

# Mars Lander

Simulations for *Easy on the right* test: ~17600

MY REPORT

HISTORY

SCORE

100%

SHARE

DETAILS →

FUEL LEFT

1851

WHAT YOU LEARNED

The solution to this puzzle lies in the following concepts. If you consider that you've acquired the skills listed below, tick the corresponding boxes (and they'll appear on your learning profile). If not, brush up on your knowledge and try again!

☒ Distances ☒ Trigonometry

Click to rate

★ ★ ★ ★ ★

Hated it.

BACK TO HOME

Validators

The validators differ from the puzzle test cases to prevent hard coded solutions. This is why you can have some fails here even if all of the tests provided in the IDE have been successfully passed.

01	Easy on the right
02	Initial speed, correct side
03	Initial speed, wrong side
04	Deep canyon
05	High ground

## Forward model details

- Action's description contains differences between previous and next thrust power and rotation so it's easier to make crossovers and mutations.

## RHEA

The algorithm described below passed all tests.

- Population size: 50,
- Action sequence length: 20,
- New populations for computing each move: 20,

## Creating new population

- 6 bests individuals proceed to next population.
- The rest is made up as follows:
  1. 2 parents are selected with roulette wheel.
  2. 2 children are created with 1-point crossover
  3. Every gene of each child can mutate with probability of  $1/2L$ , where  $L$  – sequence length

## Fitness function

I evaluated only last state of each sequence. There are several features of state that I use to calculate it's value:

- distance to landing zone – should be as small as possible
- vertical and horizontal speed – lander shouldn't fly too fast
- if landing zone is above lander's position, it should fly up.

For each feature I create part of evaluation function:

- For distance it's just  $7000 - ds$ , where  $ds$  – distance to landing zone in straight line (7000 because it's more than maximum possible distance and I want my function to be always  $\geq 0$ )
- For speed I have 2 functions (1 for vertical speed, 1 for horizontal). They can be described as follows:
  - $f(v) = 1$  if  $v \leq v_{min}$
  - $f(v) = 0$  if  $v \geq v_{max}$
  - $f(v) = 1/(v_{min} - v_{max})$  if  $v_{min} < v < v_{max}$
- For going up
  - if landing is below lander then  $f(state) = 1$
  - else  $f(state) = 1 / \text{distance\_to\_landing}$

Multiplying these 3 functions gives my fitness function.

## Program variations

I tried different approaches for fitness function and evolution but they didn't give satisfying results.

I left some unused evaluation function in my code. I tried, for example

- Evaluate somehow lander rotation – it turned out to be unnecessary
- Calculate ground distance from crash point to landing zone – it was too slow

I also tried crossover from <https://www.codingame.com/blog/genetic-algorithm-mars-lander/>, and different values of population size, sequence length etc. It didn't give better results.