The Recompute-Or-Store Alternative in reverse AD

Laurent Hascoët

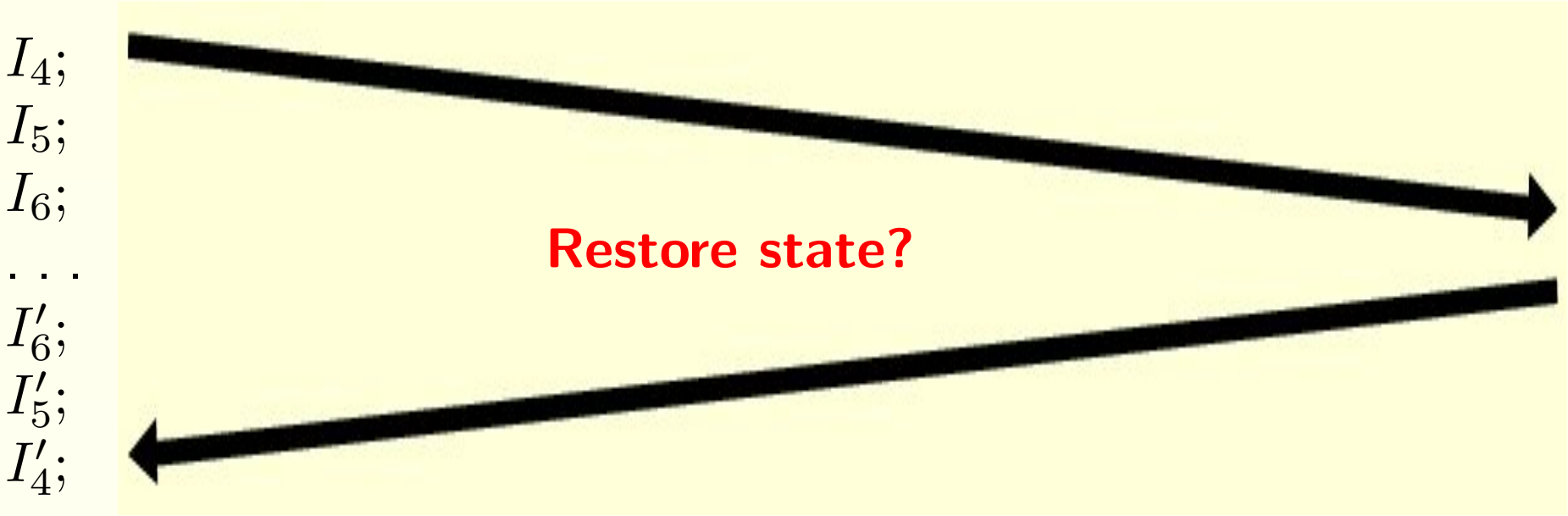
TROPICS Project, INRIA Sophia-Antipolis

AD2004, Chicago, July 19-23, 2004

Reverse AD

Time is too short to insist more on all the good sides of reverse AD...
...but let's spend some time on a key problem:

Reverse AD needs to restore memory states in the reverse of their normal appearance order.

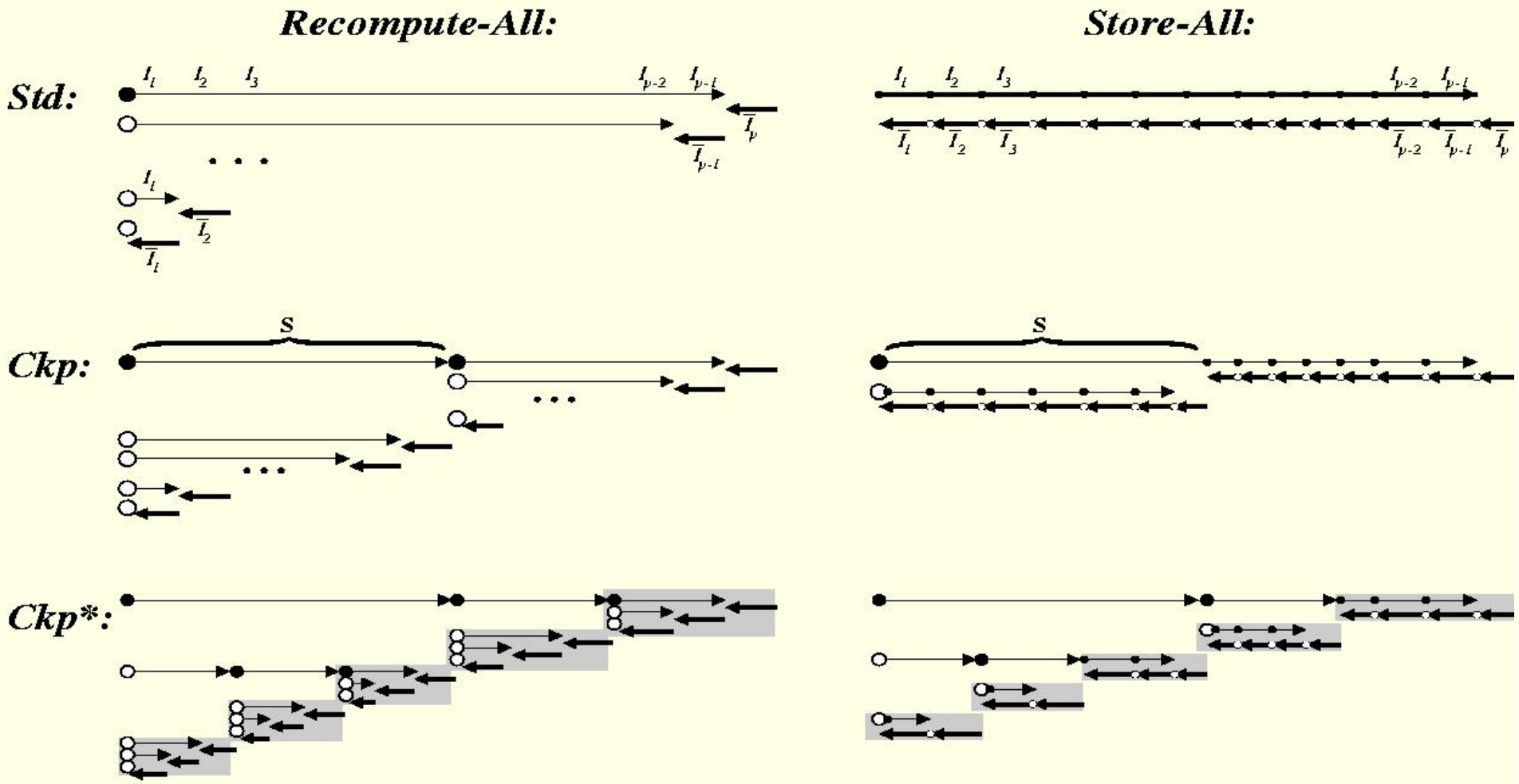


- Enough to keep us busy for a while!

Plan

- Restoring previous states is a central question.
- Answers exist: (Store-All or Recompute-All) and Checkpointing
- But we want to combine them !
- ROSA framework captures both strategies.
- Looking for Recompute-Or-Store Tradeoffs.
- Possible extensions of ROSA.

RA *vs* SA



- Checkpointing makes RA and SA grow alike,

but they are still different/incompatible in each fragment.

Combine RA and SA

- Sometimes RA is better:

```
Do i++                          Do i++
  j1 = N1(i)                      j1 ▶
...                               j1 = N1(i)
Do i--                          ...
  j1 = N1(i)                    Do i--
                                ...
                                  ▶ j1
```

RA ⇐ SA ⇒

- Sometimes SA is better:

```
x=exp1(x,y,z)                   x=exp1(x,y,z)
x=exp2(x,y,z)                   x ▶
...                             x=exp2(x,y,z)
x=exp1(x,y,z)                   ...
...                             ▶ x
x=exp1(x,y,z)                   ...
...
```

RA ⇐ SA ⇒

⇒ How can we unify and combine them?

The ROSA framework

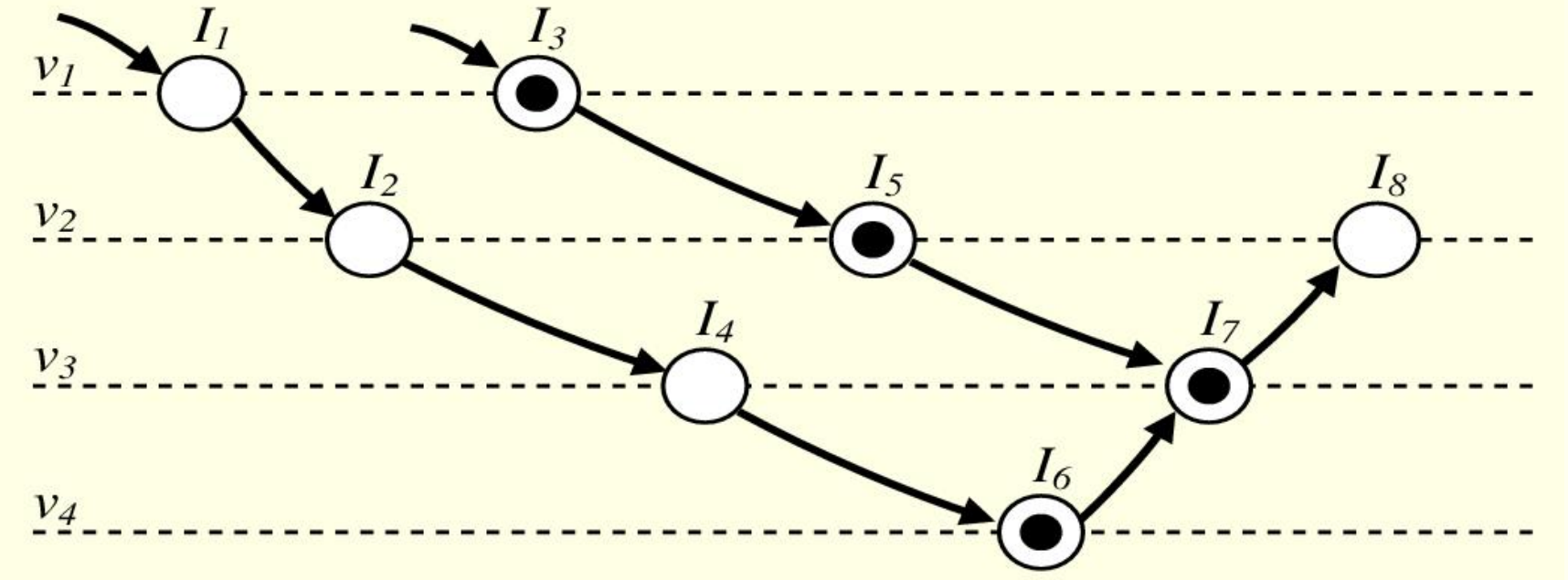
Restoration fights overwriting $\Rightarrow$ output dependences

Recomputation can be partial $\Rightarrow$ true dependences

Rescheduling instructions can help $\Rightarrow$ anti dependences

- ROSA Framework based on the ▶ data-dependence graph:

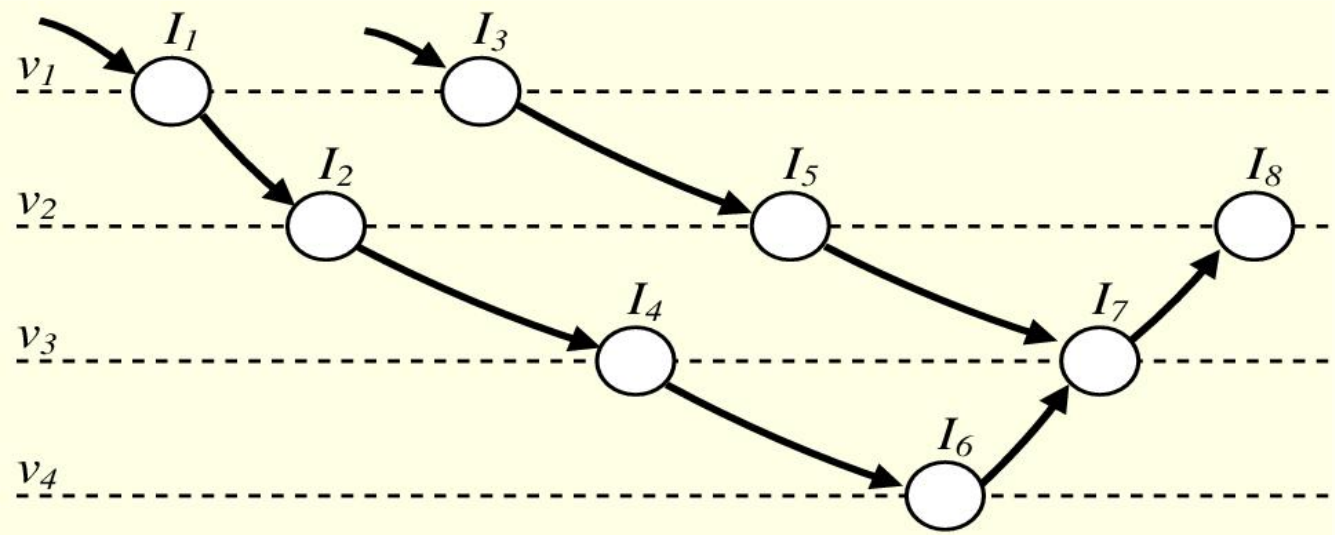
$$P = \{I_1; \quad I_2; \quad I_3; \quad I_4; \quad I_5; \quad I_6; \quad I_7; \quad I_8; \ \}$$



- Memory state represented by "exposed" nodes (black tokens).

Let's run it !

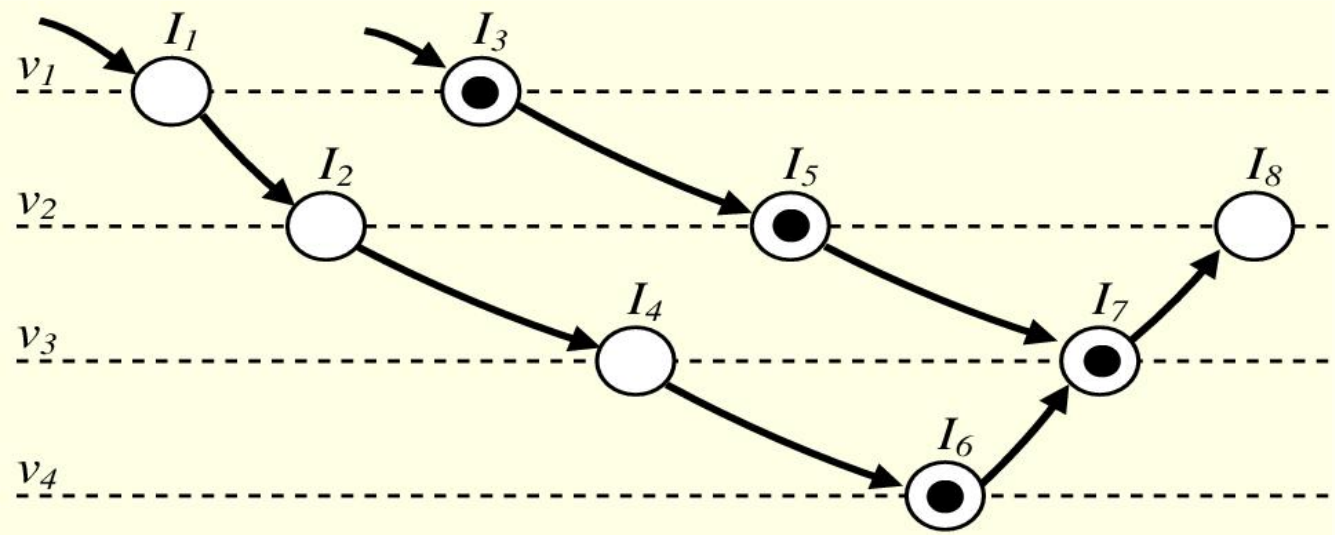
I'_8 ;



Let's run it !

I_1 ; I_2 ; I_3 ; I_4 ; I_5 ; I_6 ; I_7 ; I'_8 ;

I'_7 ;

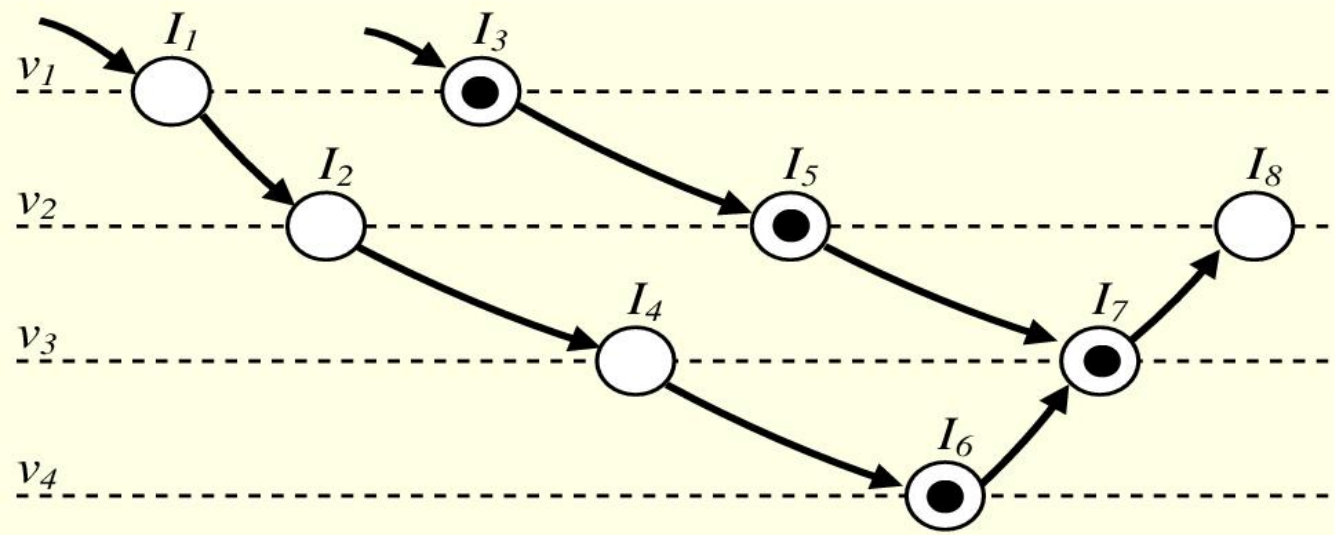


Let's run it !

I_1 ; I_2 ; I_3 ; I_4 ; I_5 ; I_6 ; I_7 ; I'_8 ;

I'_7 ;

I'_6 ;



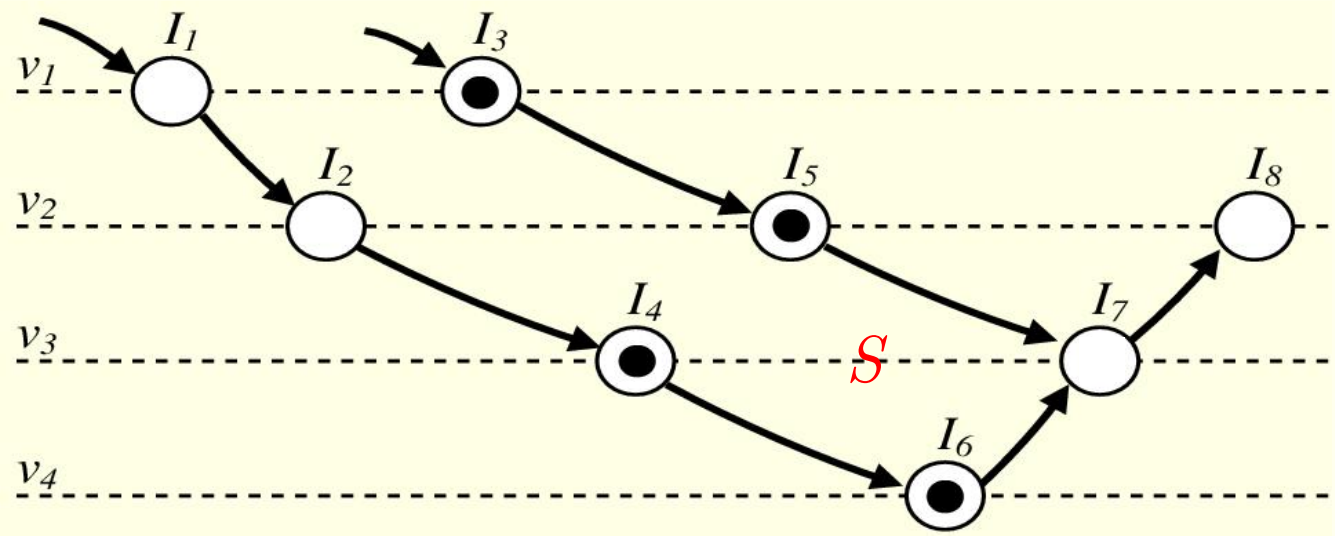
Let's run it !

$$I_1 \ ; \ I_2 \ ; \ I_3 \ ; \ I_4 \ ; \ I_5 \ ; \ I_6 \ ; \ v_3 \blacktriangleright ; \ I_7 \ ; \ I'_8 \ ;$$

$$I'_7 \ ;$$

$$\blacktriangleright \ v_3 \quad I'_6 \ ;$$

$$I'_5 \ ;$$



Let's run it !

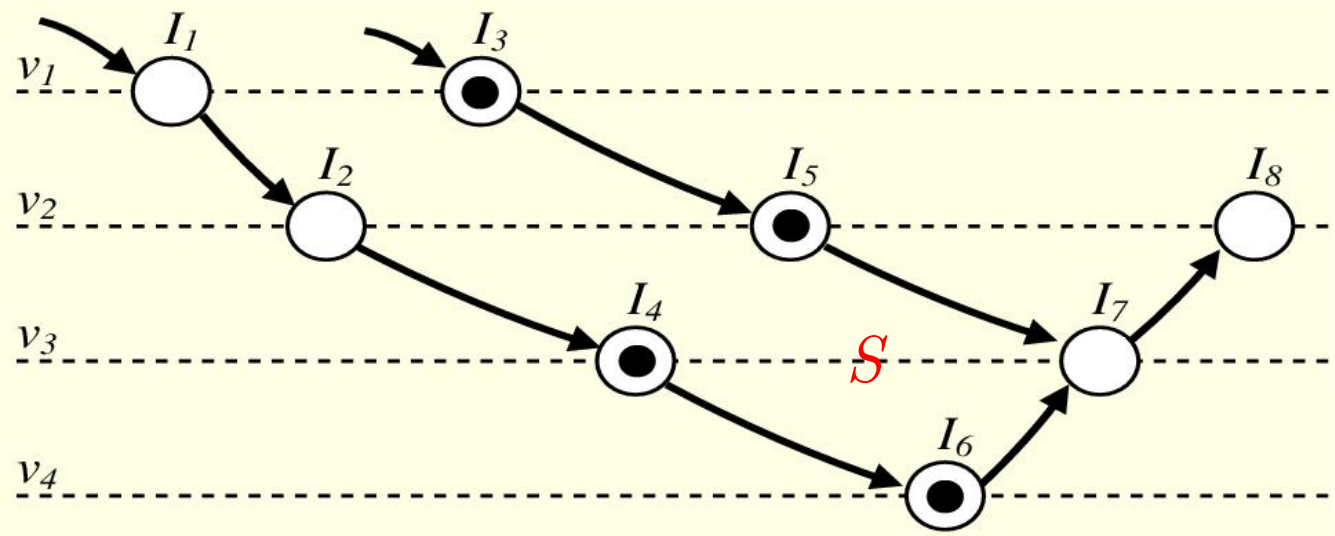
$$I_1 \ ; \ I_2 \ ; \ I_3 \ ; \ I_4 \ ; \ I_5 \ ; \ I_6 \ ; \ v_3 \blacktriangleright ; \ I_7 \ ; \ I'_8 \ ;$$

$$I'_7 \ ;$$

$$\blacktriangleright \ v_3 \quad I'_6 \ ;$$

$$I'_5 \ ;$$

$$I'_4 \ ;$$



Let's run it !

I_1 ; I_2 ; I_3 ; I_4 ; I_5 ; I_6 ; v_3 ▶; I_7 ; I'_8 ;

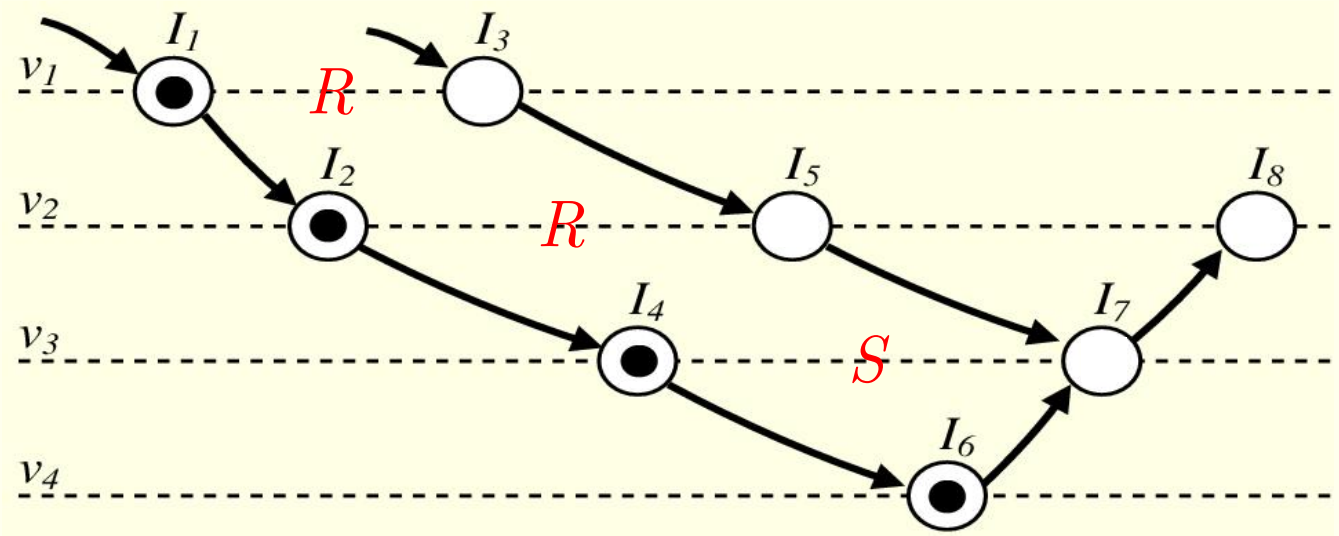
I'_7 ;

▶ v_3 I'_6 ;

I'_5 ;

I_1 ; I_2 ; I'_4 ;

I'_3 ;



Let's run it !

I_1 ; I_2 ; I_3 ; I_4 ; I_5 ; I_6 ; v_3 ▶; I_7 ; I'_8 ;

I'_7 ;

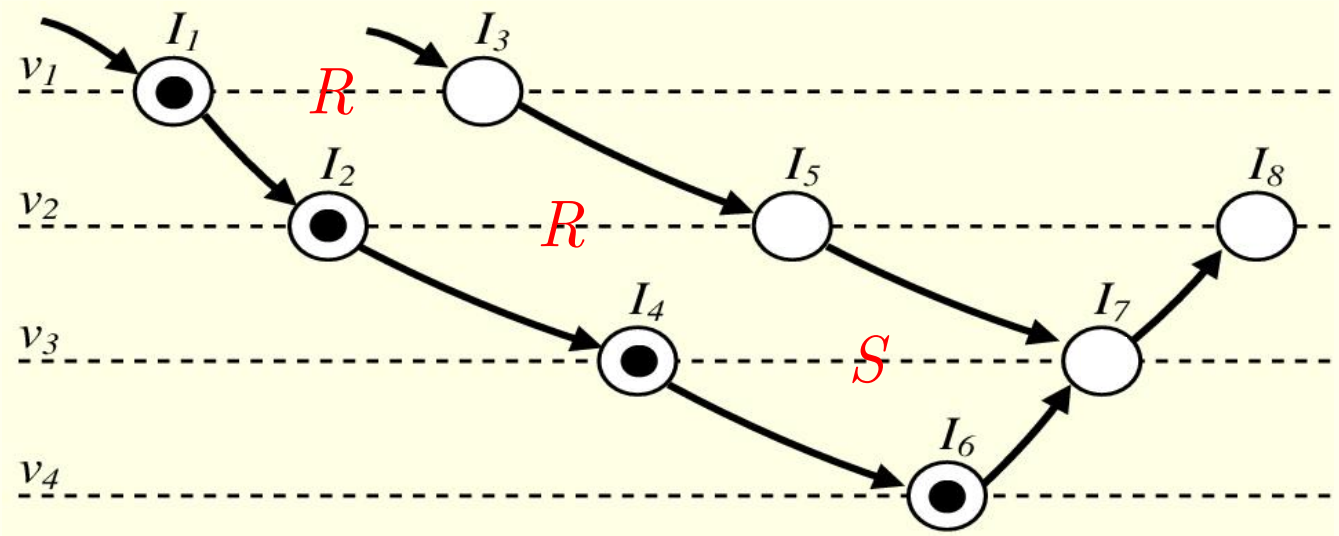
▶ v_3 I'_6 ;

I'_5 ;

I_1 ; I_2 ; I'_4 ;

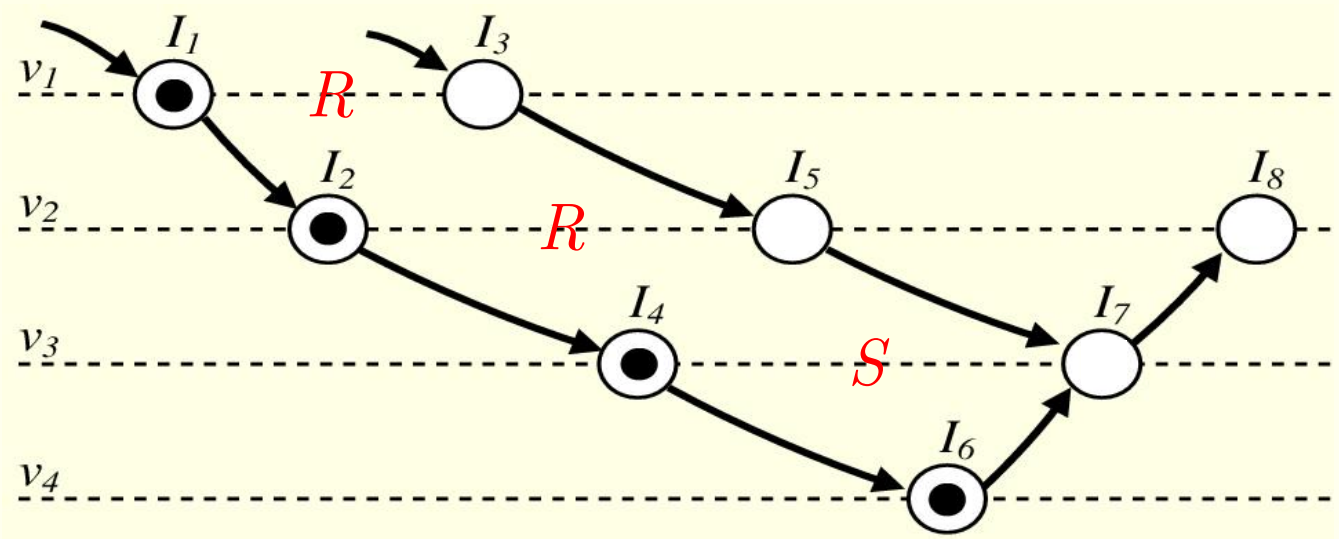
I'_3 ;

I'_2 ;



Let's run it !

$$I_1 \ ; \ I_2 \ ; \ I_3 \ ; \ I_4 \ ; \ I_5 \ ; \ I_6 \ ; \ v_3 \blacktriangleright ; \ I_7 \ ; \ I'_8 \ ;$$



$$I'_7 \ ;$$

$$\blacktriangleright v_3 \quad I'_6 \ ;$$

$$I'_5 \ ;$$

$$I_1 \ ; \ I_2 \ ; \ I'_4 \ ;$$

$$I'_3 \ ;$$

$$I'_2 \ ;$$

$$I'_1 \ ;$$

- Assign *Recompute* or *Store* to each output-dep

$\Rightarrow$ One ROSA strategy

- If all get *Recompute* $\Rightarrow$ RA with ERA.
- If all get *Store* $\Rightarrow$ SA with TBR.

First experiments

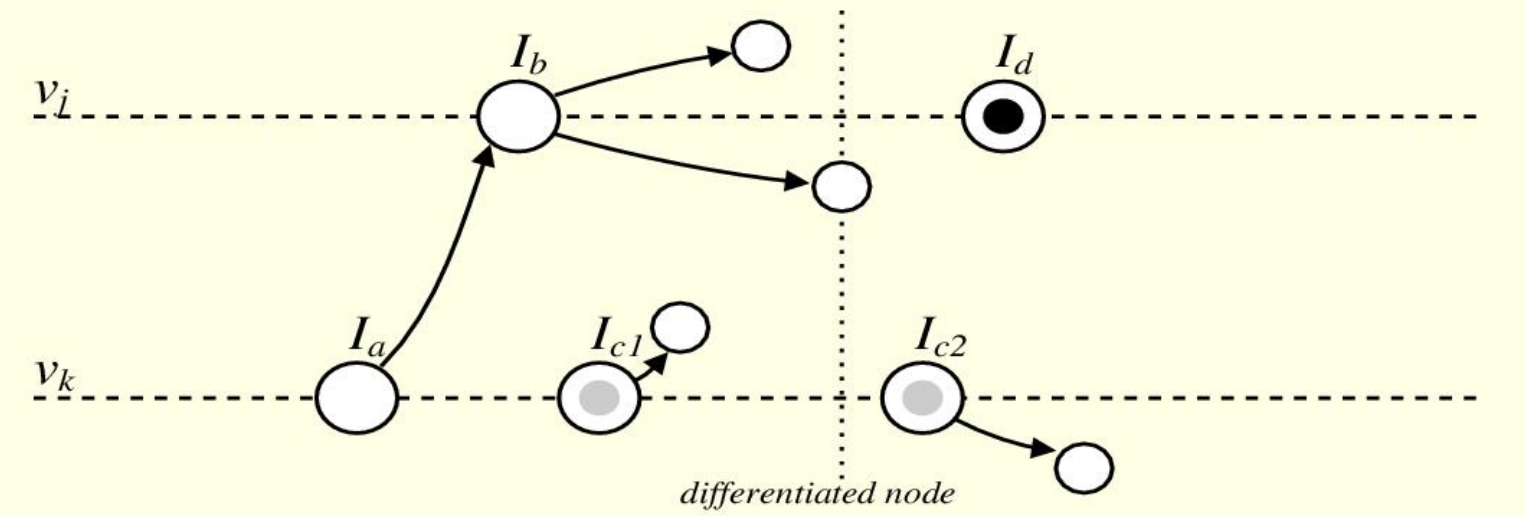
```
DO jt=1,nt
   DO k=1,4
      is = nu(k, jt)                    Recompute !
      uph(1, k) = ua(1, is)             Recompute !
      usro = 1.0d0/uph(1, k)            Recompute !
      ...
      cr2 = gam1*(uar5 - qir)/cr2       Store !
      ...
```

Storage:	9600	9500	8300	2800	200	0
Run time:	0.23	0.23	0.22	0.26	0.26	0.28

Looking for the optimal ROSA strategy

- ... is probably out of reach for large graphs.

- but we can define heuristics:

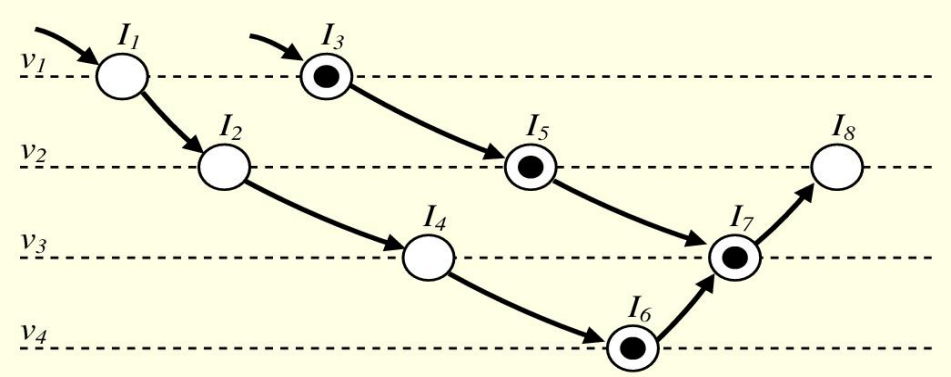


If recomputing v_j

- moves token into $I_a \leftarrow I_{c1}$: bad because I_{c1} required later
- moves token into $I_a \leftarrow I_{c2}$, ok because I_{c2} not required.

Conclusion and Further work

- ROSA framework captures both Recompute-All and Store-All strategies, plus a lot of tradeoffs in between.
- Also captures the TBR and ERA algorithms.
- Finding the optimal strategy is hard, but we can easily do better than RA and SA.
- Lots of work remaining !



- integrate checkpoints with ROSA
- allow for instructions reordering
- allow for local single-assignment transformation