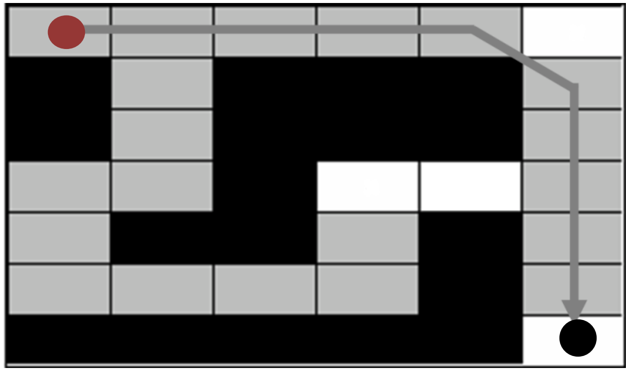
$$f(s) = g(s) + w\,h(s)$$



w=2.5
13 expansions
11 movements



w=1.5
15 expansions
11 movements



w=1.0
20 expansions
10 movements

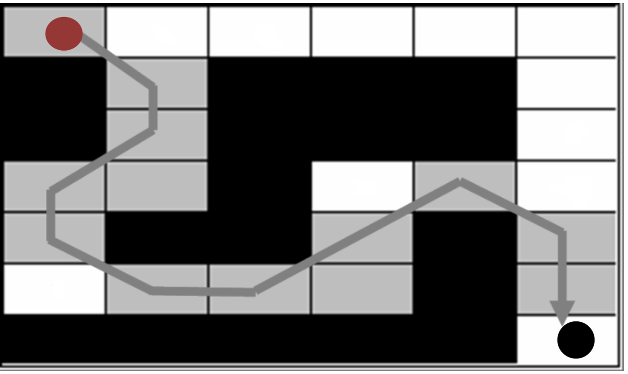
Courtesy of Maxim Likhachev

ARA*

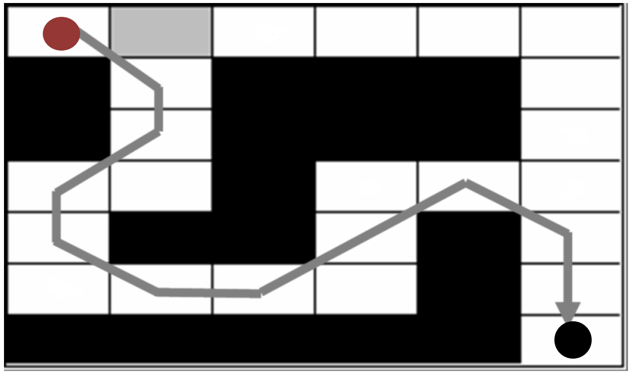
- The hunter uses weighted A*.

  The smaller the weight w, the slower the search but the shorter the path.
  Weighted A* with weight one is identical to A*.

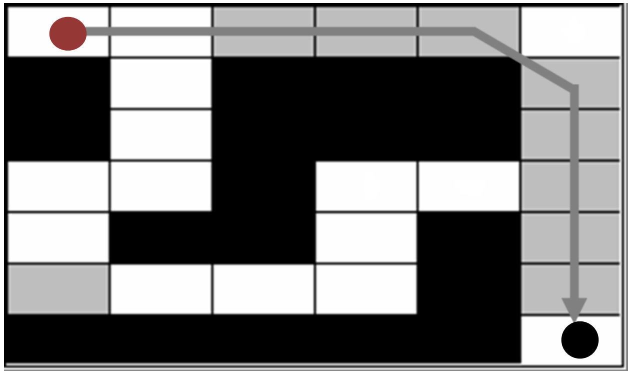
$$f(s) = g(s) + w\,h(s)$$



w=2.5
13 expansions
11 movements



w=1.5
1 expansion
11 movements

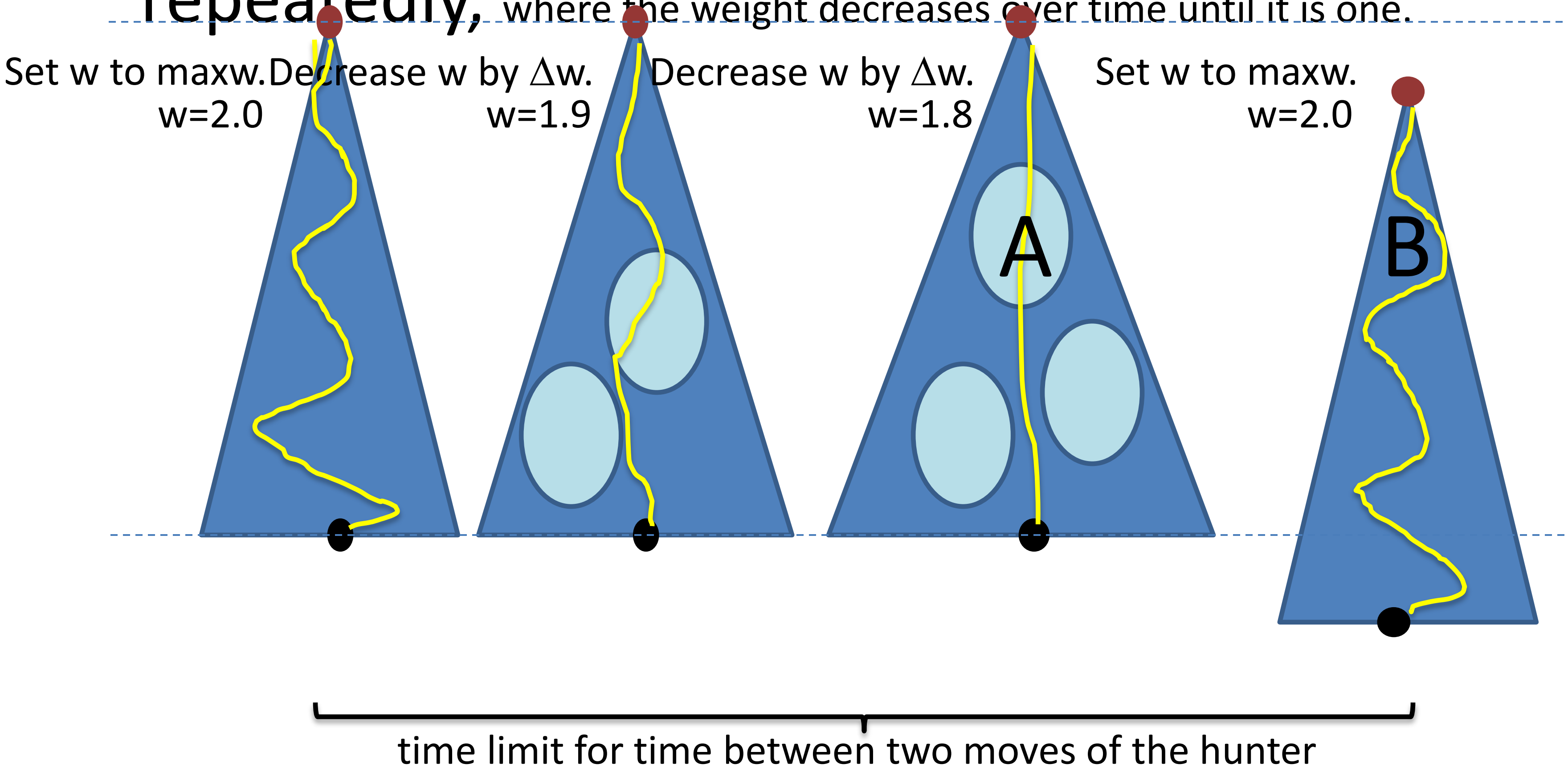


w=1.0
9 expansions
10 movements

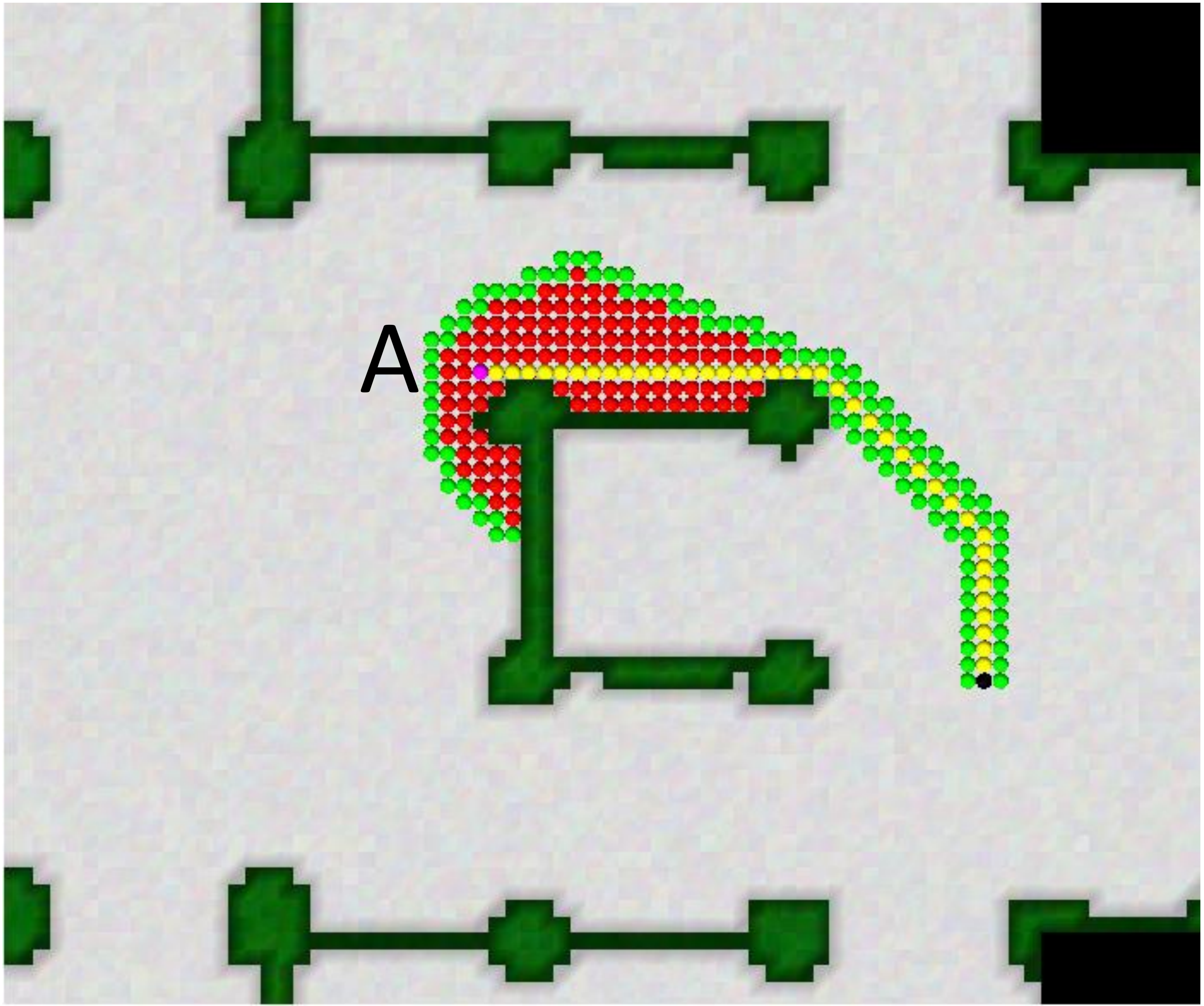
Courtesy of Maxim Likhachev

ARA*

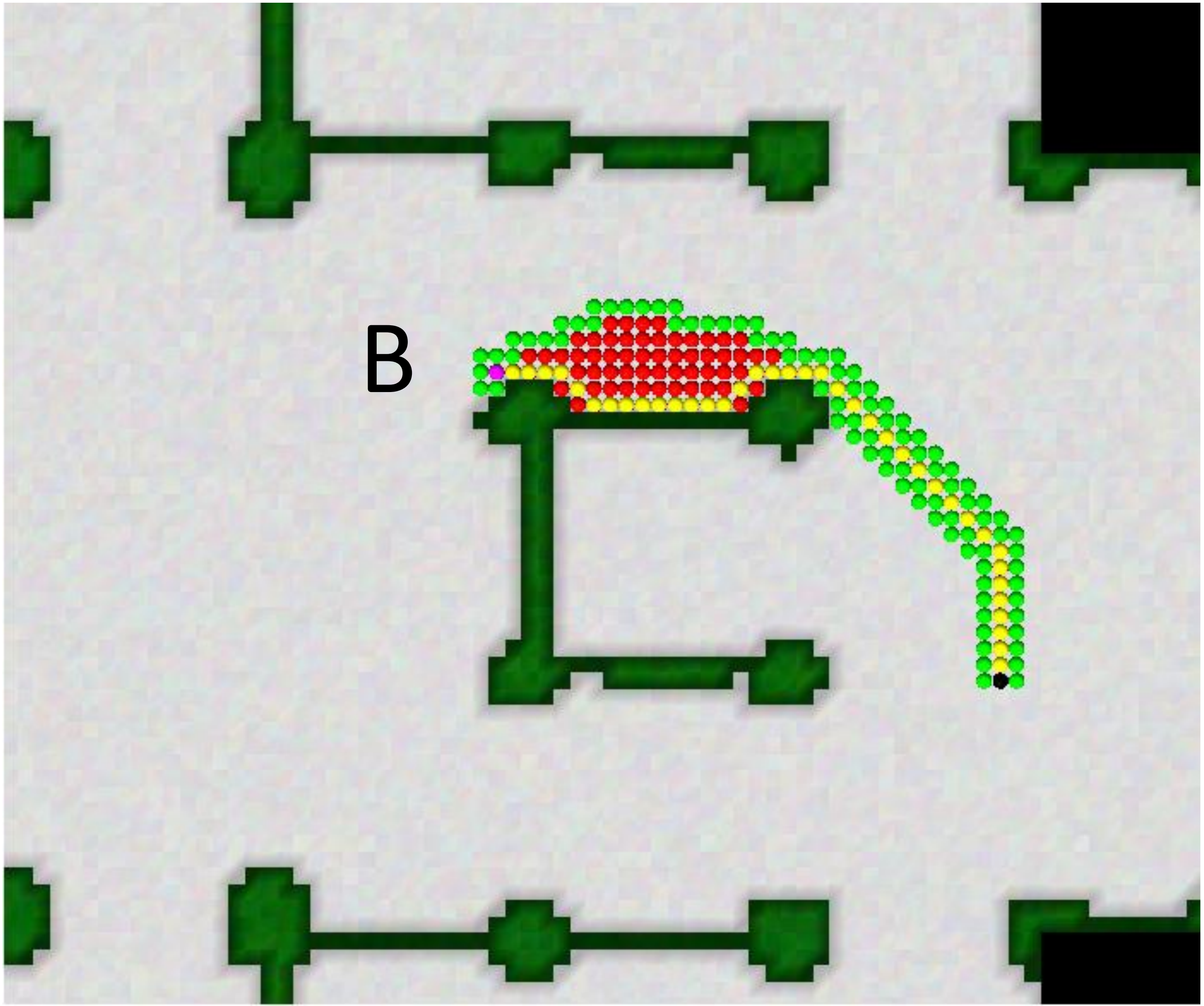
- The hunter uses incremental weighted A* repeatedly, where the weight decreases over time until it is one.



Incremental ARA* = FRA* + ARA*

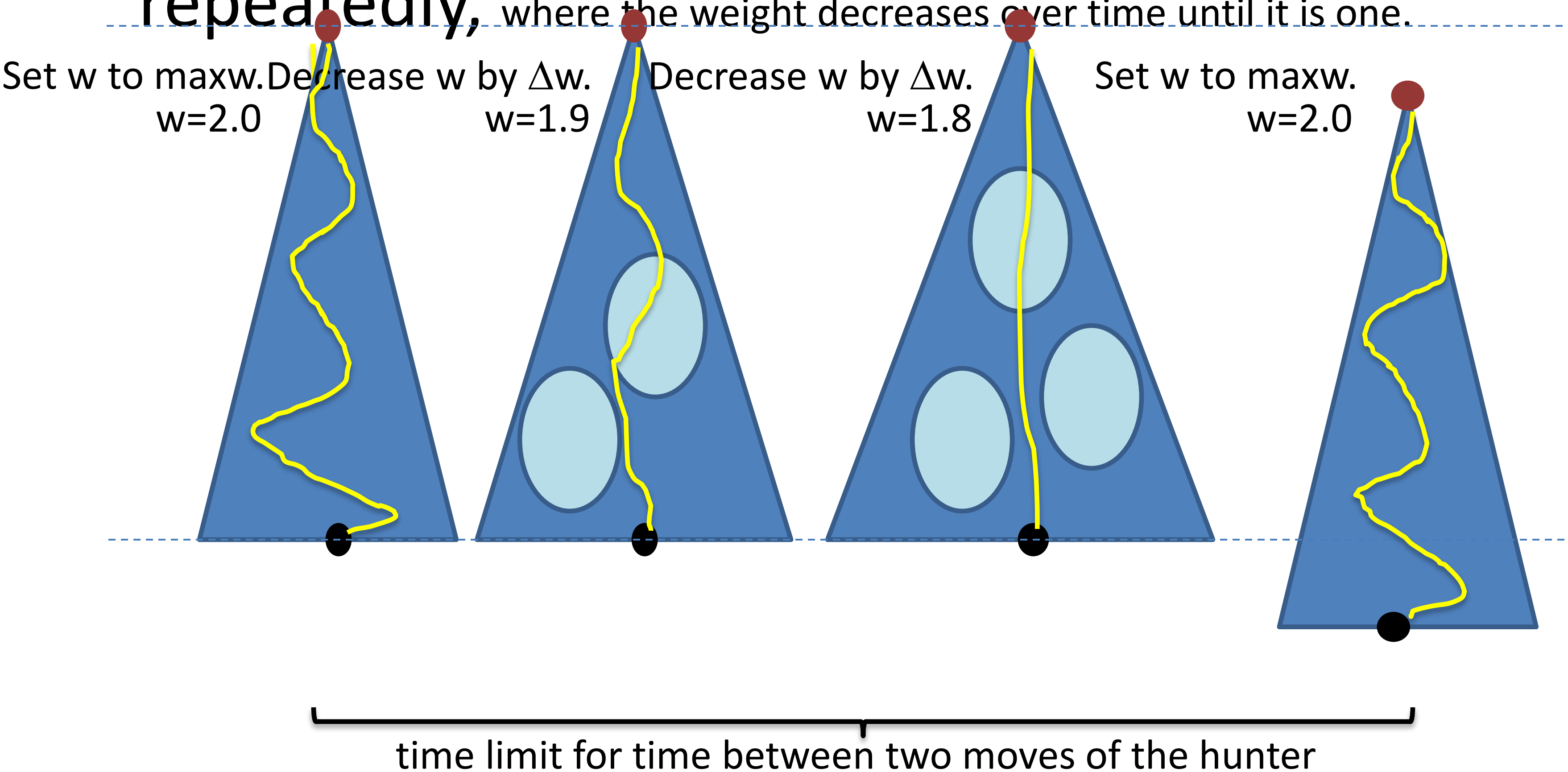


Incremental ARA* = FRA* + ARA*



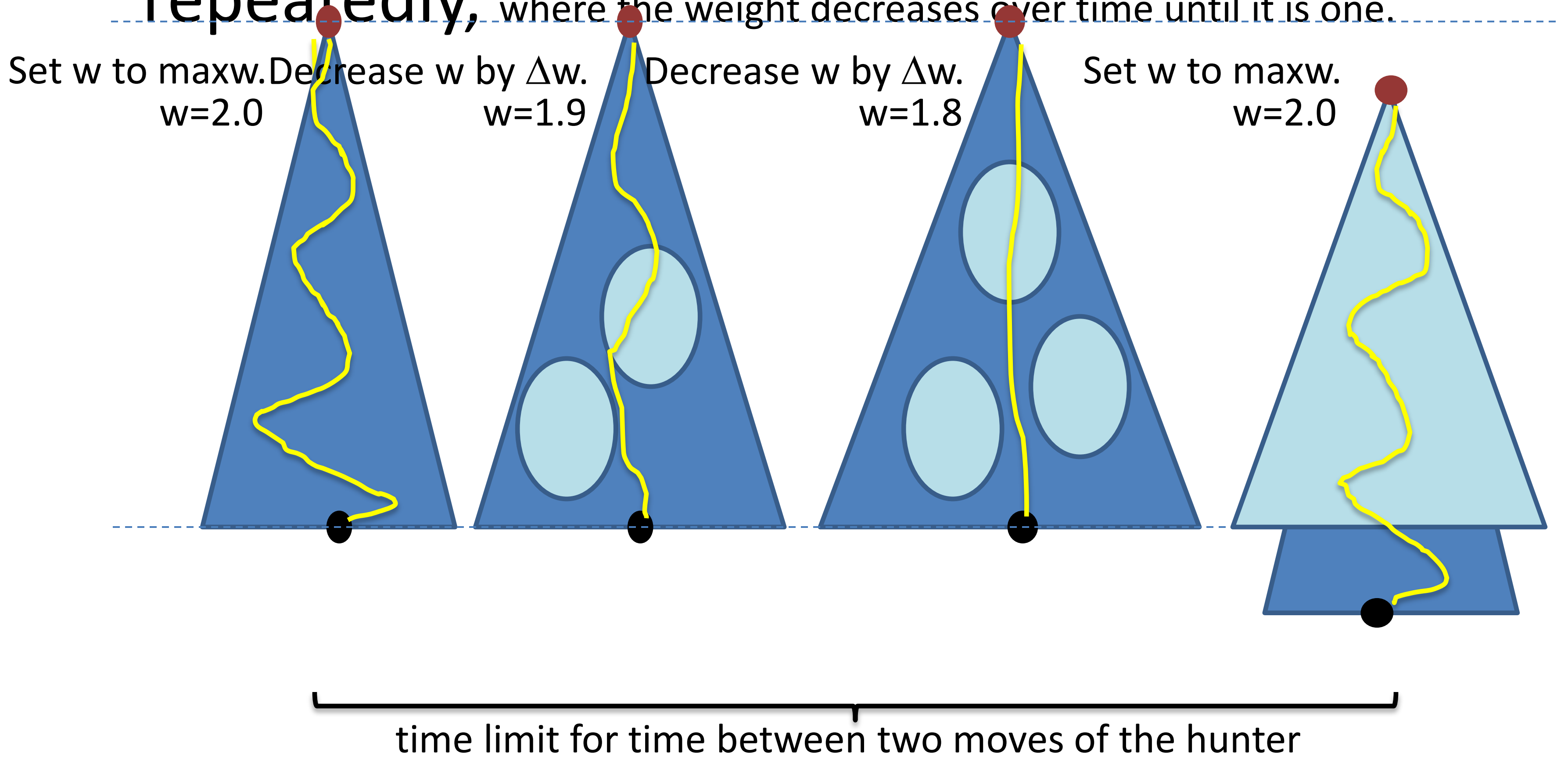
ARA* [Likhachev, Gordon, Thrun, 2003]

- The hunter uses incremental weighted A* repeatedly, where the weight decreases over time until it is one.



Incremental ARA* = FRA* + ARA*

- The hunter uses incremental weighted A* repeatedly, where the weight decreases over time until it is one.



Incremental ARA* = FRA* + ARA*

FRA*

Incremental ARA* = FRA* + ARA*

ARA*

Incremental ARA* = FRA* + ARA*

Incremental ARA* = FRA* + ARA*

```
function ImprovePath()
while g(s_goal) + ε × h(s_goal, s_goal) > min_{s∈OPEN}(g(s) + ε × h(s, s_goal))
    move s ∈ OPEN with the smallest g(s) + ε × h(s, s_goal) from OPEN to CLOSED;
    v(s) := g(s);
    forall s' ∈ Neighbor(s)
        if g(s') > v(s) + c(s, s')
            g(s') := v(s) + c(s, s');
            parent(s') := s;
            if s' ∉ CLOSED
                if s' ∉ OPEN AND s' ∉ INCONS
                    insert s' into OPEN;
            else
                move s' from CLOSED to INCONS;
if g(s_goal) = ∞
    return false;
else
    return true;

function ComputePath()
repeat
    return false if ImprovePath() = false;
    return true if ε = 1 OR the time limit has been reached;
    OPEN := OPEN ∪ INCONS;
    CLOSED := INCONS := ∅;
    ε := max(1, ε − δ_ε);

procedure Step1()
if g(s_start) ≠ v(s_start)
    g(s_start) := v(s_start);
    delete s_start from INCONS if s_start ∈ INCONS;
    delete s_start from OPEN if s_start ∈ OPEN;

procedure Step2()
if s_start ≠ previous_s_start
    parent(s_start) := NULL;
    forall s in the search tree rooted at previous_s_start but not in the subtree rooted at s_start
        v(s) := g(s) := ∞;
        parent(s) := NULL;
        delete s from INCONS if s ∈ INCONS;
        delete s from OPEN if s ∈ OPEN;
        insert s into DELETED;

procedure Step3()
forall s ∈ DELETED
    forall s' ∈ Neighbor(s)
        if g(s) > v(s') + c(s', s)
            g(s) := v(s') + c(s', s);
            parent(s) := s';
    if g(s) ≠ ∞
        insert s into OPEN;
OPEN := OPEN ∪ INCONS;
CLOSED := INCONS := DELETED := ∅;

procedure Step4()
if g(s_goal) + ε × h(s_goal, s_goal) > min_{s∈OPEN}(g(s) + ε × h(s, s_goal))
    ε := ε_max;
else
    ε := max(1, ε − δ_ε);

function Main()
forall s ∈ S
    v(s) := g(s) := ∞;
    parent(s) := NULL;
ε := ε_max;
s_start := the current cell of the hunter;
s_goal := the current cell of the target;
OPEN := CLOSED := INCONS := DELETED := ∅;
g(s_start) := 0;
insert s_start into OPEN;
while s_start ≠ s_goal
    return false if ComputePath() = false; /* Step 5 */
    previous_s_start := s_start;
    identify a path from s_start to s_goal using the parents;
    while the target has not been caught yet AND is still on the path from s_start to s_goal
        the hunter follows the path from s_start to s_goal;
    return true if the target has been caught;
    s_start := the current cell of the hunter;
    s_goal := the current cell of the target;
    Step1(); /* Step 1 */
    Step2(); /* Step 2 */
    Step3(); /* Step 3 */
    Step4(); /* Step 4 */
return true;
```

Incremental ARA* = FRA* + ARA*

- The algorithm:
  - Make the new state of the hunter locally consistent.
  - Delete all states from the search tree that are not in the subtree rooted in the new state of the hunter.
  - Add to the OPEN list all states that border non-leaf states in the search tree.
  - If the f-value of the new state of the target is no larger than the smallest f-values of all states in the OPEN list (= the search tree already contains a w-suboptimal path from the state of the hunter to the state of the target), then decrease w to max(1, w-Δw). Otherwise, set w to maxw.
  - Run an ARA* search to find an w-suboptimal path from the state of the hunter to the state of the target.
  - Move the hunter along the path until it catches the target or the target moves off the path. In the former case, stop. In the latter case, repeat.

Experimental Results

- We use 100 test cases with randomly selected unblocked connected cells for the hunter and target, namely
  - 100 four-neighbor random maps of size 1,000x1,000 with 25 percent randomly blocked cells; and
  - one four-neighbor game map of size 626x626 adapted from Warcraft III
- The target repeatedly follows a shortest path from its current cell to a randomly selected unblocked cell, skipping every tenth move.

Experimental Results

- FRA*, Repeated ARA* and Incremental ARA* were implemented in similar ways (e.g. using a binary heap).
- There is a time limit for each search but we allow the search algorithms to exceed the time limit until they find some path from the hunter to the target.
- For Repeated ARA* and Incremental ARA*, we use maxw=2.0 and Δw=0.1.

Experimental Results (Random Maps)

	Time limit	Moves per test case	First searches exceeding the time limit	Searches per time interval
FRA*	200µs	747	17.6%	-
Repeated ARA*	200µs	868	65.2%	3.29
Incremental ARA*	200µs	827	6.8%	5.47
FRA*	500µs	747	12.8%	-
Repeated ARA*	500µs	852	28.4%	5.53
Incremental ARA*	500µs	804	0.5%	7.55
FRA*	1000µs	747	3.6%	-
Repeated ARA*	1000µs	836	4.1%	7.82
Incremental ARA*	1000µs	785	0.2%	9.07

Experimental Results (Game Map)

	Time limit	Moves per test case	First searches exceeding the time limit	Searches per time interval
FRA*	200µs	545	15.5%	-
Repeated ARA*	200µs	652	69.1%	3.29
Incremental ARA*	200µs	613	6.1%	5.47
FRA*	500µs	545	10.0%	-
Repeated ARA*	500µs	645	49.0%	4.69
Incremental ARA*	500µs	596	0.7%	7.51
FRA*	1000µs	545	5.7%	-
Repeated ARA*	1000µs	639	34.3%	5.97
Incremental ARA*	1000µs	577	0.3%	8.87

Conclusions

- Replanning is important when the search problem changes.
- We begin to understand replanning for goal-directed navigation in unknown terrain towards a stationary target (where the hunter gains knowledge about the terrain).
- This paper studies replanning for goal-directed navigation in known terrain towards a moving target (where the hunter gains knowledge about the target cell).

Conclusions

- Incremental ARA* is the first incremental anytime search algorithm for moving target search in known terrain.
- Incremental ARA* can be used with smaller time limits between moves of the hunter than competing state-of-the-art moving target search algorithms.

Acknowledgements

- Thanks to Nathan Sturtevant for his test problems and HOG2 environment (see movingai.com).
- The research at USC was supported by NSF, ARO and ONR.
- The research at Singapore Management University was supported by the Singapore National Research Foundation.
- The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the sponsoring organizations.