Anthony Ventresque

anthony.ventresque@ucd.ie

Data Structures and Algorithms

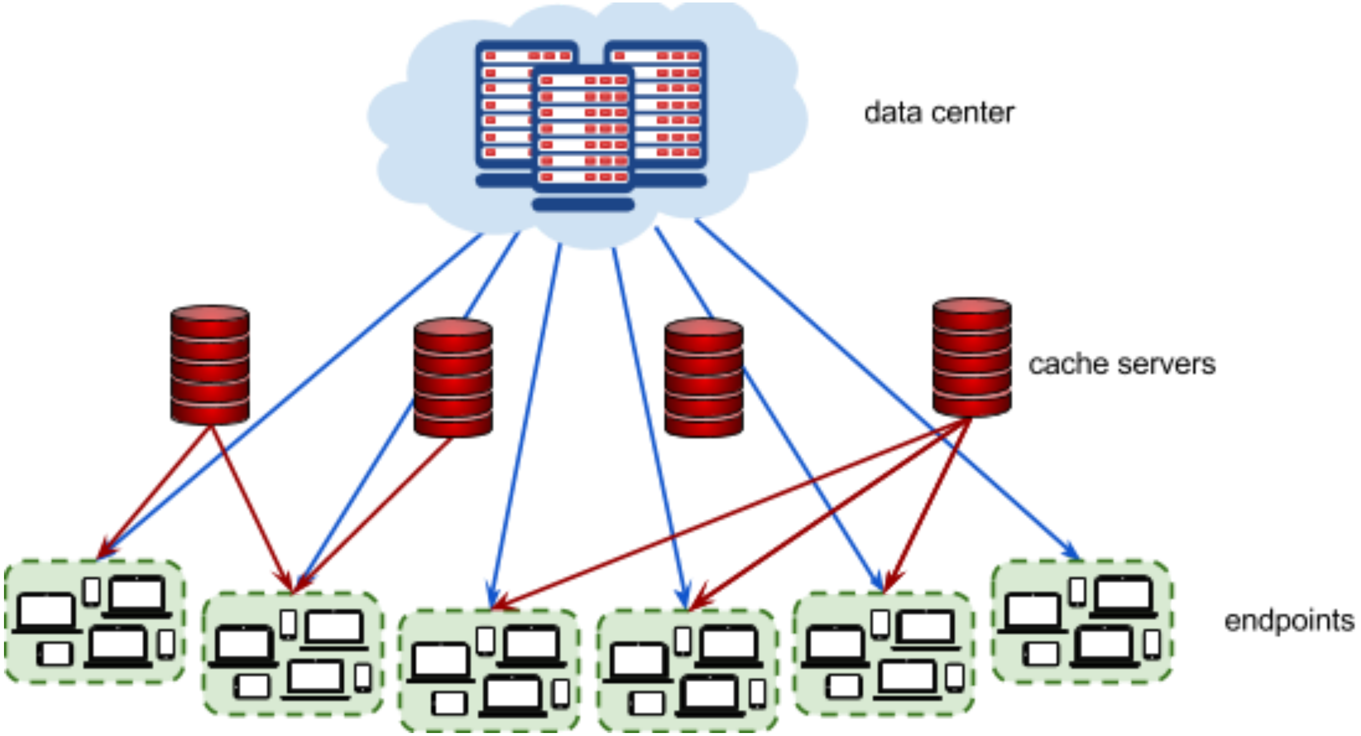
COMP20230

Project 2: Towards an Optimisation Tool

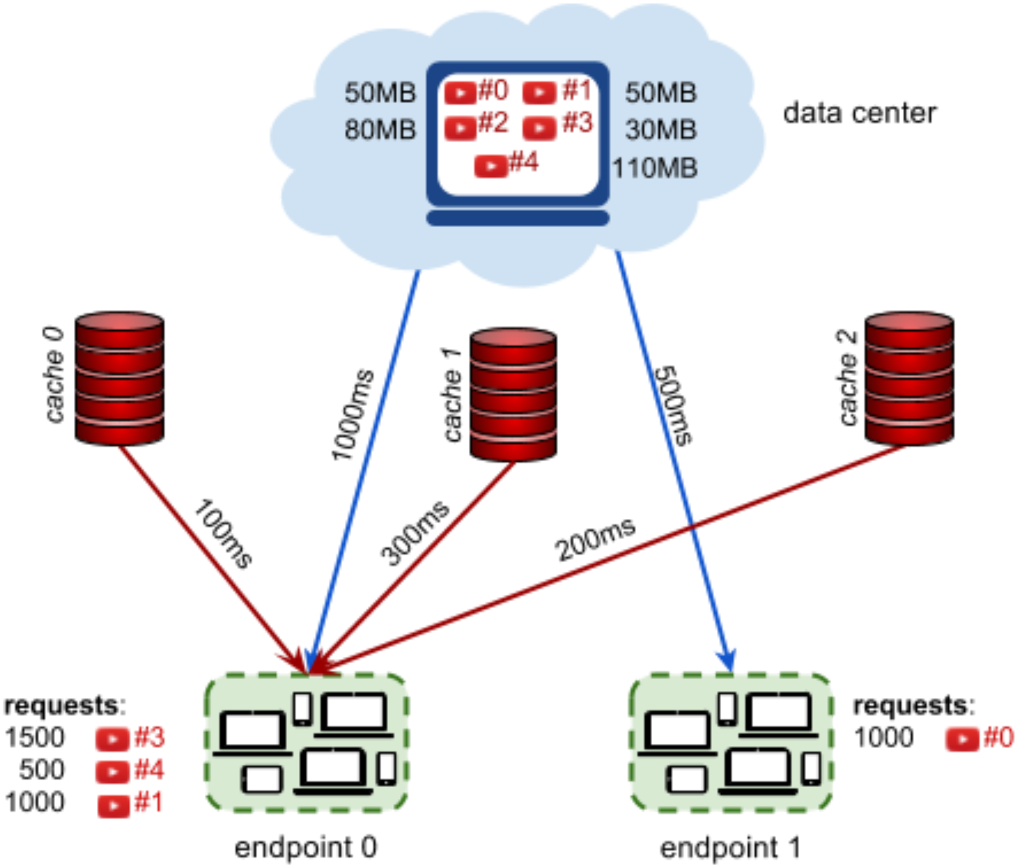
School of Computer Science, UCD Scoil na Ríomheolaíochta, UCD

GOOGLE HASH CODE 2017

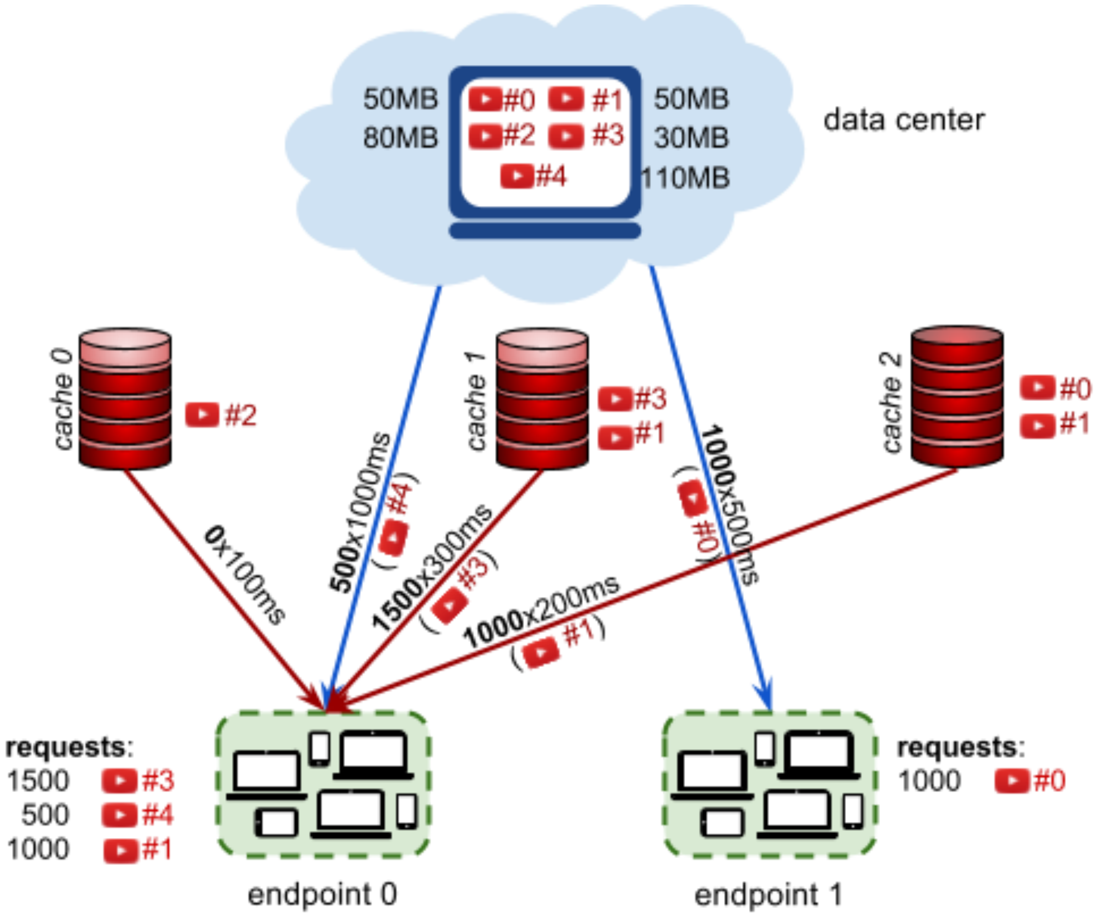
Problem Description



Problem Description (2)



Solutions


50MB
80MB
#0
#1
#2
#3
#4
50MB
30MB
110MB
data center
cache 0
#2
cache 1
#3
#1
cache 2
#0
#1
0x100ms
500x1000ms
(#4)
1500x300ms
(#3)
1000x200ms
(#1)
1000x500ms
(#0)
requests:
1500 #3
500 #4
1000 #1
endpoint 0
endpoint 1
requests:
1000 #0

Constraints

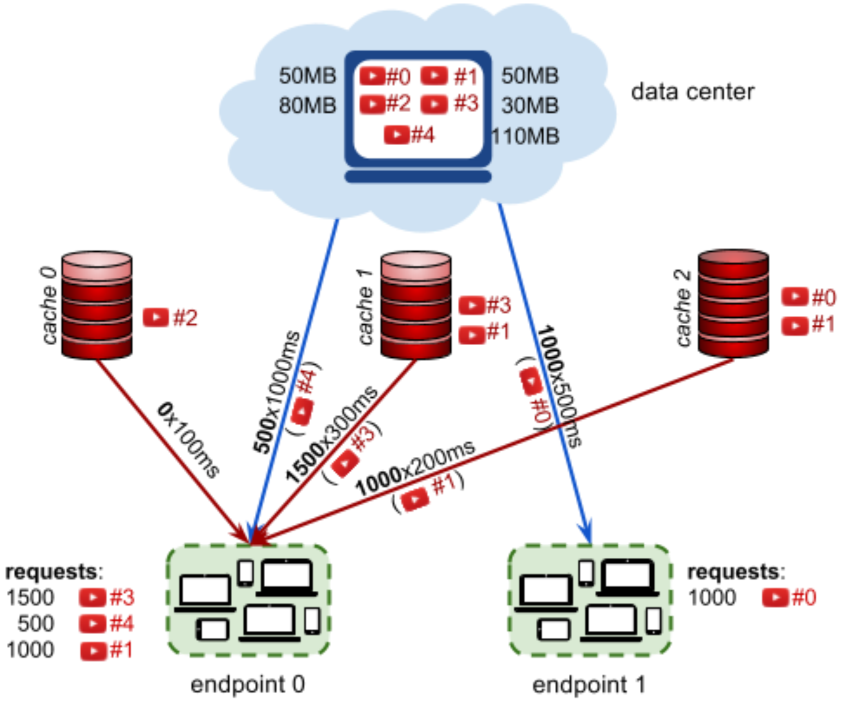
- Sum of all videos stored in a cache server ≤ capacity of the cache server

Scoring Solutions

- Scoring function
  =“cost function”
  =“utility function”
  =“fitness function”

```
score = 0
for each request q for a file f at an endpoint e
    score += number of videos in the request *
        (latency from data centre – best latency
        from a cache server serving e and hosting f)
endfor
score = score/number of requests
score *= 1000
```

Scoring Solutions



1500 x video 3 from cache 1=

1500 x (1000 - 300)

500 x video 4 from data centre=

500 x (1000 – 0)

1000 x video 1 from cache 2 =

1000 x (1000 – 200)

1000 x video 0 from data centre =

1000 x (500 – 0)

%(1500 + 500 + 1000 + 1000)

462.5 ms on average

462500 points

TOWARDS AN OPTIMISATION TOOL

Components of an Optimisation Tool

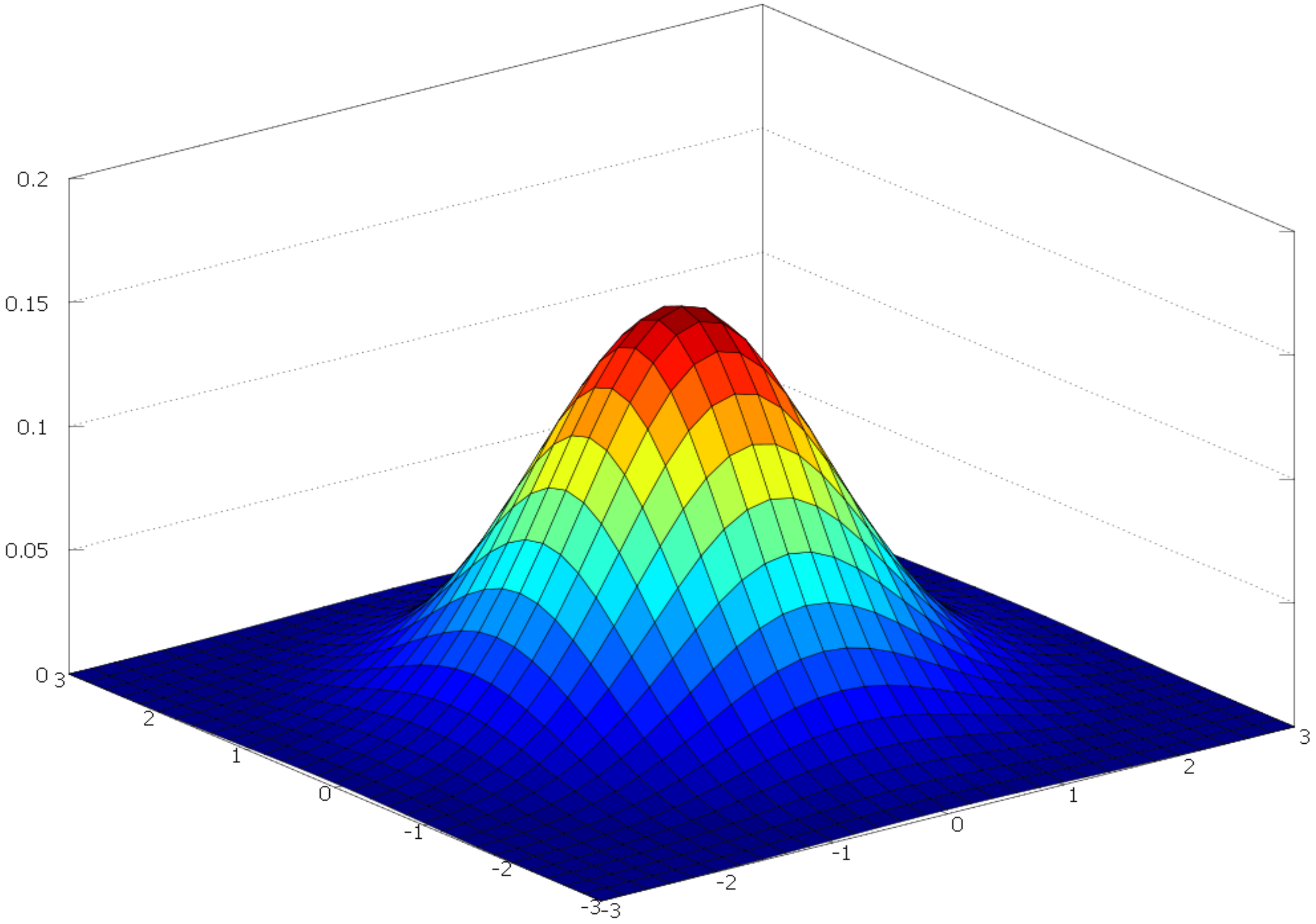
Search Algorithm

Representation

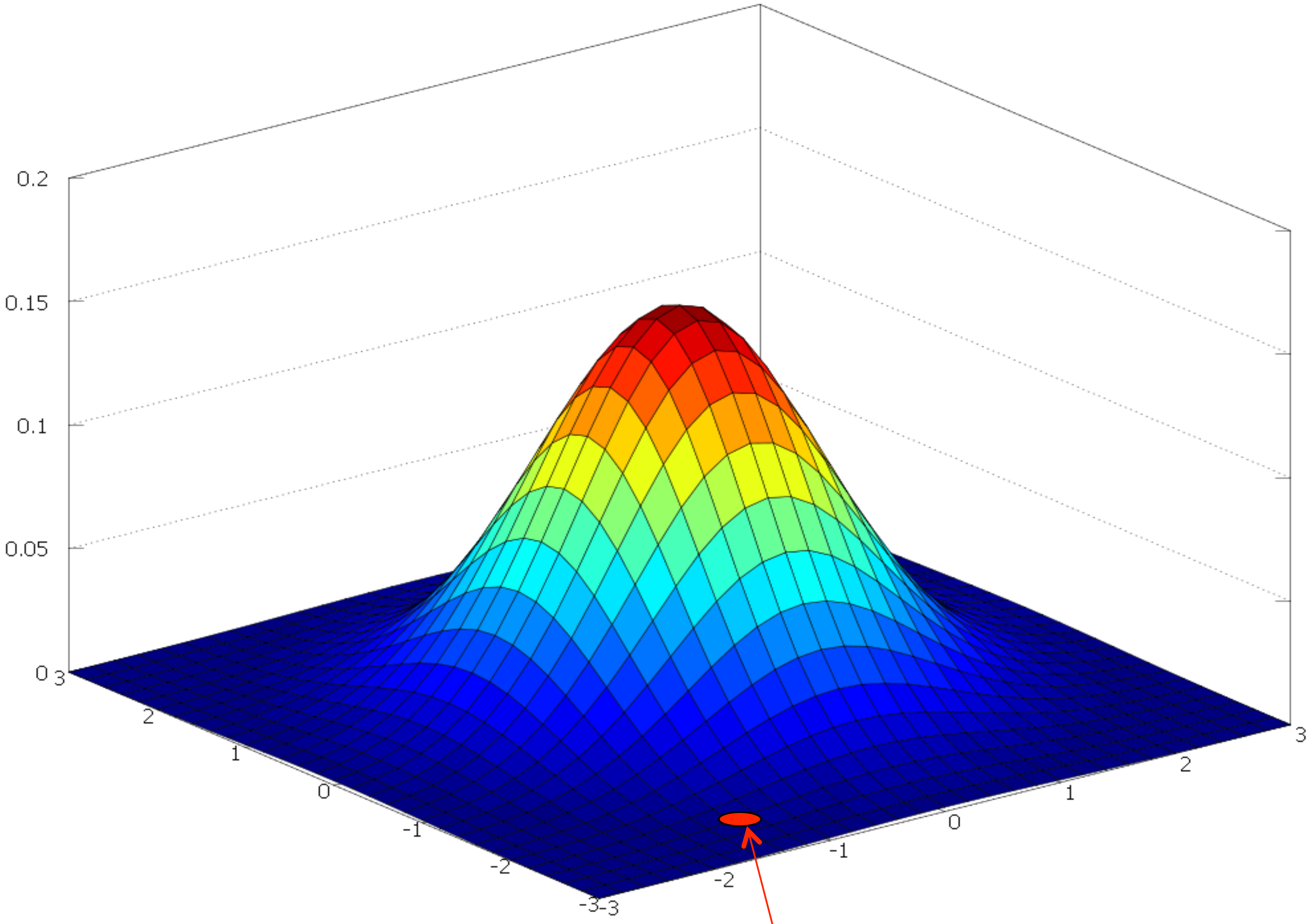
Search Operators

Fitness Function

Hill Climbing

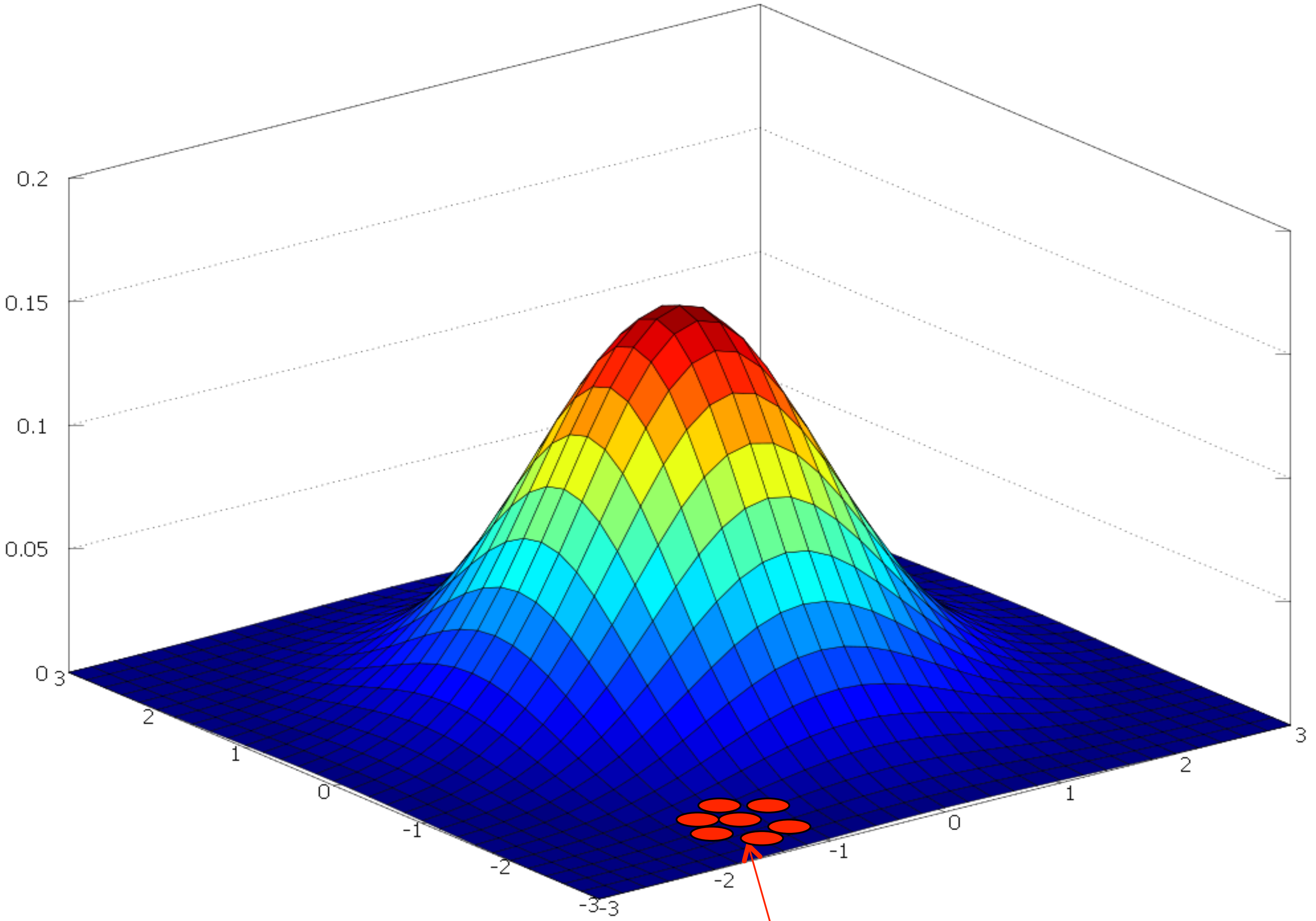


Hill Climbing



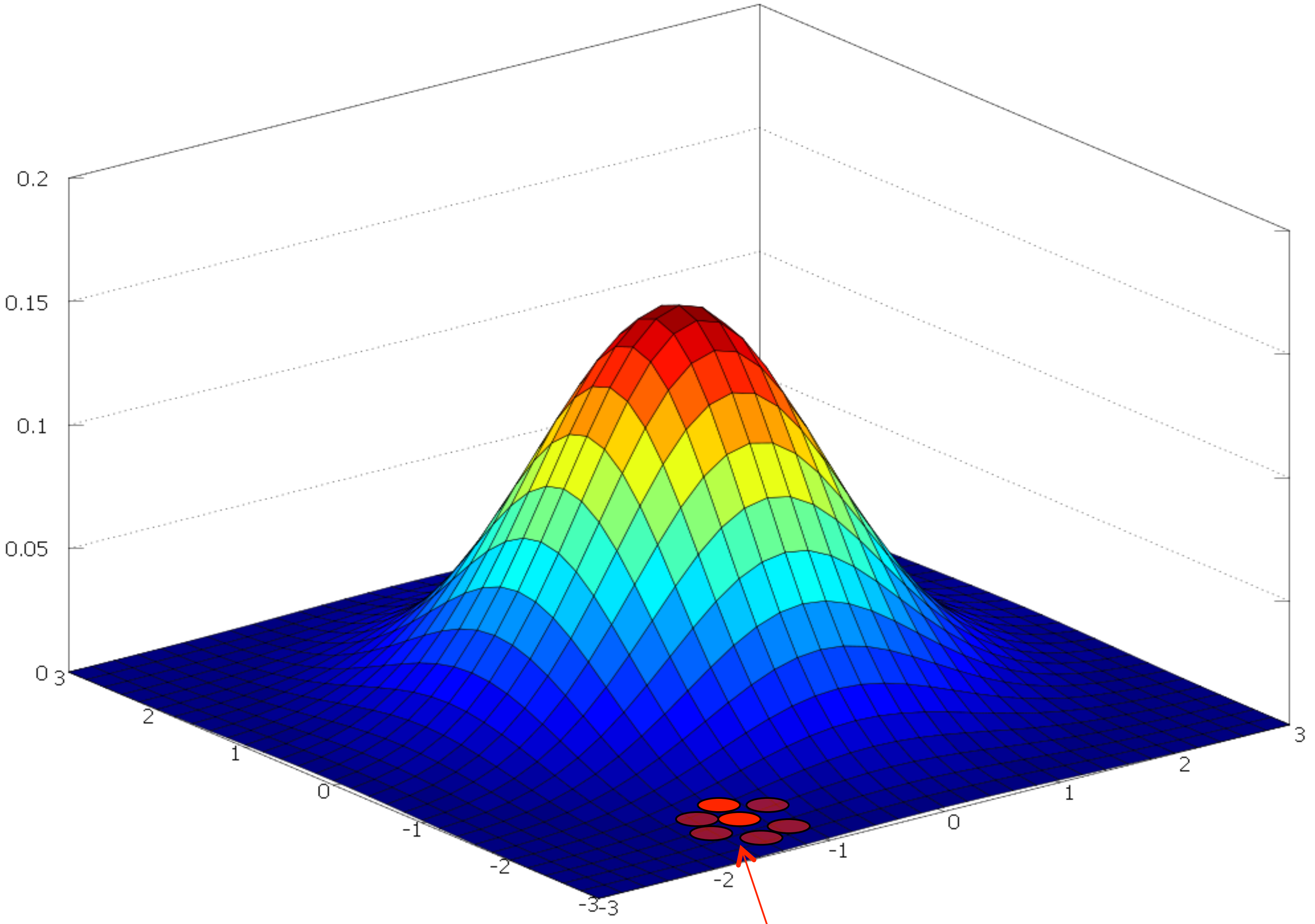
Select random value

Hill Climbing



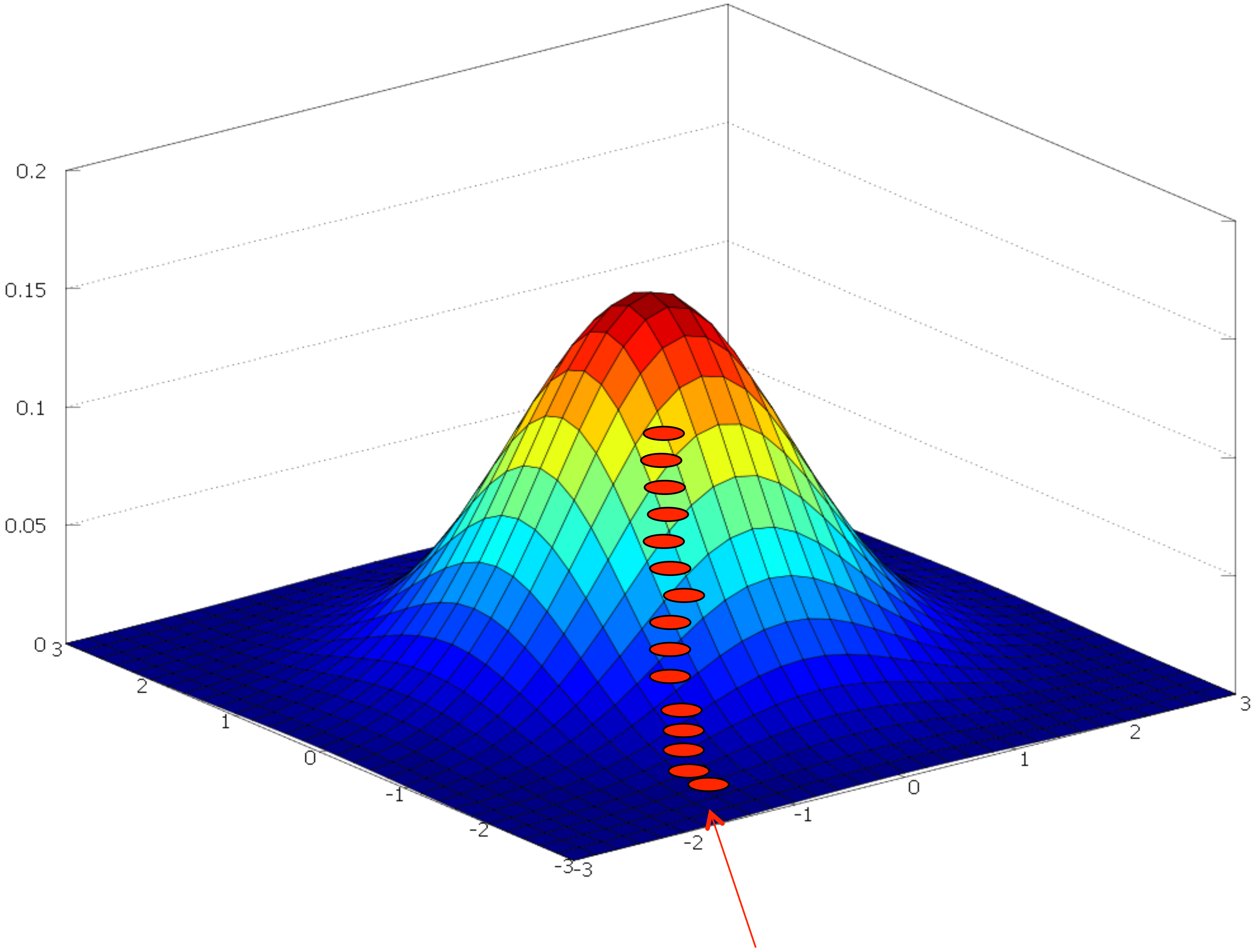
Explore neighbourhood

Hill Climbing



Choose better neighbour

Hill Climbing



Repeat

Components of an Optimisation Tool

Search Algorithm

Hill climbing

Representation

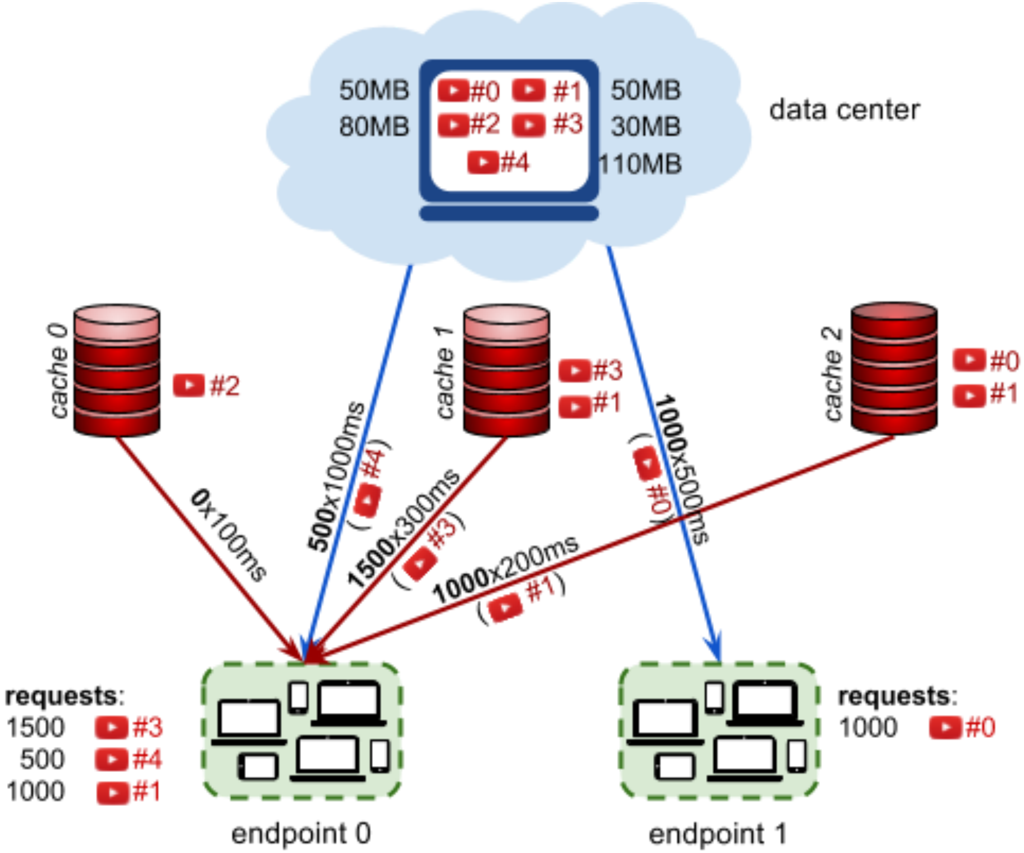
Search Operators

Fitness Function

Representation

cache0 - 2
cache1 - 1,3
cache2 – 0,1

((2),(1,3),(0,1))



A Solution

cache0 - 2
cache1 - 1,3
cache2 – 0,1

↓

((2),(1,3),(0,1)

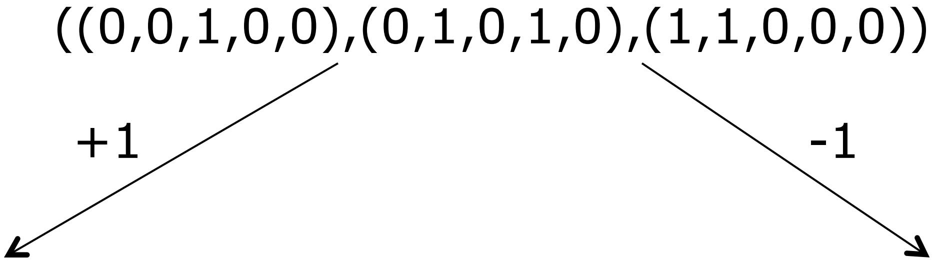
or

((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0))

Components of an Optimisation Tool

Search Algorithm	Hill climbing
Representation	2D list
Search Operators	
Fitness Function	

Search



((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0))

+1

-1

((***1***,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0))

((0,***1***,1,0,0),(0,1,0,1,0),(1,1,0,0,0))

((0,0,1,***1***,0),(0,1,0,1,0),(1,1,0,0,0))

((0,0,1,0,***1***),(0,1,0,1,0),(1,1,0,0,0))

((0,0,1,0,0),(***1***,1,0,1,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,***1***,1,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,***1***),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,0),(1,1,***1***,0,0))

((0,0,1,0,0),(0,1,0,1,0),(1,1,0,***1***,0))

((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,***1***))

((0,0,***0***,0,0),(0,1,0,1,0),(1,1,0,0,0))

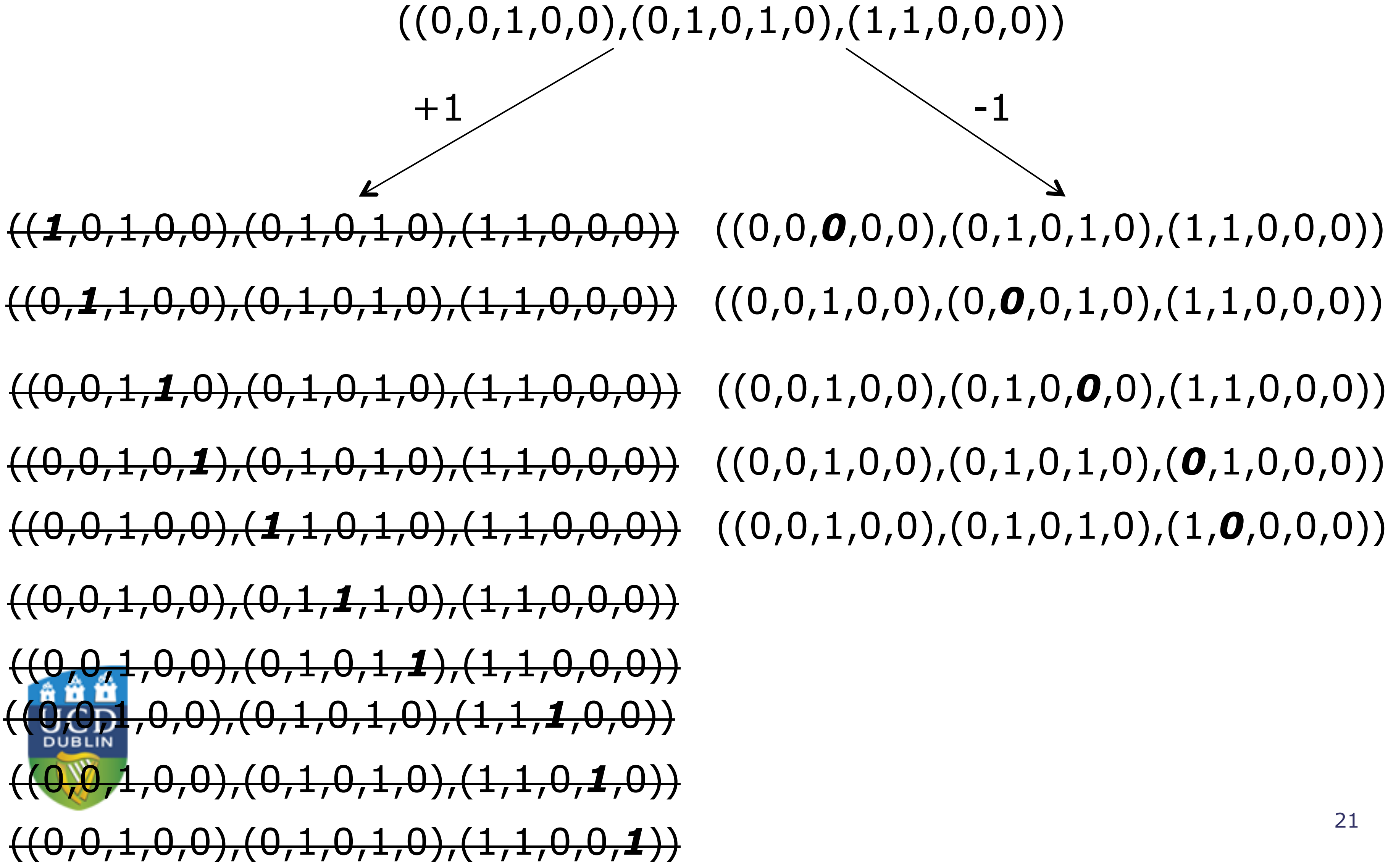
((0,0,1,0,0),(0,***0***,0,1,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,***0***,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,0),(***0***,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,0),(1,***0***,0,0,0))

Search



((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0))

+1

-1

~~((***1***,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0))~~

~~((0,***1***,1,0,0),(0,1,0,1,0),(1,1,0,0,0))~~

~~((0,0,1,***1***,0),(0,1,0,1,0),(1,1,0,0,0))~~

~~((0,0,1,0,***1***),(0,1,0,1,0),(1,1,0,0,0))~~

~~((0,0,1,0,0),(***1***,1,0,1,0),(1,1,0,0,0))~~

~~((0,0,1,0,0),(0,1,***1***,1,0),(1,1,0,0,0))~~

~~((0,0,1,0,0),(0,1,0,1,***1***),(1,1,0,0,0))~~

~~((0,0,1,0,0),(0,1,0,1,0),(1,1,***1***,0,0))~~

~~((0,0,1,0,0),(0,1,0,1,0),(1,1,0,***1***,0))~~

~~((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,***1***))~~

((0,0,***0***,0,0),(0,1,0,1,0),(1,1,0,0,0))

((0,0,1,0,0),(0,***0***,0,1,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,***0***,0),(1,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,0),(***0***,1,0,0,0))

((0,0,1,0,0),(0,1,0,1,0),(1,***0***,0,0,0))

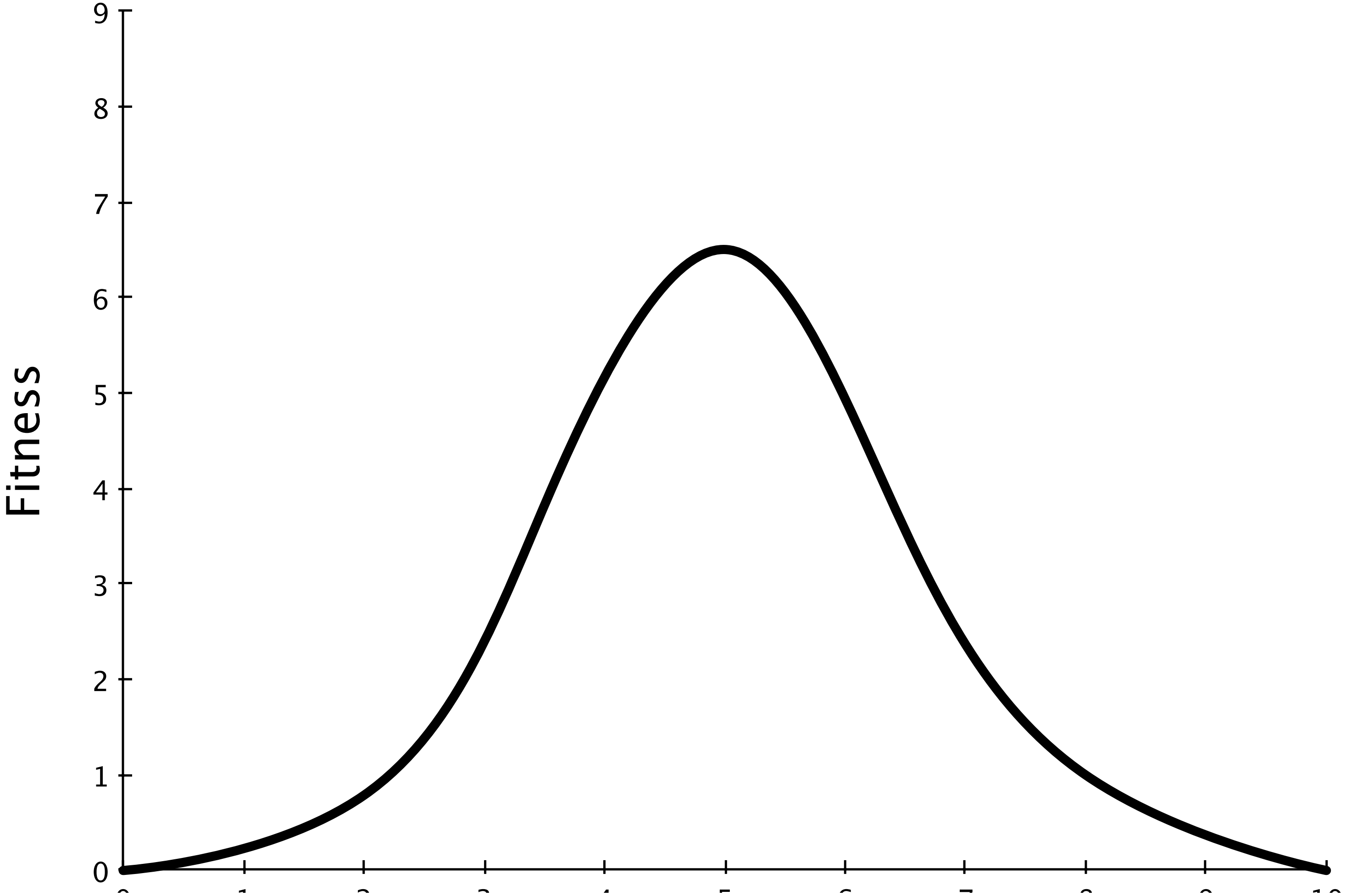
Components of an Optimisation Tool

Search Algorithm	Hill climbing
Representation	2D list
Search Operators	Neighbourhood of 2D list
Fitness Function	

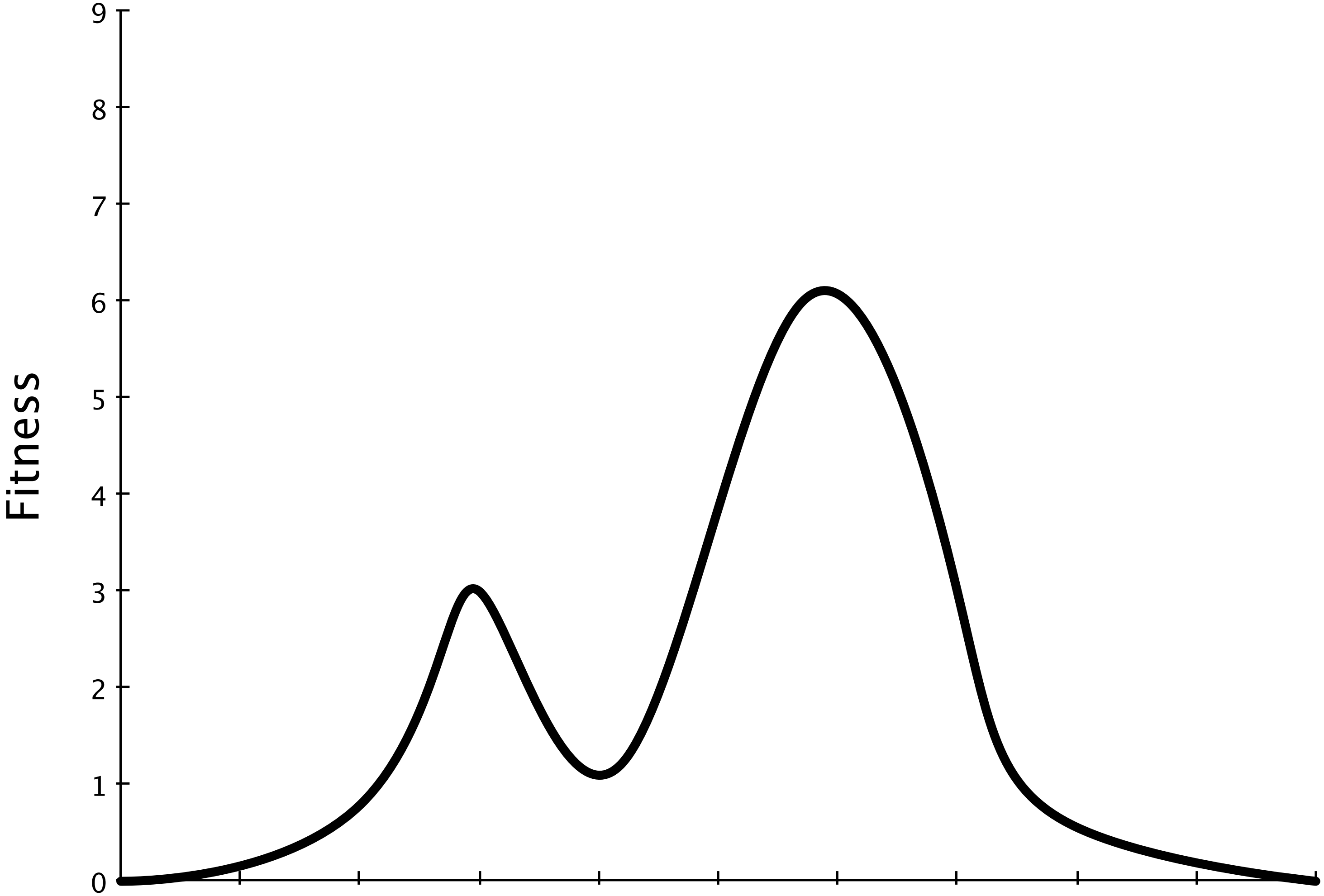
Components of an Optimisation Tool

Search Algorithm	Hill climbing
Representation	2D list
Search Operators	Neighbourhood of 2D list
Fitness Function	as given by Google

Problem with Hill-climbing



Problem with Hill-climbing



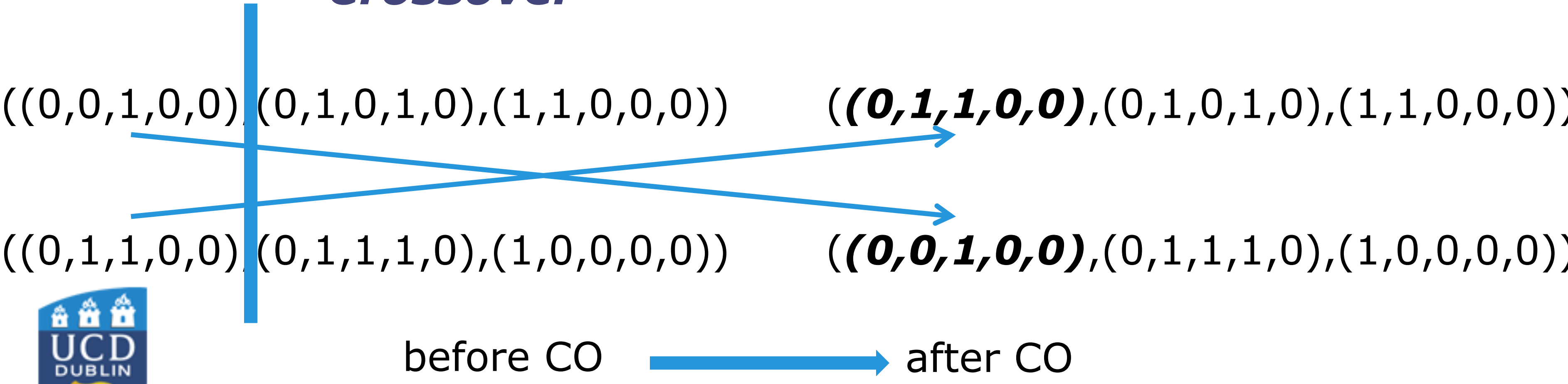
Evolutionary (Genetic) Algorithms

- Two basic operations:
  - ***Mutation***

((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0)) ((0,0,1,0,0),(0,1,***1***,1,0),(1,1,0,0,0))

before mutation → after mutation

  - ***Crossover***



((0,0,1,0,0),(0,1,0,1,0),(1,1,0,0,0)) (***(0,1,1,0,0)***,(0,1,0,1,0),(1,1,0,0,0)

((0,1,1,0,0),(0,1,1,1,0),(1,0,0,0,0)) (***(0,0,1,0,0)***,(0,1,1,1,0),(1,0,0,0,0)

before CO → after CO

Selection

- keep only the fittest individuals?
- or keep some “diversity”?
- how many individual solutions do we keep in a population?
- how often do we do crossover?
- how often do we mutate?