Compiler Construction

Coalescing

What is coalescing?

Coalescing

Some low-level form of *copy propagation*

- While building traces we tried to remove jumps
- While allocating registers, we try to remove moves
  $\Rightarrow$ This is coalescing!

For instance, SSA produces many irrelevant move operations

Coalescing

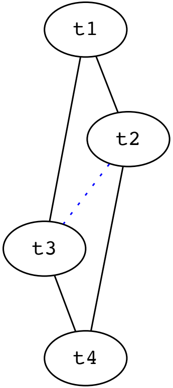
live-in: `t2`

```
t1 := ...
t2 := t1 + t2
t3 := t2
t4 := t1 + t3
t2 := t3 + t4
t1 := t2 - t4
```

live-out: `t1`

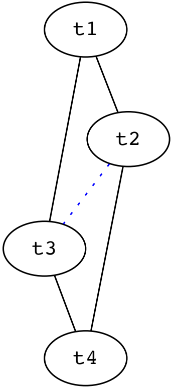
No need for a `t3` variable

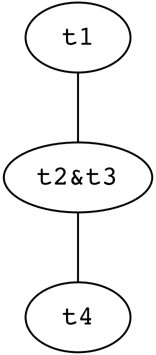
Coalescing Improves the Coloralibility



`t1` and `t4` have one neighbor less!

Coalescing Improves the Coloralibility





`t1` and `t4` have one neighbor less!

Conservative coalescing

Conservative Coalescing

Don't make it harder, i.e. don't produce nodes with higher degree

Coalesce a and b if

Briggs a&b has fewer than *k* neighbors of significant degree.

George every neighbor of a

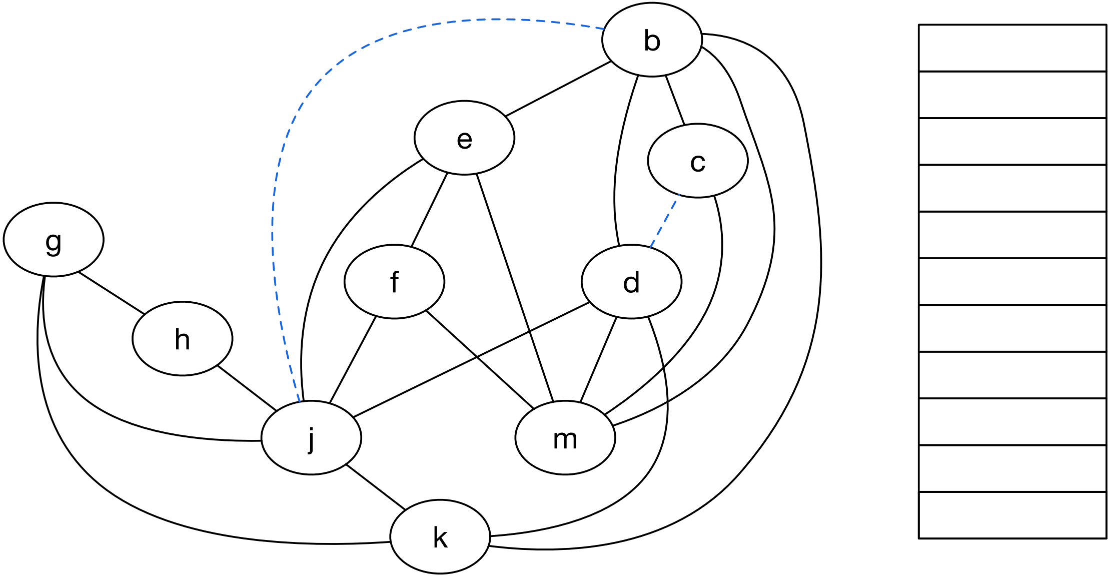
- is *either* of insignificant degree
- *or* already interfers with b

George's criterion is well suited for real registers

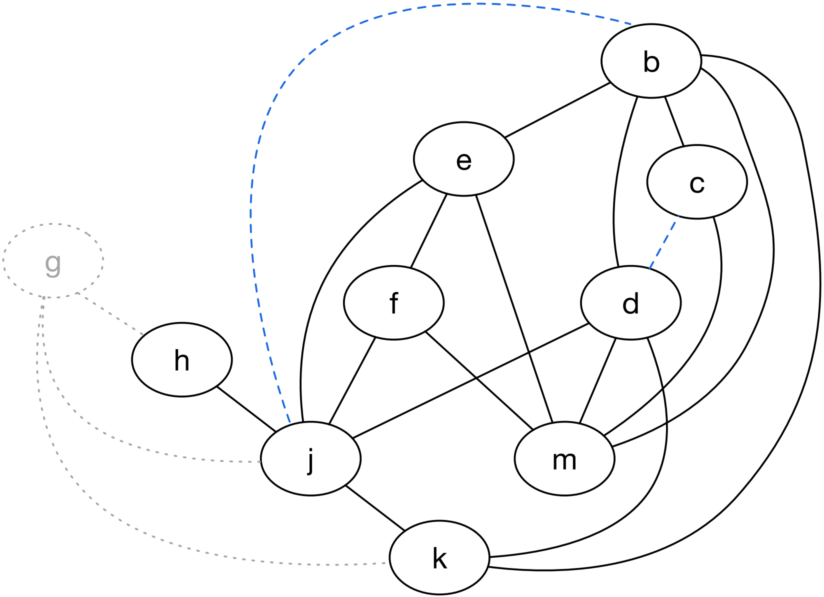
Full example

```
# live in: k j
    g := [j + 12]
    h := k - 1
    f := g * h
    e := [j + 8]
    m := [j + 16]
    b := [f]
    c := e + 8
    d := c
    k := m + 4
    j := b
# live out: d k j
```

Full example

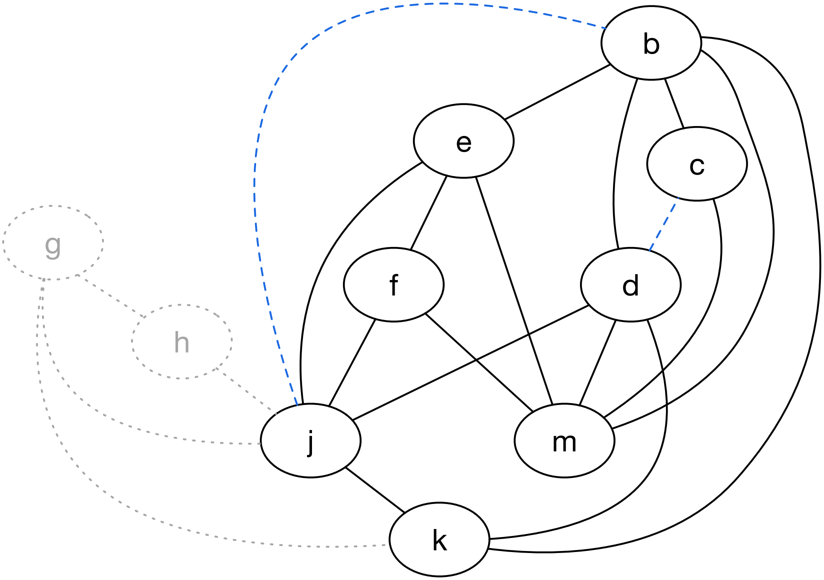


Full example



Remove g

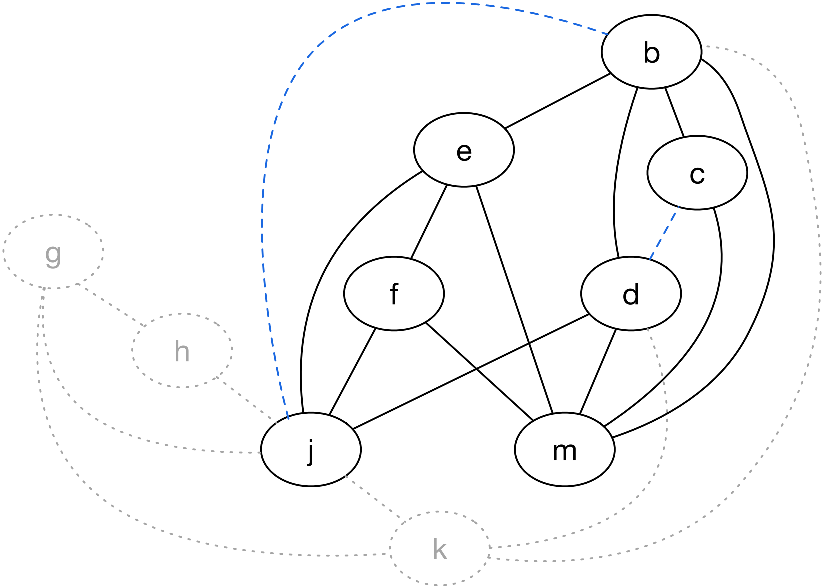
Full example



Remove h

Remove g

Full example

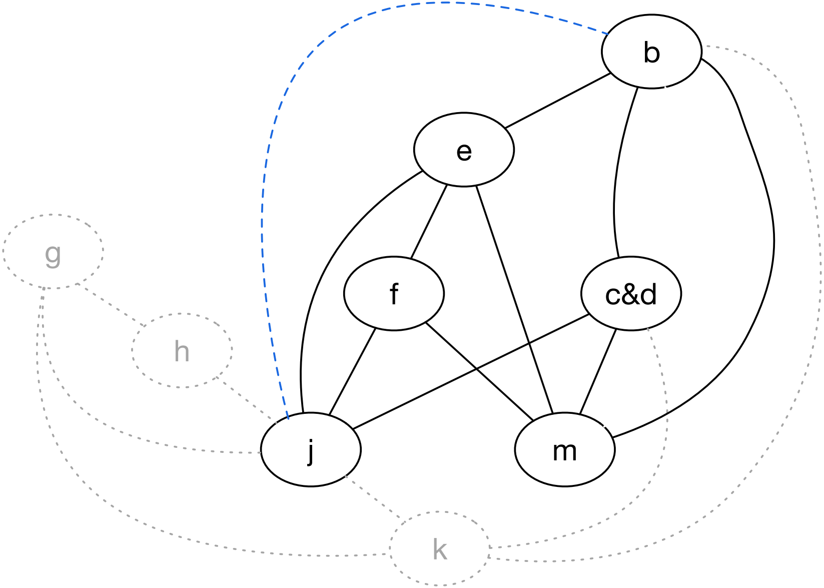


Remove k

Remove h

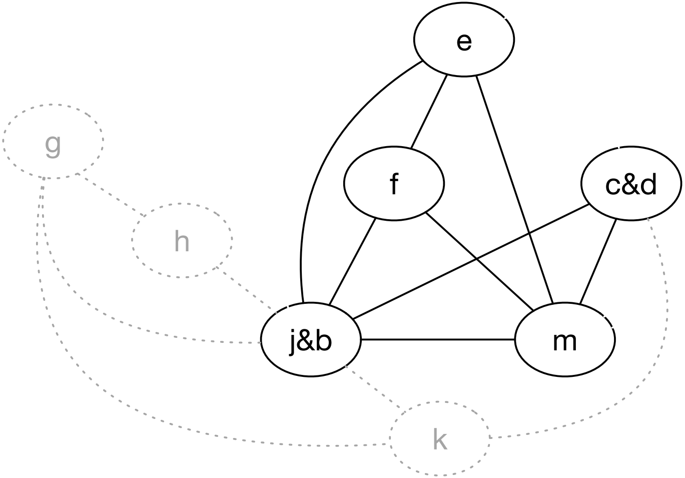
Remove g

Full example



Merge c&d
Remove k
Remove h
Remove g

Full example



Merge j&b

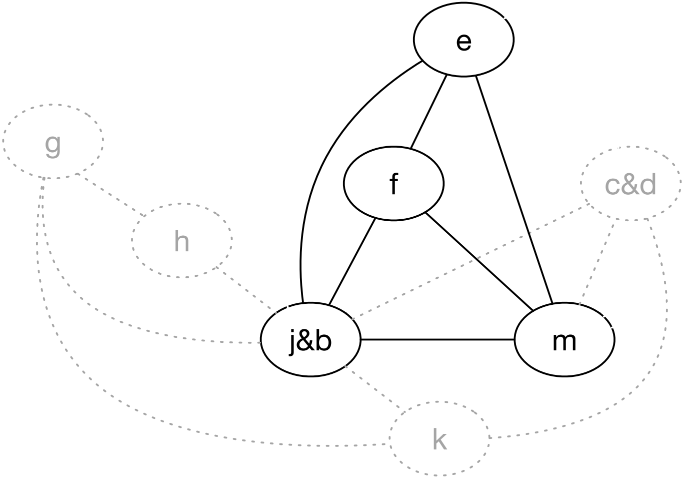
Merge c&d

Remove k

Remove h

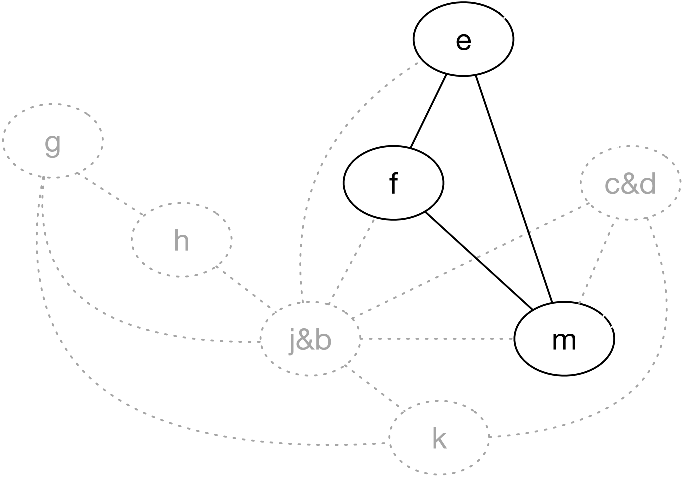
Remove g

Full example



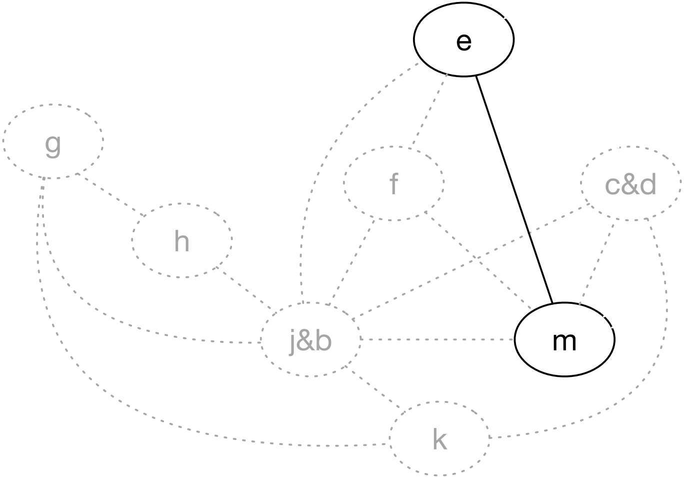
Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



Remove j&b
Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



Remove f

Remove j&b

Remove c&d

Merge j&b

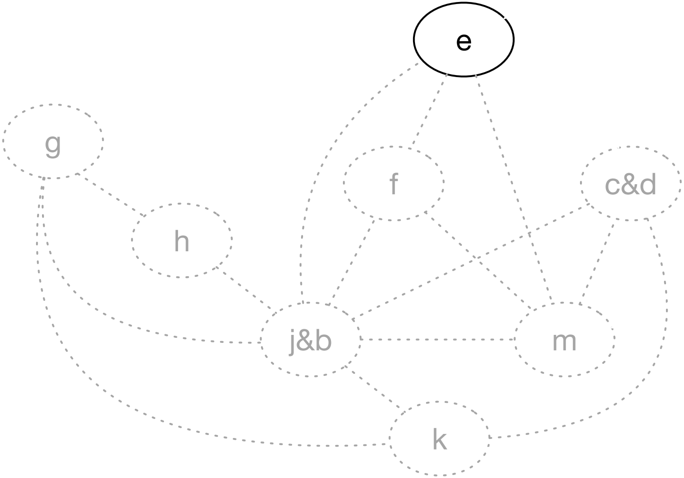
Merge c&d

Remove k

Remove h

Remove g

Full example



Remove m

Remove f

Remove j&b

Remove c&d

Merge j&b

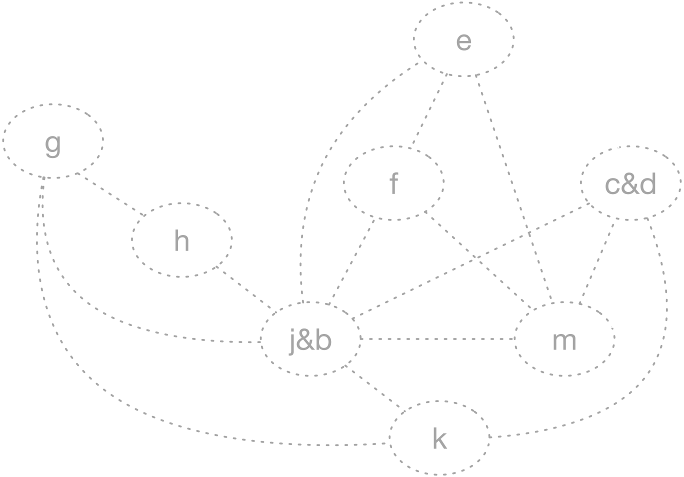
Merge c&d

Remove k

Remove h

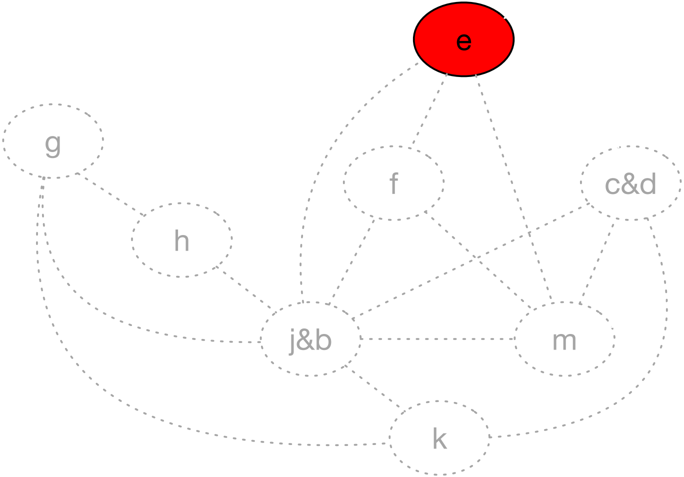
Remove g

Full example



Remove e
Remove m
Remove f
Remove j&b
Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



Remove m

Remove f

Remove j&b

Remove c&d

Merge j&b

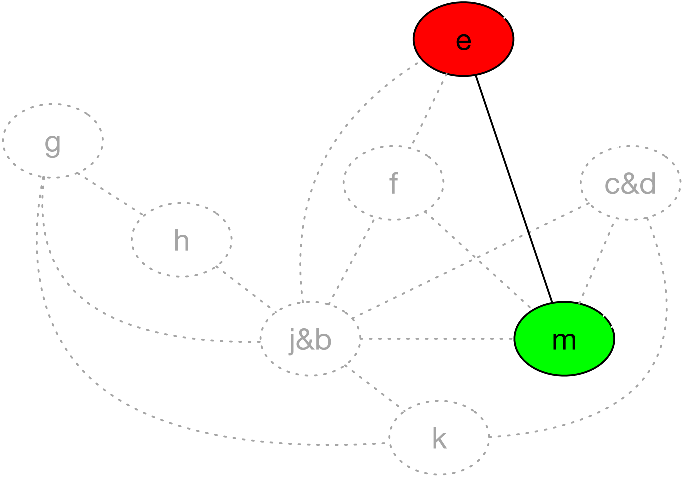
Merge c&d

Remove k

Remove h

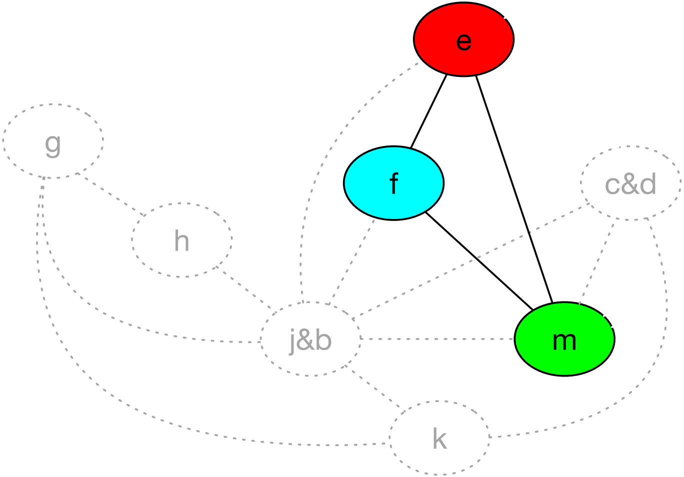
Remove g

Full example



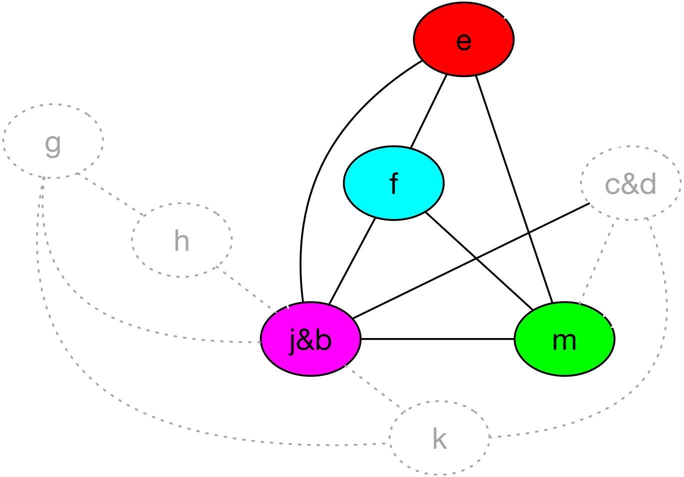
Remove f
Remove j&b
Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



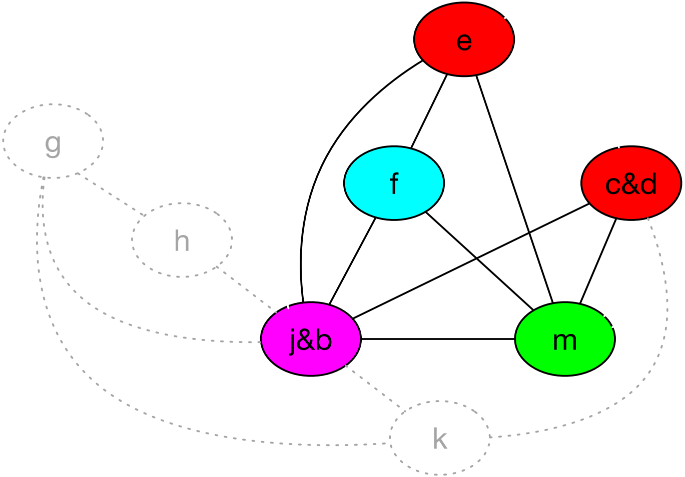
Remove j&b
Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



Remove c&d
Merge j&b
Merge c&d
Remove k
Remove h
Remove g

Full example



Merge j&b

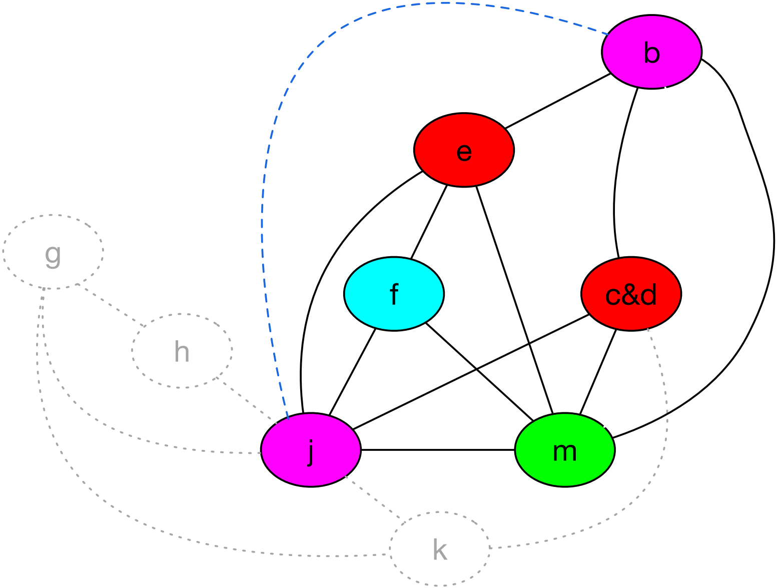
Merge c&d

Remove k

Remove h

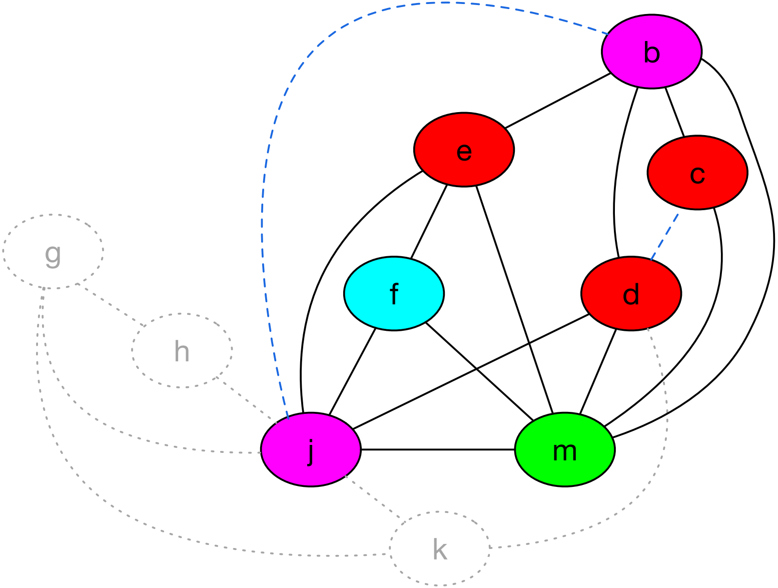
Remove g

Full example



Merge c&d
Remove k
Remove h
Remove g

Full example

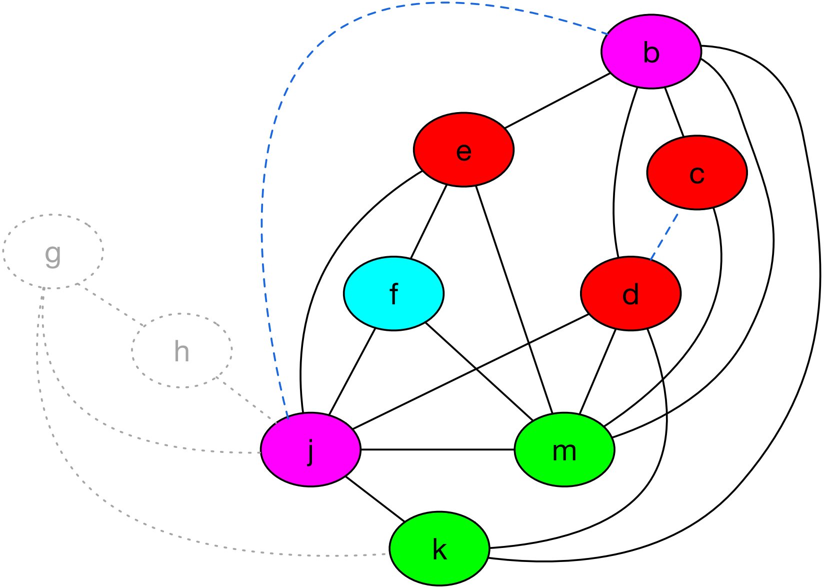


Remove k

Remove h

Remove g

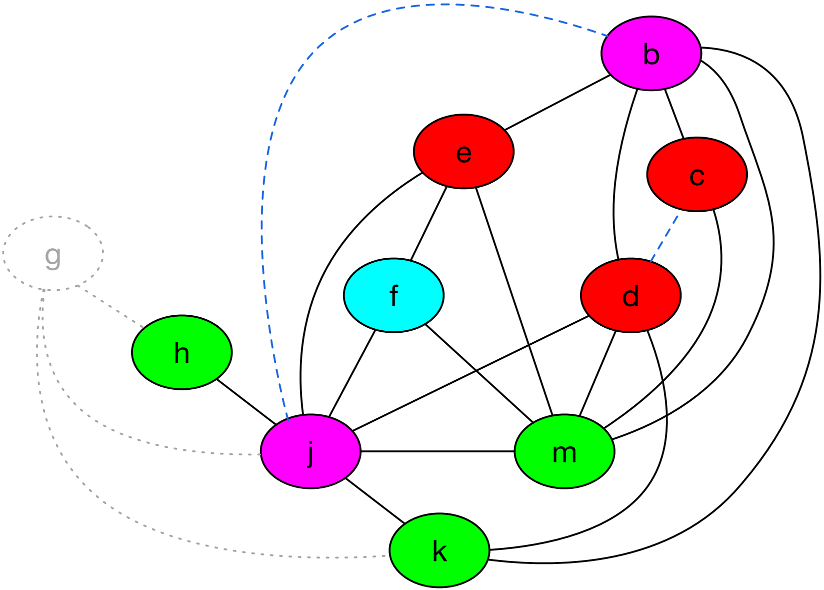
Full example

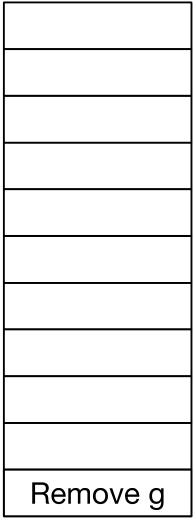


Remove h

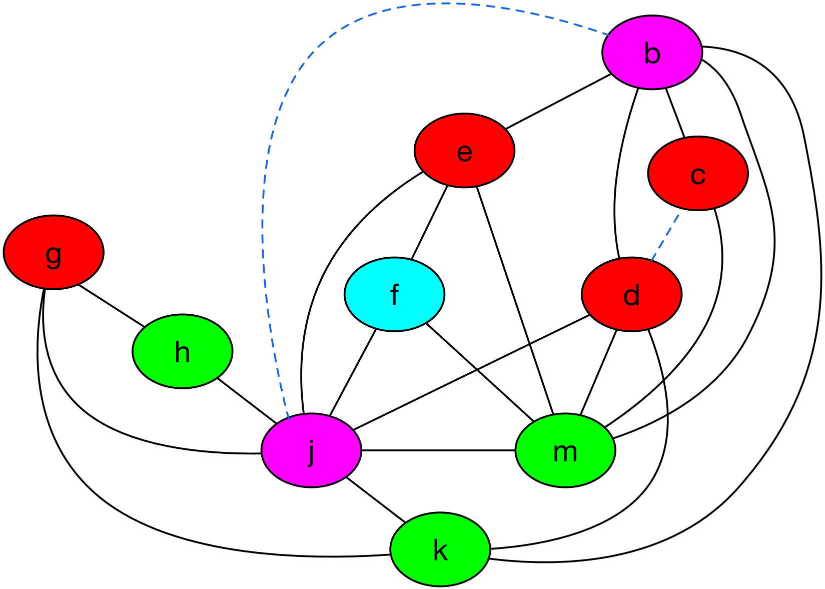
Remove g

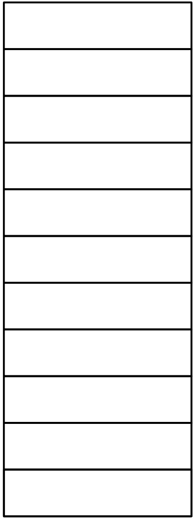
Full example




Remove g

Full example



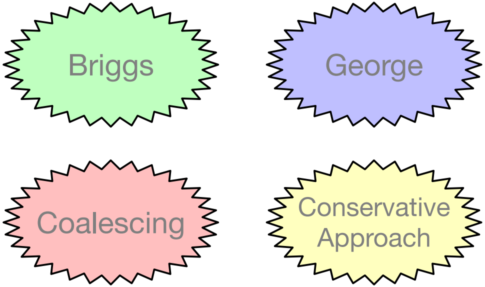


Interference Graph: Result

```
live in: k j
  g := [j + 12]
  h := k - 1
  f := g * h
  e := [j + 8]
  m := [j + 16]
  b := [f]
  c := e + 8
  d := c
  k := m + 4
  j := b
live out: d k j
```

```
live in: r2 r4
  r1 := [r4 + 12]
  r2 := r2 - 1
  r3 := r1 * r2
  r1 := [r4 + 8]
  r2 := [r4 + 16]
  r4 := [r3]
  r1 := r1 + 8
# r1 := r1
  r2 := r2 + 4
# r4 := r4
live out: r1 r2 r4
```

Summary



Briggs

George

Coalescing

Conservative Approach