Fitness Landscape

Shin Yoo
CS454, Autumn 2018, School of Computing, KAIST

Recap

- We need three key elements for SBSE
  - Representation: how we express candidate solutions for storage
  - Fitness Function: how we compare candidate solutions for selection
  - Operators: how we modify candidate solutions for trial-and-error

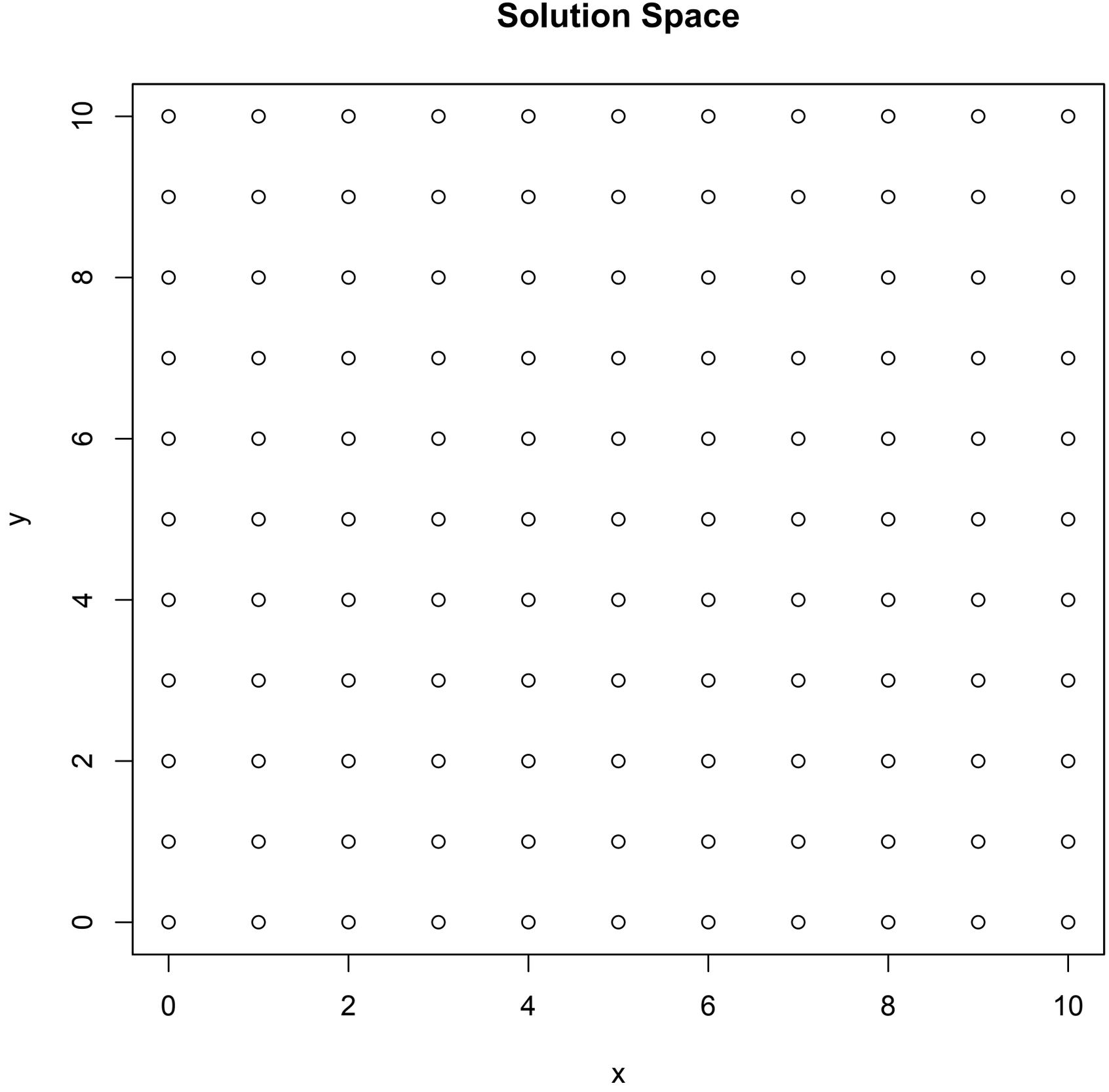
Fitness Landscape

- A spatial view of the search: there is no guarantee that the **actual** optimisation you are working on can be easily visualised spatially. However, this visual analogy is a useful tool when discussing the distribution of the fitness across possible solutions.

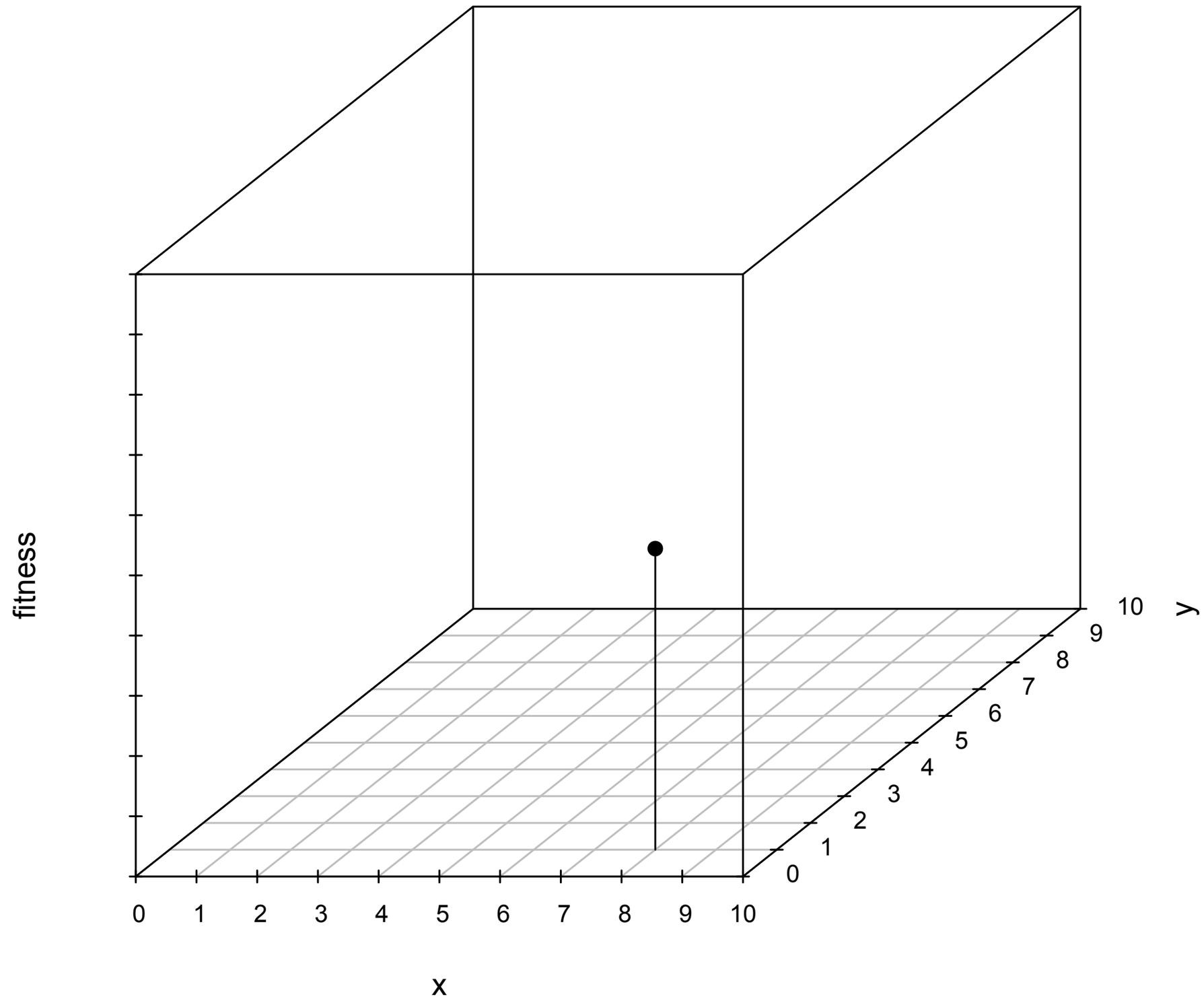
- Given a solution space S (a hyperplane), and a fitness function F, a fitness landscape is a hyper dimensional surface that represents F: $S \rightarrow \mathbb{R}$

Fitness Landscape

- Let's use a fake problem:
  - Given `0 ≤ x ≤ 10`, `0 ≤ y ≤ 10`, find `(x, y)` such that `x + y = 10`.

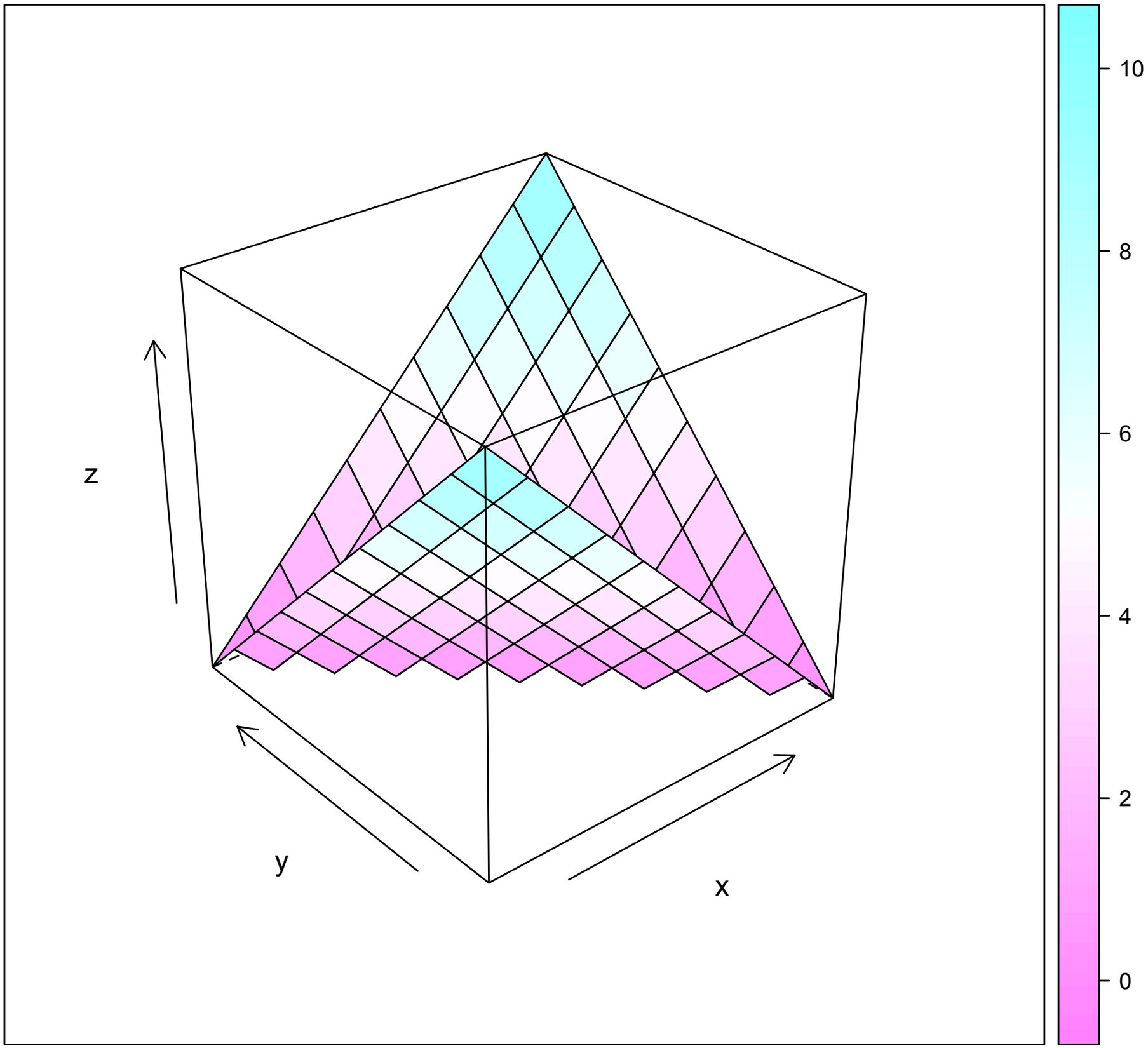

Solution Space
y
10
8
6
4
2
0
0
2
4
6
8
10
x

A single point in fitness landscape



Fitness for Fake Problem

- Given `(x, y)`, **how far** are we from **solving the problem**?
    - We solve the problem when `x + y == 10`
    - If the current sum of x and y are s, we are `|10 - s|` far away from solving the solution
    - `f(x, y) = |10 - (x + y)|`
    - Minimise the above function until it becomes 0.

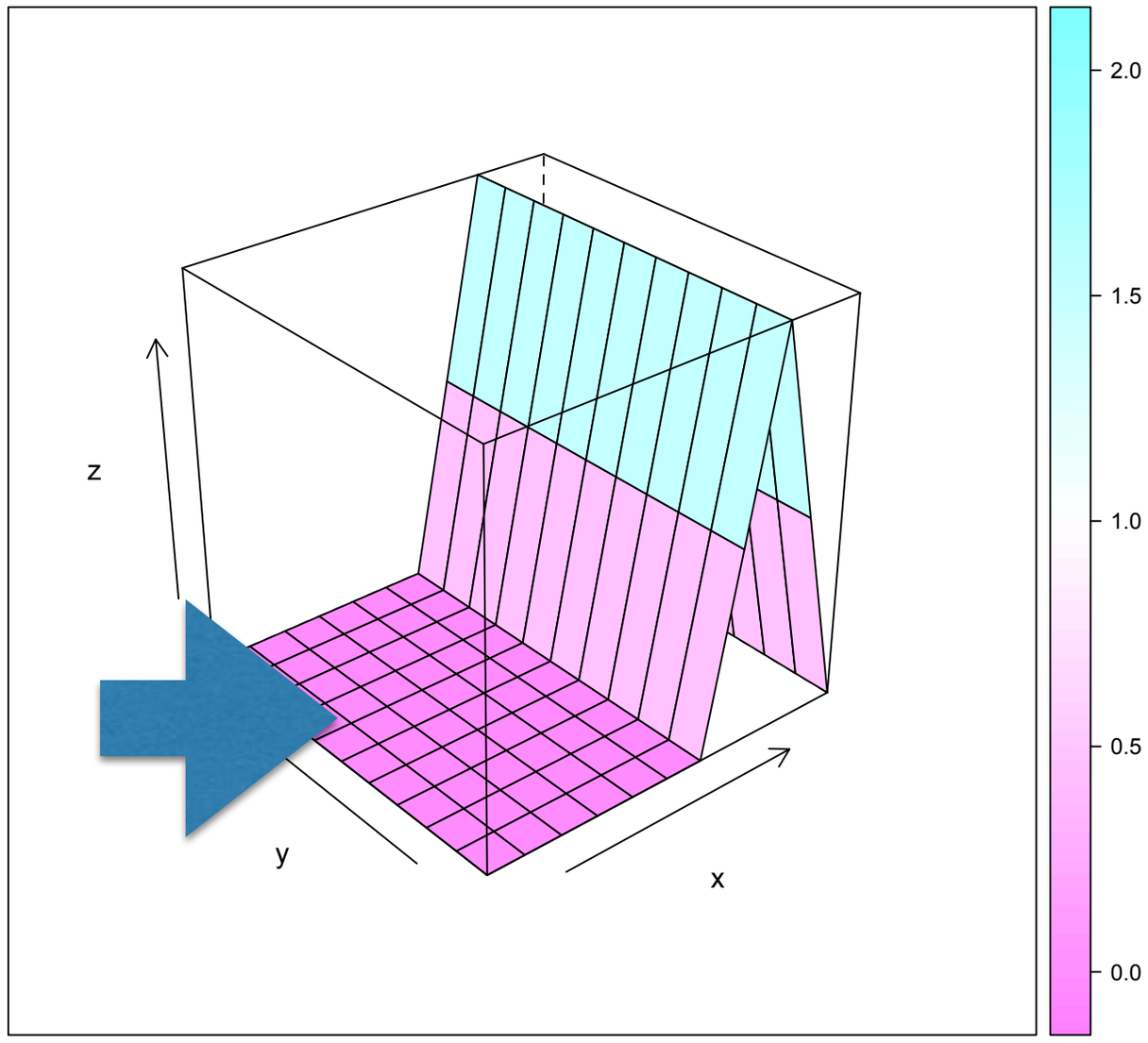

z
y
x
10
8
6
4
2
0

Properties of Landscape

- Size: small/large but also finite/(effectively) infinite
- Flatness: is there a large plateau?
- Ruggedness: how many local optima should we expect?
- Discreteness: continuous numeric, discrete numeric, combinatoric

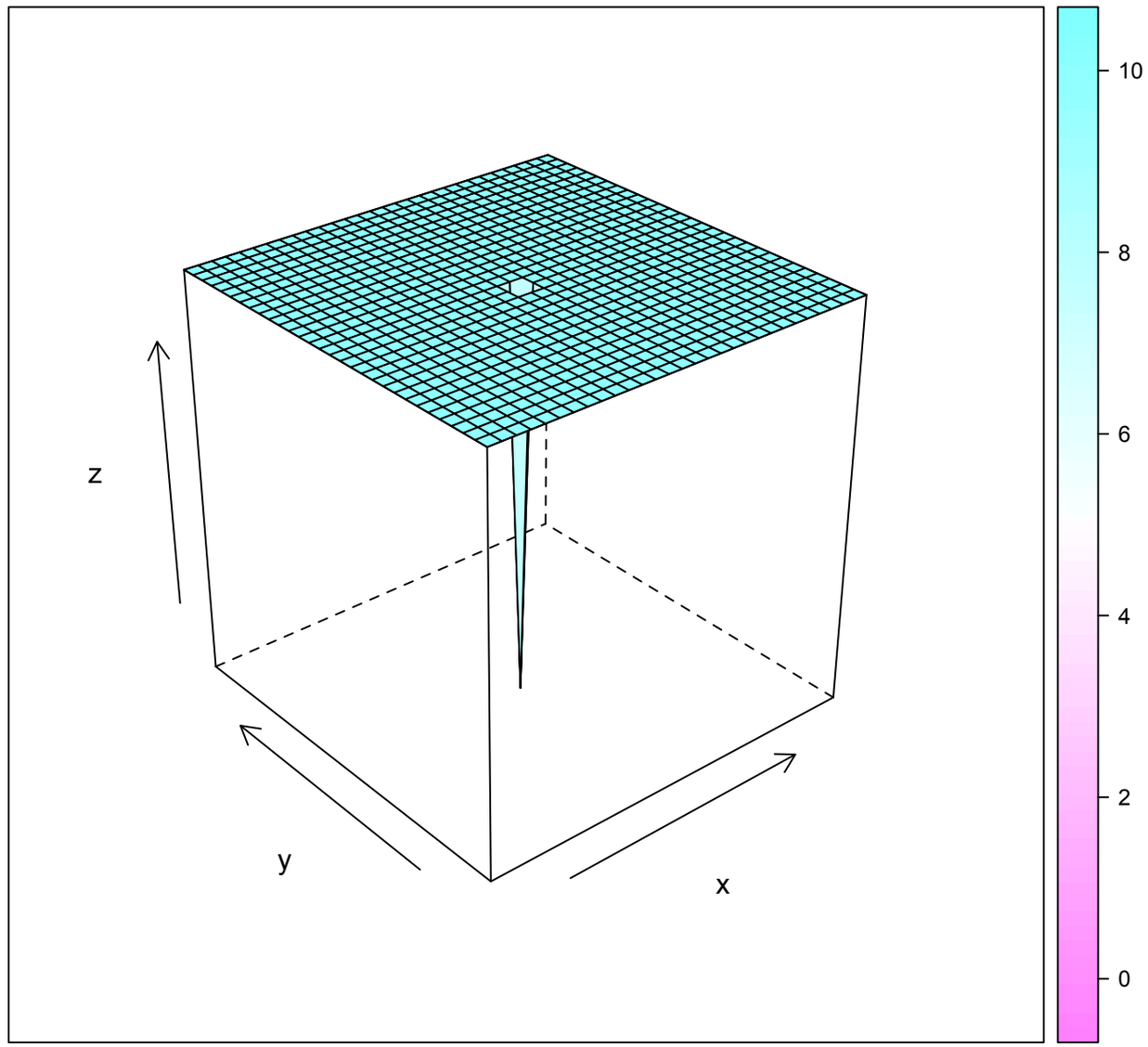
Plateau

- Large, flat region that does not exhibit any gradient.
- Suppose current solution as well as others generated by operators all fall in a plateau.
- There is no guidance; hard to escape.



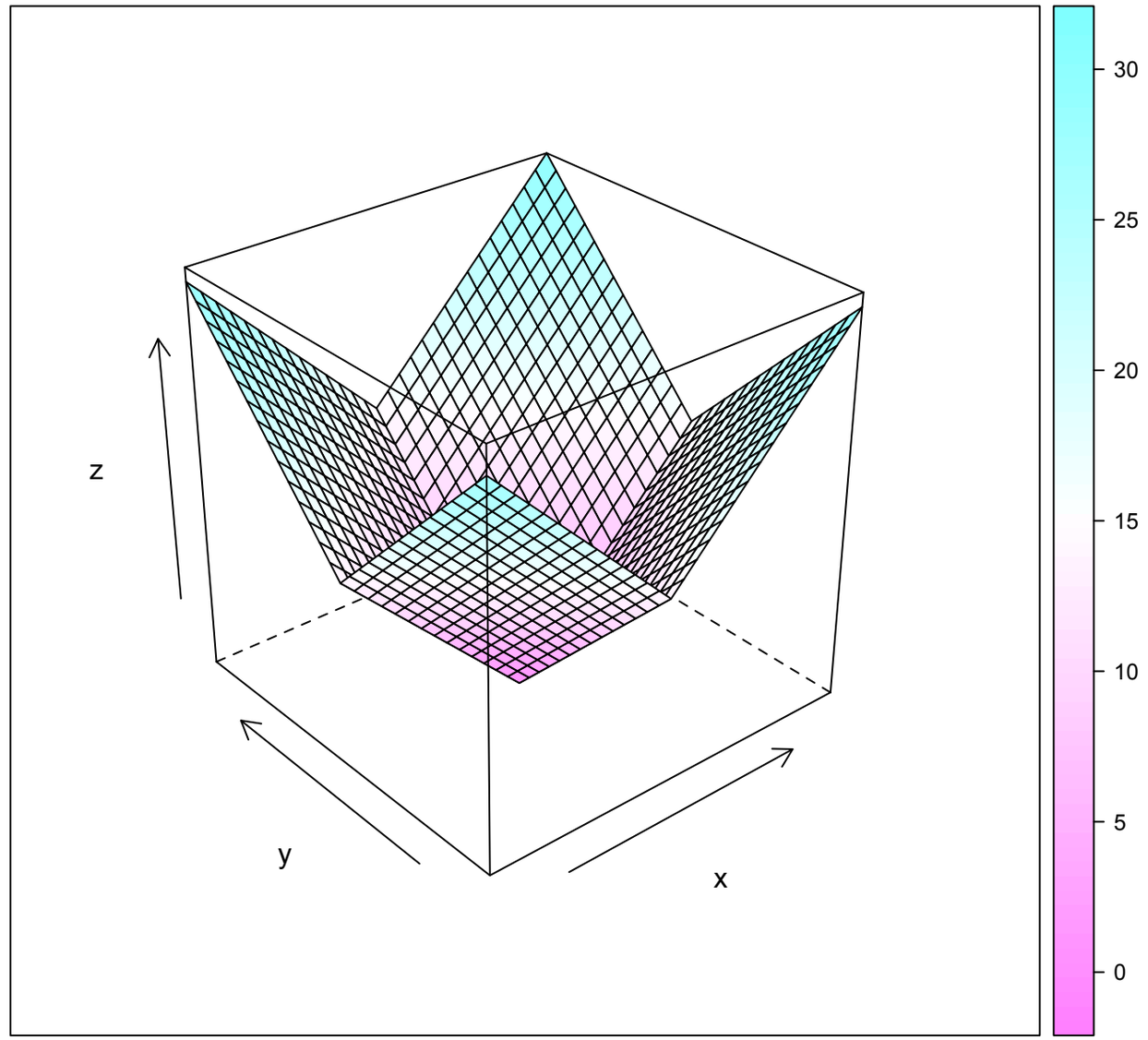
Needle in the Haystack

- Worst landscape to search.
- Can be avoided by transforming the problem and/or designing better fitness functions
- To search for (x, y) = (15, 15):
  - `f1(x, y) = (x==15 && y == 15) ? 0 : 10`



Needle in the Haystack

- f2(x, y) = |x-15| + |y-15|



(..later application in testing)

```
bool flag = (x == 42);
...
if(flag){
   //do some computation
   //that needs to be tested
}
```

```
...
if(x == 42){
   //do some computation
   //that needs to be tested
}
```

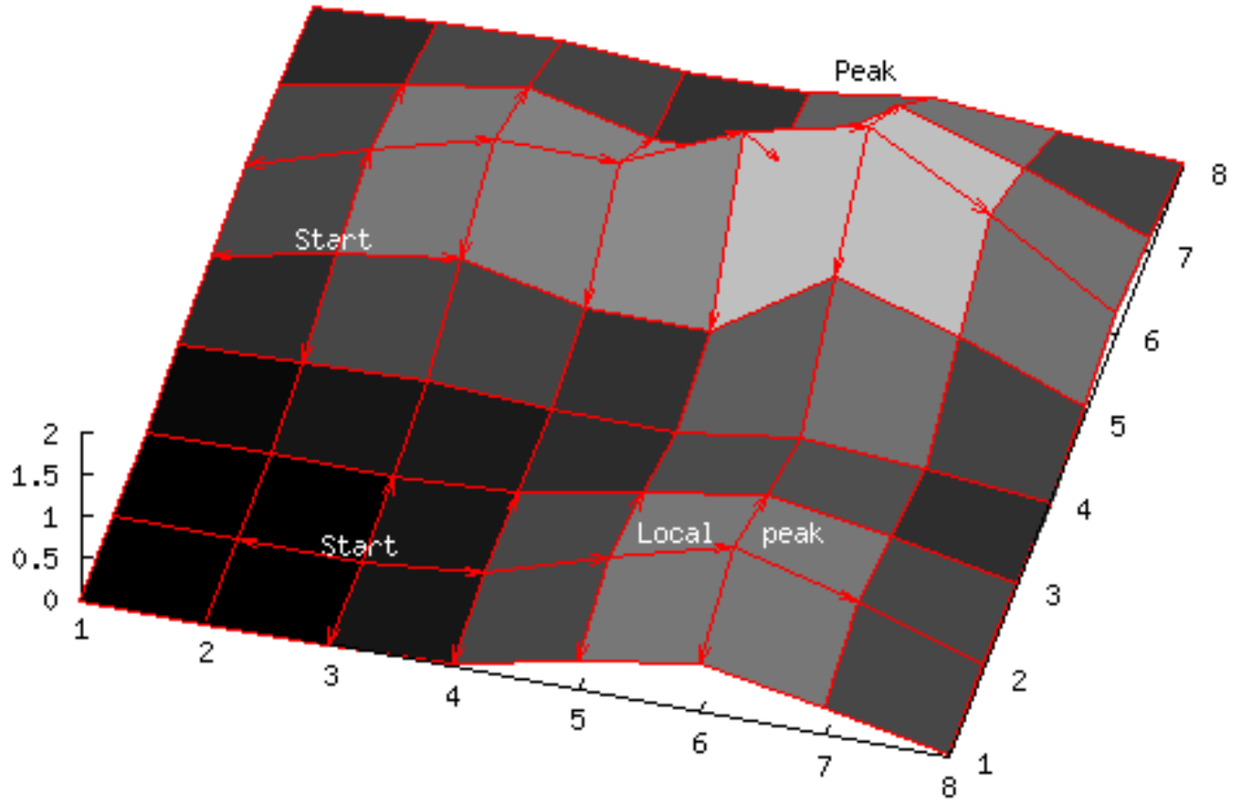
```
...
if(|x -42| == 0){
   //do some computation
   //that needs to be tested
}
```

M. Harman, L. Hu, R. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper. Testability trans- formation. IEEE Transactions on Software Engineering, 30(1):3–16, Jan. 2004.

Local vs. Global Optima

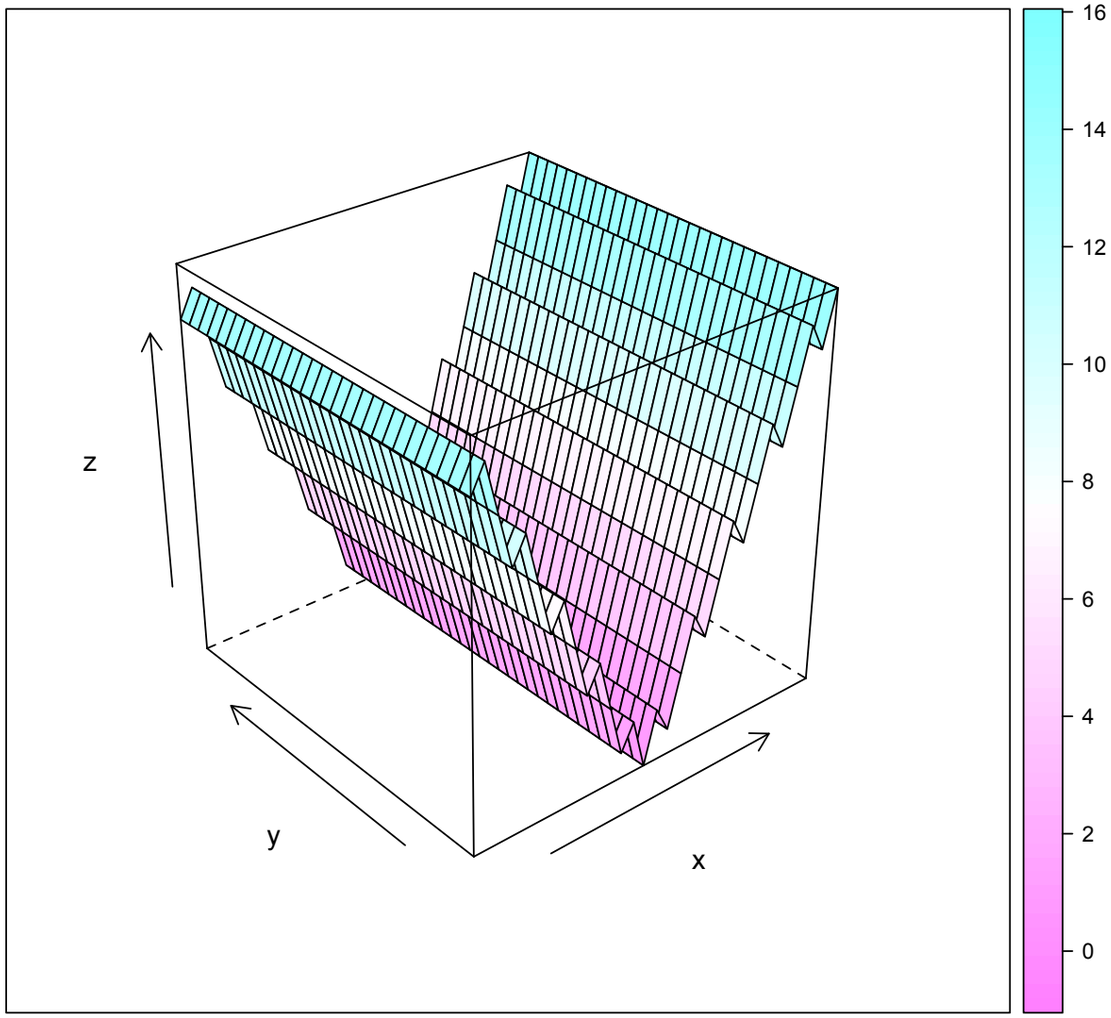
- Local optima: better fitness than any surrounding region, but not the best possible fitness

- Global optima: better fitness than any other point in the landscape

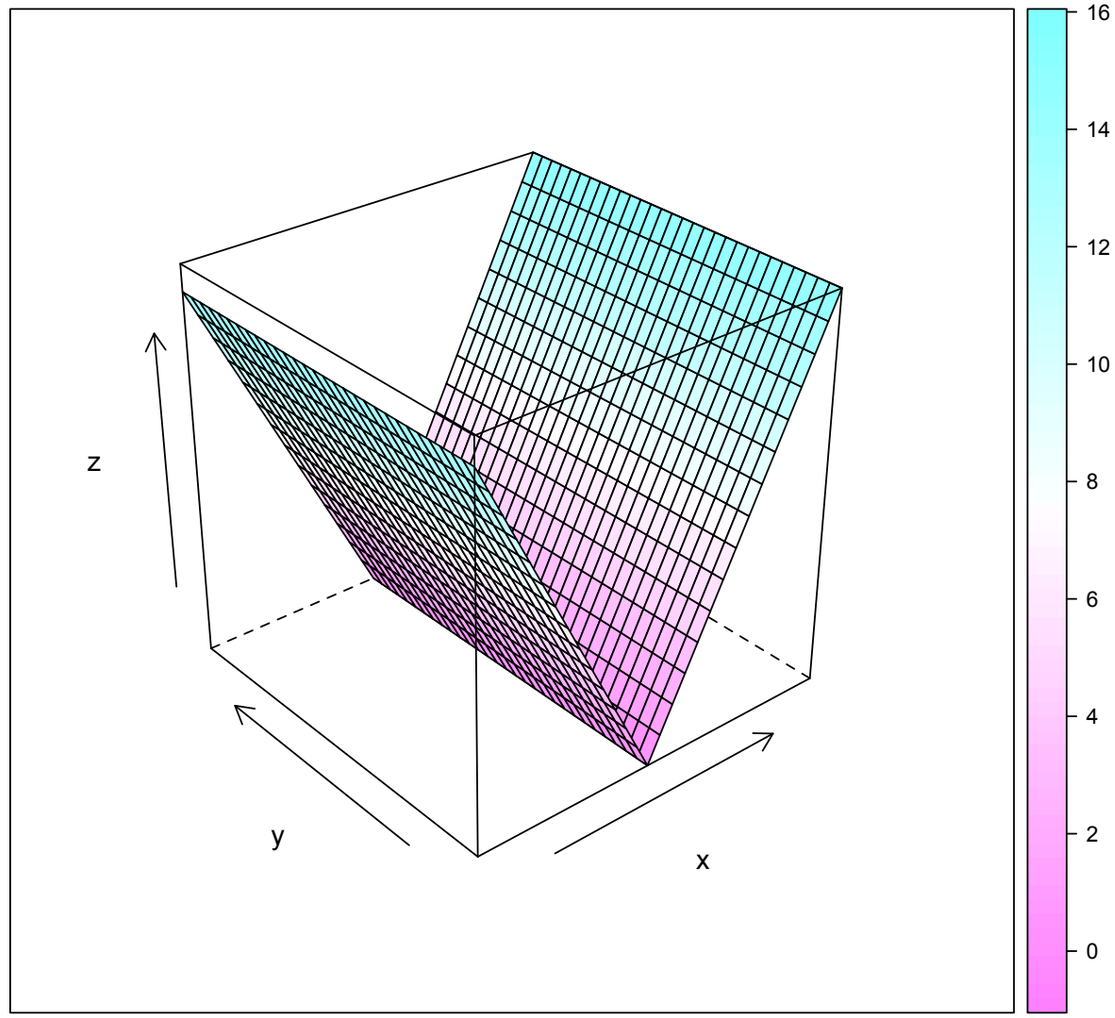


Foundation of Genetic Programmin

Ruggedness



Easy to get stuck
in one of many local optima



Smooth descent

Discrete Fitness Landscape

- In case of `(x, y) = (15, 15)`, it is (relatively) obvious what the neighbouring solutions are.
  - `(14, 15), (16, 15), (15, 14), (15, 16)`
  - `(16, 16), (14, 14), (16, 14), (14, 16)`

- What if we are searching for non-numeric solution?
  - Set membership (e.g. Do I include this requirement or not?, Do I execute this test case or not?)
  - Permutations (e.g. In which order should I execute this test suite?)
  - Highly structured data (e.g. To test this compiler, which program should I use as input?)