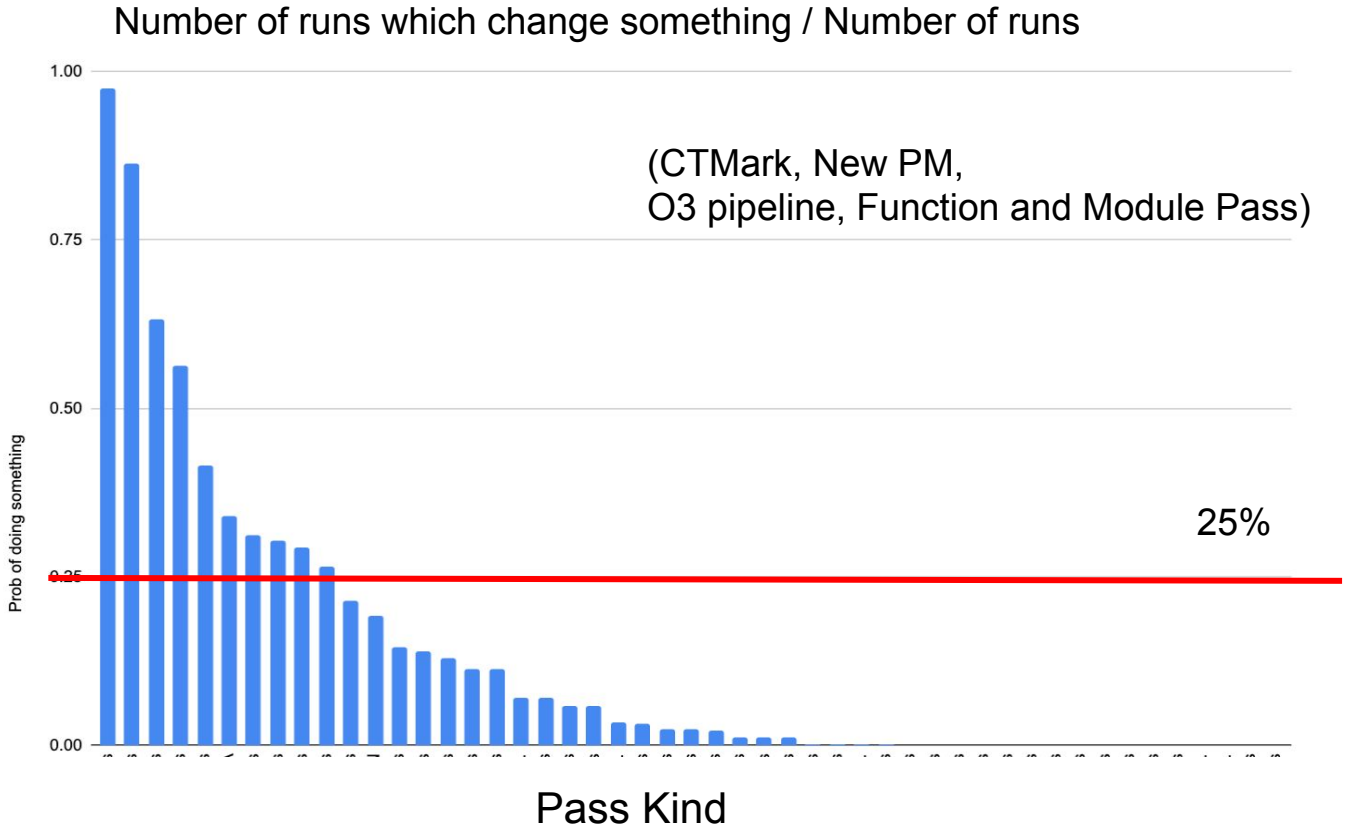
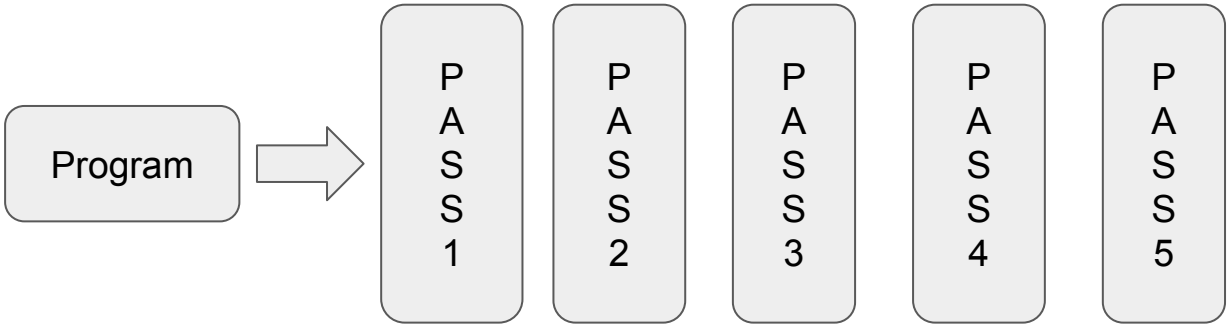
From Implicit Pass Dependencies to Effectiveness Prediction

Hideto Ueno	*University of Tokyo*
Johannes Doerfert	*Argonne National Laboratory*
Giorgis Georgakoudis	*Lawrence Livermore National Laboratory*
EJ Park	*Los Alamos National Laboratory*
Tarindu Jayatilaka	*University of Moratuwa*

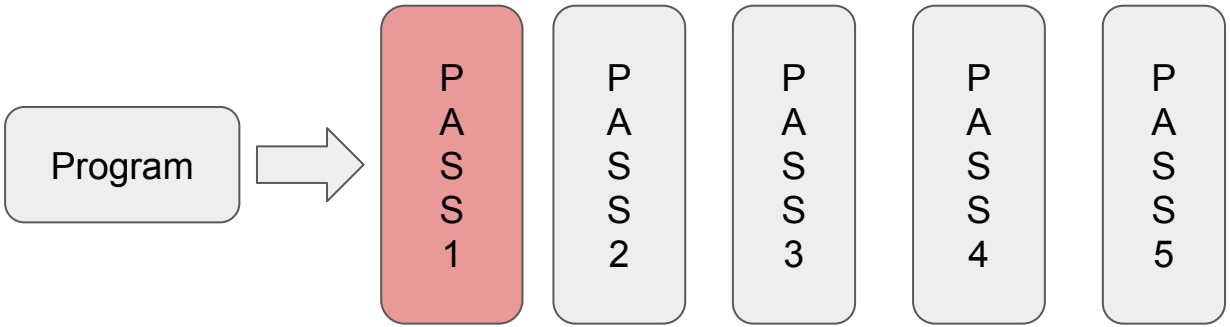
Background : How often do optimizations work?



Motivation : Skip passes to save compile time

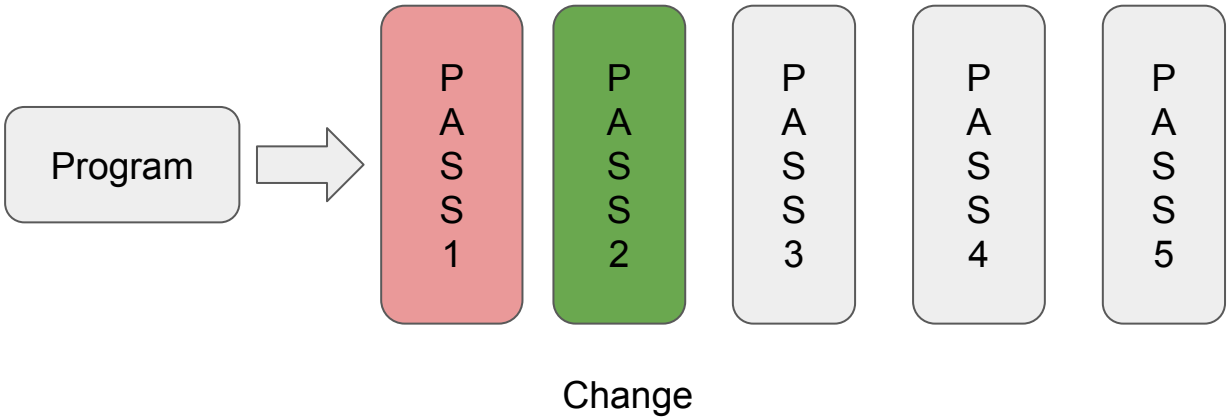


Motivation : Skip passes to save compile time

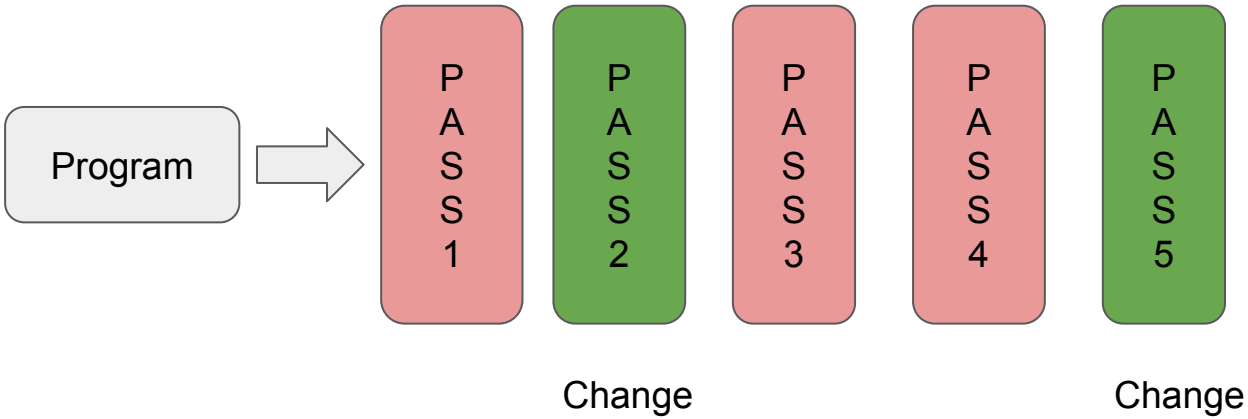


No Change

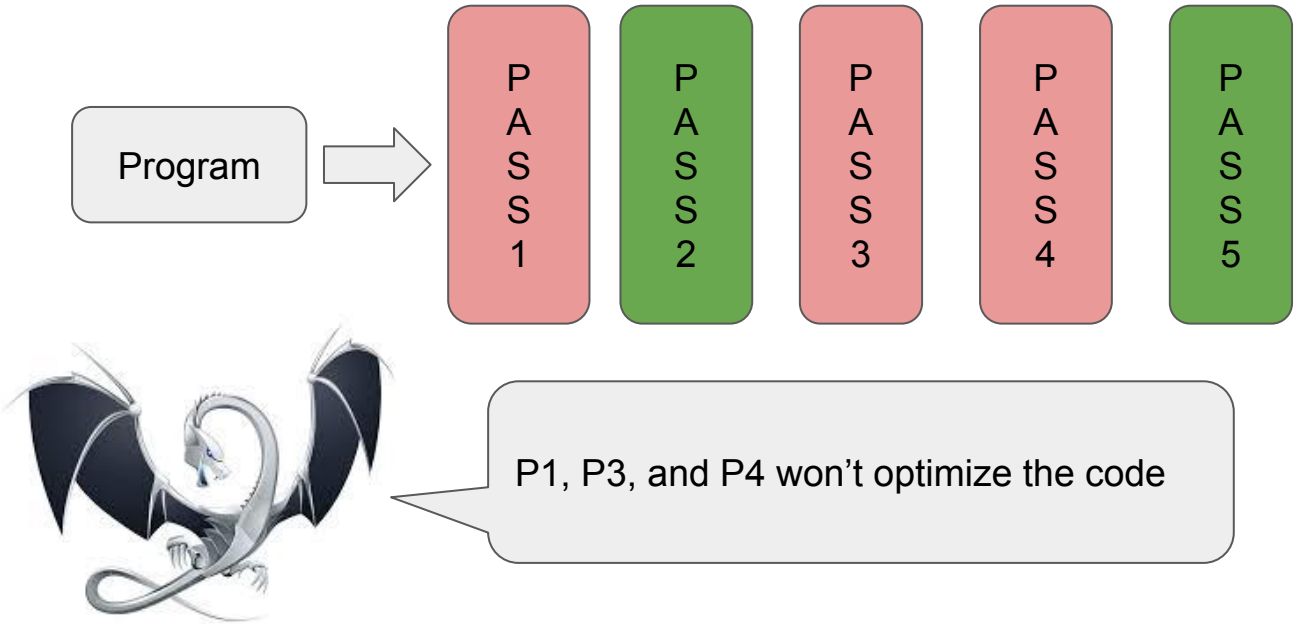
Motivation : Skip passes to save compile time



Motivation : Skip passes to save compile time

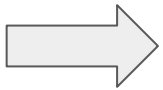


Motivation : Skip passes to save compile time



Code Feature

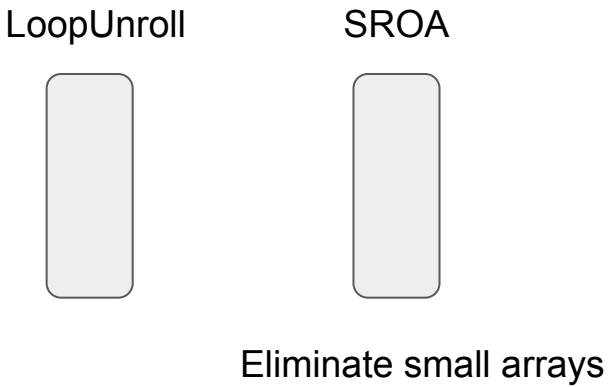
```
define i32 @fact(i32)  {
  %2 = icmp slt i32 %0, 2
  br i1 %2, label %7, label %3

bb1:
  %4 = add nsw i32 %0, -1
  %5 = tail call i32 @fact(i32 %4)
  %6 = mul nsw i32 %5, %0
  ret i32 %6

bb2:
  ret i32 %0
}
```



```
{
  “InstructionCount”: 7,
  “BasicBlockCount”:3,
  “CallCount”:1,
  “RetCount”:2,
  …
}
```

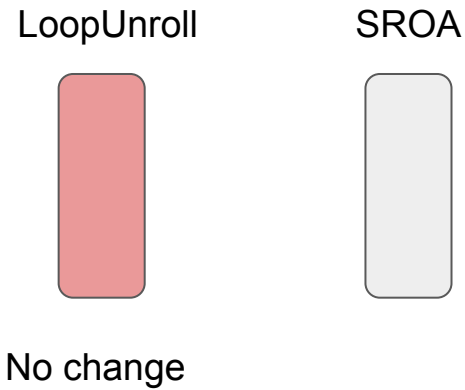
Pass Dependencies

- Most recent pass results have strong dependencies



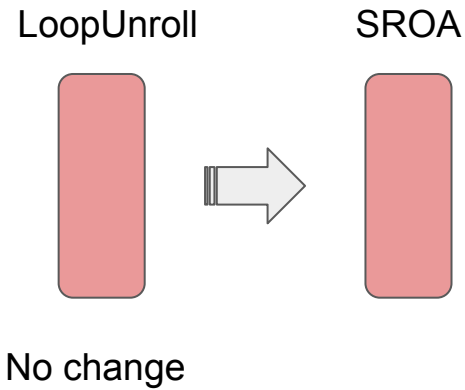
Pass Dependencies

- Most recent pass results have strong dependencies



Pass Dependencies

- Most recent pass results have strong dependencies



Prediction Model

- Recent Pass Results × Code Feature Vector → Predictions

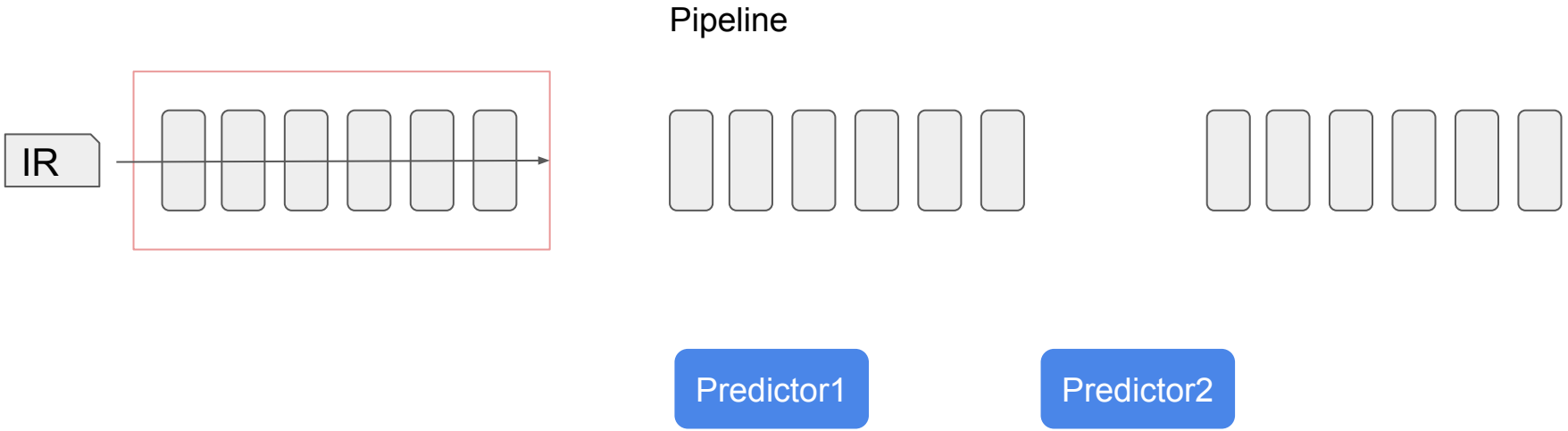
1, 4, and 6th pass changed IR

{ “InstructionCount”: .., “BasicBlockCount”: .. , etc .. }

Should run only 4th pass

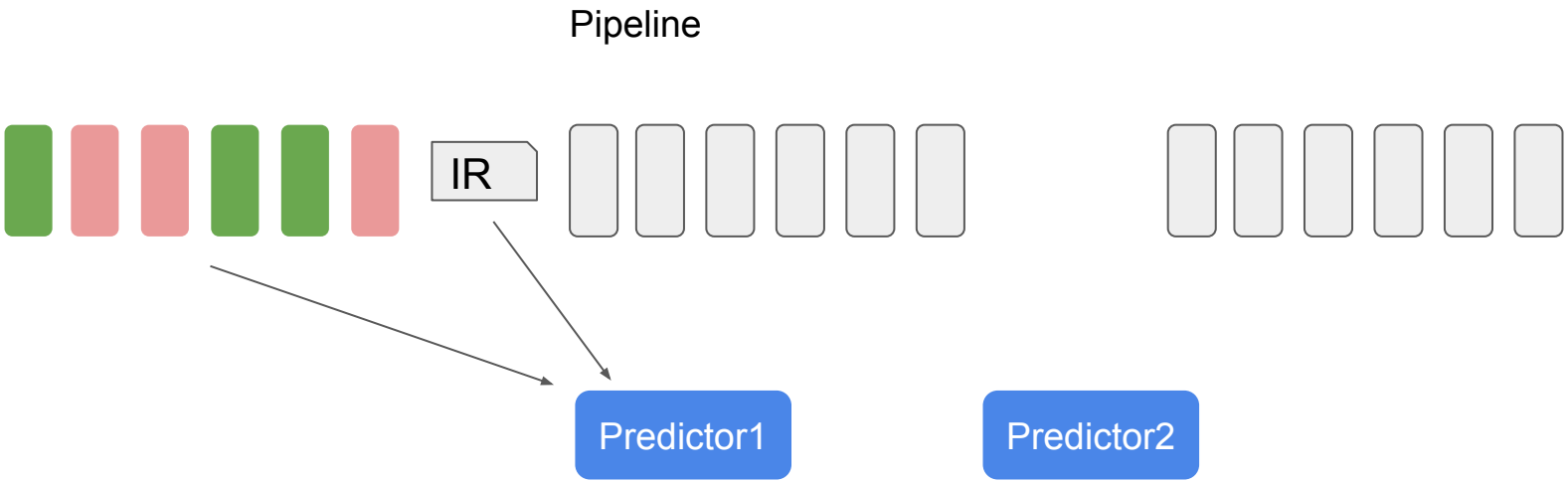
Embedding into pipeline

- We embed a predictor into specific points in the pipeline (4 points in `buildFunctionSimplificationPipeline`)



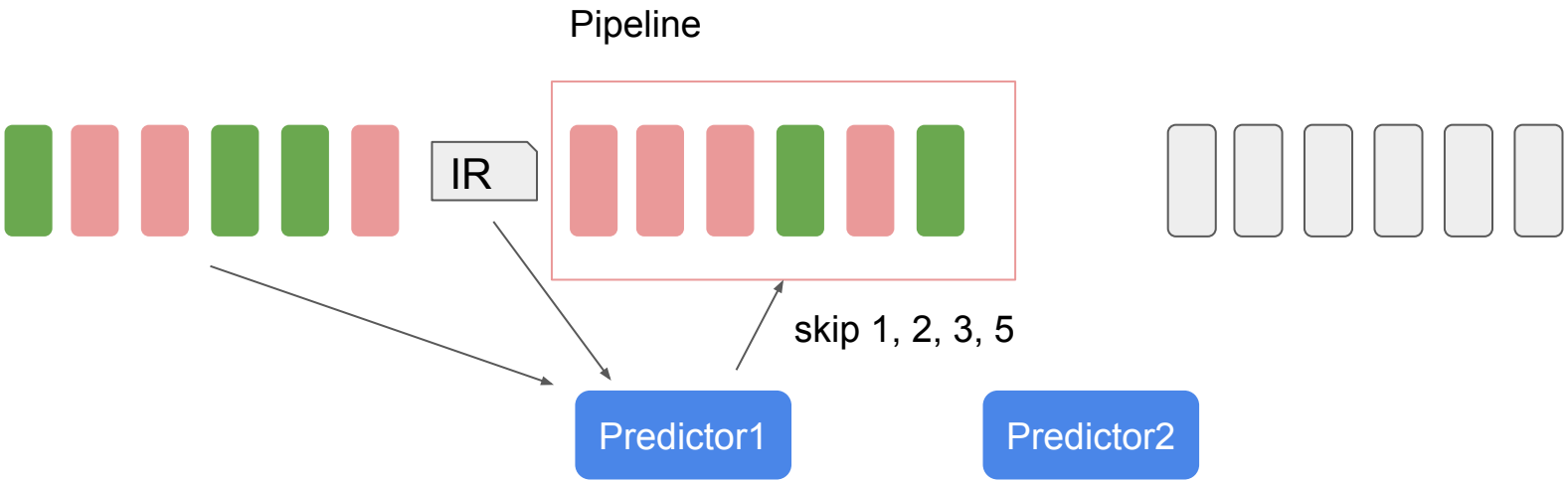
Embedding into pipeline

- We embed a predictor into specific points in the pipeline (4 points in `buildFunctionSimplificationPipeline`)



Embedding into pipeline

- We embed a predictor into specific points in the pipeline (4 points in `buildFunctionSimplificationPipeline`)



Current Result

- Relative changes(Instruction counts) to baseline, CTMark
  (Trained with test-suite/MultiSource + SingleSource expect for CTMark)

O3

threshold	compile time	execution time
prob = 0.5	-2.74%	+0.1%
prob = 0.9	-4.93%	+0.51%

aggresive

- If threshold is set higher, we skip passes more aggresively

```
bool should_run = model->modification_probability() > threshold;
```

Current Result

- Relative changes(Instruction counts) to baseline, CTMark
  (Trained with test-suite/MultiSource + SingleSource expect for CTMark)

O3

threshold	compile time	execution time
prob = 0.5	-2.74%	+0.1%
prob = 0.9	-4.93%	+0.51%

aggresive ↓

- If threshold is set higher, we skip passes more aggresively

```
bool should_run = model->modification_probability() > threshold;
```

Thank you for listening!

Hideto Ueno (uenoku.tokotoko [at] gmail.com)