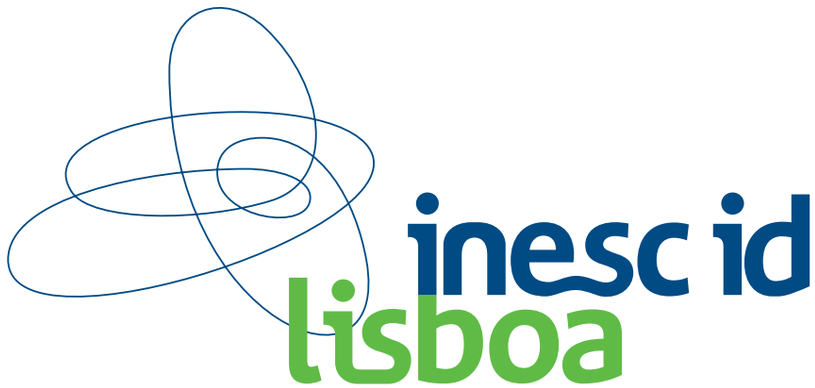
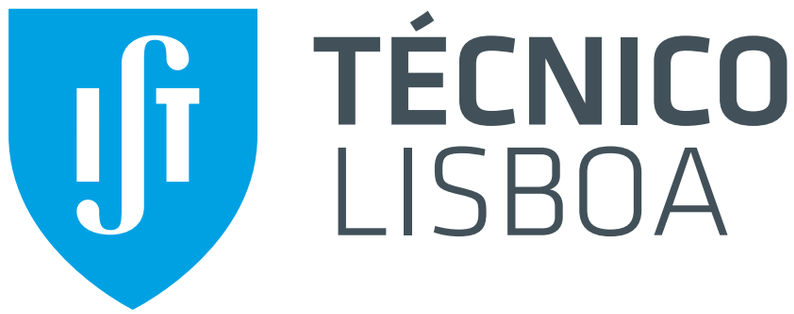
# Self-Tuning Intel TSX

3rd Euro-TM Workshop on Transactional Memory

**Nuno Diegues** and Paolo Romano



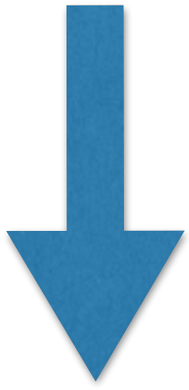

to appear on the 11th USENIX ICAC 2014

# Using TSX

```
_xbegin
// your transactional code

_xend
```

# Using TSX

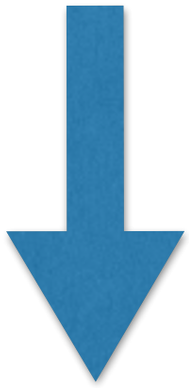
```
_xbegin
// your transactional code

_xend
```



**May Abort**

# Using TSX

```
_xbegin
// your transactional code
_xend
```



**May Abort**

- Data contention
- Forbidden instructions
- Hardware buffers' capacity
- Signals and faults

# Using TSX

```
_xbegin

// your transactional code

_xend
```

**May Abort**

**Transparently Restarts**

- Data contention
- Forbidden instructions
- Hardware buffers' capacity
- Signals and faults

# Using TSX

**Best-effort nature**
**we cannot rely exclusively on TSX**

# Best-effort nature

*Not *that* specific to Intel TSX. IBM HTMs apply partly here too*

# Best-effort nature

Not *that* specific to Intel TSX. IBM HTMs apply partly here too

```
begin:
unsigned int status = _xbegin
if (status == ok)
   goto code

goto begin
```

```
code:
// your transactional code


   _xend
```

# Best-effort nature

```
begin:
unsigned int status = _xbegin
if (status == ok)
   goto code          // fast path

goto begin
```

```
code:
// your transactional code

   _xend              // fast path
```

*Not *that* specific to Intel TSX. IBM HTMs apply partly here too*

# Best-effort nature

```
begin:
unsigned int status = _xbegin
if (status == ok)
   goto code          // fast path
if (shouldRetry) // retry policy
   goto begin
```

```
code:
// your transactional code

if (shouldRetry)
   _xend              // fast path
```

*Not *that* specific to Intel TSX. IBM HTMs apply partly here too*

# Best-effort nature

```
begin:
unsigned int status = _xbegin
if (status == ok)
   goto code          // fast path
if (shouldRetry) // retry policy
   goto begin
else
   acquire(lock) // fallback

code:
// your transactional code

if (shouldRetry)
   _xend              // fast path
else
   release(lock) // fallback
```

*Not *that* specific to Intel TSX. IBM HTMs apply partly here too*

# Best-effort nature

```
begin:
unsigned int status = _xbegin
if (status == ok)
   goto code          // fast path
if (shouldRetry) // retry policy
   goto begin
else
   acquire(lock) // fallback

code:
// your transactional code

if (shouldRetry)
   _xend              // fast path
else
   release(lock) // fallback
```

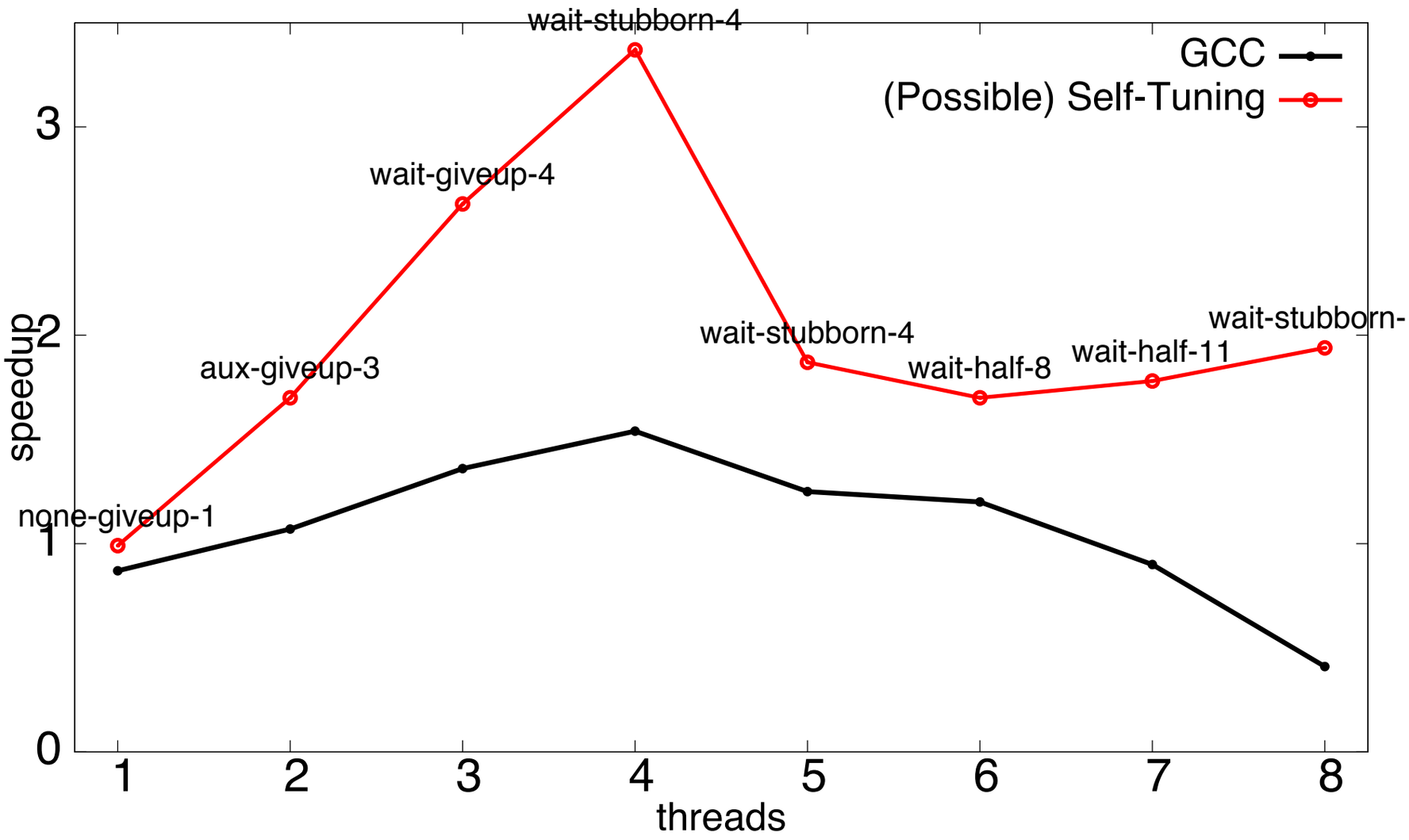
Not *that* specific to Intel TSX. IBM HTMs apply partly here too

Transactions need to be aware of this

# Summary of issues

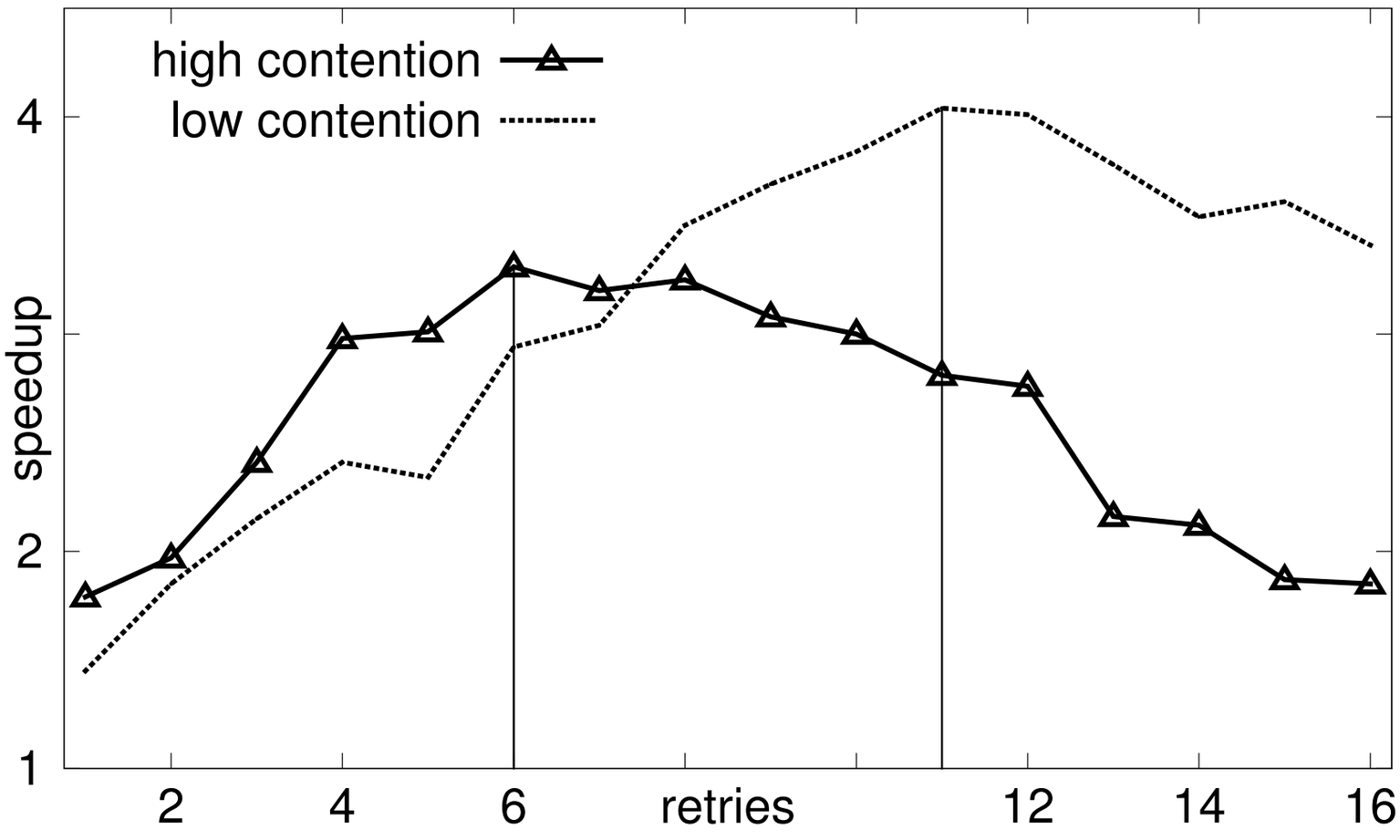
- Lemming effect
- Number of attempts
- Retry policy
- Management of fall-back

# Summary of issues

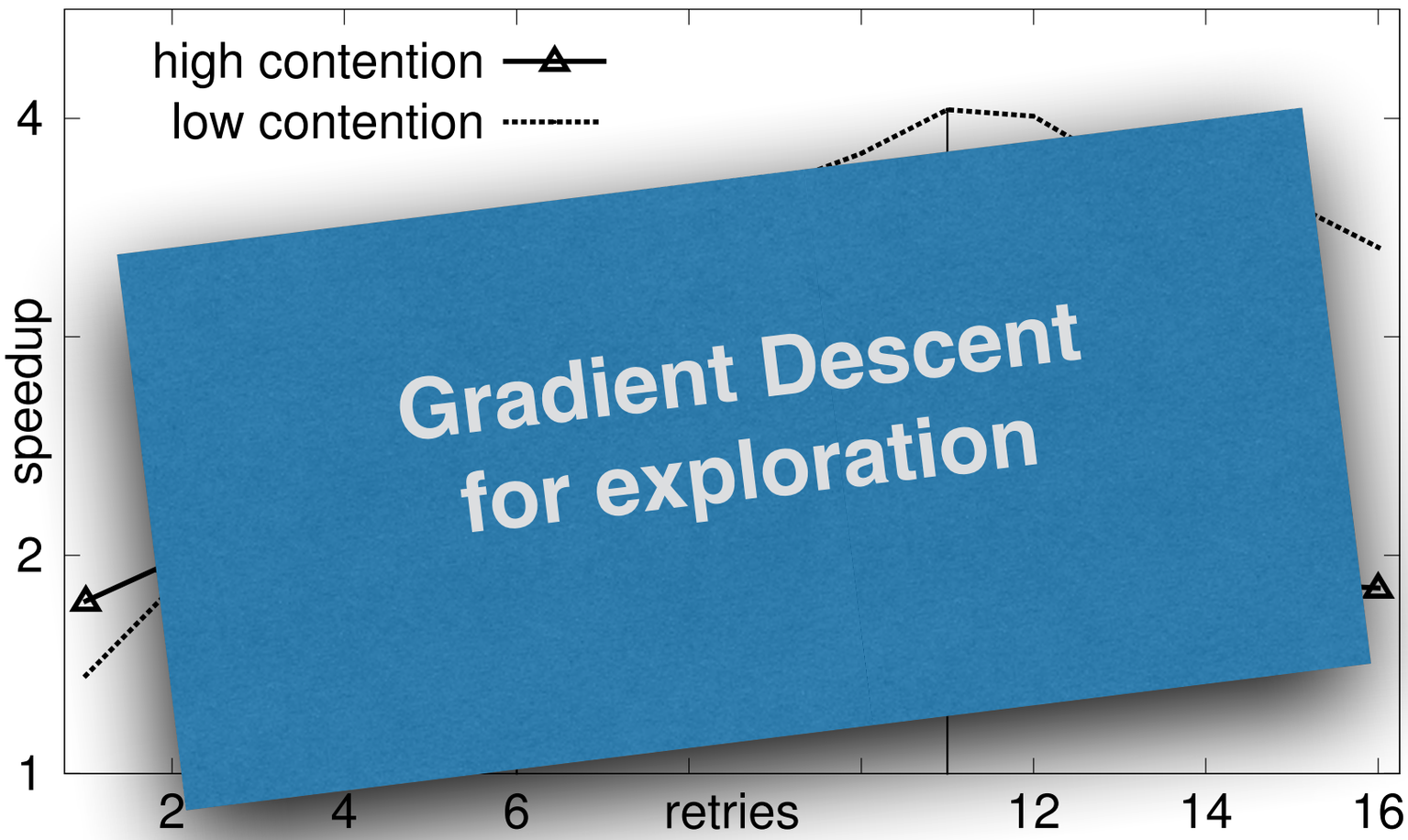



Genome from STAMP suite

# Number of attempts

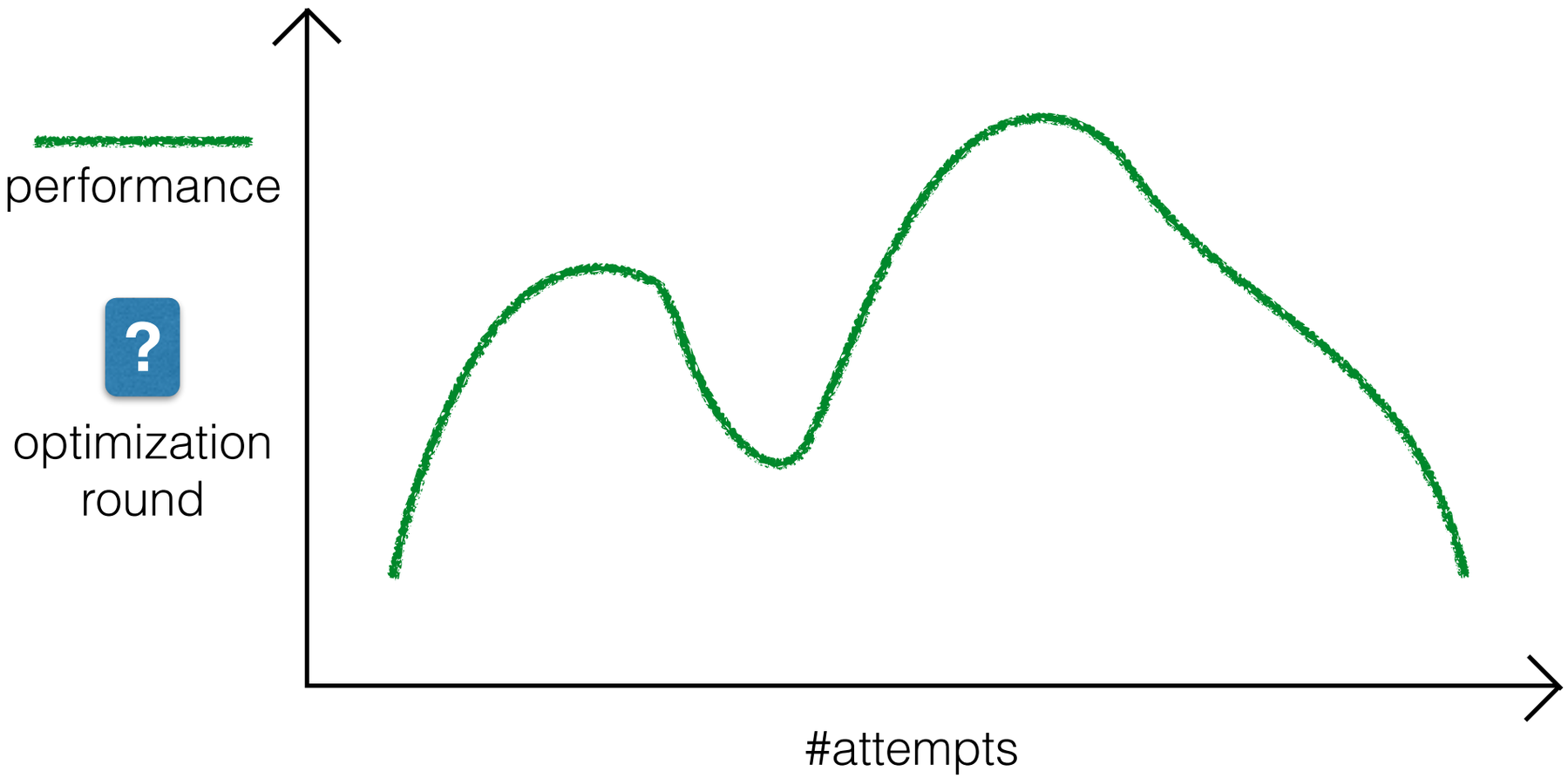


# Number of attempts



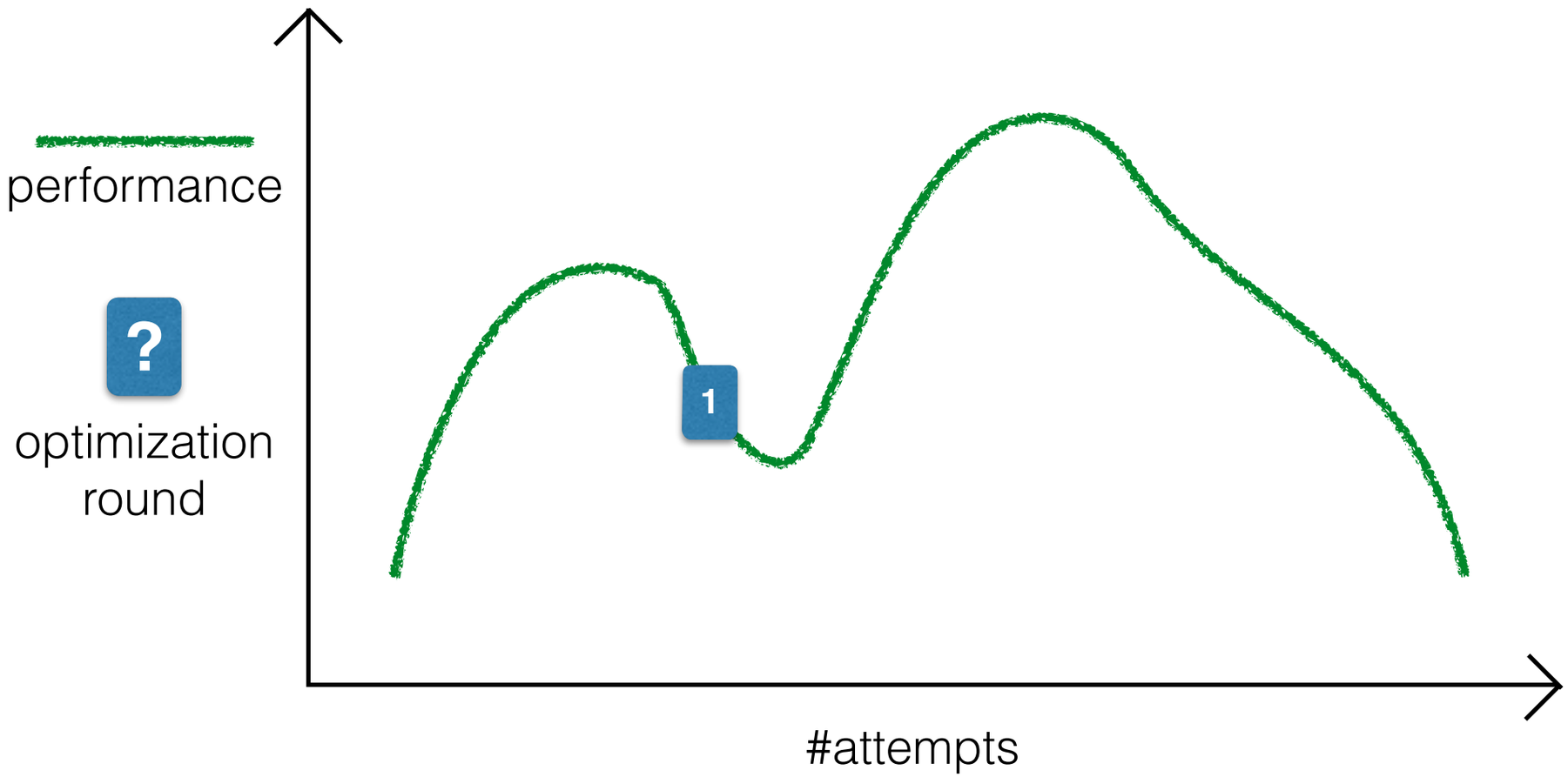
# Gradient Descent

## tuning the number of attempts



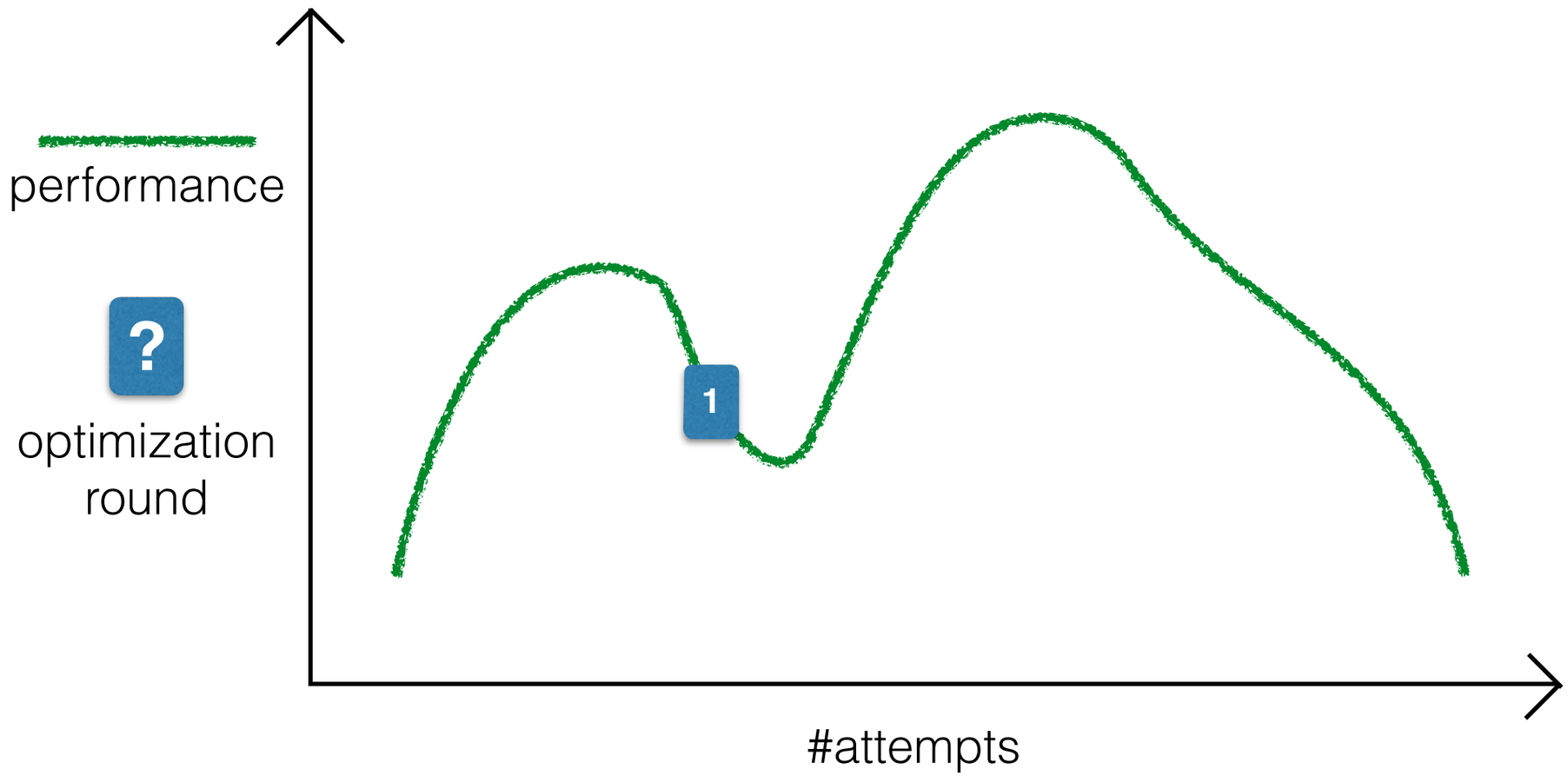
# Gradient Descent

## tuning the number of attempts



# Gradient Descent

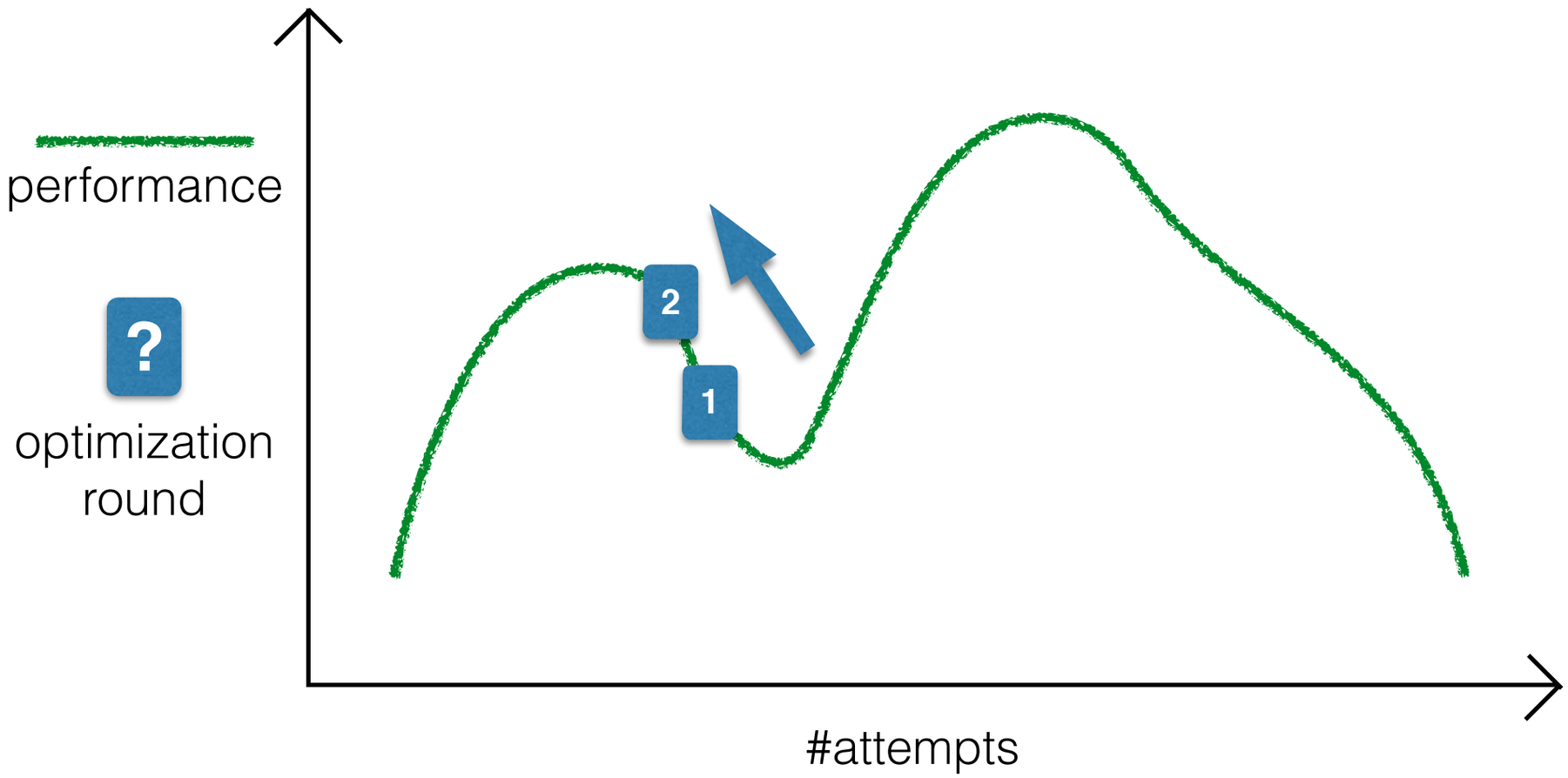
## tuning the number of attempts




randomly search some direction; explore it while profitable

# Gradient Descent

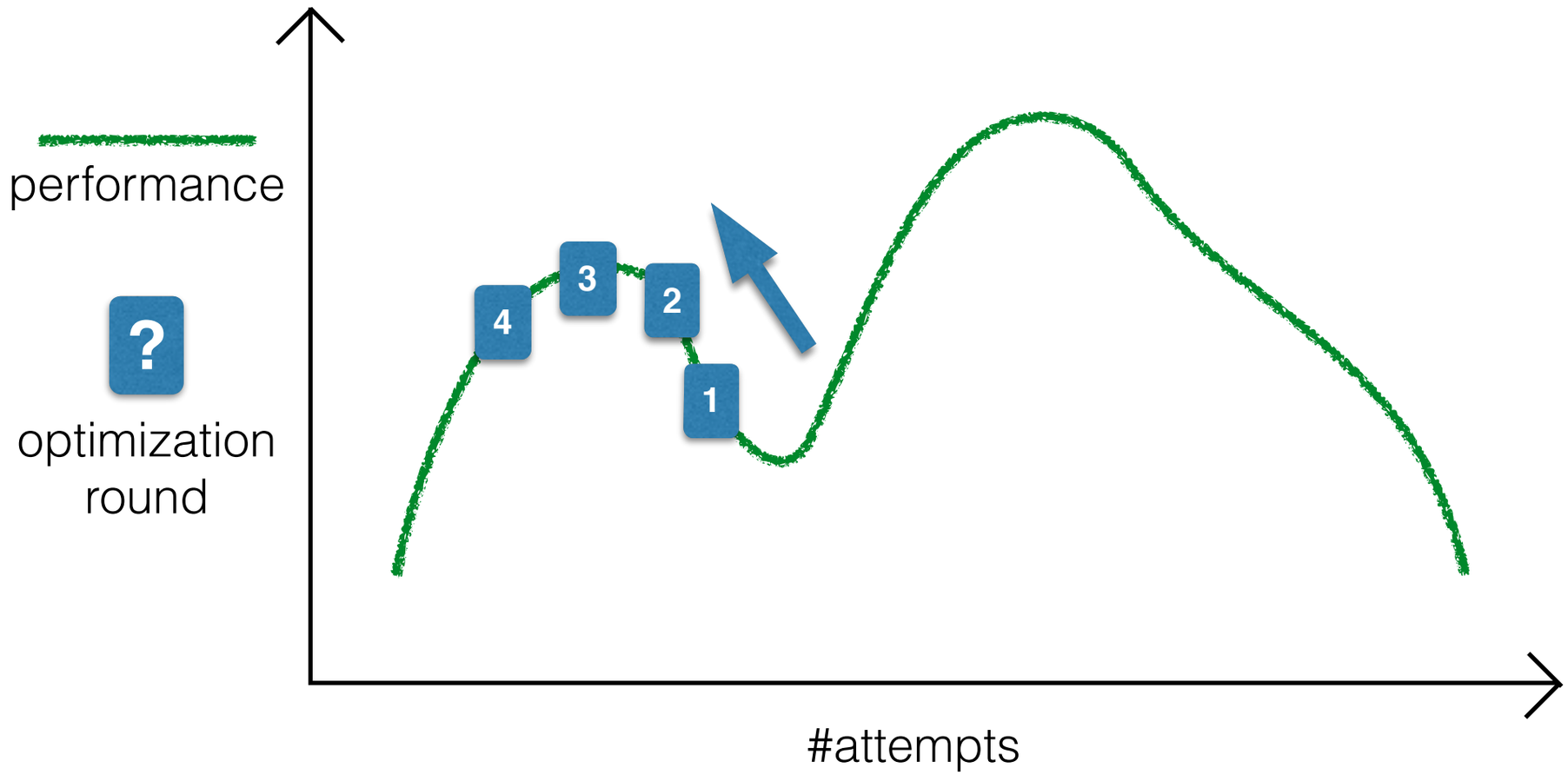
## tuning the number of attempts




randomly search some direction; explore it while profitable

# Gradient Descent

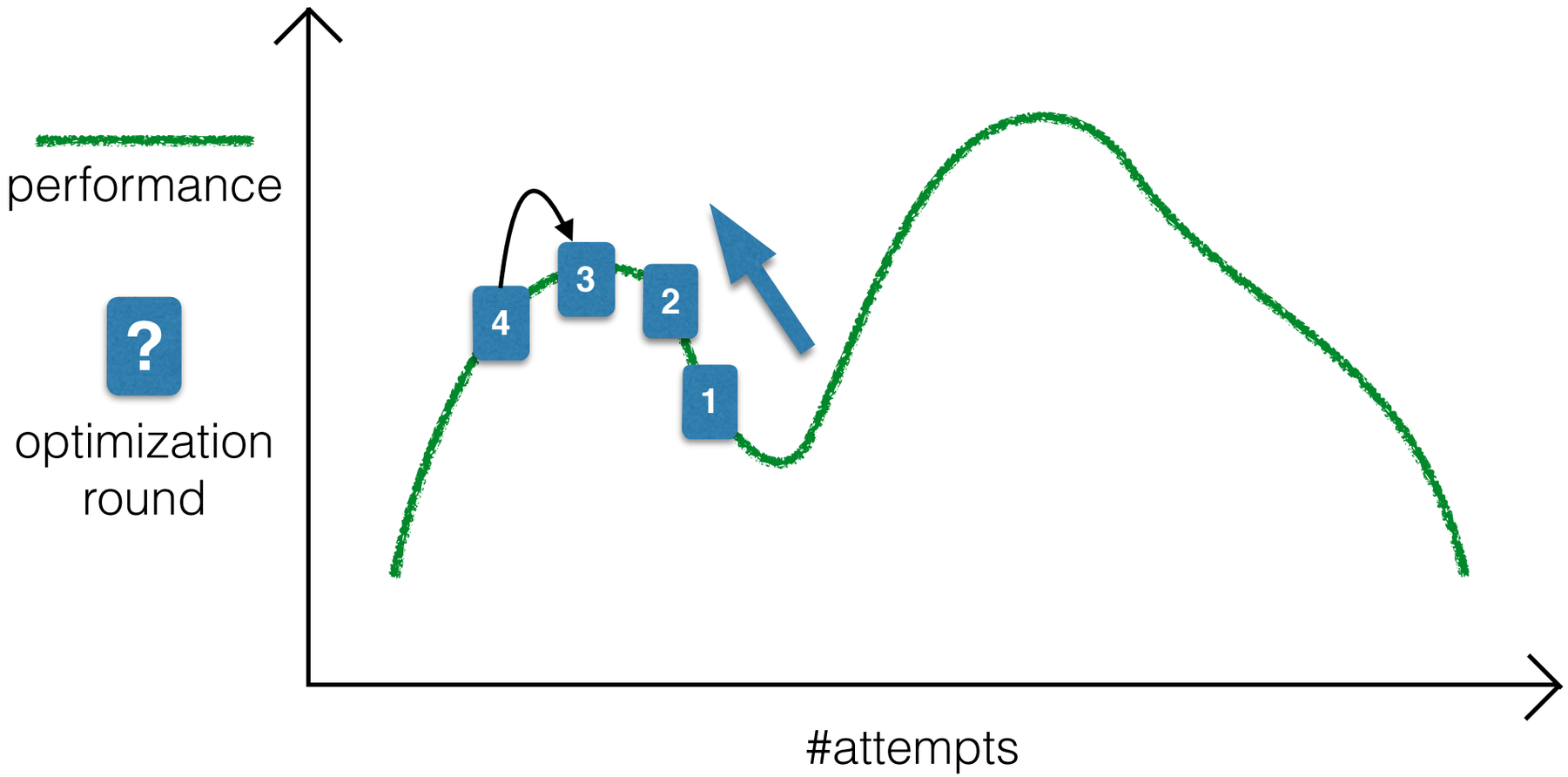
## tuning the number of attempts




randomly search some direction; explore it while profitable

# Gradient Descent

## tuning the number of attempts

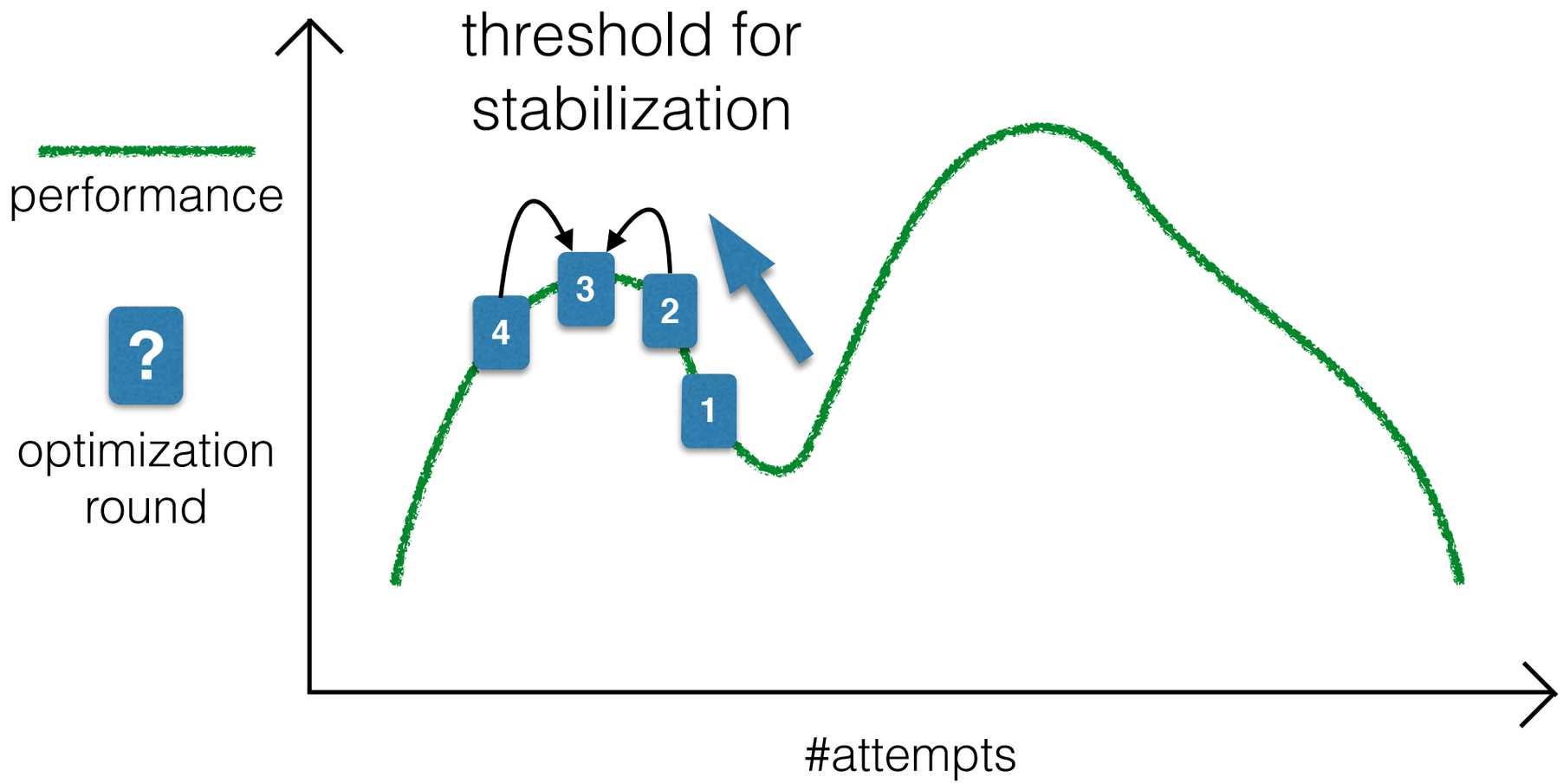



randomly search some direction; explore it while profitable

revert direction when not profitable

# Gradient Descent

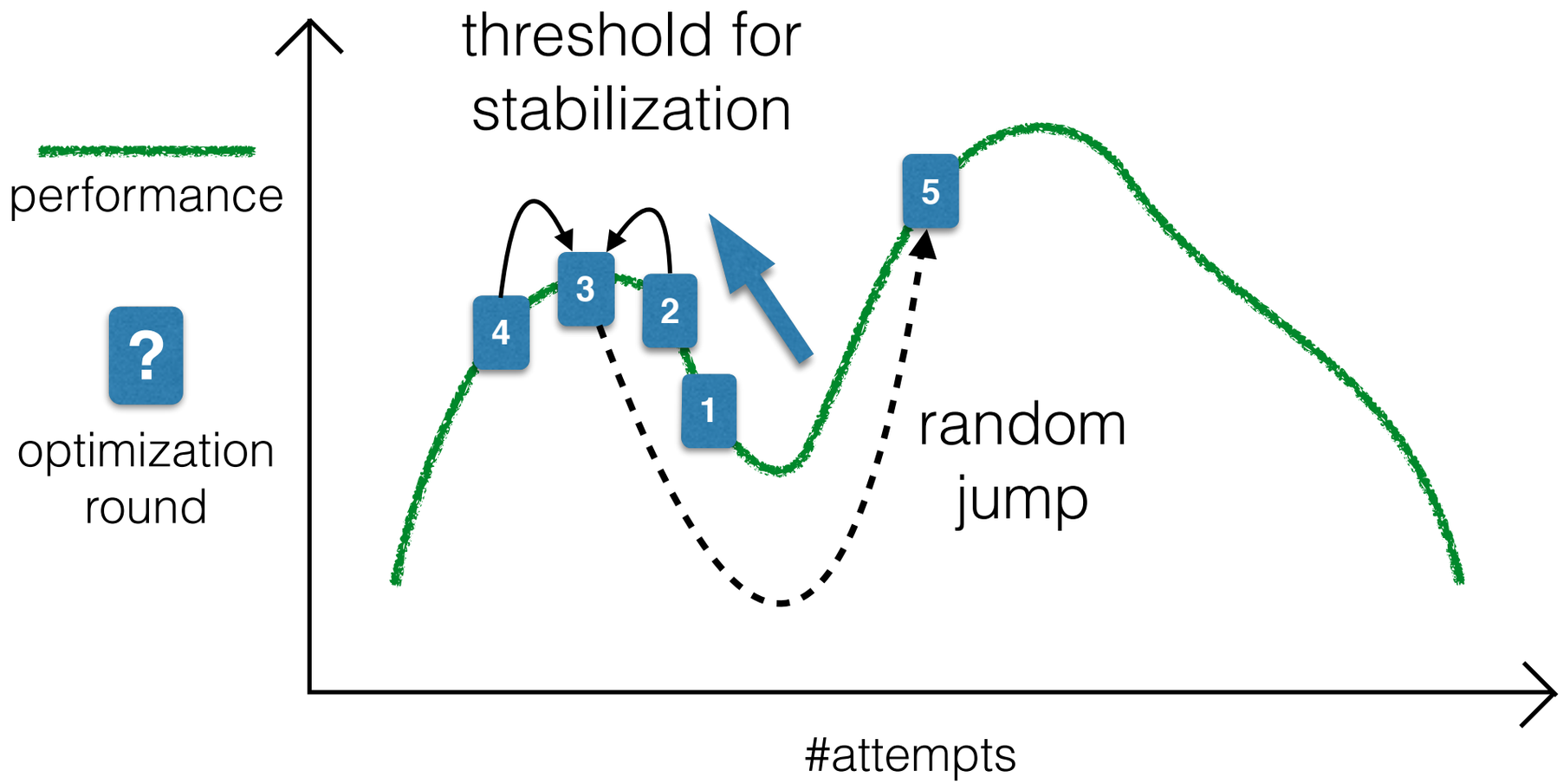
## tuning the number of attempts




randomly search some direction; explore it while profitable
revert direction when not profitable

# Gradient Descent

## tuning the number of attempts



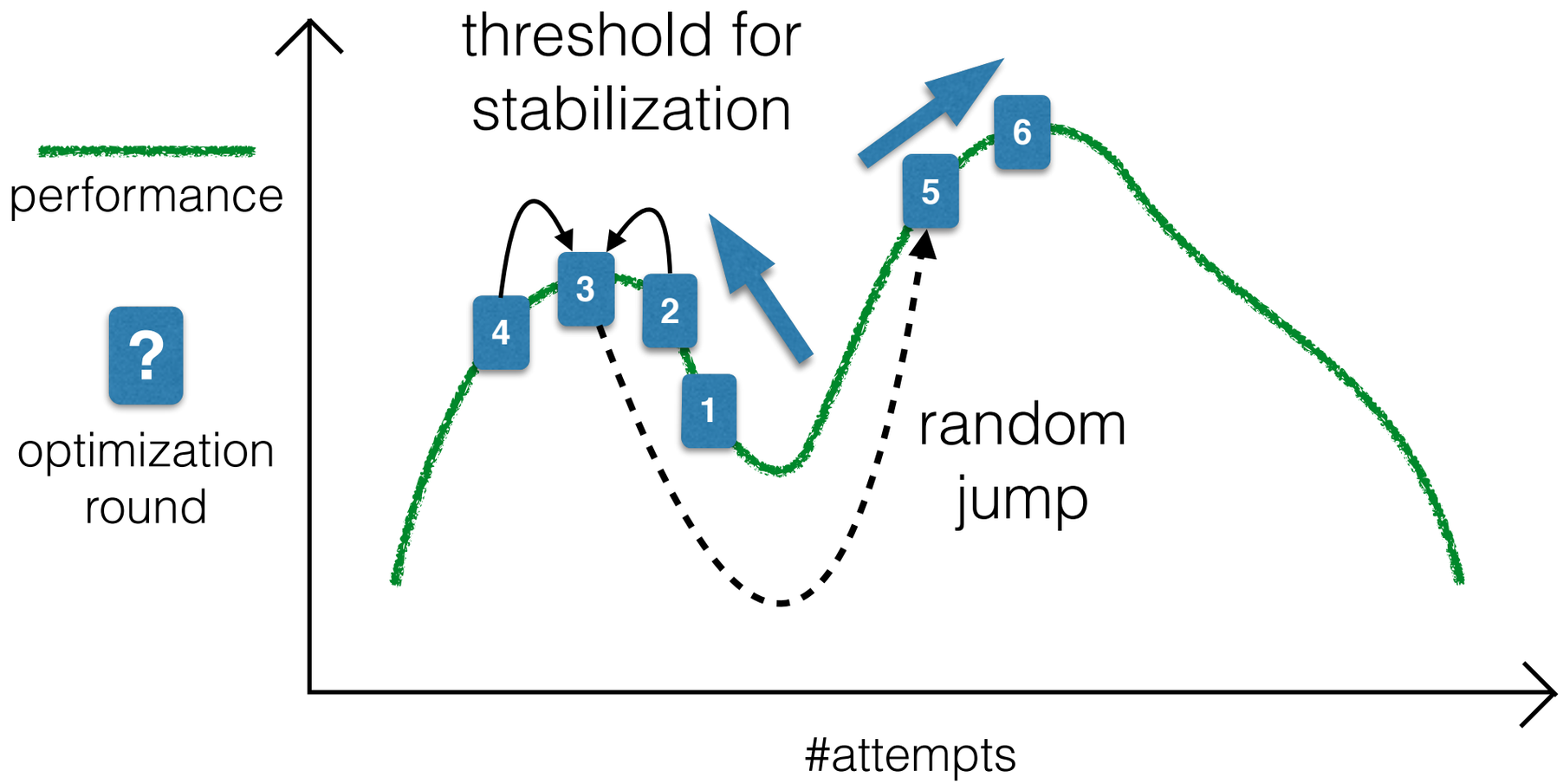

randomly search some direction; explore it while profitable

revert direction when not profitable

random jumps to avoid local minima

# Gradient Descent

## tuning the number of attempts



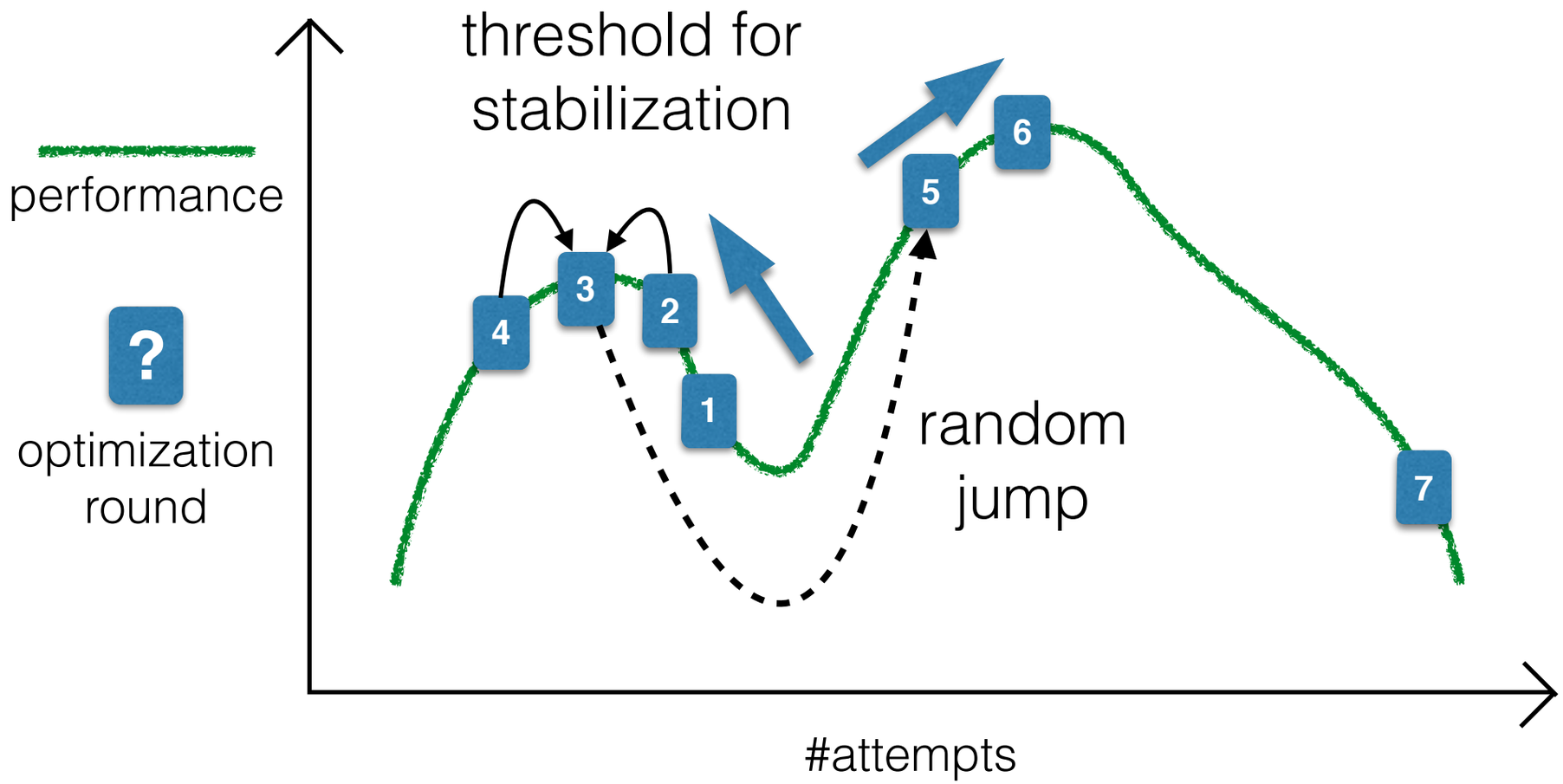

randomly search some direction; explore it while profitable

revert direction when not profitable

random jumps to avoid local minima

# Gradient Descent

## tuning the number of attempts



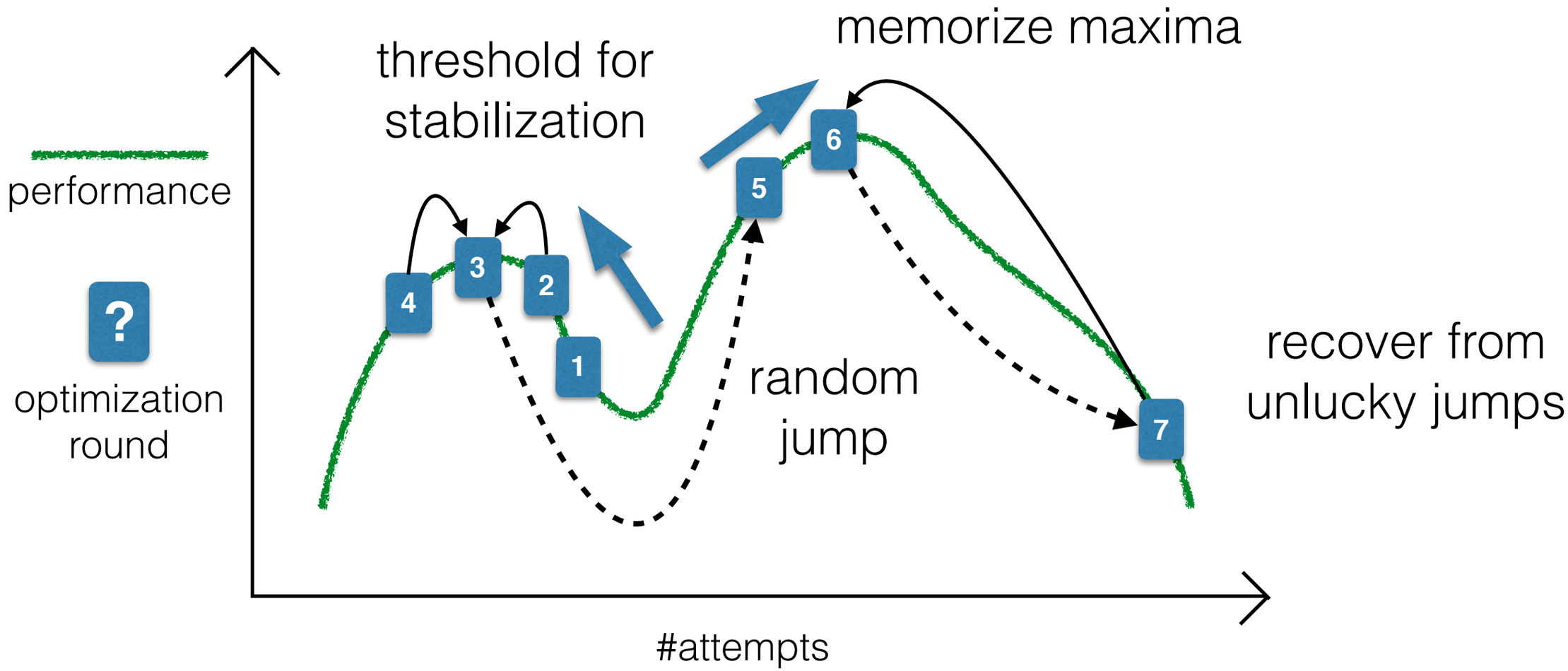

randomly search some direction; explore it while profitable

revert direction when not profitable

random jumps to avoid local minima

# Gradient Descent

## tuning the number of attempts




randomly search some direction; explore it while profitable

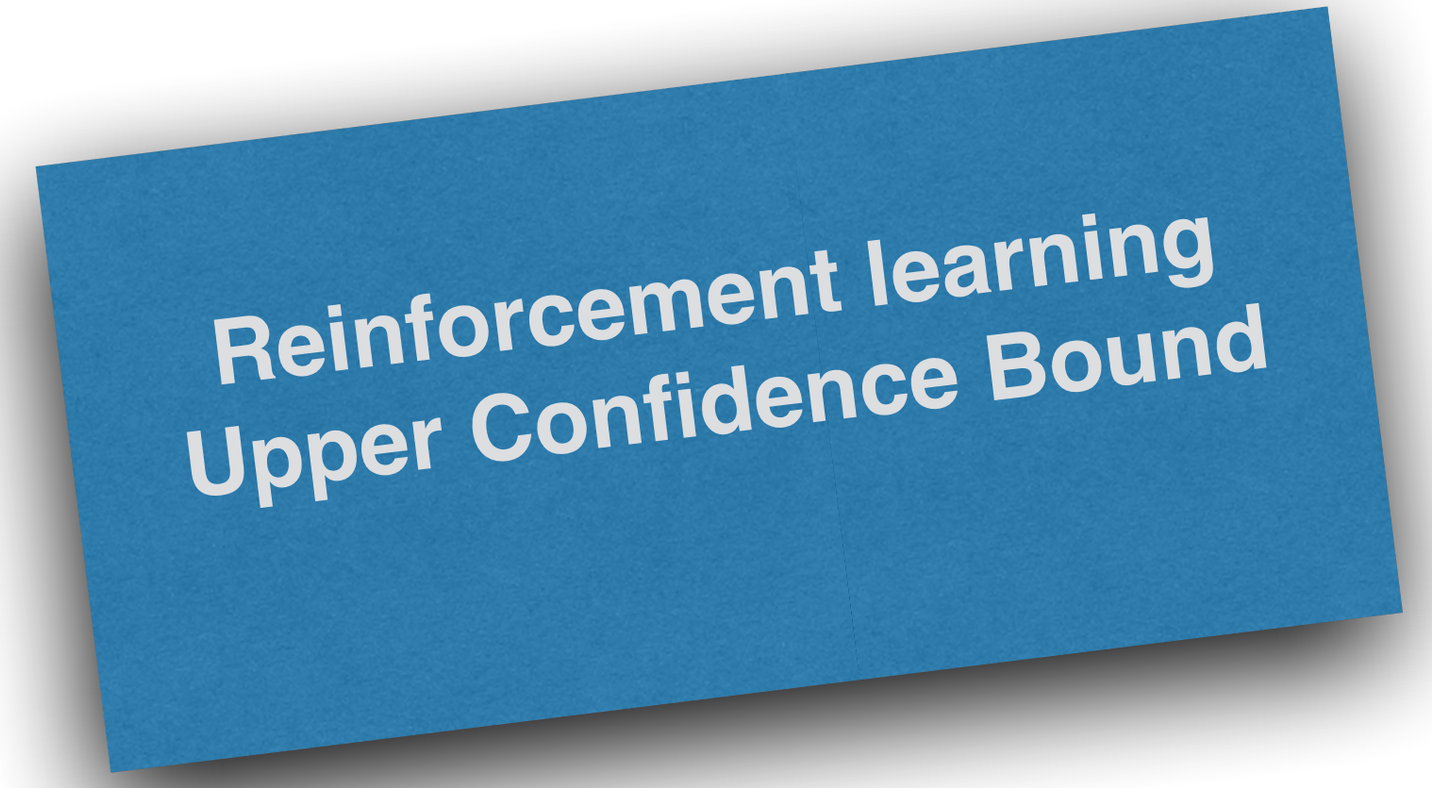
revert direction when not profitable

random jumps to avoid local minima

# Retry policy

- Give up on capacity aborts?
- How should we “consume” the attempts’ budget?
- How to manage the fall-back?

# Retry policy



# UCB

## tuning the retry policy

# UCB

## tuning the retry policy


A quest for **exploration** vs **benefit** from current knowledge

# UCB

## tuning the retry policy


A quest for **exploration** vs **benefit** from current knowledge

UCB adapts the strategy to maximize reward
Logarithmic bound on the optimization error

# UCB

tuning the retry policy

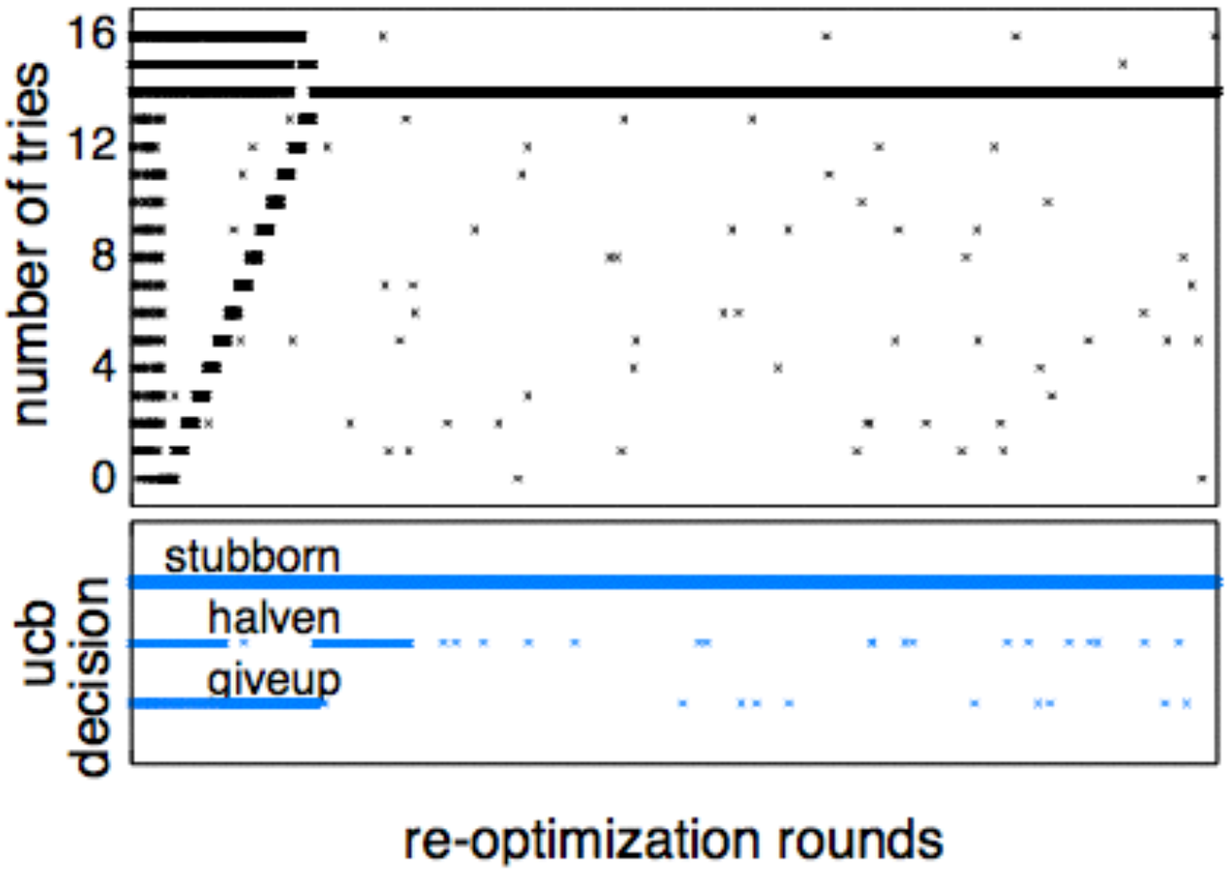
Model the belief about capacity aborts:

- giveup — exhaust attempts
- half — drops half the attempts
- stubborn — decrements attempts

Reward:
function of processor cycles (RDTSC)

# Adaptation

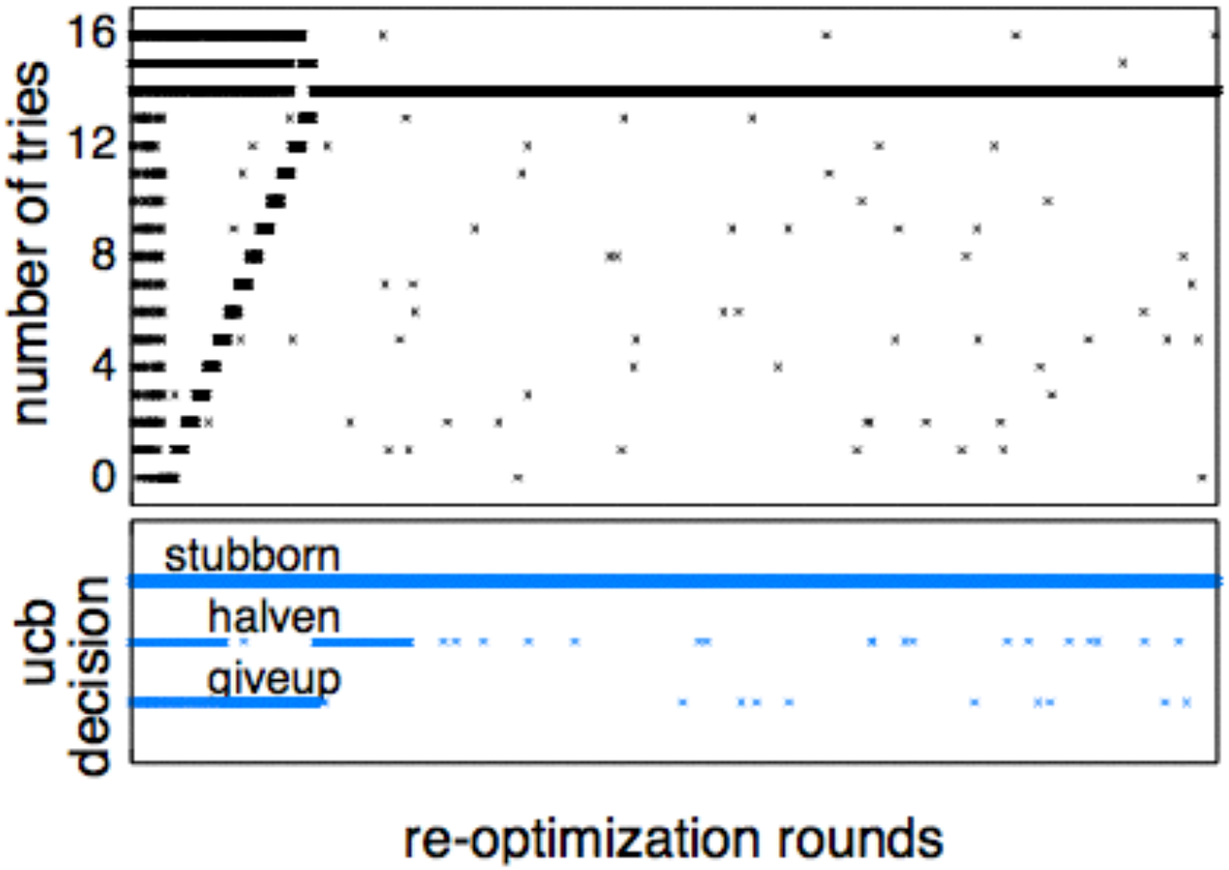
## of one atomic block in Yada



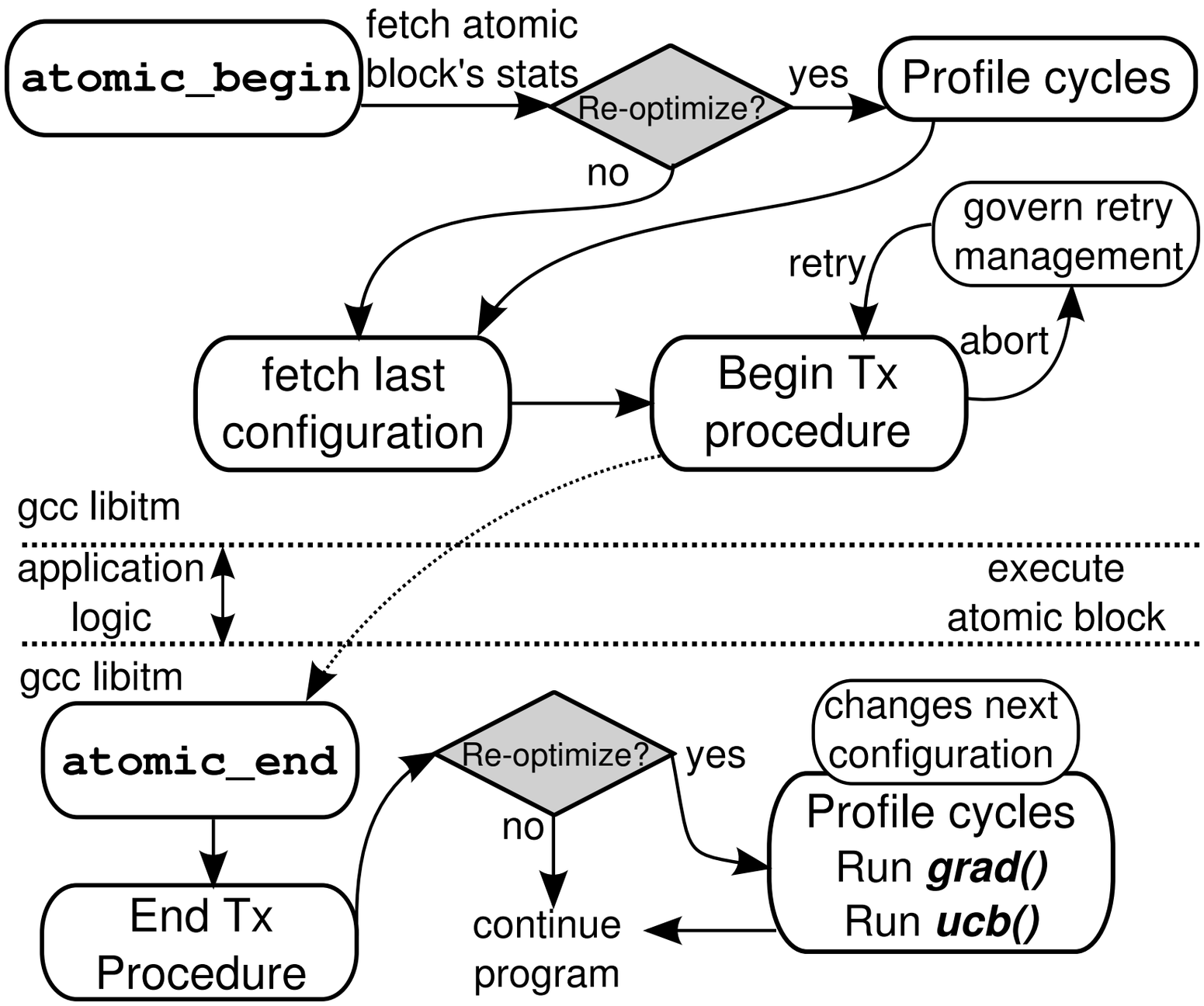
# Adaptation

## of one atomic block in Yada

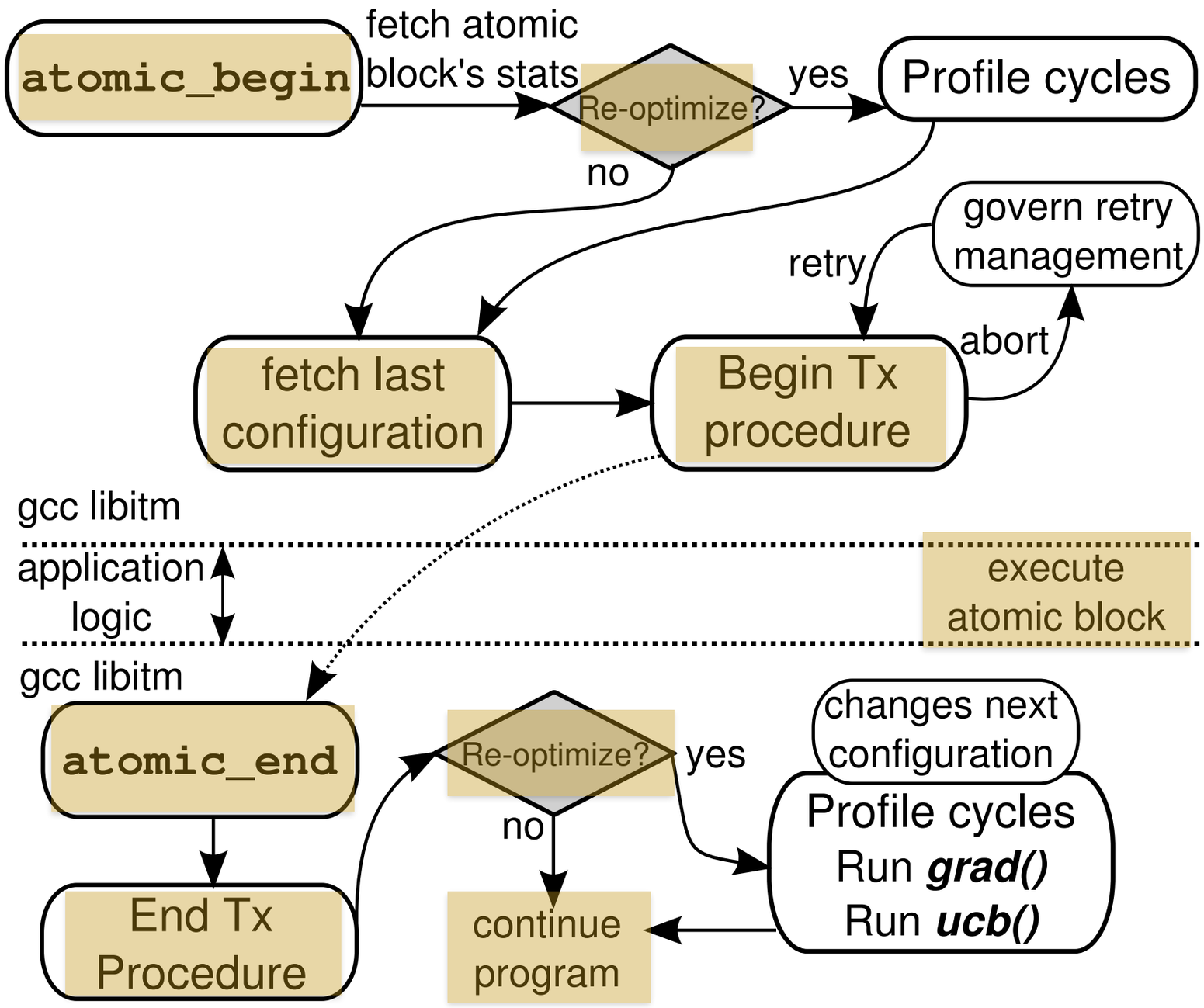
**optimizers are *not* independent**



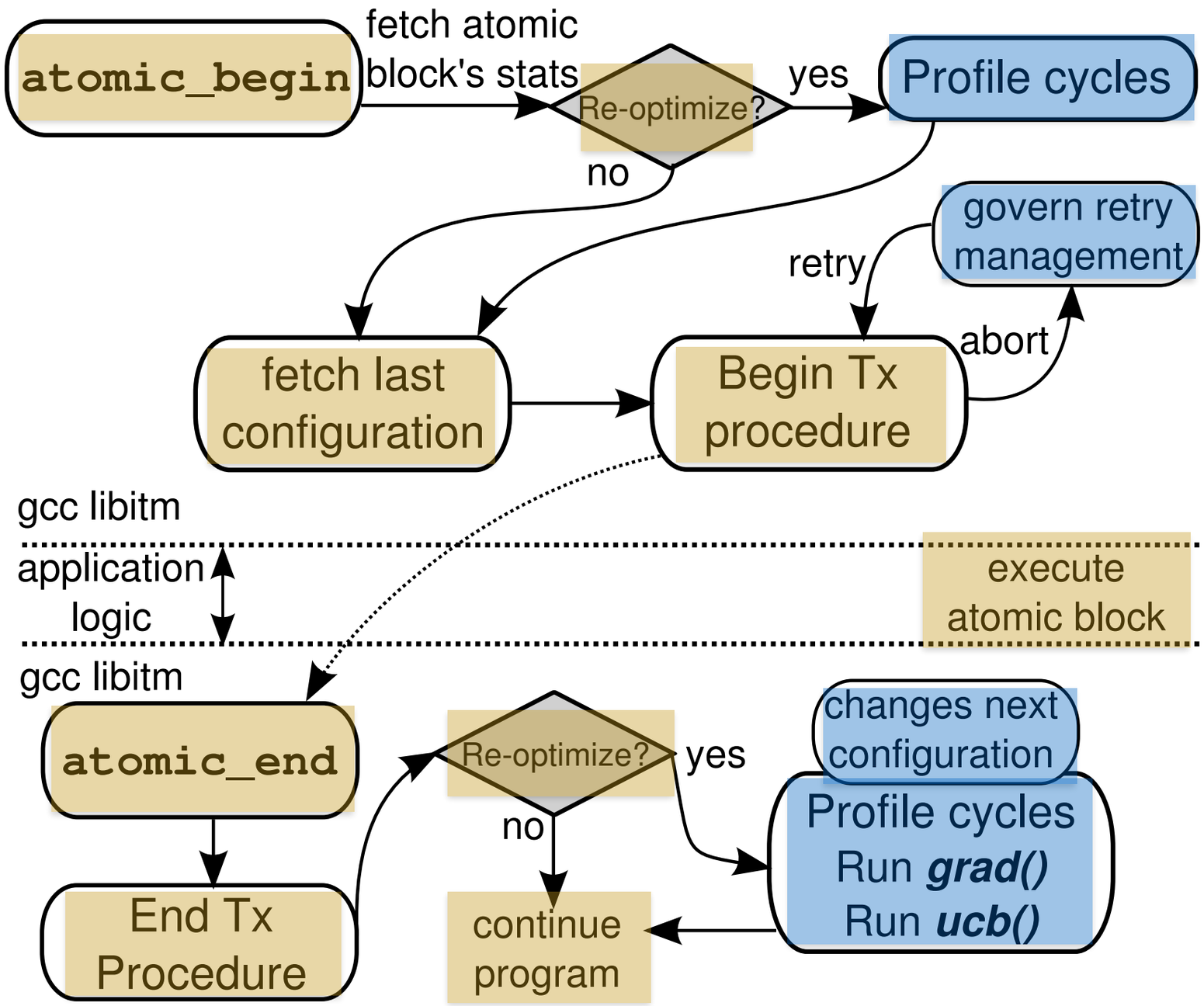
# Transparency to the User



# Transparency to the User



# Transparency to the User



# Summary of Evaluation

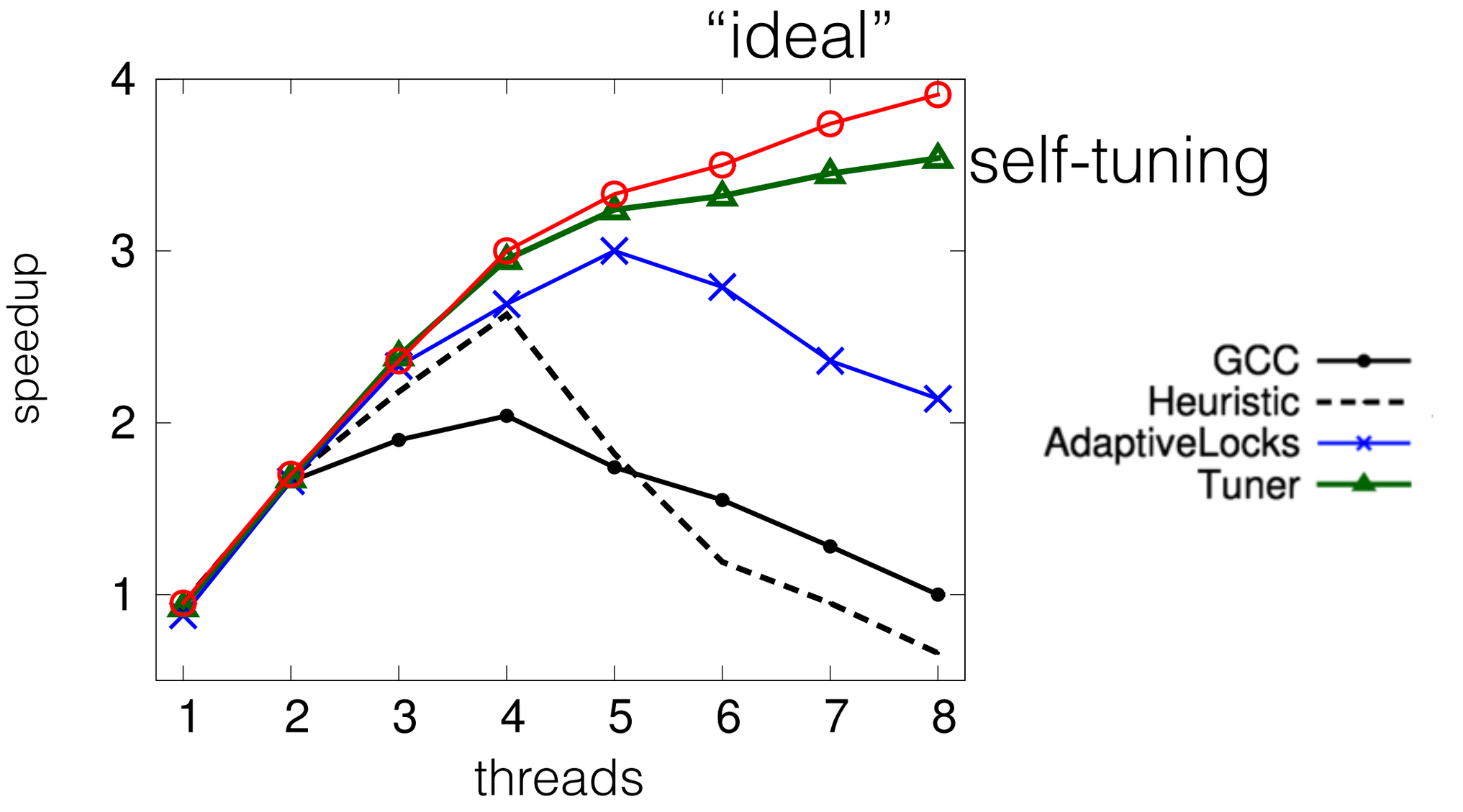
| Algorithms | threads | | | |
|---|---|---|---|---|
| | 2 | 4 | 6 | 8 |
| GCC | | | | |
| HEURISTIC | | | | |
| ADAPTIVELOCKS | | | | |
| TUNER | | | | |
| Best Variant | | | | |

# Summary of Evaluation

| Algorithms | threads | | | |
|---|---|---|---|---|
| | 2 | 4 | 6 | 8 |
| GCC | 1.25 | 1.74 | 1.51 | 1.29 |
| HEURISTIC | 1.46 | 2.01 | 1.37 | 1.28 |
| ADAPTIVELOCKS | 1.26 | 1.19 | 1.10 | 1.11 |
| TUNER | 1.46 | 2.25 | 2.34 | 2.54 |
| Best Variant | 1.51 | 2.35 | 2.41 | 2.66 |

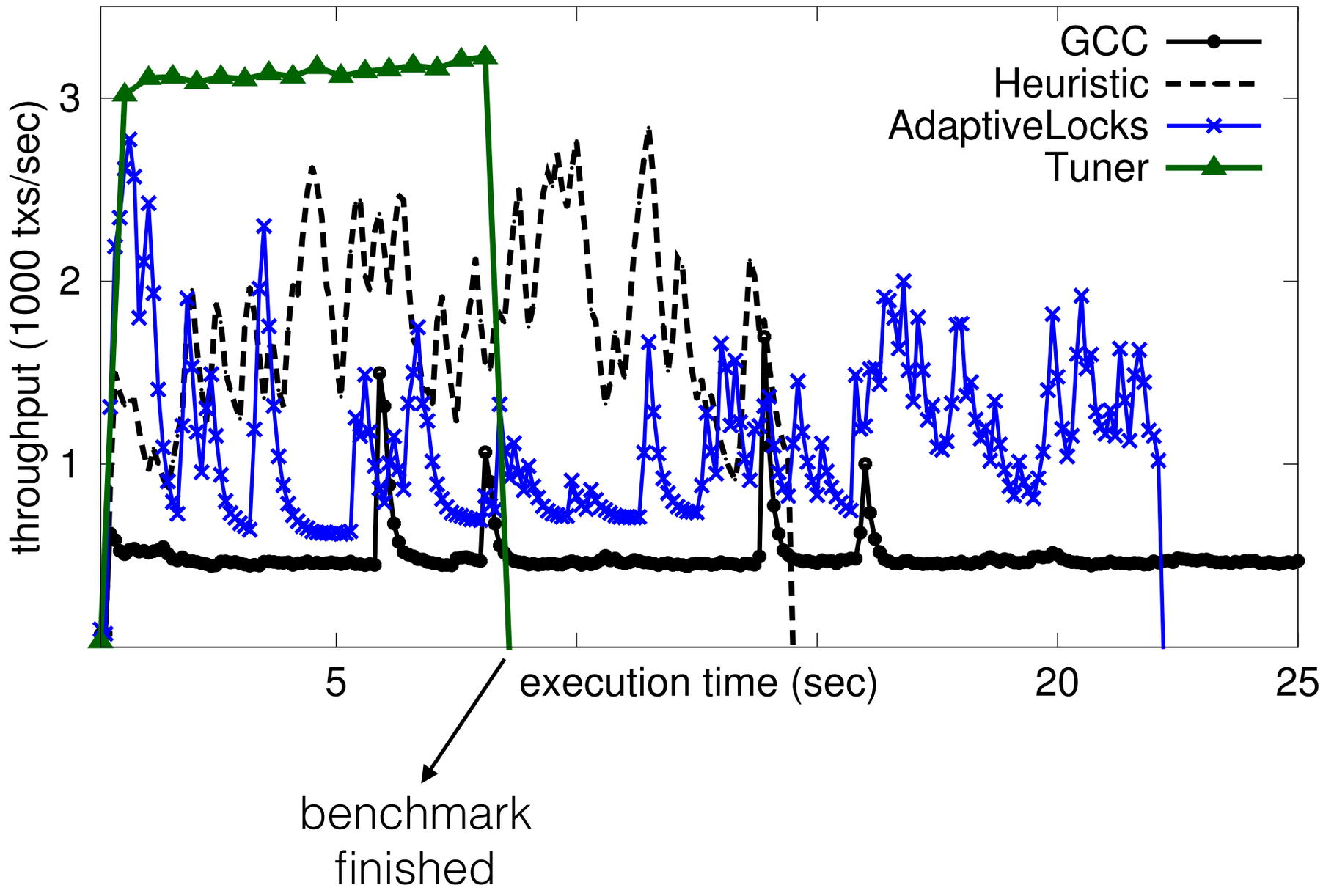
# Peek view on results

## Intruder from STAMP



# Peek view on results

## Yada with 8 threads



# Summary

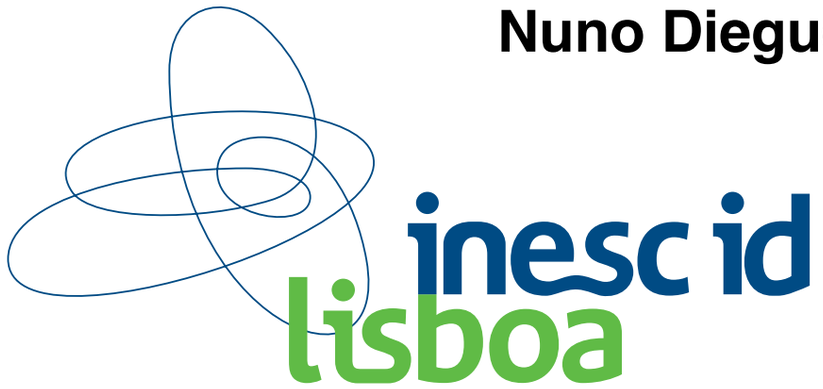
- Best-effort HTMs need proper tuning
- No one-size fits all
- We used lightweight exploration/learning techniques
- Transparent to the programmer

# Thank you!

Questions?

Self-Tuning Intel TSX

**Nuno Diegues** and Paolo Romano




TÉCNICO LISBOA

to appear on the 11th USENIX ICAC 2014