Adjoint computation and Backpropagation

Julien Herrmann
Inria Center of Bordeaux, France

14th Scheduling for Large Scale Systems Workshop

Who's who

Automatic Differentiation

Paul Hovland (Argonne)

Navjot Kukreja (Imperial College)

Krishna Narayanan (Argonne)

Machine Learning (I)

Alexis Joly (Inria)

Machine Learning (II)

Alena Shilova (Inria)

Scheduling

Guillaume Pallez (Inria)

Olivier Beaumont (Inria)

Julien Herrmann (Inria)

Ice-sheet model (I)

"In climate modelling, Ice-sheet models use numerical methods to simulate the evolution, dynamics and thermodynamics of ice sheets." (wikipedia)

Model Algorithm (single timestep)

1. Evaluate driving stress $\tau_d = \rho g h \nabla s$
2. Solve for velocities

   DO i = 1, max_iter

   i. Evaluate nonlinear viscosity ν_i from iterate $\mathbf{u}_i$

   ii. Construct stress matrix $A\{\nu_i\}$

   iii. Solve linear system $A\, \mathbf{u}_{i+1} = \tau_d$

   iv. (Exit if converged)

   ENDDO
3. Evolve thickness (continuity eqn)

Automatic differentiation (AD) tools generate code for adjoint of operations

Credit: Daniel Goldberg

"In climate modelling, Ice-sheet models use numerical methods to simulate the evolution, dynamics and thermodynamics of ice sheets." (wikipedia)

Model Algorithm (single timestep)

1. Evaluate driving stress $\tau_d = \rho g h \nabla s$
2. Solve for velocities

   DO i = 1, max_iter

   i. Evaluate nonlinear viscosity ν_i from iterate $\boldsymbol{u}_i$

   ii. Construct stress matrix $A\{\nu\}$

   iii. Solve linear system $A\, \boldsymbol{u}_{i+1} = \boldsymbol{\tau_d}$

   iv. (Exit if converged)

   ENDDO
3. Evolve thickness (continuity eqn)

Automatic differentiation (AD) tools generate code for adjoint of operations

Simpler Version:

proc Model Algorithm($u_0, \boldsymbol{y}$)
begin
  Do stuff;
  for $i = 0$ *to* n **do**
    $\boldsymbol{u_{i+1} = f_i(u_i)}$;
    Do stuff;
  end
  /* $F(u_0) = f_n \circ f_{n-1} \circ \ldots \circ f_0(u_0)$ */
  Compute $\boldsymbol{\nabla F(u_0) y}$;
end

Credit: Daniel Goldberg

Ice-sheet model (II)

A quick reminder about the gradient:

$$F(u_0) = f_n \circ f_{n-1} \circ \ldots \circ f_1 \circ f_0(u_0)$$

$$\begin{aligned}\boldsymbol{\nabla} F(u_0)\boldsymbol{y} &= Jf_0(u_0)^T \cdot \boldsymbol{\nabla}(f_n \circ f_1)(u_1) \cdot \boldsymbol{y} \\ &= \boxed{Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}}\end{aligned}$$

$$\begin{aligned}Jf^T &= \text{ Transpose Jacobian matrix of } f; \\ u_{i+1} &= f_i(u_i) = f_i\left(f_{i-1} \circ \ldots \circ f_0(u_0)\right).\end{aligned}$$

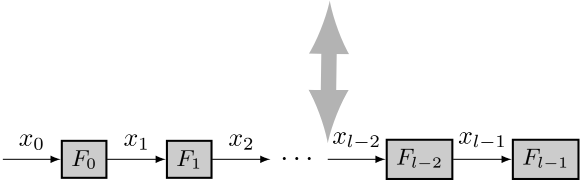
Adjoint computation

$$\begin{aligned} F_i(x_i) &= x_{i+1} \quad && i < l \\ \bar{F}_i(x_i, \bar{x}_{i+1}) &= \bar{x}_i \quad && i \leq l \end{aligned}$$

Adjoint computation

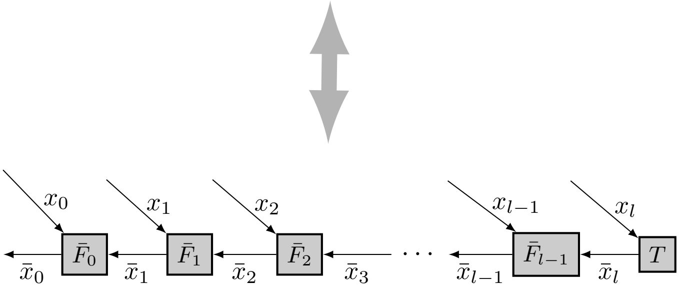
$$\boldsymbol{F_i(x_i) = x_{i+1} \quad i < l}$$

$$\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \qquad i \leq l$$



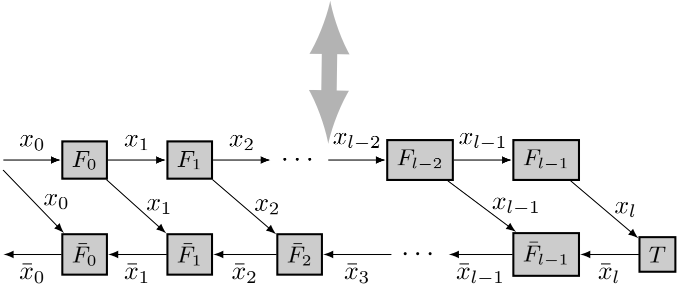
Adjoint computation

$$F_i(x_i) = x_{i+1} \quad i < l$$
$$\boldsymbol{\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \quad i \leq l}$$

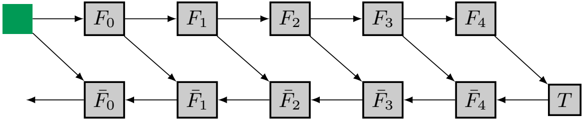


Adjoint computation

$$\begin{aligned} F_i(x_i) &= x_{i+1} \quad & i < l \\ \bar{F}_i(x_i, \bar{x}_{i+1}) &= \bar{x}_i \quad & i \leq l \end{aligned}$$



Model of execution

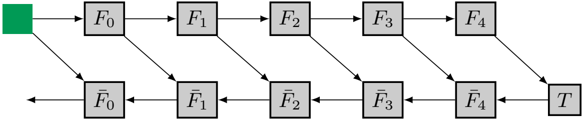


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

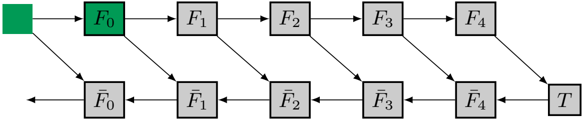


Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

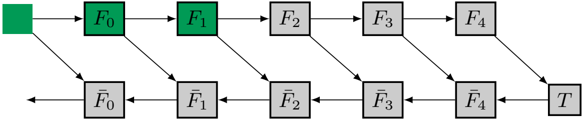
Model of execution



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

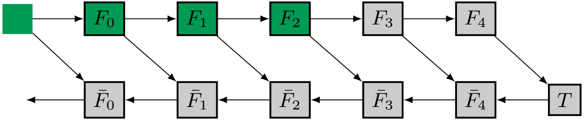


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

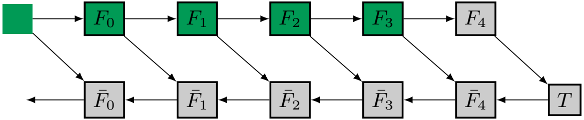
Model of execution



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

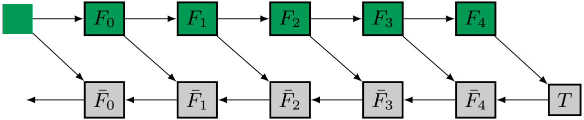


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

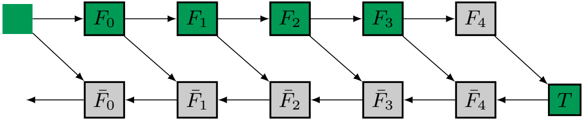
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

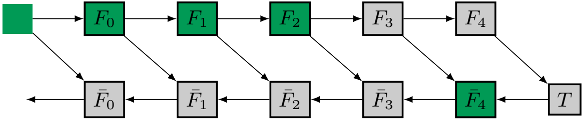
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



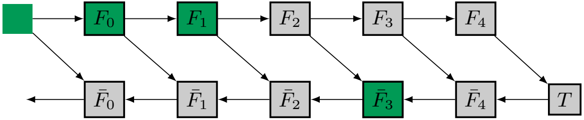
Peak Mem

6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



Peak Mem

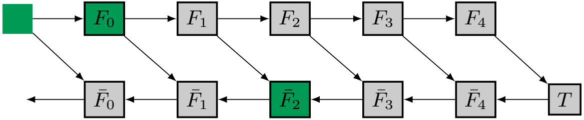
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

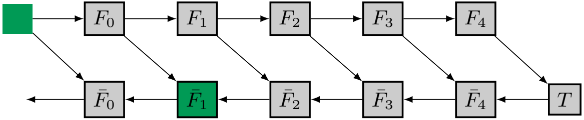
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



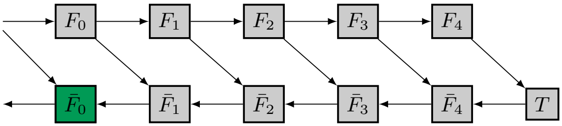
Peak Mem

6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



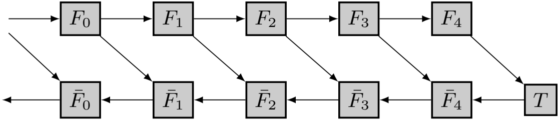
Peak Mem

6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



Peak Mem

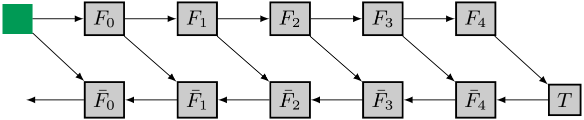
6

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



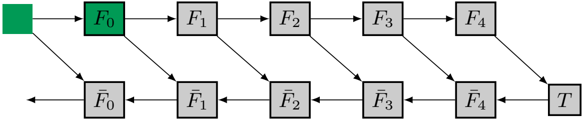
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

Model of execution



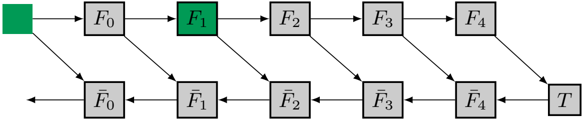
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

Model of execution

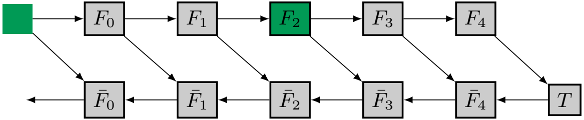


Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

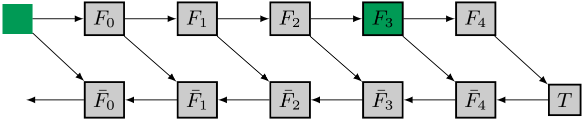


Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		



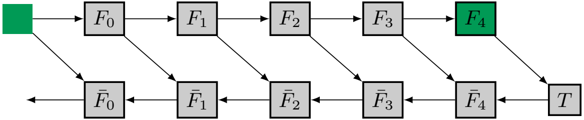
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

Model of execution



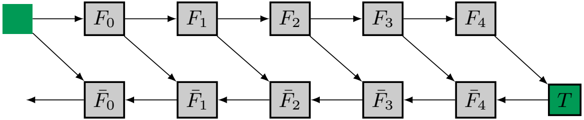
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

Model of execution

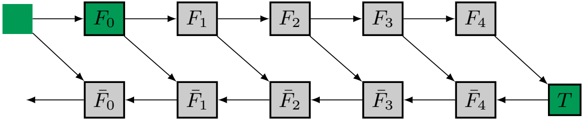


Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		



Peak Mem

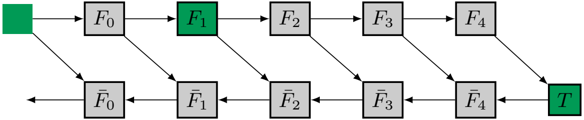
3

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

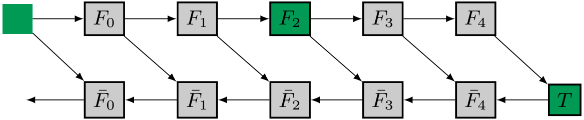
3

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



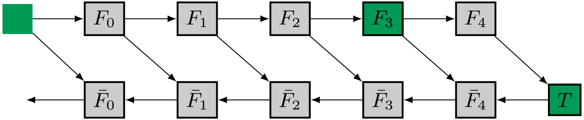
Peak Mem

3

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store “none”		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



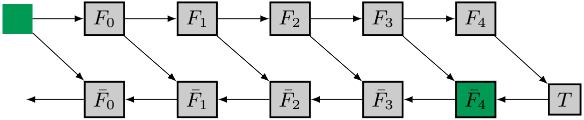
Peak Mem

3

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



Peak Mem

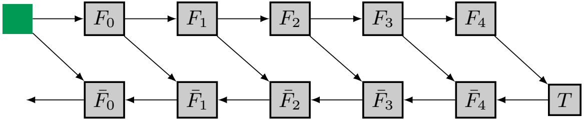
3

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$

Model of execution



Peak Mem

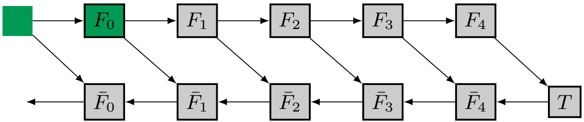
4

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store “none”	$$$	$
Checkpoint	$$	$$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

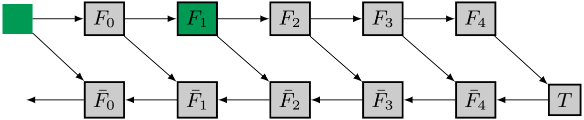
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store "none"	$$$	$
Checkpoint	$$	$$

Model of execution



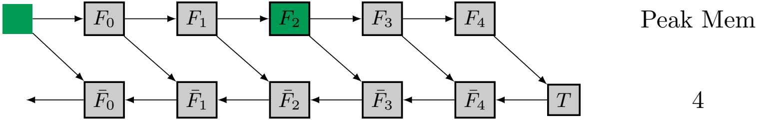
Peak Mem

4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

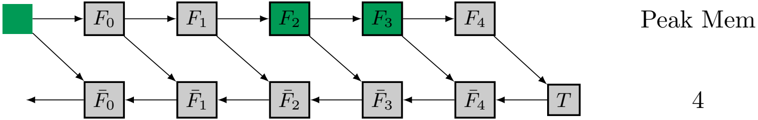
- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



Example of execution

Strategy	Time	Space
Store all	$	$$$
Store "none"	$$$	$
Checkpoint	$$	$$

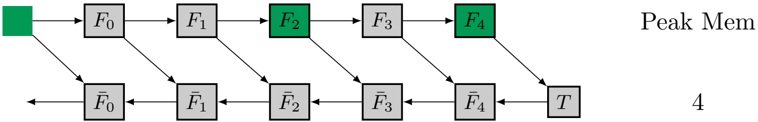
- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

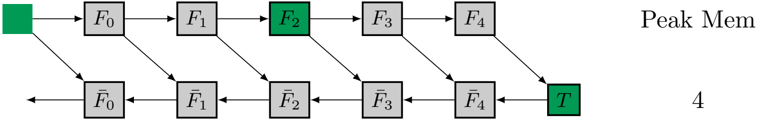


Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution

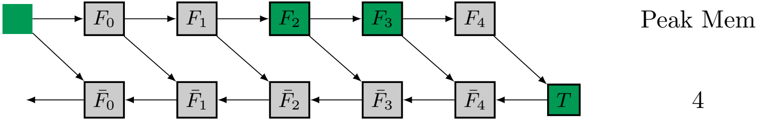


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

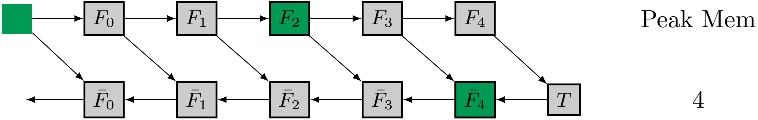
Model of execution



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

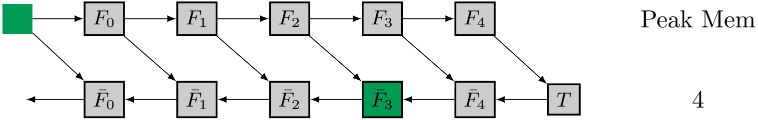
Strategy	Time	Space
Store all	$	$$$
Store "none"	$$$	$
Checkpoint	$$	$$



Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

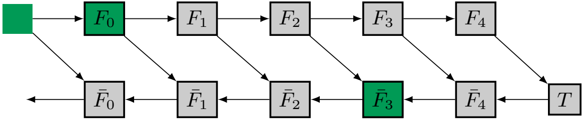


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store "none"	$$$	$
Checkpoint	$$	$$

Model of execution



Peak Mem

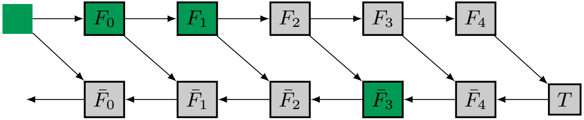
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



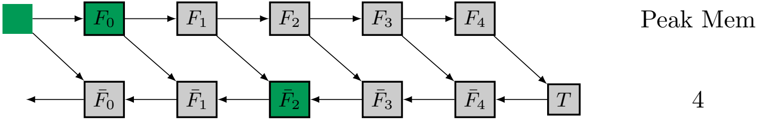
Peak Mem

4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store "none"	$$$	$
Checkpoint	$$	$$

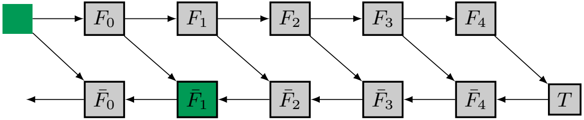


Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

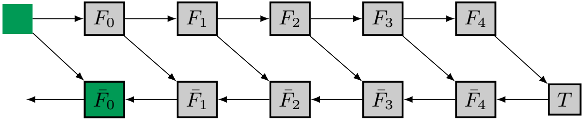
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



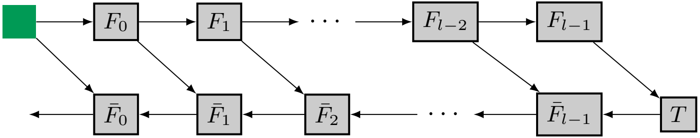
- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
Checkpoint	\$\$	\$\$

We want to minimize the makespan of:

		Initial state:
AC graph:	size l	
Steps:	u_f, u_b	
Memory:	$c_m,\ w_m = r_m = 0,$	$\mathcal{M}_{\text{ini}} = \{x_0\}$
Storage k:	$c_k, w_k, r_k,$	$S_{\text{ini}} = \emptyset$



Question: How to organize the reverse execution of intermediate steps? What do we store, what do we recompute?

- Store all: memory expensive
- Recompute all: compute expensive
- Intermediate status?

Bounded memory

Griewand and Walther, 2000: Revolve(l, c_m), optimal algorithm with c_m memory slots.

5

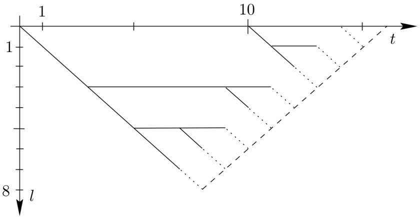


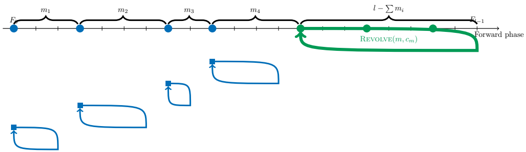
Figure 1.3: Example of Parallel Reversal Using 3 Checkpoints

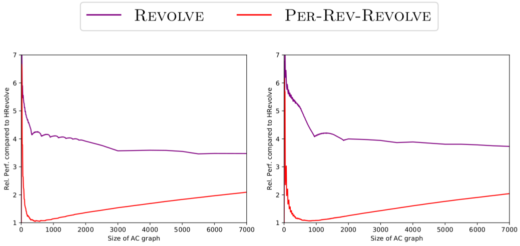
Source: Andrea Walther's PhD thesis, 1999

Aupy, Herrmann, Hovland, Robert, 2015: Optimal algorithm for two level of storage: cheap bounded memory and costly unbounded disks.

Aupy, Herrmann, 2019: Library of optimal schedules for any number of storage level.

(`https://gitlab.inria.fr/adjoint-computation`)



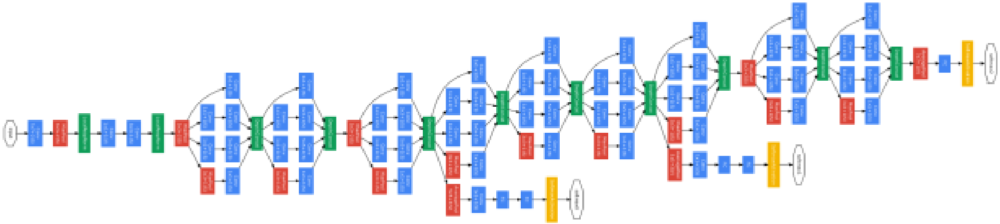


(a) *arch_1.txt*

(b) *arch_2.txt*

Fig. 5. Relative performance of the heuristics compared to the optimal solution on hierarchical platforms for large graph sizes.

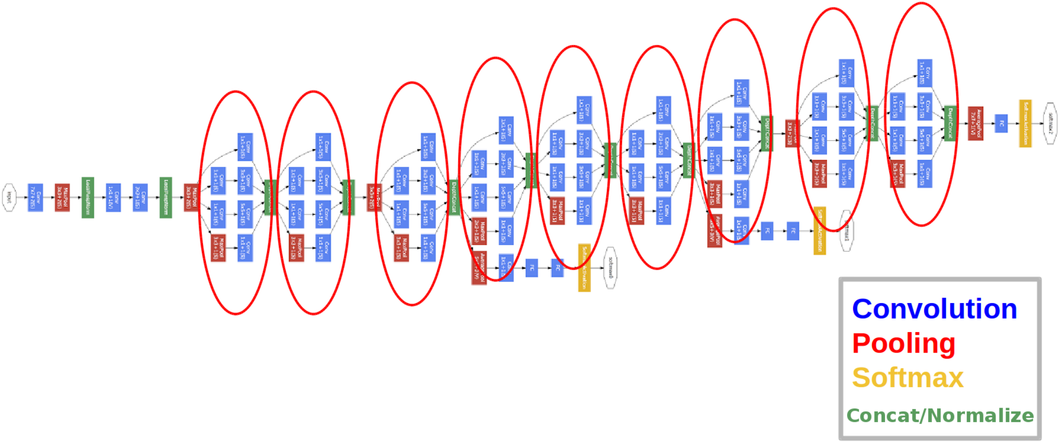
GoogleNet graph:



Source : Internet :s

Relation to IA? (I)

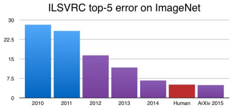
GoogleNet graph:



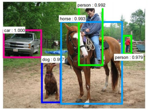
Derivatives in machine learning

“Backprop” and gradient descent are at the core of all recent advances

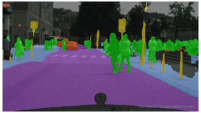
Computer vision



Top-5 error rate for ImageNet (NVIDIA devblog)

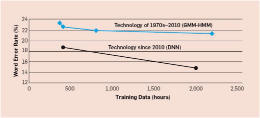


Faster R-CNN (Ren et al. 2015)



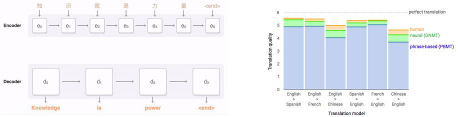
NVIDIA DRIVE PX 2 segmentation

Speech recognition & synthesis



Word error rates (Huang et al., 2014)

Machine translation



Google Neural Machine Translation System (GNMT)

4

What directions for AI?

While the core of the algorithms remain similar, the problematics are different:

- Shallower graphs ($O(100 - 1000)$ levels).
- Cost functions (time/memory) are not necessarily uniform.
- **Graphs with more structure than chains.**
- Multi-Learners/Hyperparameter tuning (independent graphs executed simultaneously), shared memory?
- Etc.

Dir. for AI: Graph structure II

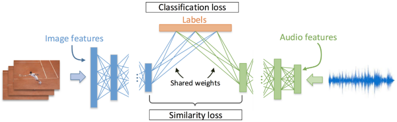


Fig. 2. Schematic of the used architecture.

Source: Surís et al., *Cross-Modal Embeddings for Video and Audio Retrieval*, 2018

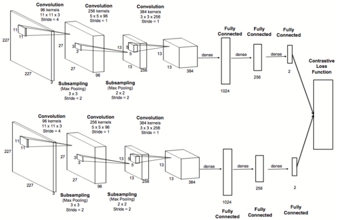
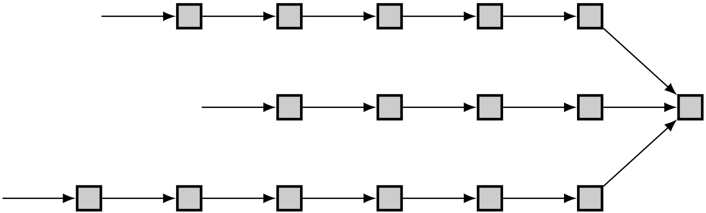


Figure 1: Siamese Neural Network Architecture

Source: Rao et al., *A Deep Siamese Neural Network (...)*, 2016

Pitchfork graph (aka join graphs):



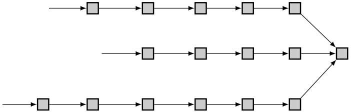
Theorem (Aupy, Beaumont, Herrmann, Shilova, 2019)

Given a bounded memory and a pitchfork with a bounded number of "teeth", we can find in polynomial time the solution that backpropagates it in minimal time.

A Grasp of the proof? (I)

Three phase algorithm:

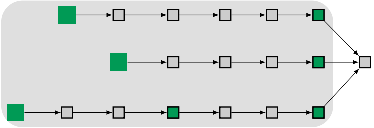
1. Forward phase
2. Turn
3. Backward phase



A Grasp of the proof? (I)

Three phase algorithm:

1. **Forward phase**
2. Turn
3. Backward phase

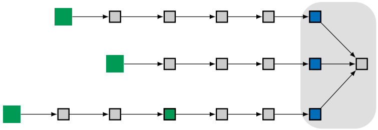


- Traverse all branches. Write some intermediate data

A Grasp of the proof? (I)

Three phase algorithm:

1. Forward phase
2. **Turn**
3. Backward phase

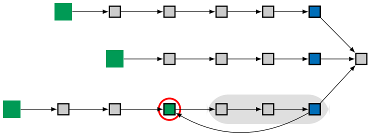


- Traverse all branches. Write some intermediate data
- *Backpropagate* the handle of the pitchfork

A Grasp of the proof? (I)

Three phase algorithm:

1. Forward phase
2. Turn
3. **Backward phase**



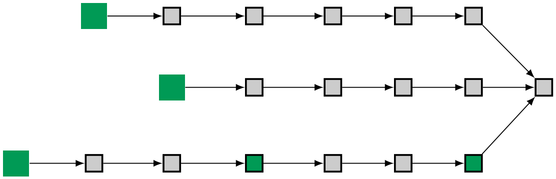
- Traverse all branches. Write some intermediate data
- *Backpropagate* the handle of the pitchfork
- Iteratively, read some checkpointed data from one of the branches, backpropagate a subset of the graph (can write additional intermediate data)

A Grasp of the proof? (II)

It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.



A Grasp of the proof? (II)

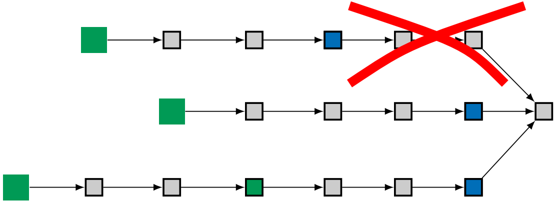
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Stability 1)

If F_i is "backpropagated", then there are no F_j for $i \leq j$.



A Grasp of the proof? (II)

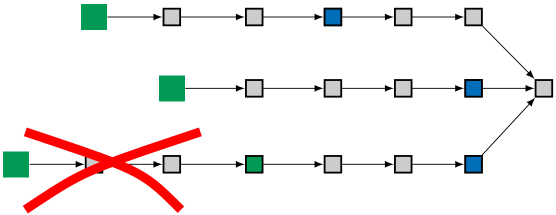
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Checkpoint persistence)

If x_i is stored, until F_i is "backpropagated", there are no F_j for $j < i$.



A Grasp of the proof? (II)

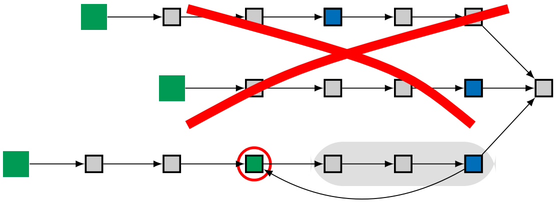
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Stability 2)

If x_i is read, then there are no F_j on other branches until it is backpropagated.



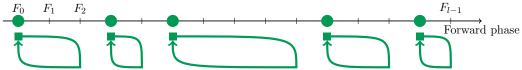
A Grasp of the proof? (II)

It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

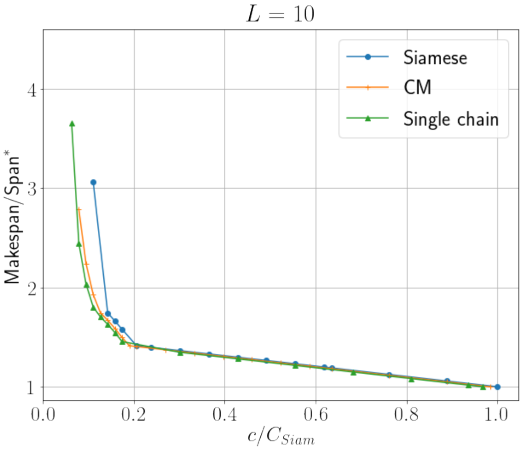
which give us a multi-phase approach.

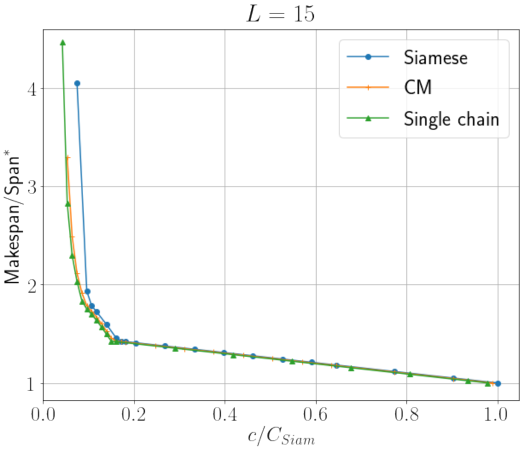
In this case, for a given forward phase, we get a multi-phase backward phase:



- Where do we schedule the checkpoints in the forward phase?
- In which order do we execute the subsegment on each branch?

Makespan overhead





Is it worth it?

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)

- From an adjoint perspective: **Yes!**
  With a memory of size $O(M)$:
  - *Store All* can execute a graph of size $O(M)$ in time $O(M)$;
  - *Revolve* can execute a graph of size $O(e^M)$ in time $O(Me^M)$!
  - *H-Revolve* inproves performance by a factor of magnitude.

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)
- From an adjoint perspective: **Yes!**
  With a memory of size $O(M)$:
  - *Store All* can execute a graph of size $O(M)$ in time $O(M)$;
  - *Revolve* can execute a graph of size $O(e^M)$ in time $O(Me^M)$!
  - *H-Revolve* inproves performance by a factor of magnitude.
- Machine Learning perspective: deeper networks!

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)
- From an adjoint perspective: **Yes!**
  With a memory of size $O(M)$:
  - *Store All* can execute a graph of size $O(M)$ in time $O(M)$;
  - *Revolve* can execute a graph of size $O(e^M)$ in time $O(Me^M)$!
  - *H-Revolve* inproves performance by a factor of magnitude.
- Machine Learning perspective: deeper networks!

Thanks