OPTIMIZING THE A* ALGORITHM: A PATH TO EFFICIENT UAV NAVIGATION

Ian Henriques

Seminole High School

DRONES IN DISASTERS
(BACKGROUND)

Photo Credit: NBC News

Photo Credit: V-Drone Pro

Photo Credit: Forbes

Drone Navigation Algorithms

- Closed source
- Proprietary

→ Limited to developed world

DRONES IN DISASTERS
(BACKGROUND)

Photo Credit: Haaretz

Unknown Surroundings

Crowded Environment

CRASHES

Research Goal: Open-source navigation system for autonomous drones

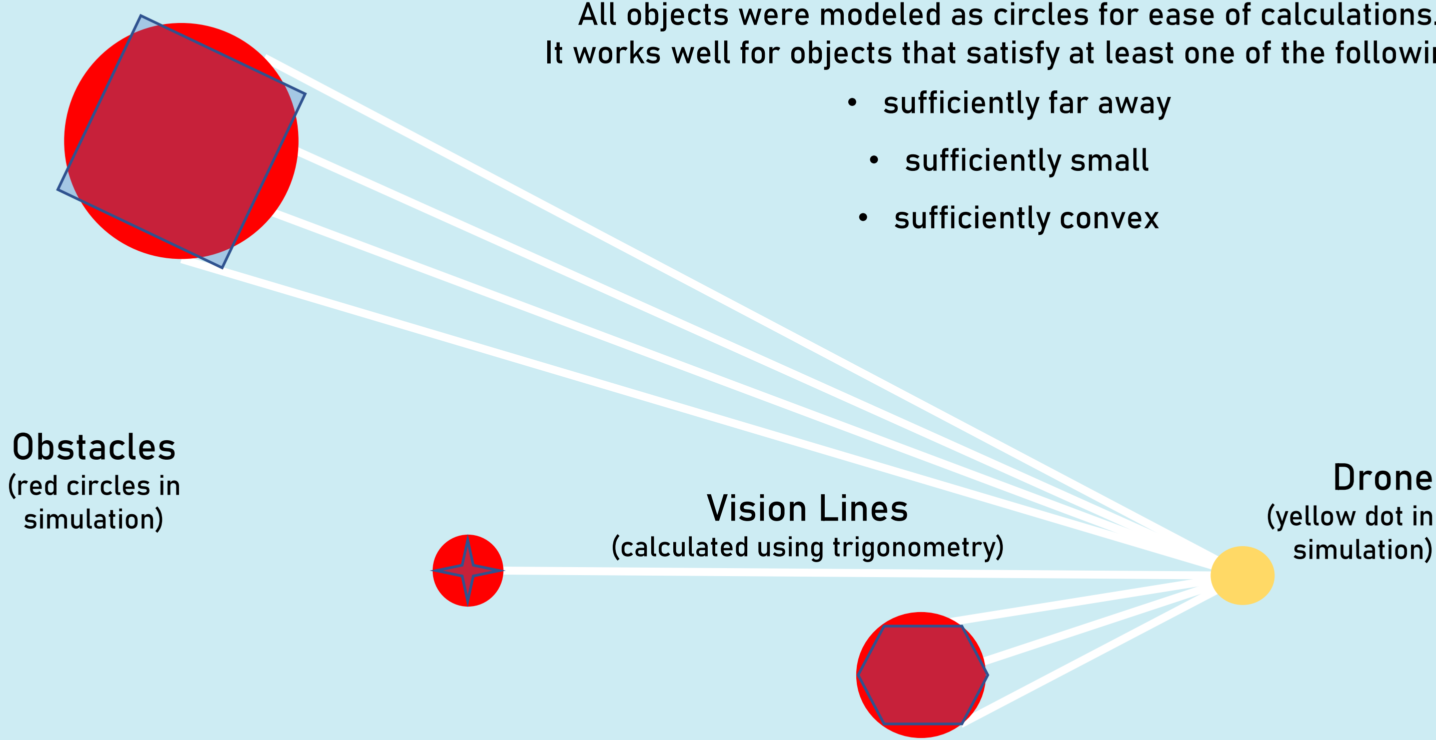
- COLLISION AVOIDANCE
- FUEL EFFICIENCY

SIMULATION ENVIRONMENT

- **PHYSICAL PROPERTIES OF DRONES AND OBSTACLES**
- **FUEL EFFICIENCY CALCULATIONS**

All objects were modeled as circles for ease of calculations.
It works well for objects that satisfy at least one of the following:

- sufficiently far away
- sufficiently small
- sufficiently convex



RELEVANT PHYSICS TO FUEL CONSUMPTION (IAN HENRIQUES)

Let $D = 0.254\ m$ (diameter of rotor blade), $\rho = 1.225\ kg/m^3$ (air density at sea level), $m = 2.57\ kg$ (mass of drone), and $g = 9.81\ m/s^2$ (gravitational acceleration). There are also velocity (v) and acceleration (a) values for the drone's current state.

Then $F_d = (0.07512)v^2$ (drag force on drone based on the drone's speed), $T = \sqrt{(F_d + ma)^2 + (mg)^2}$ (thrust), and $\alpha = \tan^{-1}\left(\frac{F_d+ma}{mg}\right)$ (the angle of the drone).

According to the body of knowledge, there is a quantity known as induced velocity (v_i) which obeys

$$v_i = \frac{2T}{\pi n D^2 \rho \sqrt{(v\cos\alpha)^2 + (v\sin\alpha + v_i)^2}} \cdots ①$$

After solving for it, one can attain

$$P = T(v\sin\alpha + v_i) \cdots ②$$

as the energy per second, or power, consumed by the drone.

Newton's method was used to find the solution to the following function, rearranged from ①:

$$f(v_i) = v_i^4 - 2v(\sin\alpha)v_i^3 + v^2v_i^2 - \left(\frac{T}{2\pi D^2\rho}\right)^2$$

The derivative of this function is

$$f'(v_i) = 4v_i^3 - 6v(\sin\alpha)v_i^2 + 2v^2v_i$$

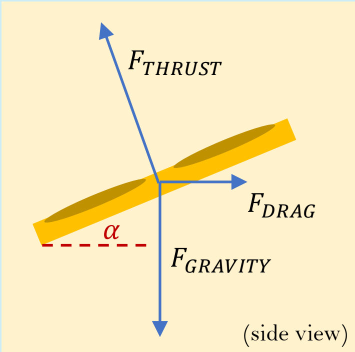
Let $v_{i,0} = 140$ to represent that 140 m/s (a starting value chosen since it is a suitable upper bound) is the intial guess of the induced velocity. The rule used is:

$$v_{i,n+1} = v_{i,n} - \frac{f(v_{i,n})}{f'(v_{i,n})}$$

where n is the current guess. When the guess reaches a desirable accuracy, it is accepted.

Then ② is used to determine the power. From this, two auxiliary calculations are used. One is energy per frame, calculated as EPS $= P/(10\text{ fps})$ due to a frame rate of 10 fps. Next, there was the energy per distance, calculated as EPD $= P/d$ with a trip distance of d.

Accurate Fuel Efficiency Modeling



- **Analysis of Forces** (PHYSICS)
- **Momentum Theory** (PHYSICS)
- **Differentiation** (CALCULUS)
- **Newton's Method** (CALCULUS)

ORIGINAL A-STAR
(GRID-BASED ALGORITHM)

Popularity

Efficiency

Good fit for my model

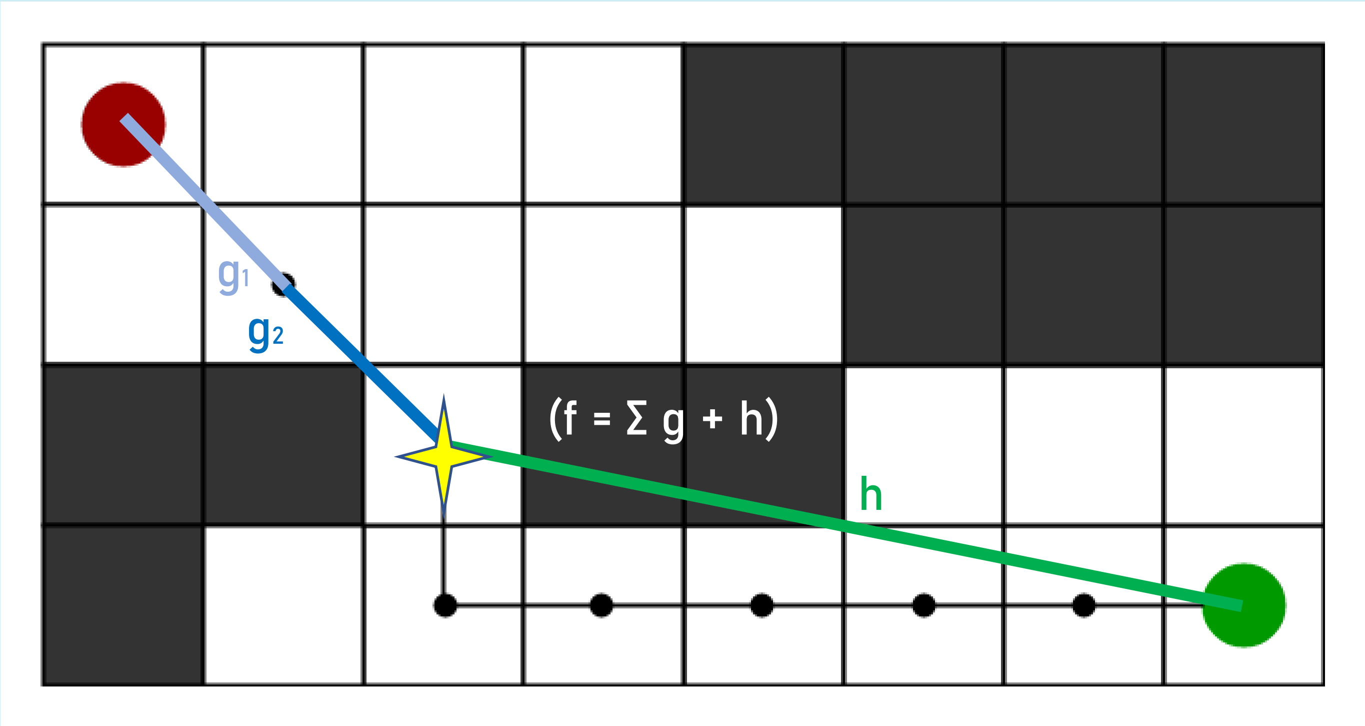
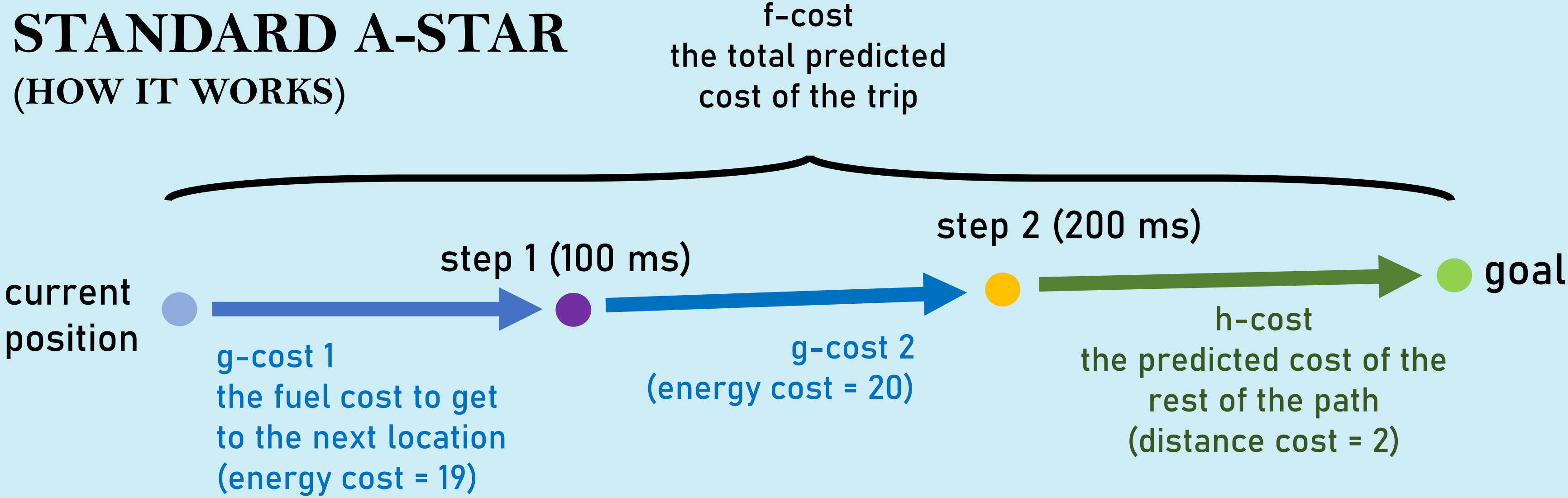


Photo Credit: GeeksforGeeks

STANDARD A-STAR
(HOW IT WORKS)

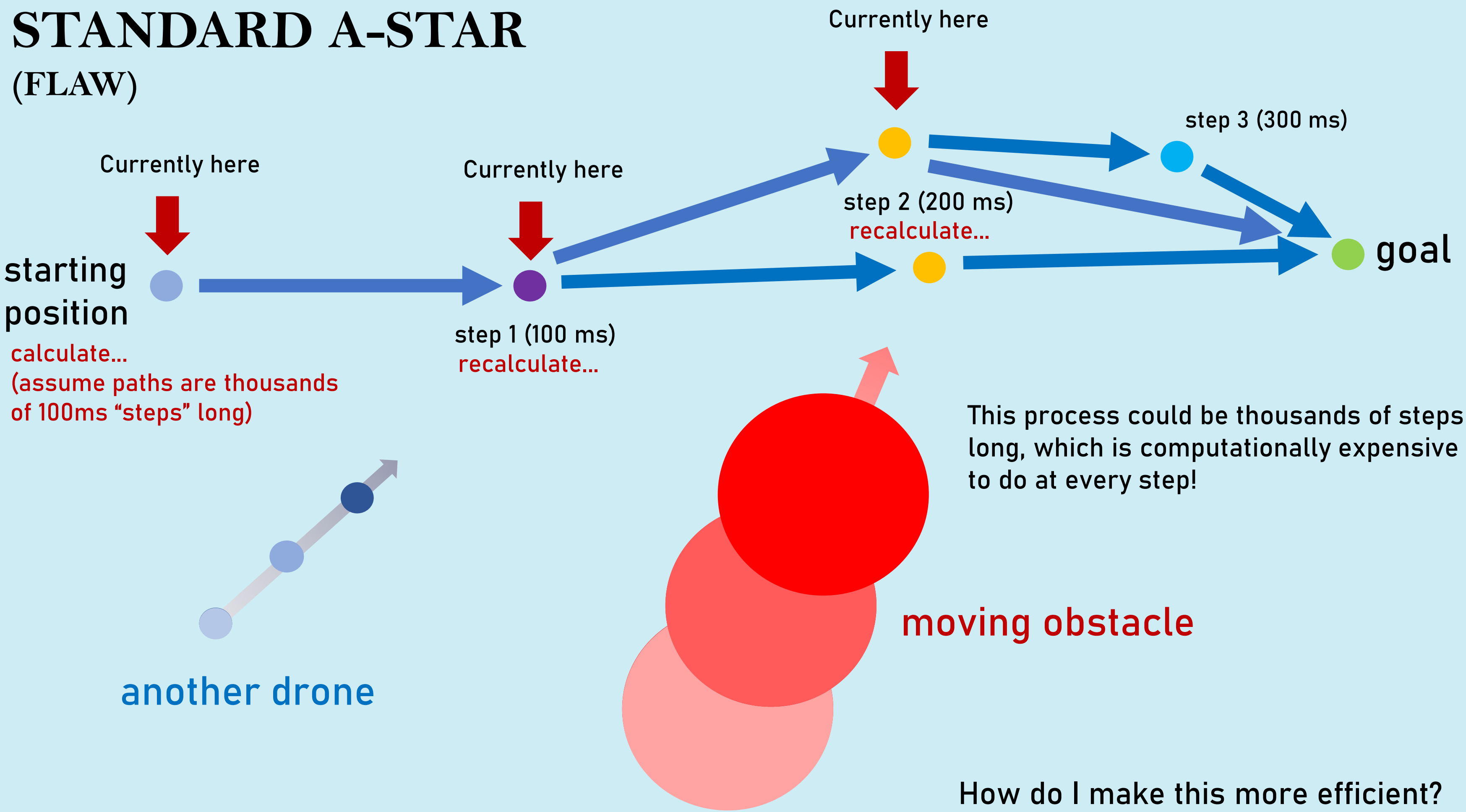


KINEMATICS +
FUEL EFFICIENCY
CALCULATIONS

$$f = (\Sigma\ g) + h = (19 + 20) + 2 = \underline{41}$$

Locations with the lowest f-cost should always be explored first.

STANDARD A-STAR
(FLAW)

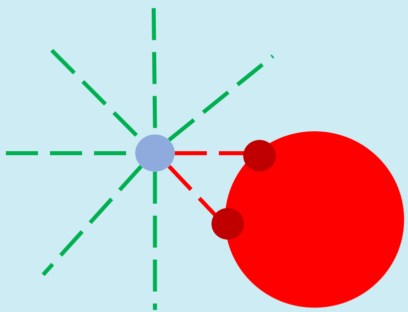


This process could be thousands of steps long, which is computationally expensive to do at every step!

How do I make this more efficient?

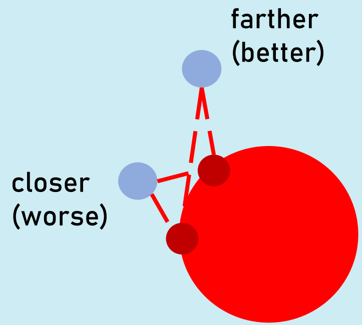
A-STAR VARIANT
(DYNAMIC ENVIRONMENTS)

Action 1
Find estimates of where obstacles are, or danger points



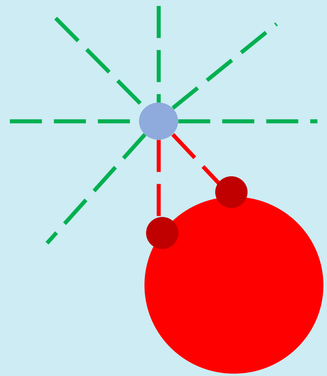
Number of rays is based on a parameter which I called the *input size*

Action 2
Based on obstacles, evaluate future short-term locations



This ranking is determined by a value which I called the *penalty*

Action 3
Move to chosen location, recalculate often

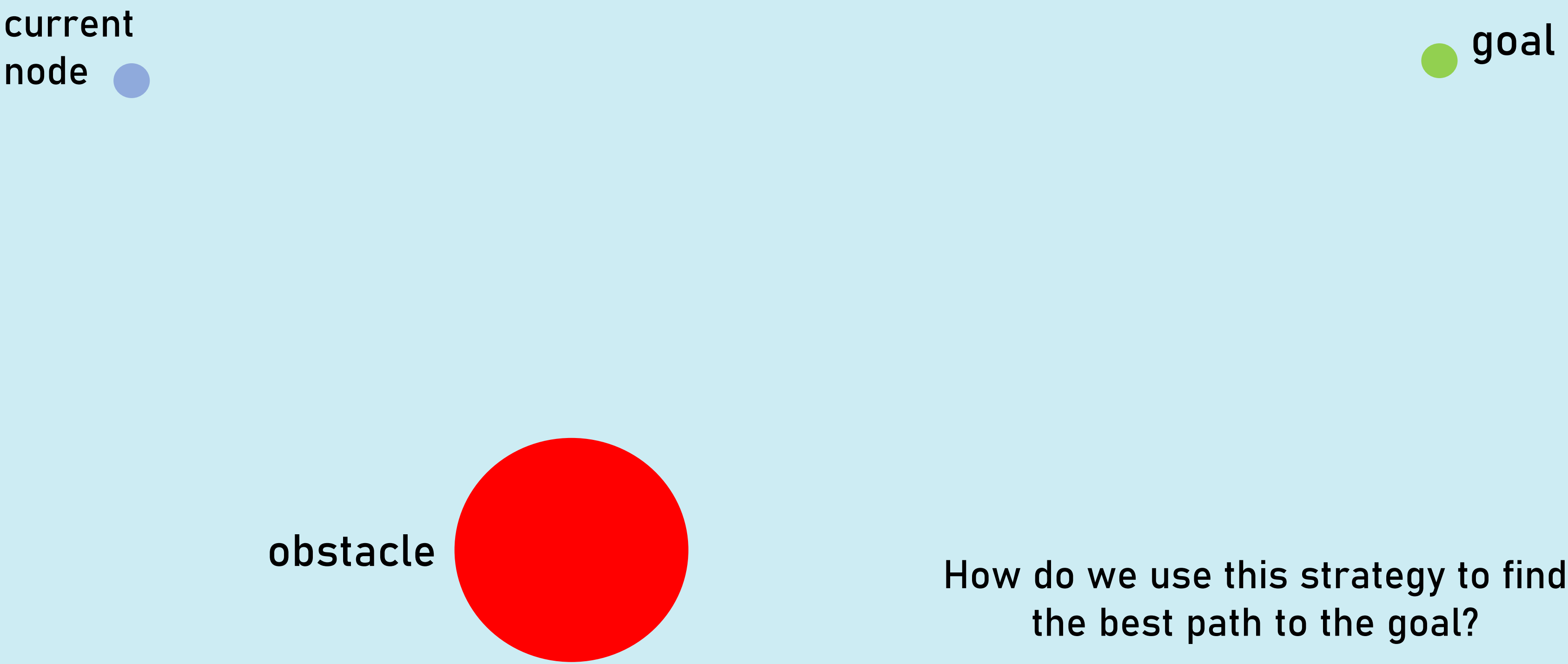


The number of steps done before recalculation will be called the *step size*

A-STAR VARIANT
(EXAMPLE)

AT START

$(f = \Sigma\, g + h + p)$

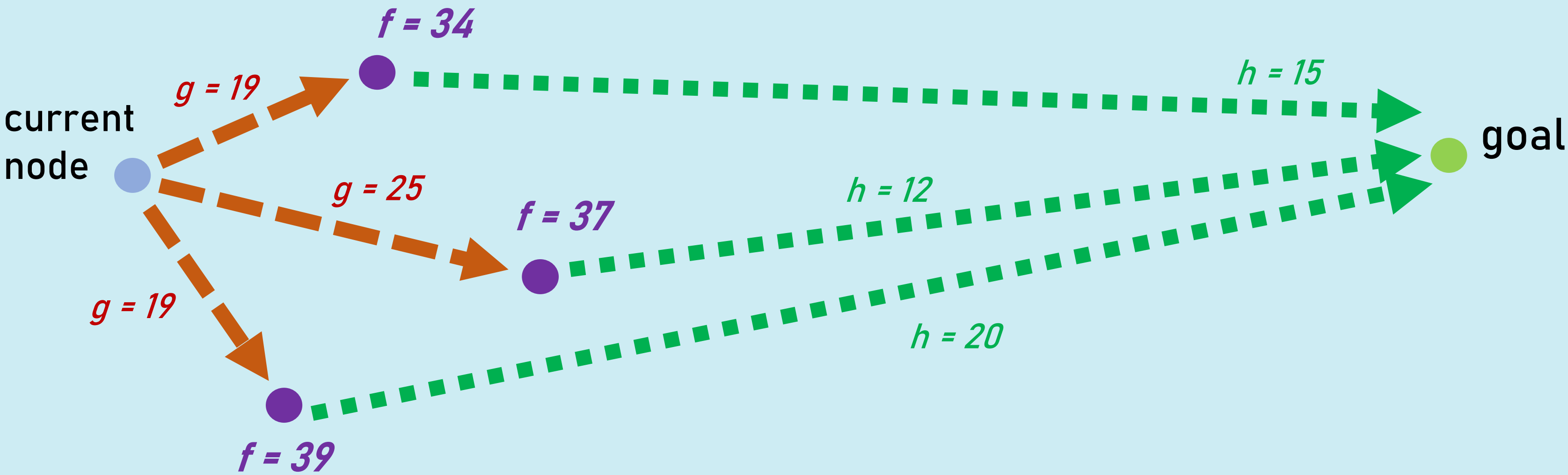


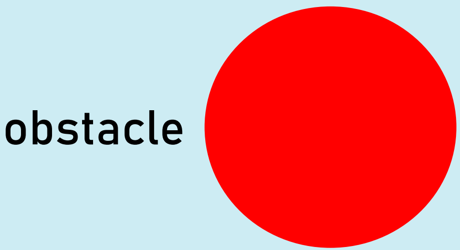
How do we use this strategy to find the best path to the goal?

A-STAR VARIANT
(EXAMPLE)

FIRST STEP

$(f = \Sigma\, g + h + p)$



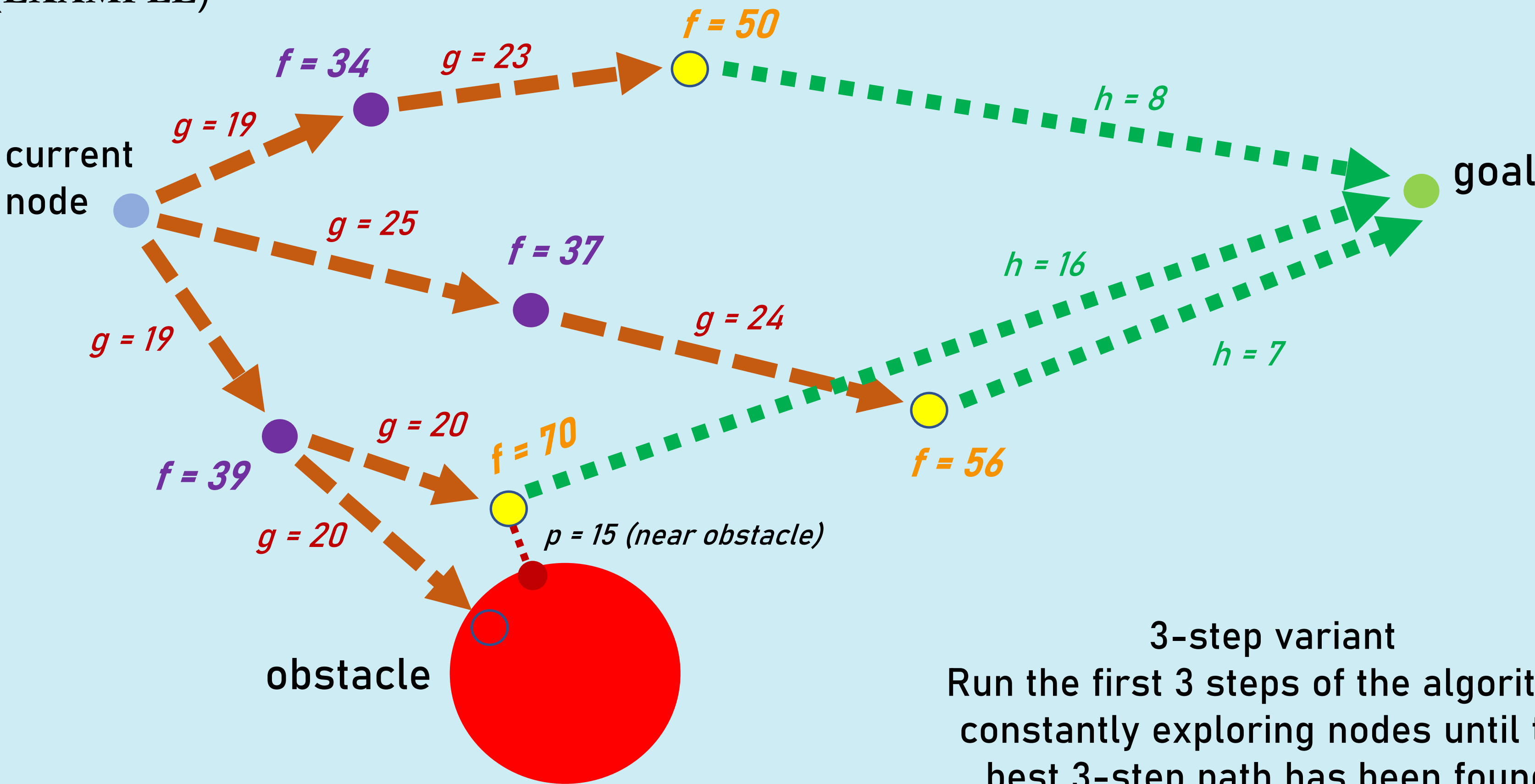


3-step variant
Run the first 3 steps of the algorithm, constantly exploring nodes until the best 3-step path has been found.

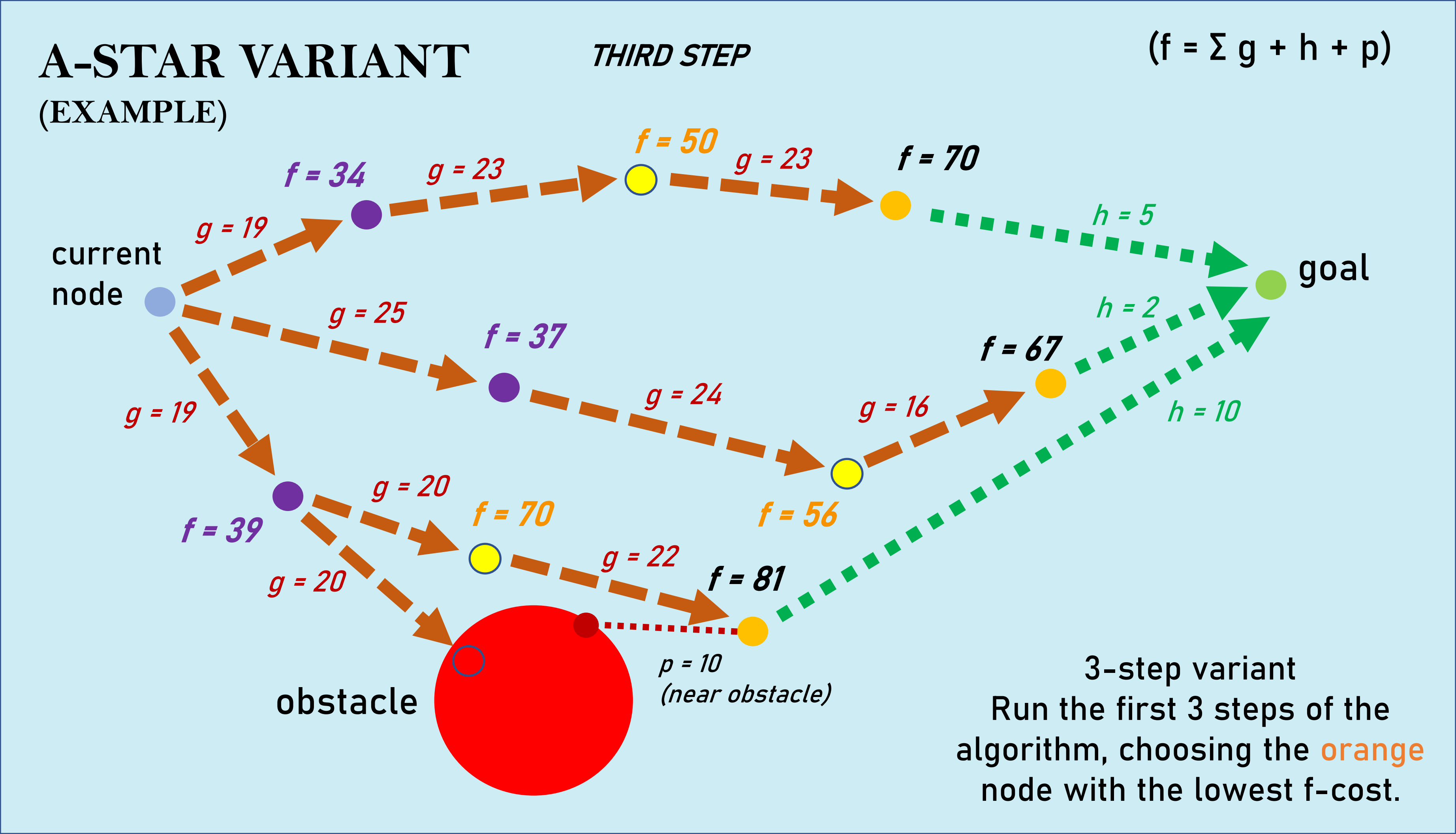
A-STAR VARIANT
(EXAMPLE)

SECOND STEP

(f = Σ g + h + p)



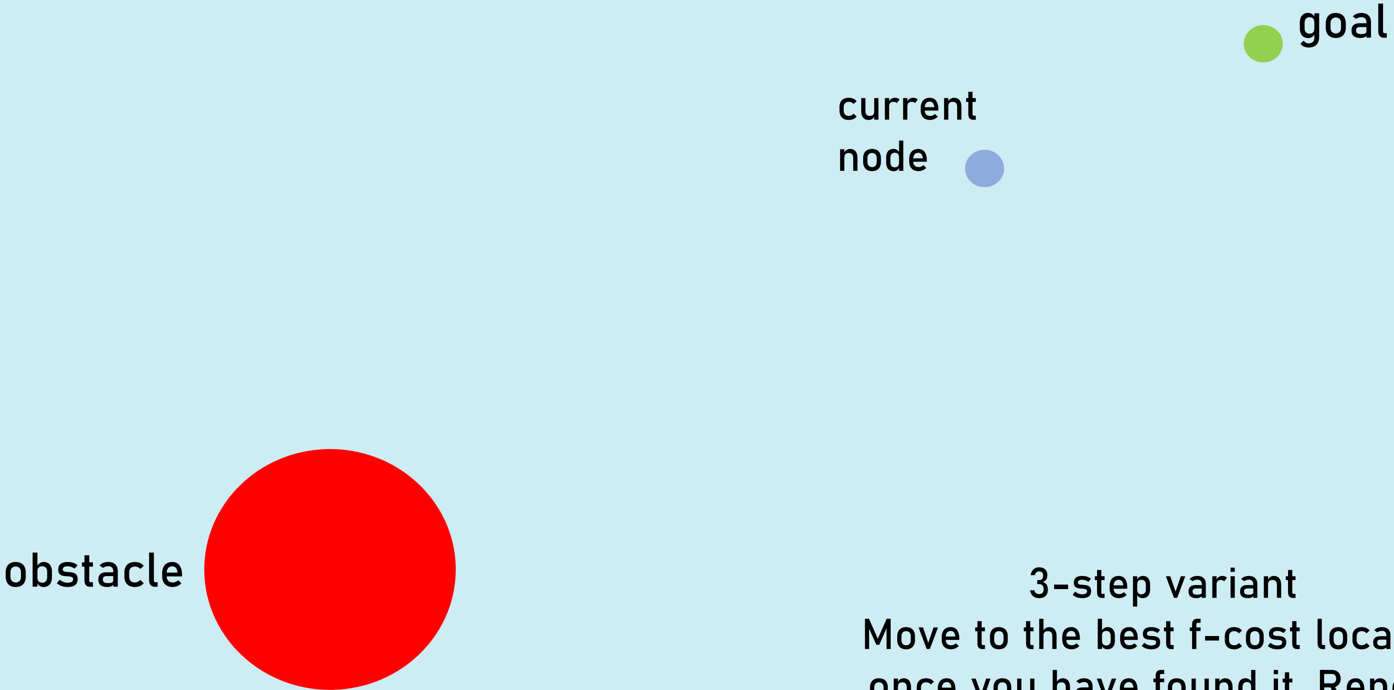
3-step variant
Run the first 3 steps of the algorithm, constantly exploring nodes until the best 3-step path has been found.


A-STAR VARIANT
(EXAMPLE)
THIRD STEP
(f = Σ g + h + p)
current node
goal
obstacle
f = 34
f = 37
f = 39
f = 50
f = 70
f = 67
f = 56
f = 70
f = 81
g = 19
g = 25
g = 19
g = 23
g = 23
g = 24
g = 16
g = 20
g = 20
g = 22
h = 5
h = 2
h = 10
p = 10
(near obstacle)
3-step variant
Run the first 3 steps of the algorithm, choosing the orange node with the lowest f-cost.

A-STAR VARIANT
(EXAMPLE)

AFTER

$(f = \Sigma\, g + h + p)$

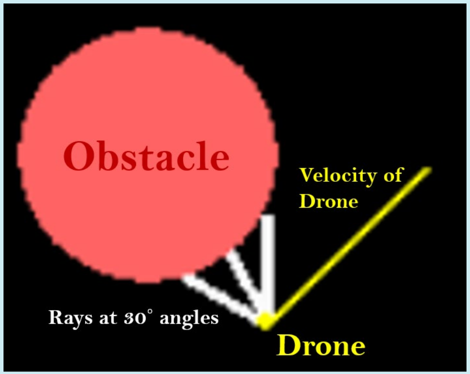


3-step variant
Move to the best f-cost location
once you have found it. Repeat.

A-STAR VARIANT
(IMPLEMENTATION)

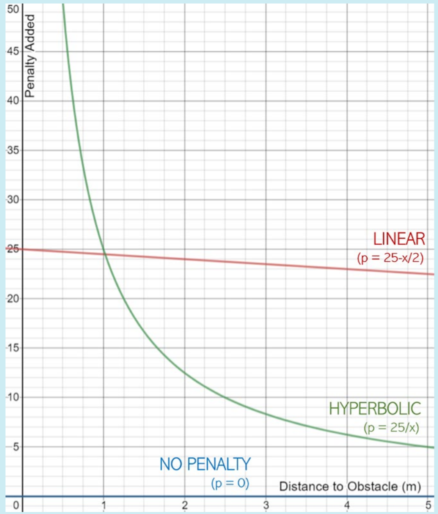
35 total combinations

Action 1
Find estimates of where obstacles are, or "danger points"



3 input sizes:
12, 24, 36

Action 2
Based on obstacles, evaluate future short-term locations



3 types of penalties:
no penalty, linear, hyperbolic

Action 3
Recalculate often

The past few slides demonstrated how this works

5 step sizes:
1, 2, 3, 4, 5

A-STAR VARIANT
(TESTING)

8 SCENARIOS (15 trials each)

- No obstacles
- 1 large, stationary obstacle
- 1 large, stationary obstacle with 2 small, moving obstacles
- 4 small, moving obstacles
- 8 small, moving obstacles
- 12 small, moving obstacles
- 16 small, moving obstacles
- Swarm with 5 copies of itself

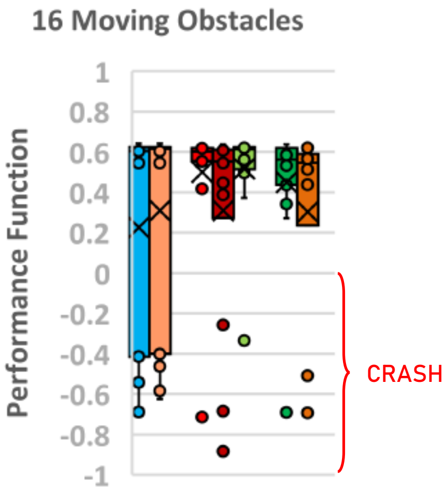
Performance function

- if (crashed)
  - ranked by distance from goal (negative value)
- else if (reached goal)
  - ranked by fuel efficiency (positive value)
- else
  - ranking of 0

A-STAR VARIANT
(ANALYSIS)

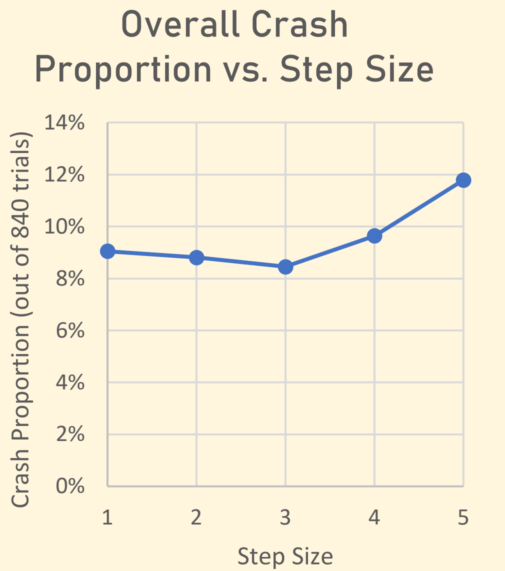
Individual graph analysis

- Compared by least crashes
- Compared by lowest spread

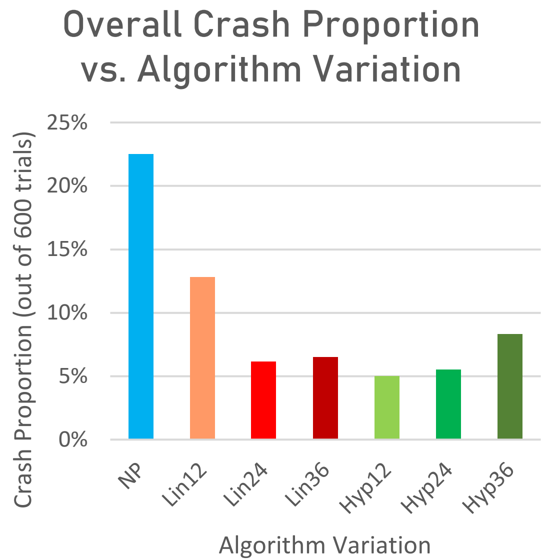


(example from step size 3)

Step size analysis



Penalty & inputs analysis



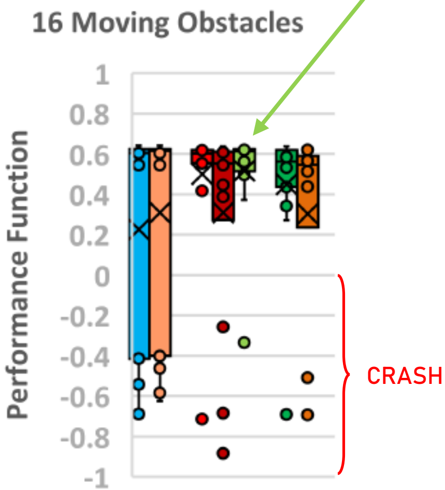
A-STAR VARIANT
(ANALYSIS)

CONCLUSION

- <u>Step size 3, hyperbolic penalty, input size 12</u> was shown to perform the best overall (only 3 crashes out of 120 trials, a crash rate of 2.5%)
- <u>The blue distributions (standard A*)</u> performed the worst overall, crash rate 23%

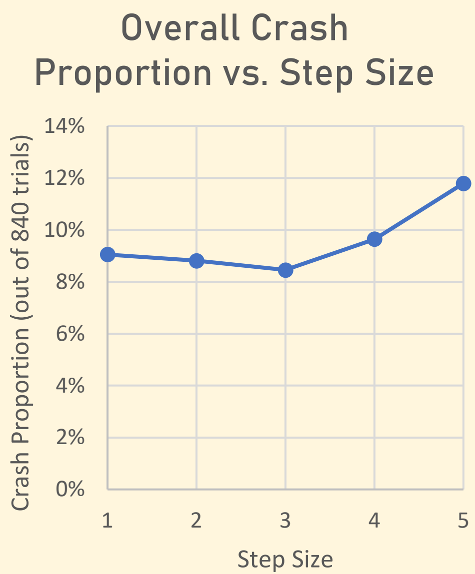
Individual graph analysis

- Compared by least crashes
- Compared by lowest spread

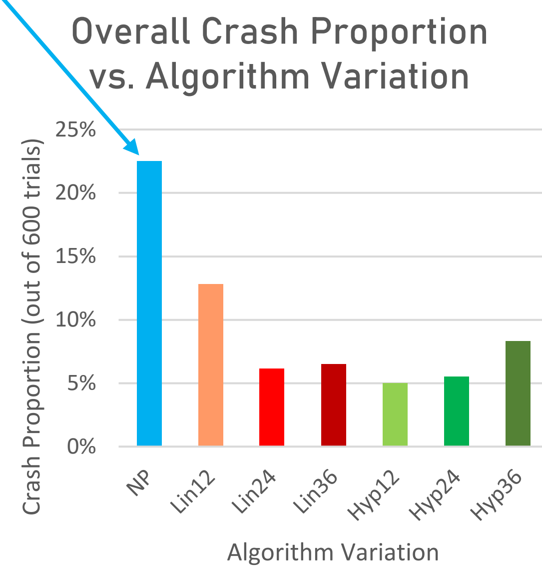


(example from step size 3)

Step size analysis



Penalty & inputs analysis



A-STAR VARIANT
(IMPLICATIONS & FUTURE STUDIES)

Implications

- These modifications to the A* algorithm performed better than the standard algorithm
- This model incorporated fuel efficiency calculations using Momentum Theory, which hasn't been done before and can increase the accuracy of fuel planning
- Using kinematics to evaluate future paths rather than a grid, the simulation showed realistic motion.
- Overall, the combination of simulation enhancements and algorithmic modifications increases the viability of A* as an algorithm to control drones in natural disaster relief efforts

Future Work

- Support for 3-D simulation that can take into account terrain and turbulence
- Artificial Intelligence to help drones adapt to even more complex environments
- Implementation of the algorithm in a physical drone to test effectiveness