MANCHESTER
1824
The University of Manchester

IGWOD / LEANBIN

LeanBin: Harnessing Lifting and Recompilation to Debloat Binaries

39th IEEE/ACM International Conference on Automated Software Engineering
California, CA, USA, 27 October - 1 November 2024

Igor Wodiany, Antoniu Pop, Mikel Lujan
University of Manchester
<firstname>.<lastname>@manchester.ac.uk

MANCHESTER 1824
The University of Manchester

LeanBin

Binary Lifting

Reverse Engineering
Program Analysis
Patching
Recompilation

Binary Debloating

Security
Attack Surface Reduction
Specialisation

Hybrid Approach

↓

Static Analysis + Dynamic Analysis

↓

Fast + Precise

MOTIVATION

(With related work)

MANCHESTER 1824
The University of Manchester

Software is Imperfect

- Binaries and libraries can be exploited
- Bigger the binary, bigger the attack surface
- Often the whole binary is not needed (legacy code, unused functionality, etc.)

Software Debloating

- Remove unused code or keep what is needed
- Software debloating / specialisation can create new binaries with a smaller attack surface

Software Debloating

- Source code is not always available
- But debloating binaries directly lacks portability
- Binaries can be lifted to higher-level IR first, then be debloated and recompiled

Software Debloating

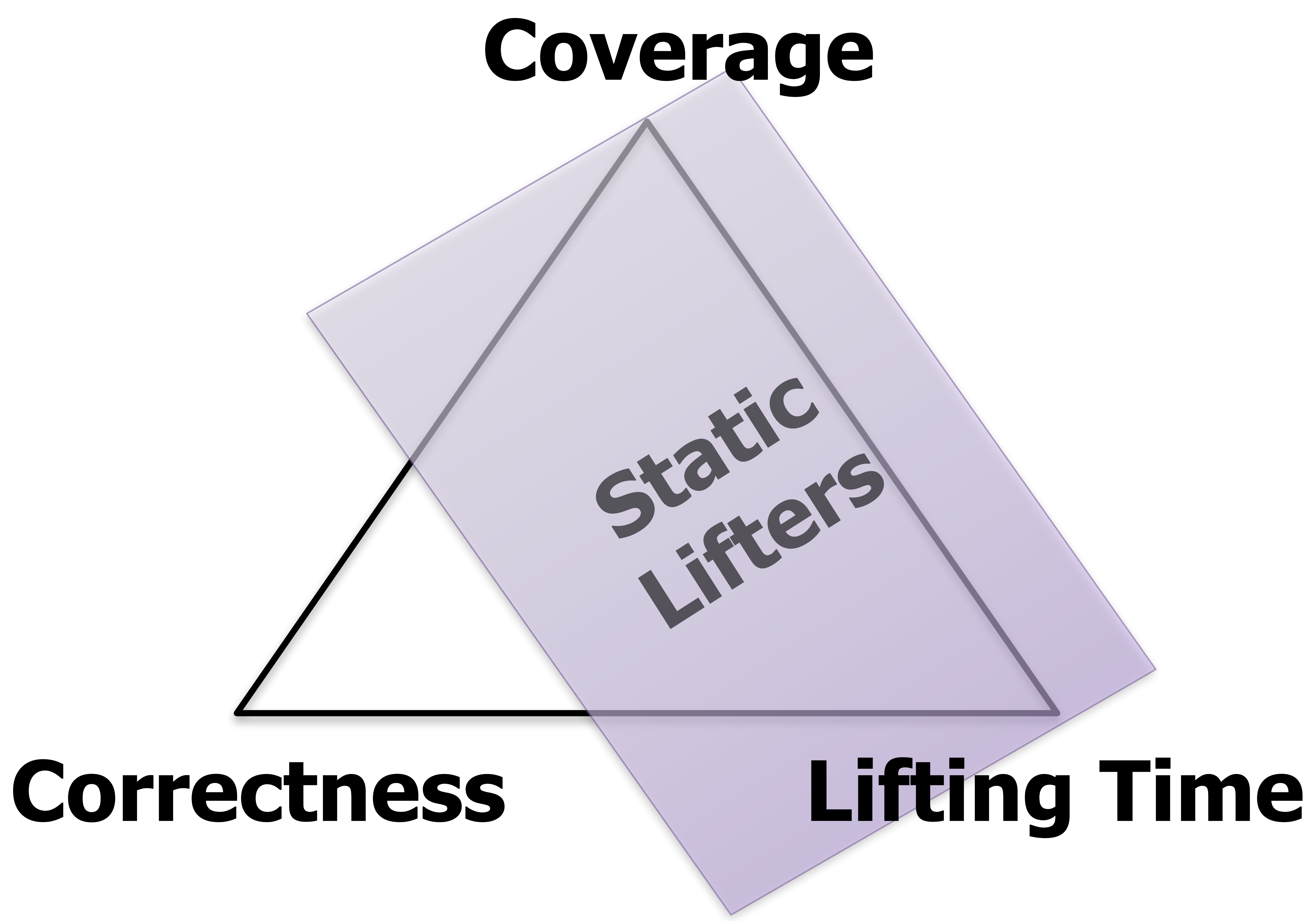
- **Only BinRec combines flexibility of binary debloating and reuse of the compiler infrastructure by the means on binary lifting and recompilation**
- **… But BinRec has limitations in the way it does lifting**

Into Lifting

So let’s talk about binary lifters…

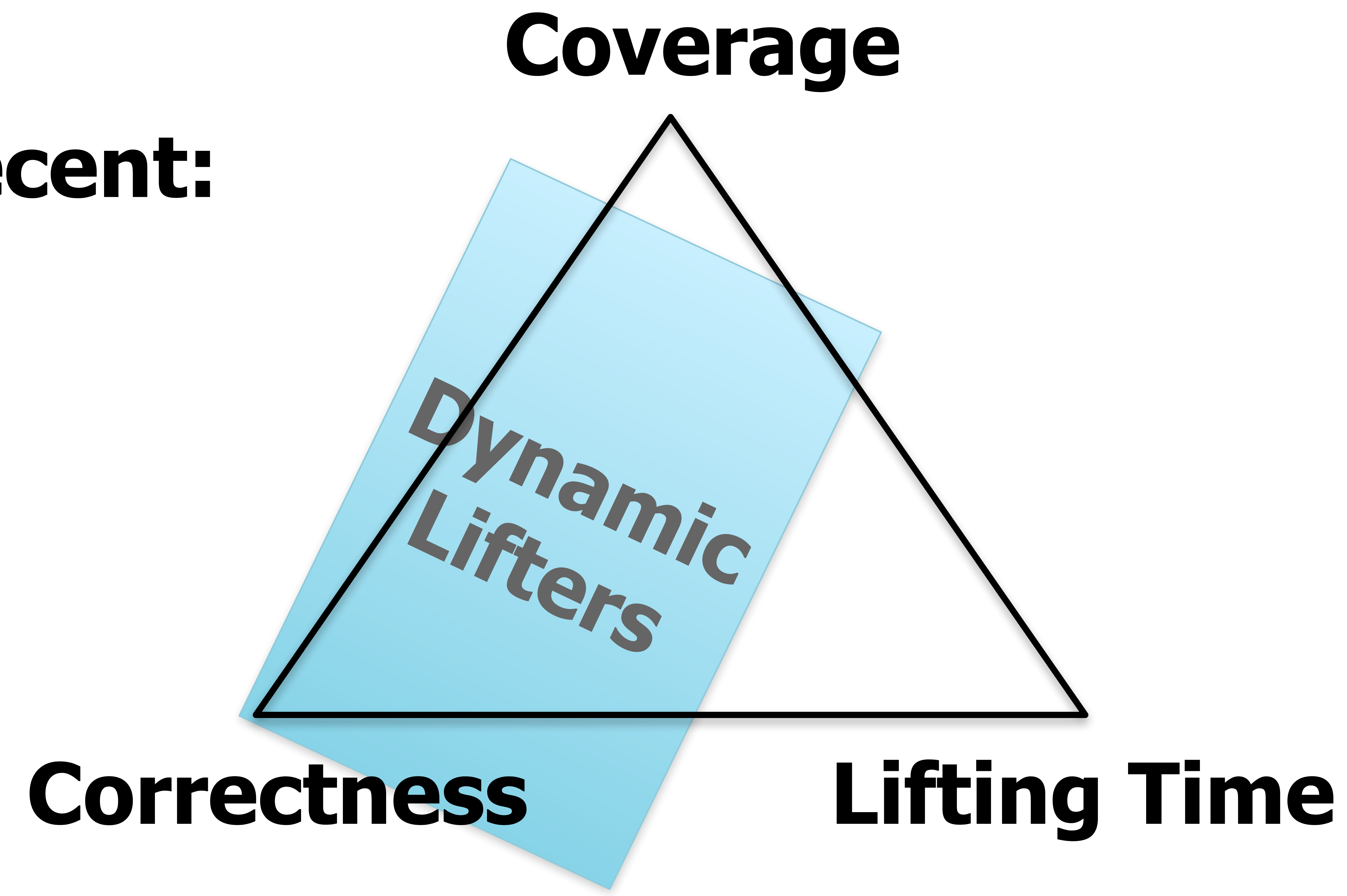
Binary Lifting

- **Static lifter are prevalent:**
  - **Fast**
  - **High coverage**
  - **Use heuristics**



Binary Lifting

- Dynamic lifters are more recent:
  - Slower
  - Heuristic free
  - Limited coverage



Question

Can we combine benefits of both to support binary debloating?

OUR APPROACH

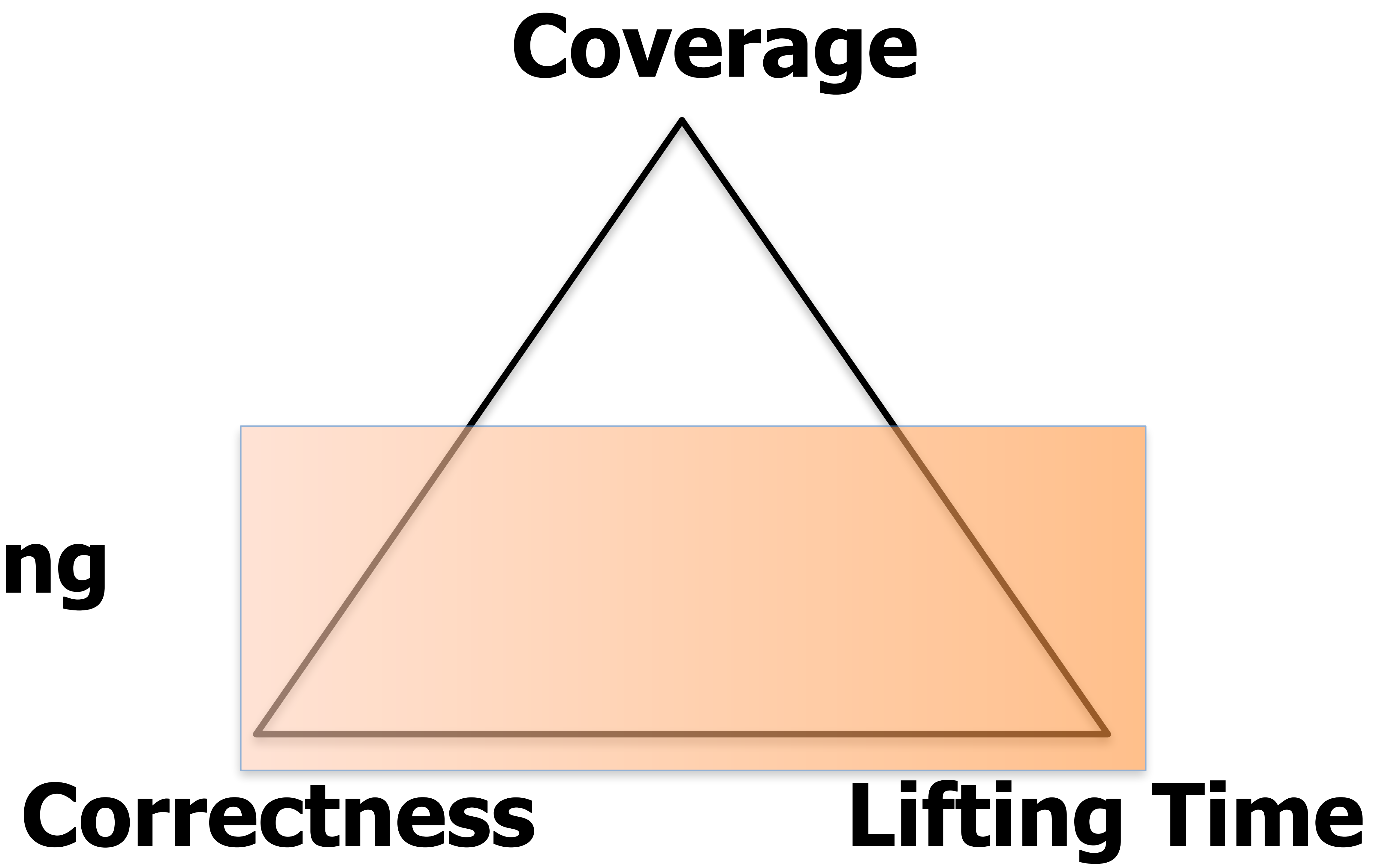
Our Approach

- Indirect control-flow cannot always be statically analysed…
  - So use dynamic analysis to discover indirect control flow
- But direct branches can…
  - So use heuristc-free static analysis to expand the control-flow beyond dynamic analysis

The University of Manchester

Hybrid Lifting

- **Novel hybrid lifting:**
  - **Fast**
  - **Can have limited coverage**
    - **But that is okay for debloating**
  - **Heuristic free**



System Overview

DYNAMIC
STATIC
01
BINARY
CONCRETE EXECUTION TRACER
A
CONCRETE INPUTS
?
STATIC ANALYSIS
B
CFG STRUCTURING
C
CODE GENERATION
D
C CODE
CLANG
LLVM IR

DEBLOATING STRATEGIES

(With an example)

Debloating Strategies

```
example:
  cmp x0, x1
  b.eq cond1
  add x2, x0, x1
  b end_cond

cond1:
  sub x2, x0, x1

end_cond:
  cmp x0, x1
  b.eq func1
  bl foo
  b end_func

func1:
  bl bar

end_func:
  ret
```

Not Executed = Undiscovered

Dynamic

`x0 = 1 and x1 = 1`

Debloating Strategies

```
example:
  cmp x0, x1
  b.eq cond1
  add x2, x0, x1
  b end_cond

cond1:
  sub x2, x0, x1

end_cond:
  cmp x0, x1
  b.eq func1
  bl foo
  b end_func

func1:
  bl bar

end_func:
  ret
```

Statically Analysed

Branches outside the function so not followed

Dynamic + Static 1

`x0 = 1 and x1 = 1`

Only follow internal control flow

MANCHESTER 1824
The University of Manchester

Debloating Strategies

```
example:
   cmp x0, x1
   b.eq cond1
   add x2, x0, x1
   b end_cond

cond1:
   sub x2, x0, x1

end_cond:
   cmp x0, x1
   b.eq func1
   bl foo
   b end_func

func1:
   bl bar

end_func:
   ret
```

All direct branches followed

Dynamic + Static 2

`x0 = 1 and x1 = 1`

Follow all control flow

Debloating Strategies

```
example:
   cmp x0, x1
   b.eq cond1
   add x2, x0, x1
   b end_cond

cond1:
   sub x2, x0, x1

end_cond:
   cmp x0, x1
   b.eq func1
   blr x0 ✗
   b end_func

func1:
   blr x1

end_func:
   ret
```

Direct branches can be followed

`x0 = 1 and x1 = 1`

Indirect branch instead

Not executed indirect branches cannot be followed

Executed indirect branches can be followed

MANCHESTER
1824
The University of Manchester

IMPLEMENTATION

MANCHESTER
1824
The University of Manchester

Implementation

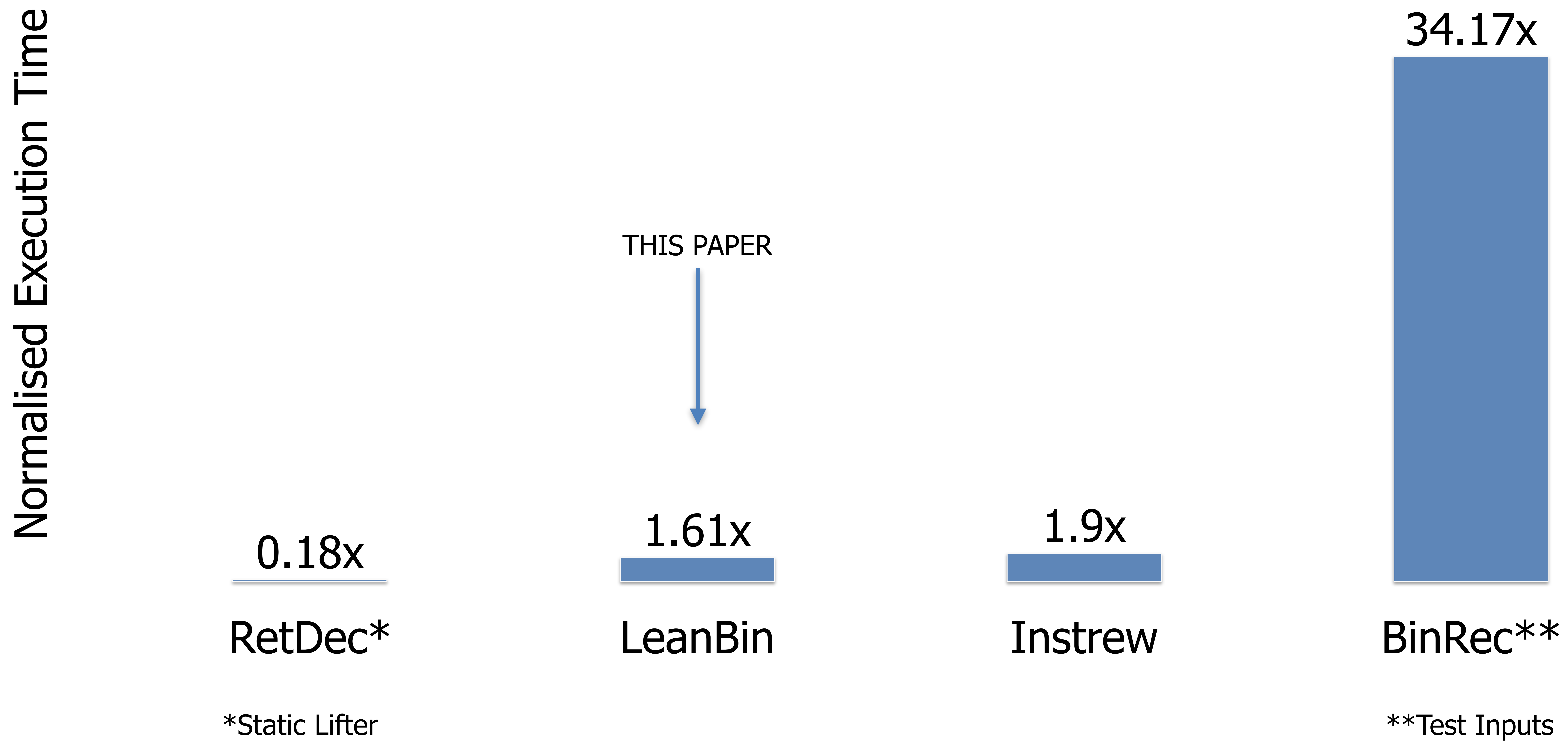
- Targets 64-bit ARM (AArch64) binaries and libraries
- Supports optimised binaries with non-trivial control flow (indirect branches, callbacks)

MANCHESTER
1824
The University of Manchester

RESULTS

MANCHESTER 1824
The University of Manchester

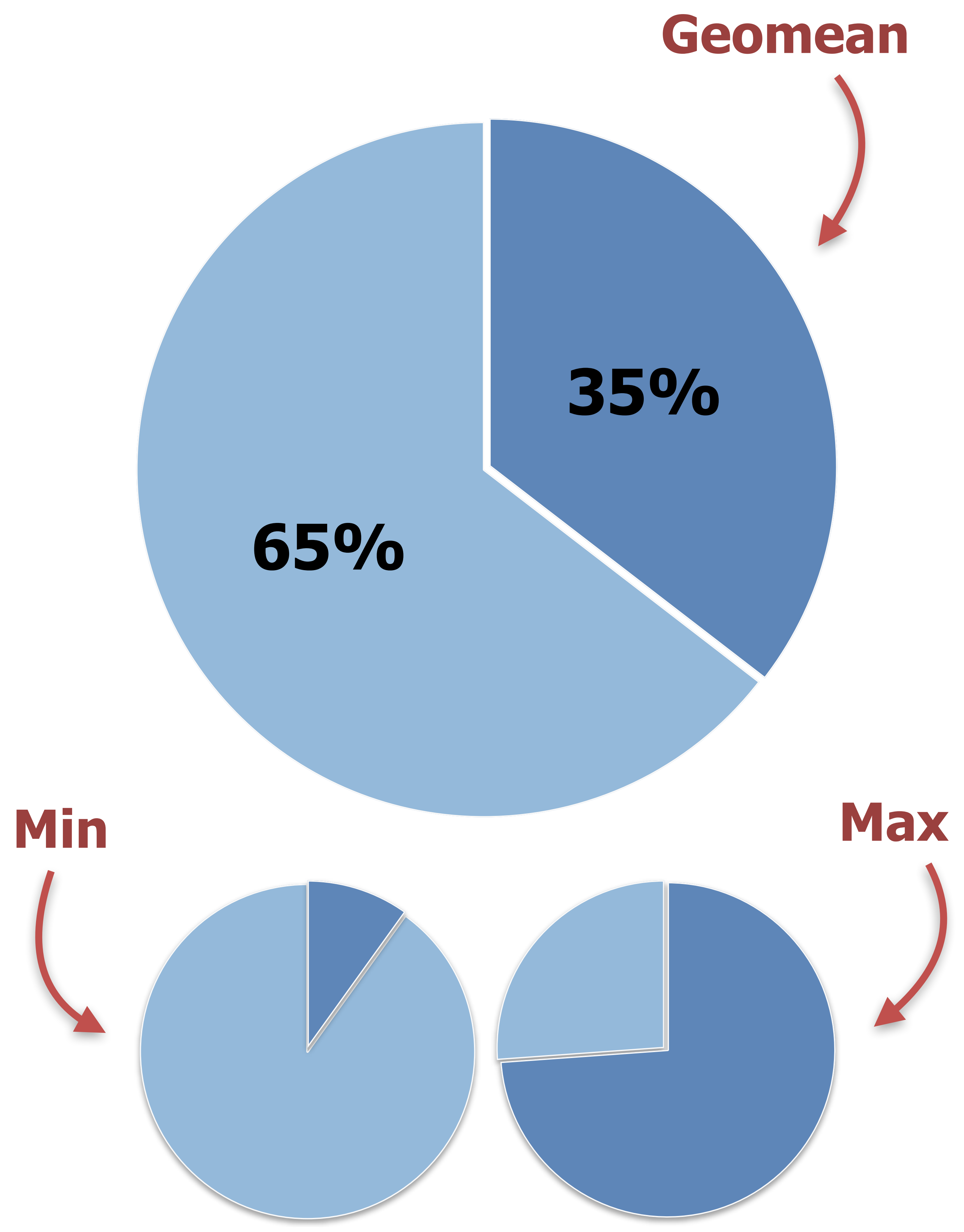
Performance Overhead of Lifting



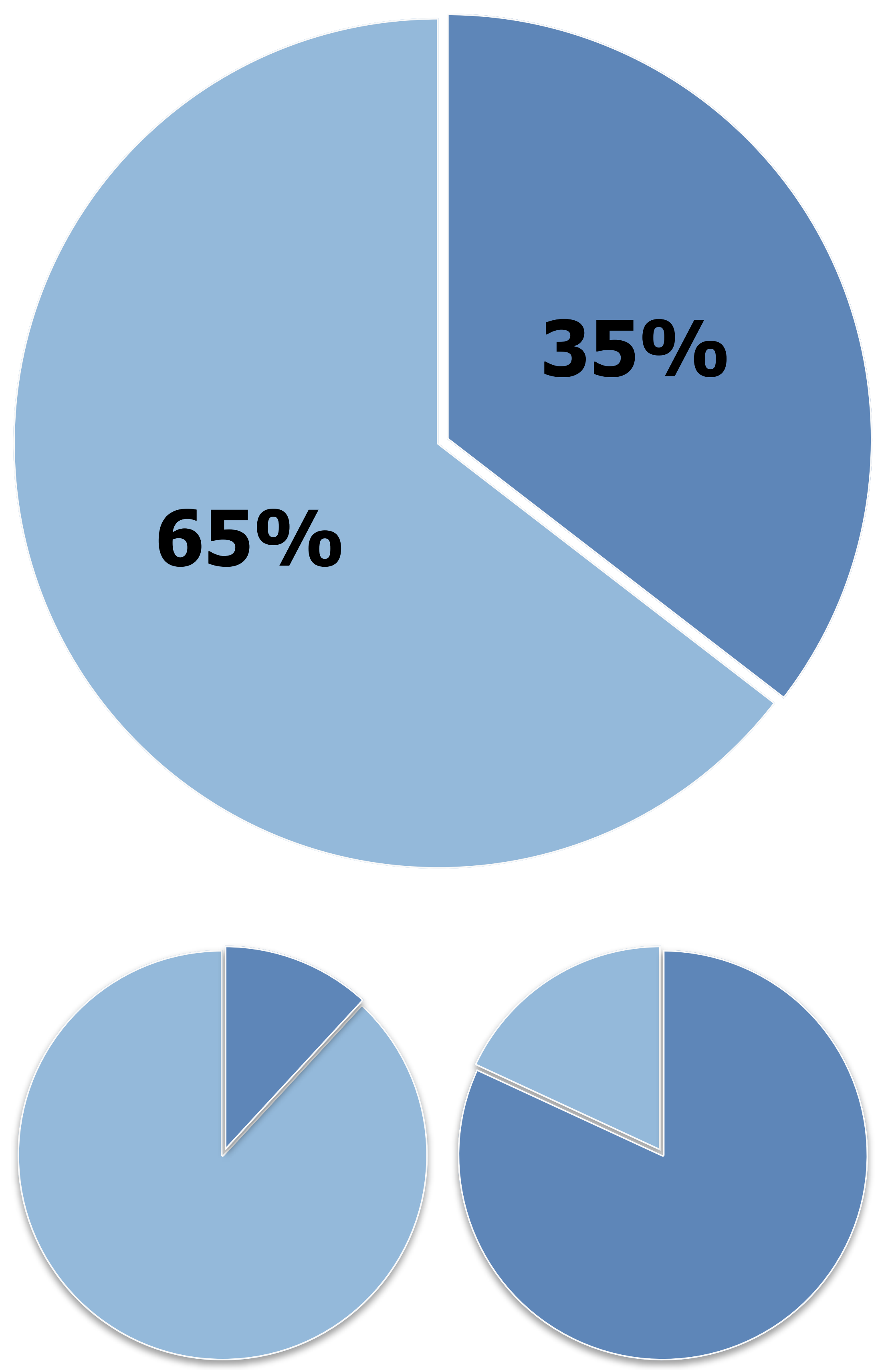
Coverage

SPEC CPU2006 INT

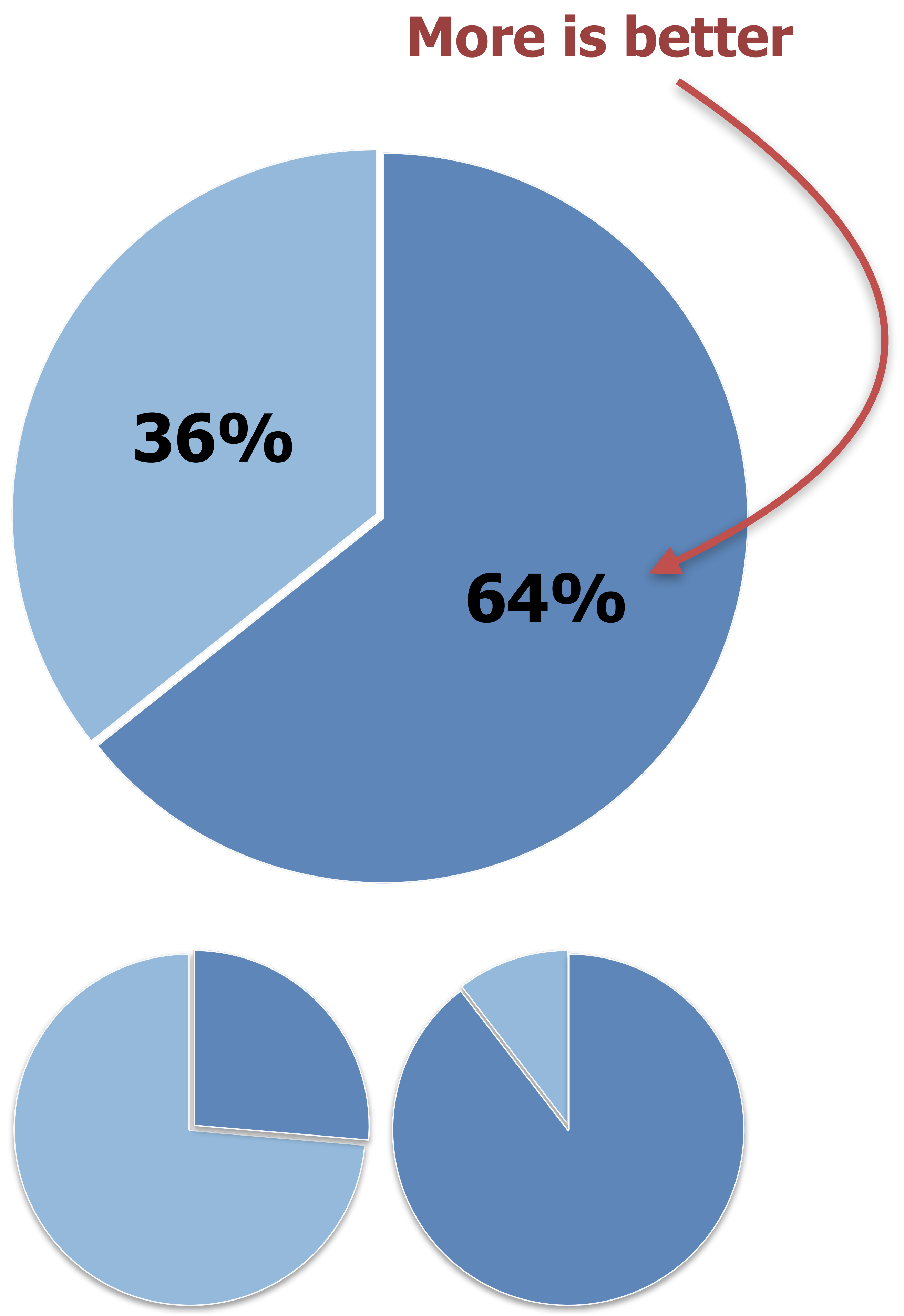
● Discovered ● Undiscovered



Dynamic



Dynamic + Static 1

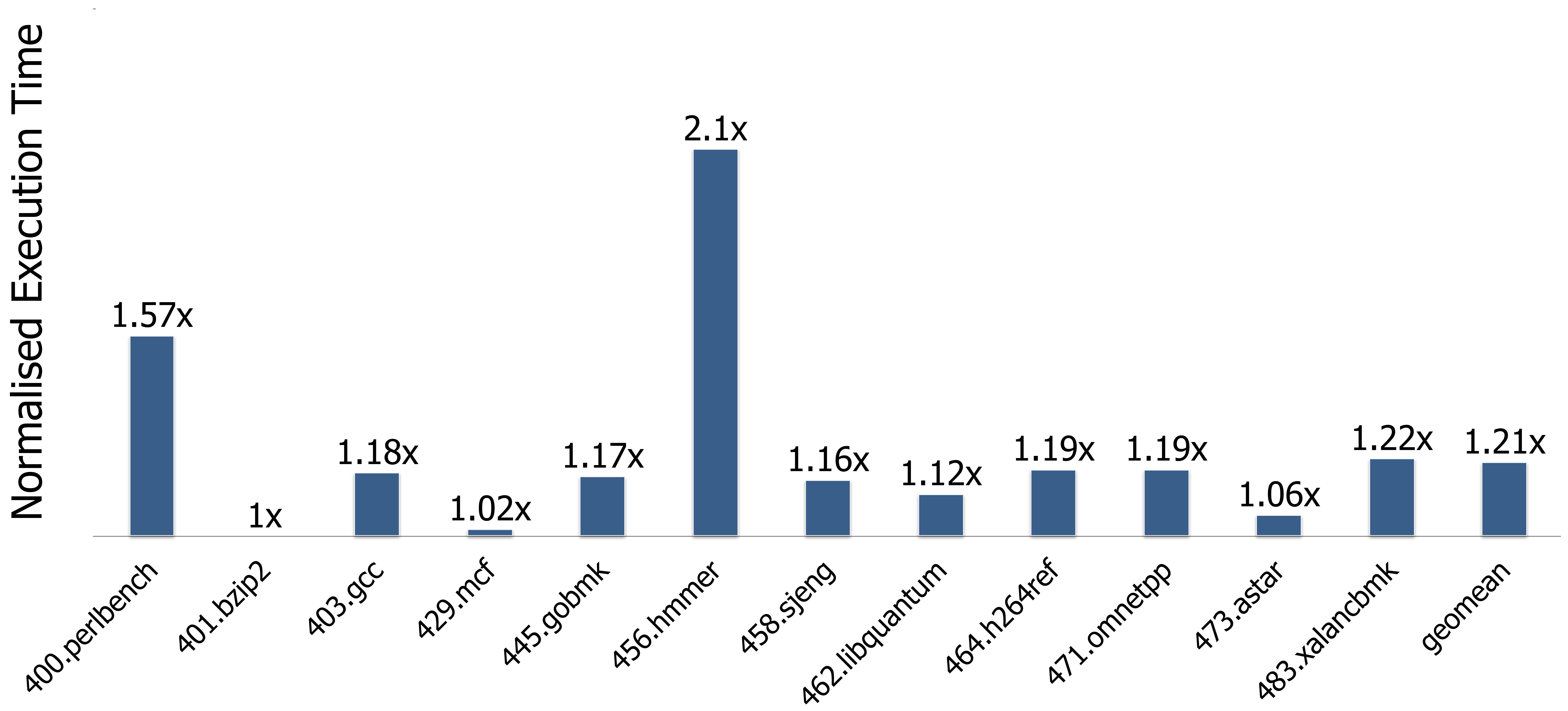


Dynamic + Static 2

MANCHESTER
1824
The University of Manchester

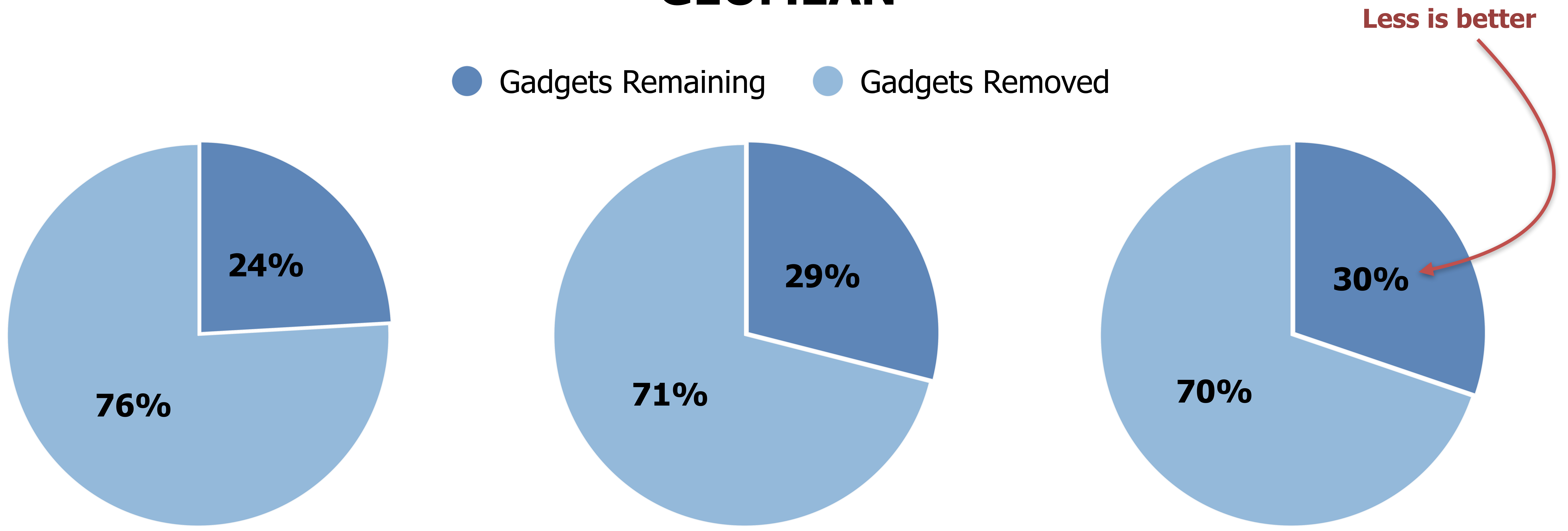
Debloated Binary Performance

SPEC CPU2006 INT



Gadgets

SPEC CPU2006 INT GEOMEAN

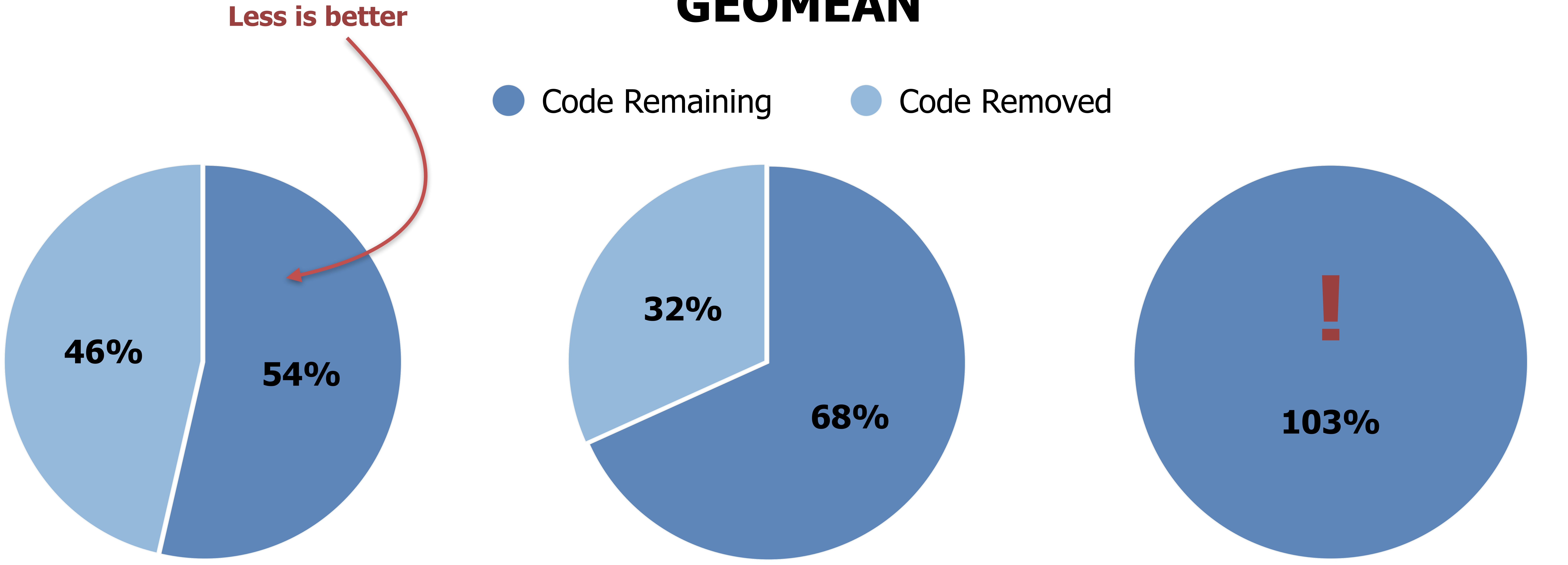


Dynamic **Dynamic + Static 1** **Dynamic + Static 2**

MANCHESTER 1824
The University of Manchester

Code Size

SPEC CPU2006 INT GEOMEAN



Dynamic

Dynamic + Static 1

Dynamic + Static 2

CONCLUSIONS

Conclusions

- First binary debloater based on novel hybrid lifting combining dynamic and heuristic-free static analysis
- Open-source implementation for ARM 64-bit (AArch64) binaries

MANCHESTER 1824
The University of Manchester

Thanks!

UK Research and Innovation

ROYAL ACADEMY OF ENGINEERING

THE ROYAL SOCIETY

EPSRC
Engineering and Physical Sciences Research Council

MoatE (10017512) and Soteria (75243)

MANCHESTER 1824

The University of Manchester

CODE OPEN SOURCE ON GITHUB

IGWOD / LEANBIN

(APACHE 2.0 LICENSE)