FUNCTION MERGING by SEQUENCE ALIGNMENT

Rodrigo Rocha, Pavlos Petoumenos, Zheng Wang, Murray Cole, Hugh Leather

up to 25% smaller code

Merge Operation

```c
int f1(int a, int b) {
  int var = a*b + 10;
  int result = foo(var);
  printf("result: %d\n", result);
  return result;
}
```

```c
void f2(int a, int b) {
  int var = a*b + b;
  int result = foo(var);
  printf("result: %d\n", result);
  if (result==0)
    printf("result is zero\n");
}
```

Input Functions

Function: f1

```
label: entry
%r0 = mul %a, %b
%r1 = add %r0, 10
%r2 = call foo(%r1)
%r3 = call printf(@str,%r2)
ret %r2
```

Function: f2

```
label: entry
%x0 = mul %a, %b
%x1 = add %x0, %b
%x2 = call foo(%x1)
%x3 = call printf(@str,%x2)
%x4 = icmp eq %x2, 0
br %x4, %if.then, %if.end
```

```
label: if.then
%x5 = call printf(@str)
br %if.end
```

```
label: if.end
ret void
```

Function Linearization

```
label: entry
%r0 = mul %a, %b
%r1 = add %r0, 10
%r2 = call foo(%r1)
%r3 = call printf(@str,%r2)
ret %r2
```

```
label: entry
%x0 = mul %a, %b
%x1 = add %x0, %b
%x2 = call foo(%x1)
%x3 = call printf(@str,%x2)
%x4 = icmp eq %x2, 0
br %x4, %if.then, %if.end
label: if.then
%x5 = call printf(@str)
br %if.end
label: if.end
ret void
```

Sequence Alignment

	label	mul	add	call foo	call printf	icmp eq	br	label	call	br	label	ret
	0	−1	−2	−3	−4	−5	−6	−7	−8	−9	−10	−11	−12
label	−1	2	1	0	−1	−2	−3	−4	−5	−6	−7	−8	−9
mul	−2	1	4	3	2	1	0	−1	−2	−3	−4	−5	−6
add	−3	0	3	6	5	4	3	2	1	0	−1	−2	−3
call foo	−4	−1	2	5	8	7	6	5	4	3	2	1	0
call printf	−5	−2	1	4	7	10	9	8	7	6	5	4	3
ret	−6	−3	0	3	6	9	9	8	7	6	5	4	6

Match: +2
Mismatch: −1
Gap: −1

```
label: entry
%r0 = mul %a, %b
%r1 = add %r0, 10
%r2 = call foo(%r1)
%r3 = call printf(@str,%r2)
```

```
label: entry
%x0 = mul %a, %b
%x1 = add %x0, %b
%x2 = call foo(%x1)
%x3 = call printf(@str,%x2)
%x4 = icmp eq %x2, 0
br %x4, %if.then, %if.end
label: if.then
%x5 = call printf(@str)
br %if.end
label: if.end
```

```
ret %r2
```

Mergeable / Non-Mergeable

Code Generation

```
label: b1
%m0 = mul %a, %b
%t0 = select %0, 10, %b
%m1 = add %m0, %t0
%m2 = call foo(%m1)
%m3 = call printf(@str,%m2)
br %0, %b2, %b3
```

Renaming Table

entry	b1
r0	m0
x0	m0
r1	m1
x1	m1
r2	m2
x2	m2
r3	m3
x3	m3

f1

```
label: b2
br %b4
```

f2

```
label: b3
%x4 = icmp eq %m2, 0
br %x4, %if.then, %if.end
```

```
label: if.then
%x5 = call printf(@str)
br %if.end
```

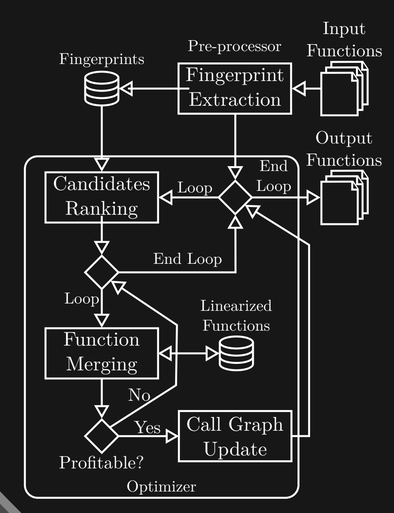
```
label: if.end
br %b4
```

```
label: b4
ret %m2
```

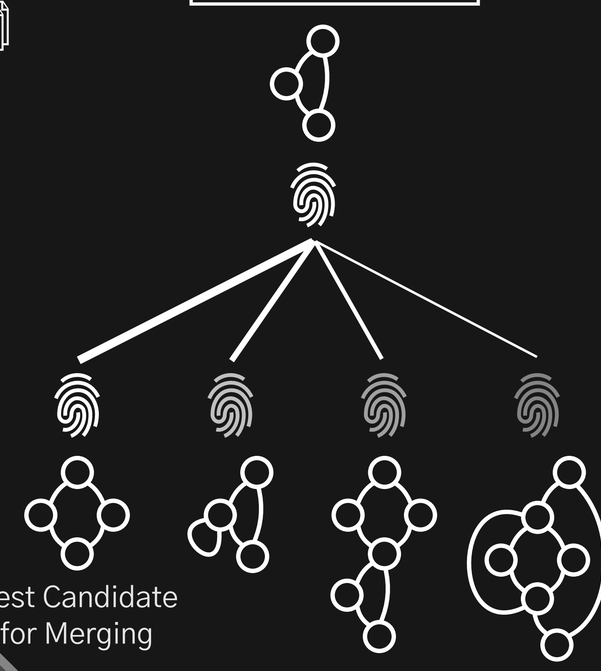
Output Function
f1 & f2 Merged
reduces code size!!

Ranking-based Exploration Mechanism

The Architecture of the New Function Merging Pass



Function Fingerprint

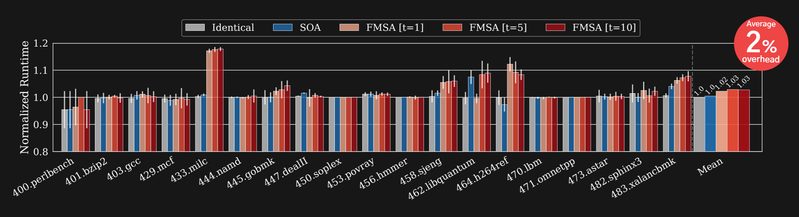


Best Candidate for Merging

Evaluation

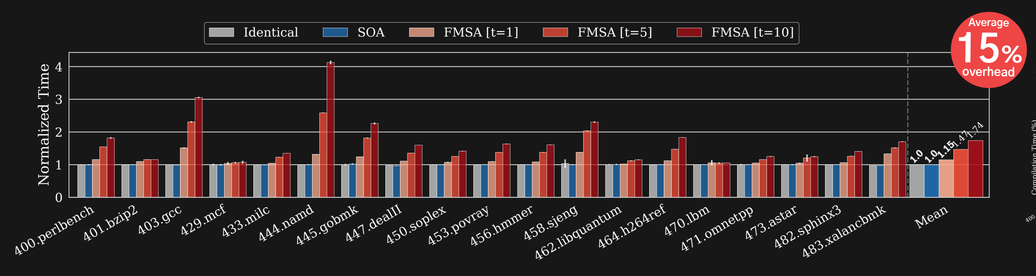
Runtime Overhead

negligible in most cases

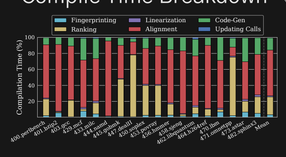


Compilation Time Overhead

small overhead for the best cost–benefit setting

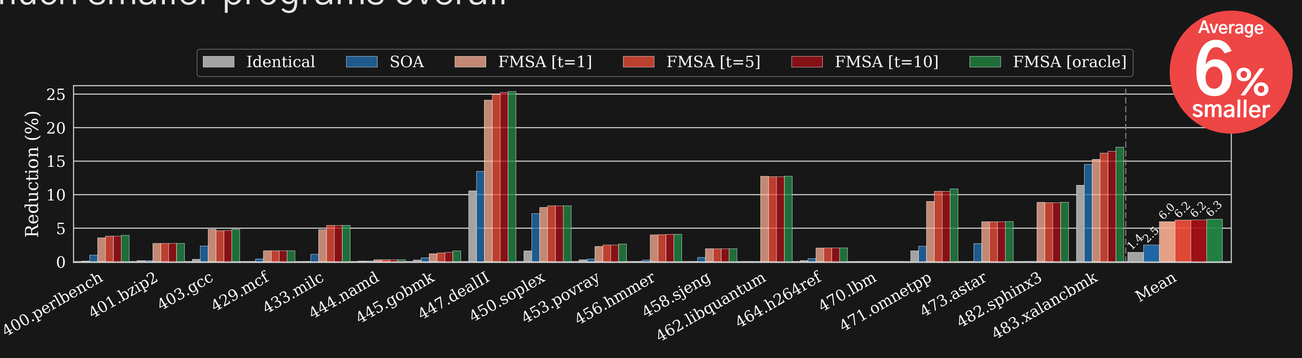


Compile Time Breakdown



Code Size Reduction

much smaller programs overall



Merging Parameters