Efficient Implicit Parallel Patterns for Geographic Information System

Kevin Bourgeois, Sophie Robert, Sébastien Limet, Victor Essayan

June 21, 2017

LIFO/Géo-hyd

Table of contents

Introduction

Context

- Large volumes of heterogeneous data
- Improved models and simulations

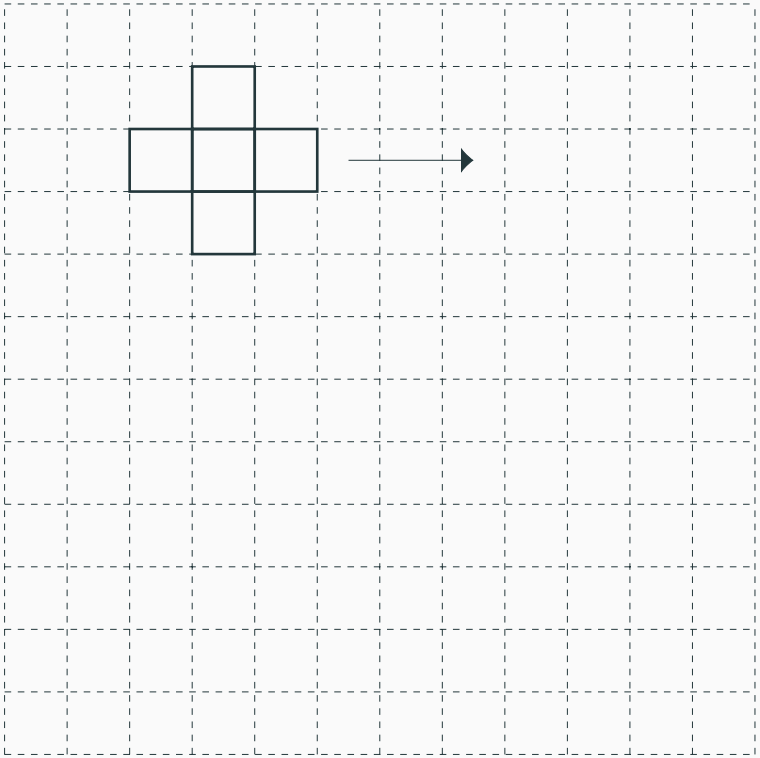
Context

- Large volumes of heterogeneous data
- Improved models and simulations
- Runtime increased

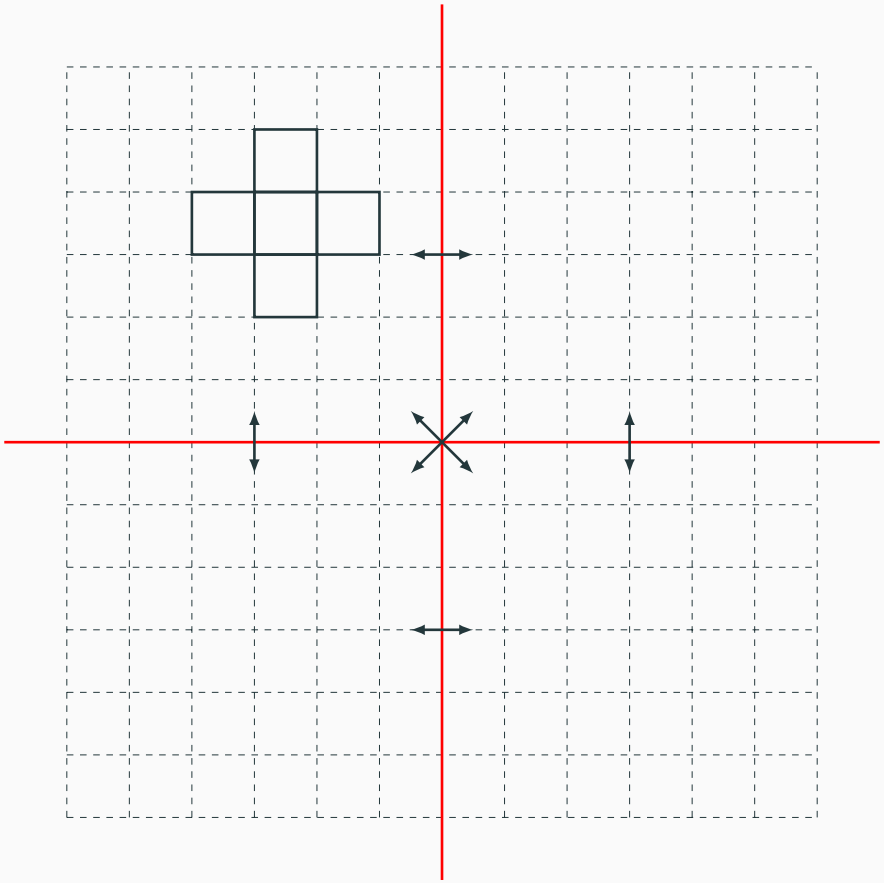
Context

- Large volumes of heterogeneous data
- Improved models and simulations
- Runtime increased
- Parallelization is essential

Sequential is easy



Hard to parallelize



Any solutions ?

Hard to parallelize and so what ?

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use
  - Add new functions need experts

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use
  - Add new functions need experts
- Patterns/Skeletons

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use
  - Add new functions need experts
- Patterns/Skeletons
  - Customizable by the user which provides the function to apply

Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use
  - Add new functions need experts
- Patterns/Skeletons
  - Customizable by the user which provides the function to apply
  - Can be composed basic skeletons to create complex skeletons

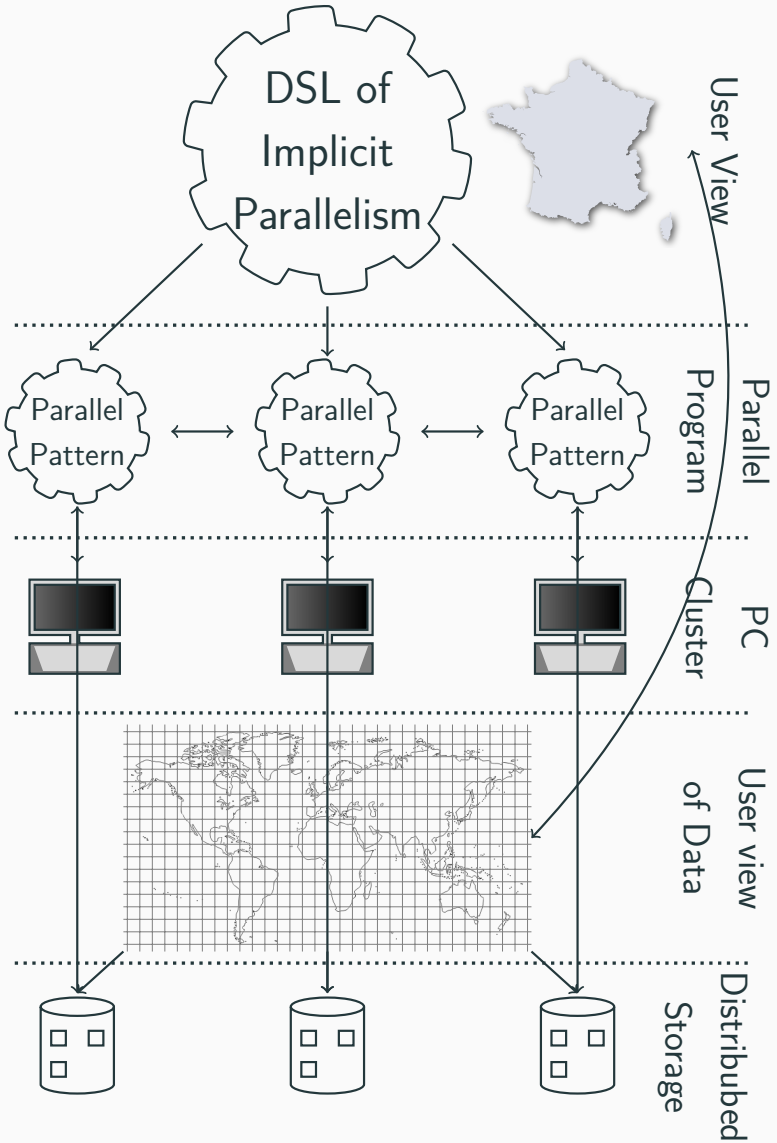
Any solutions ?

Hard to parallelize and so what ?

- Ask to a HPC-expert
  - Maybe the best way to get an optimized program
  - Communication is not always easy
  - The scientist depends a lot on the expert $\rightarrow$ Durabilty ?
- Parallel libraries
  - More or less easy to use
  - Add new functions need experts
- Patterns/Skeletons
  - Customizable by the user which provides the function to apply
  - Can be composed basic skeletons to create complex skeletons
  - Limited to the number of implemented skeletons

Our goals

We have design a new layer based parallel framework.



The watershed study

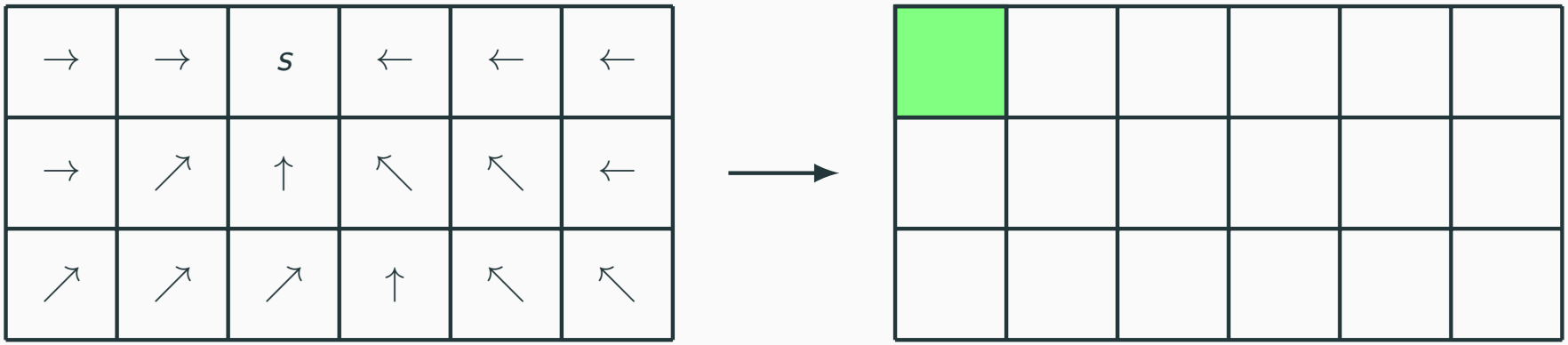
The watershed computation is a very common task in GIS domain that consists in several steps :

- The flow directions $\rightarrow$ stencil pattern
- The flow accumulations and the watershed labelization $\rightarrow$ difficult to predict and to parallelize

The parallelization of these algorithms allows us to extract a new pattern : the *Pre-Determinated Dependencies* (PDD) pattern

PDD Pattern

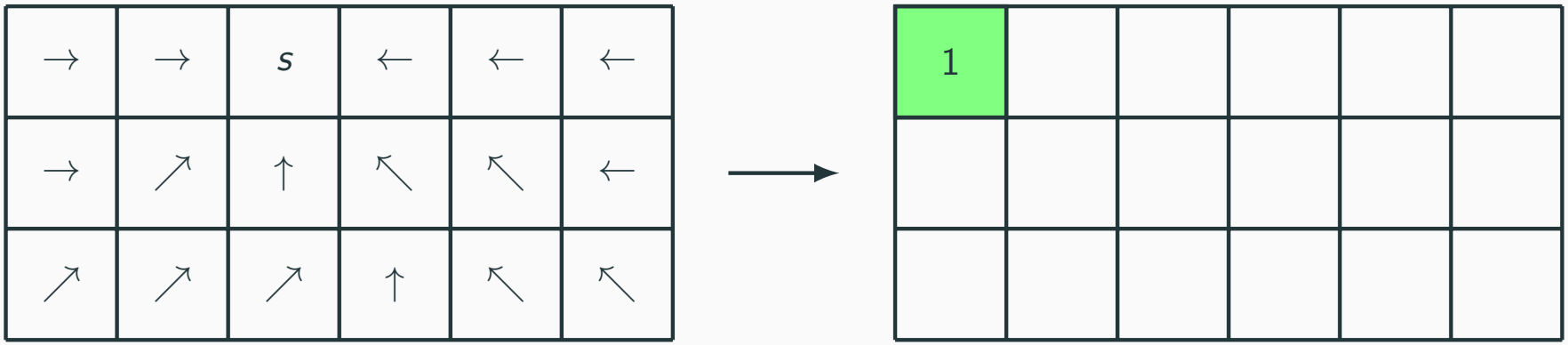
Study case : The flow accumulations



Flow directions

Flow accumulations

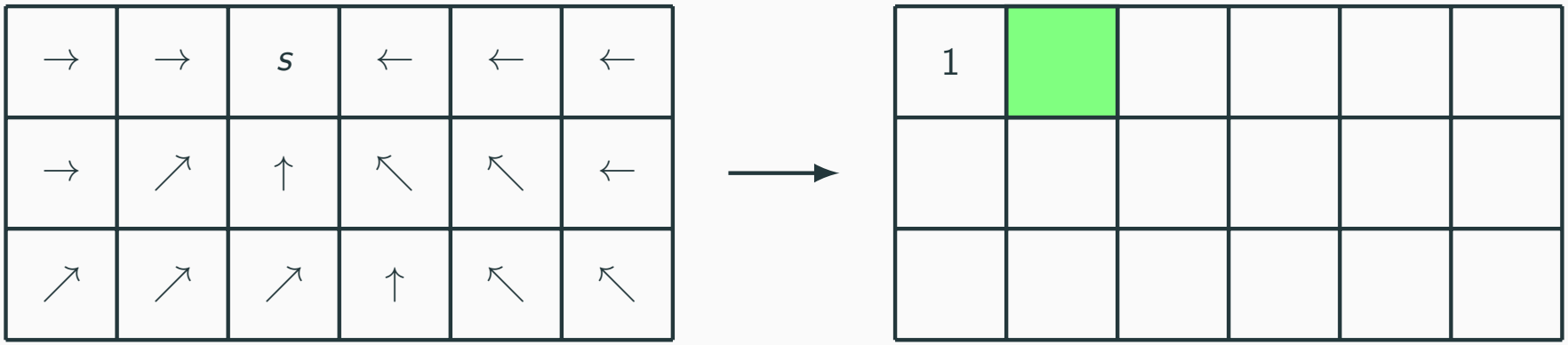
Study case : The flow accumulations



Flow directions

Flow accumulations

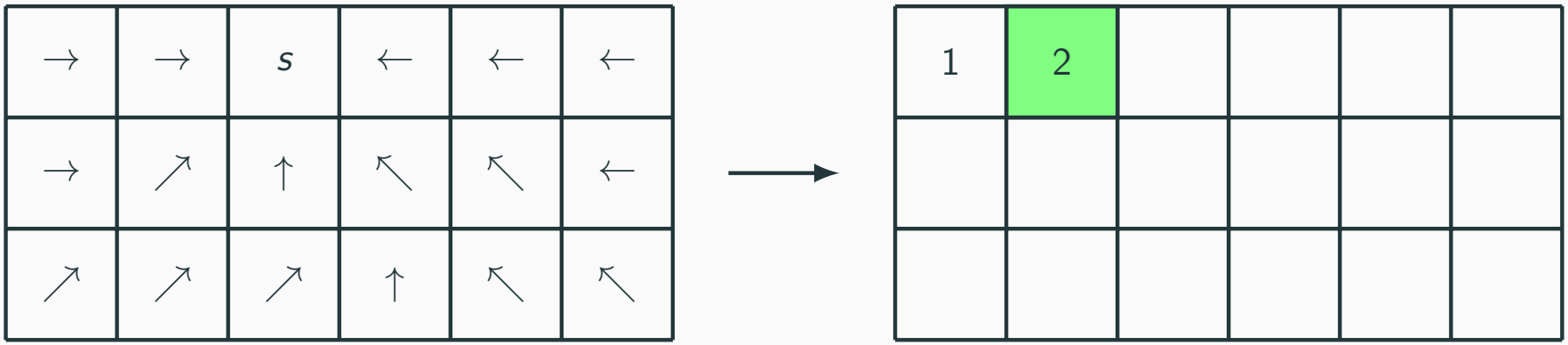
Study case : The flow accumulations



Flow directions

Flow accumulations

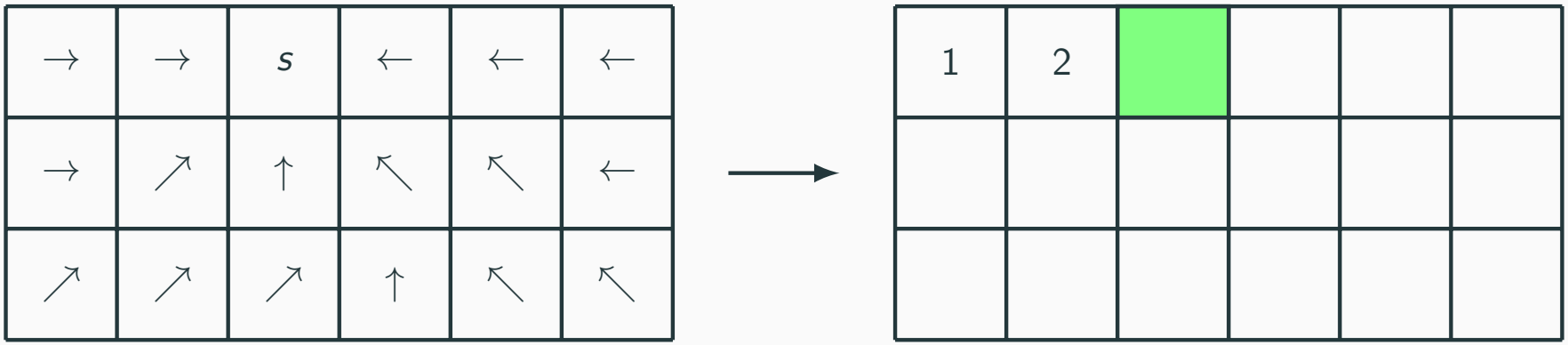
Study case : The flow accumulations



Flow directions

Flow accumulations

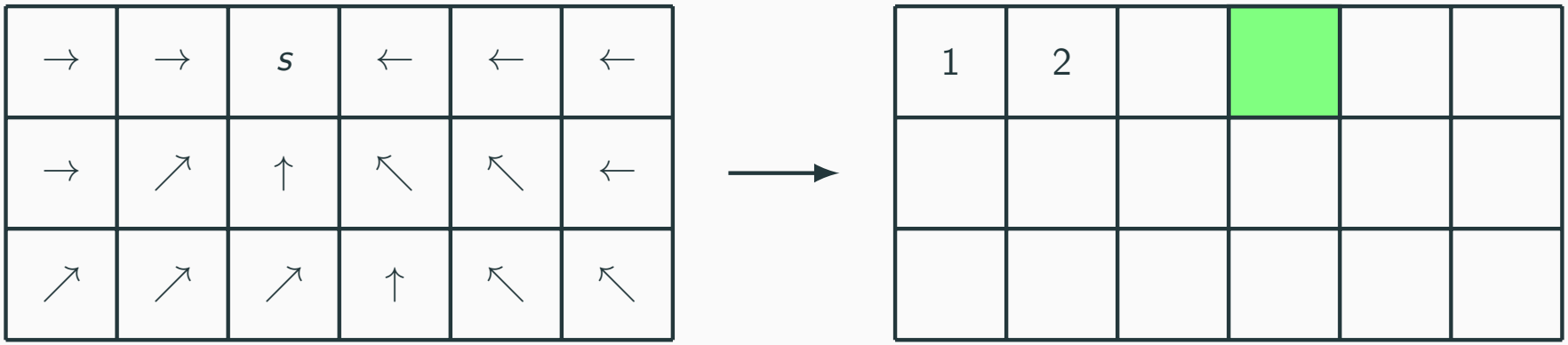
Study case : The flow accumulations



Flow directions

Flow accumulations

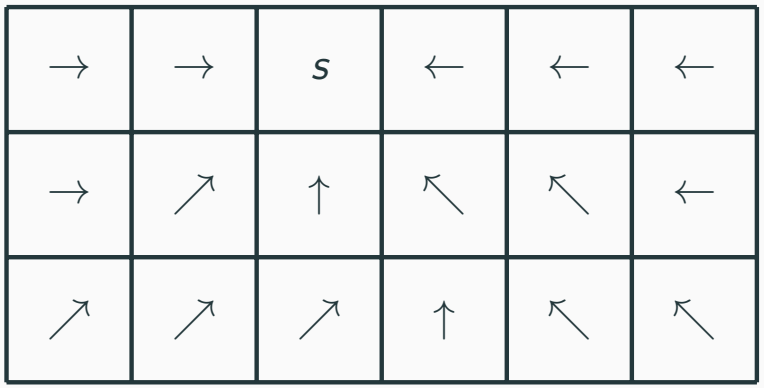
Study case : The flow accumulations



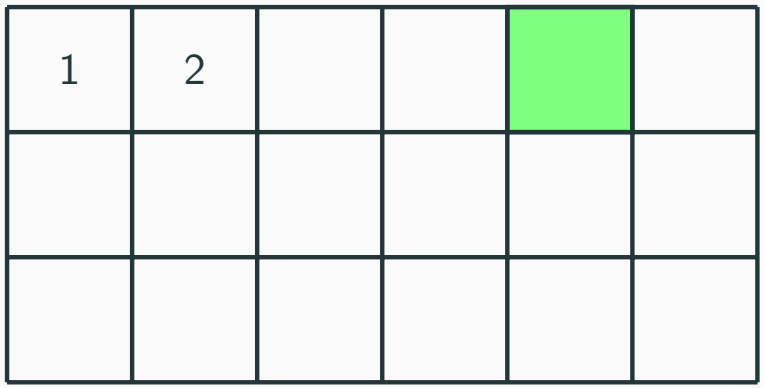
Flow directions

Flow accumulations

Study case : The flow accumulations

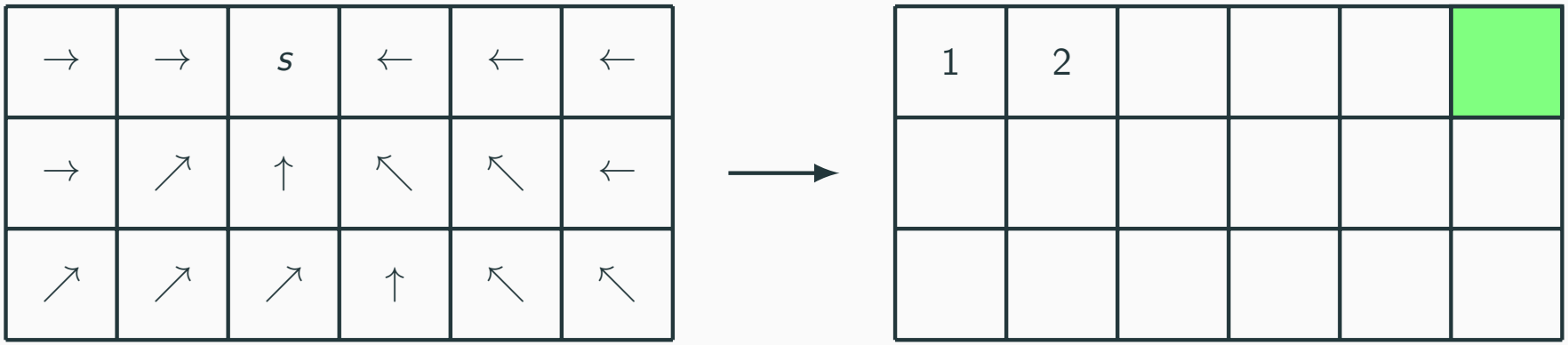


Flow directions



Flow accumulations

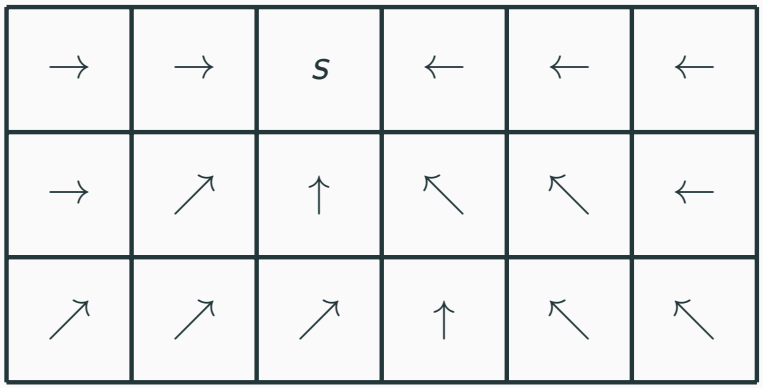
Study case : The flow accumulations



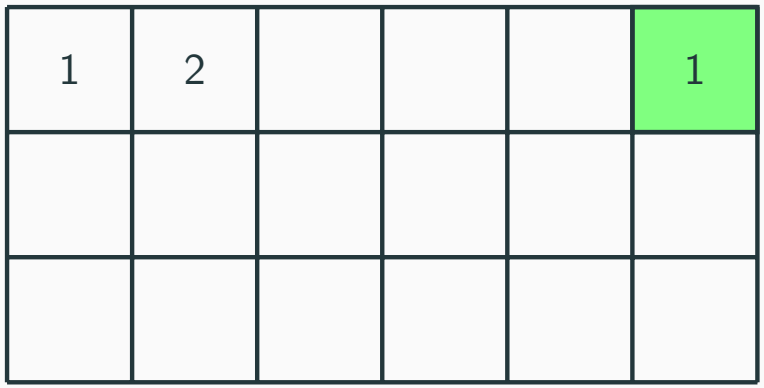
Flow directions

Flow accumulations

Study case : The flow accumulations

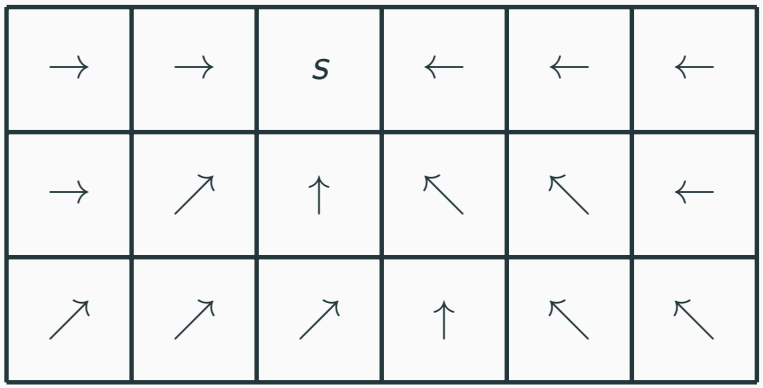


Flow directions

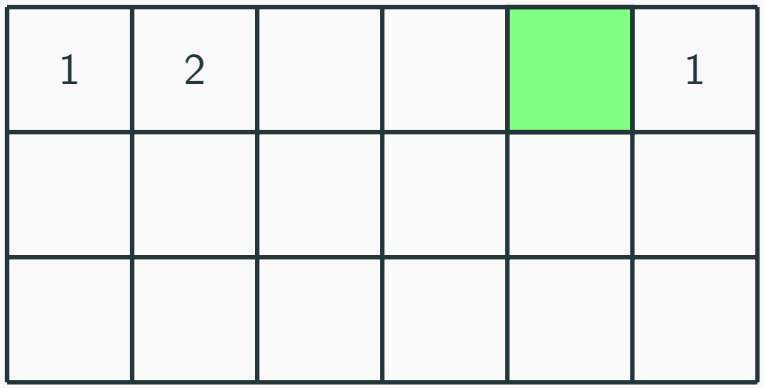


Flow accumulations

Study case : The flow accumulations

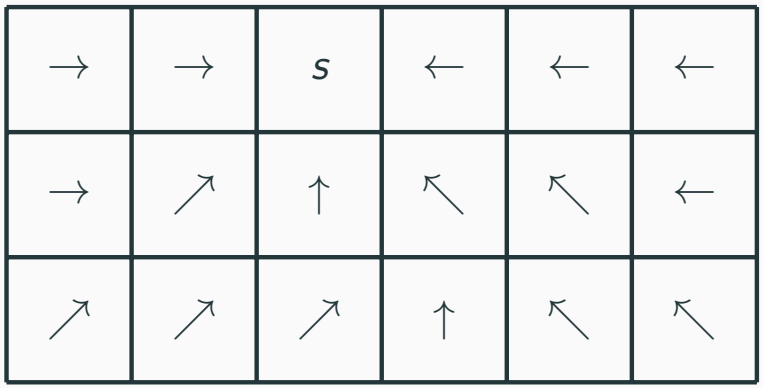


Flow directions

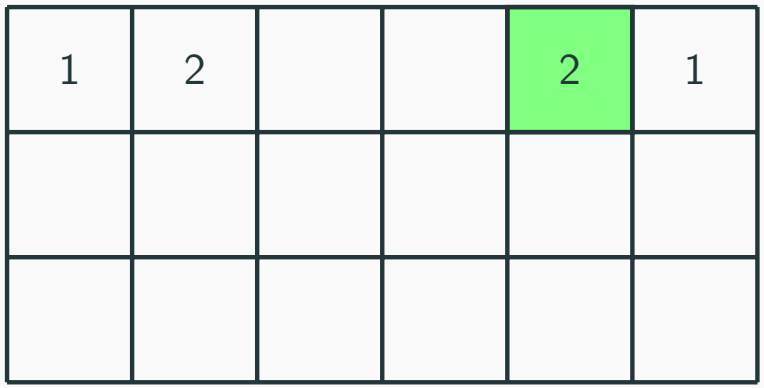


Flow accumulations

Study case : The flow accumulations

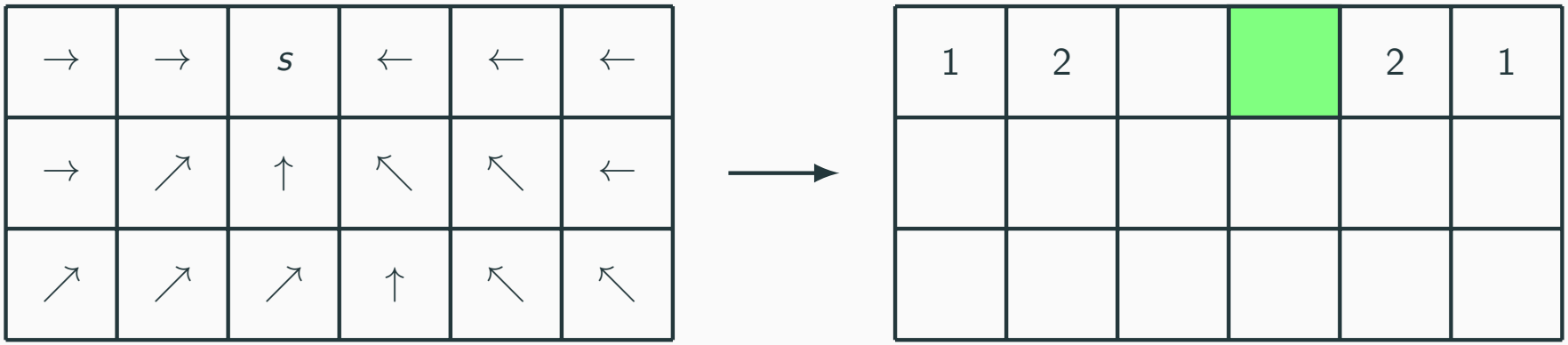


Flow directions



Flow accumulations

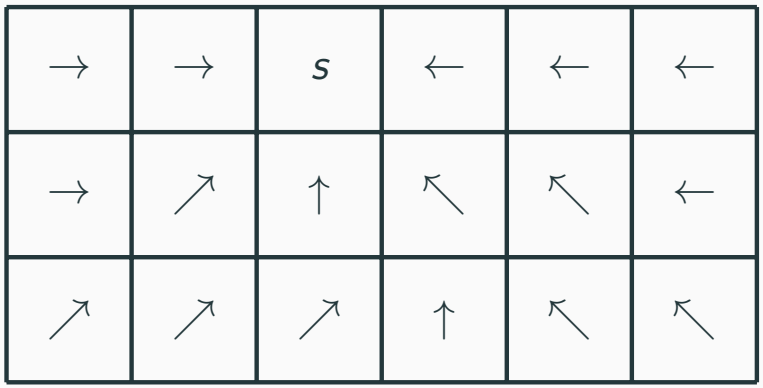
Study case : The flow accumulations



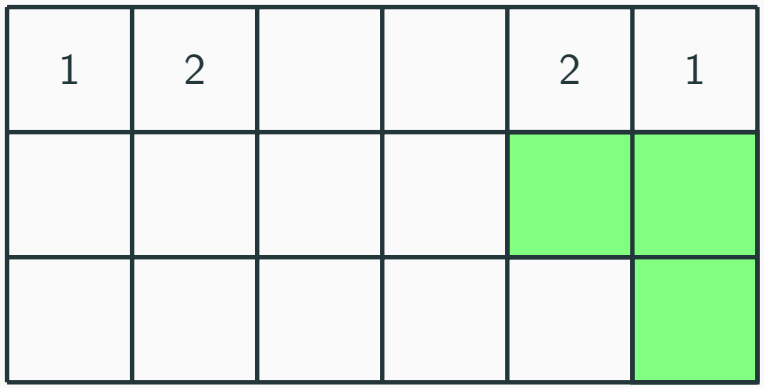
Flow directions

Flow accumulations

Study case : The flow accumulations

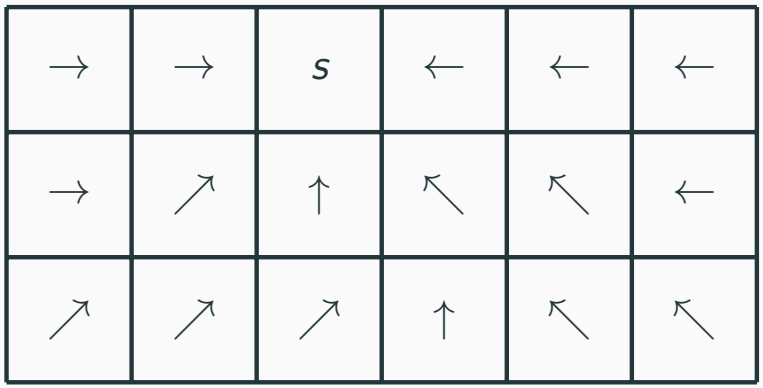


Flow directions

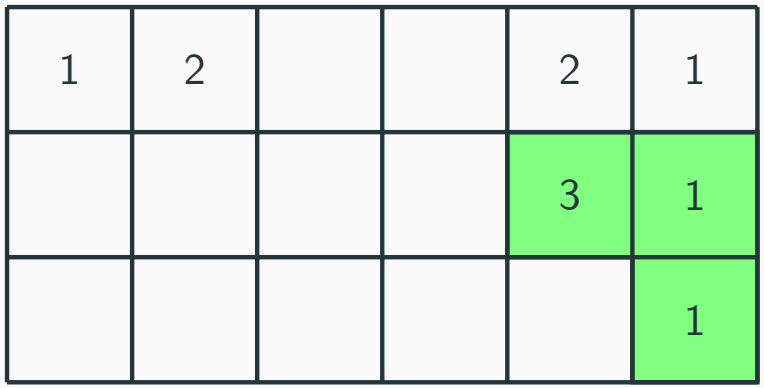


Flow accumulations

Study case : The flow accumulations

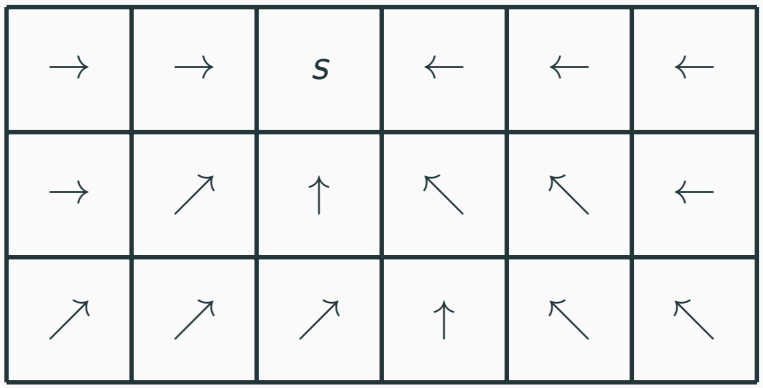


Flow directions

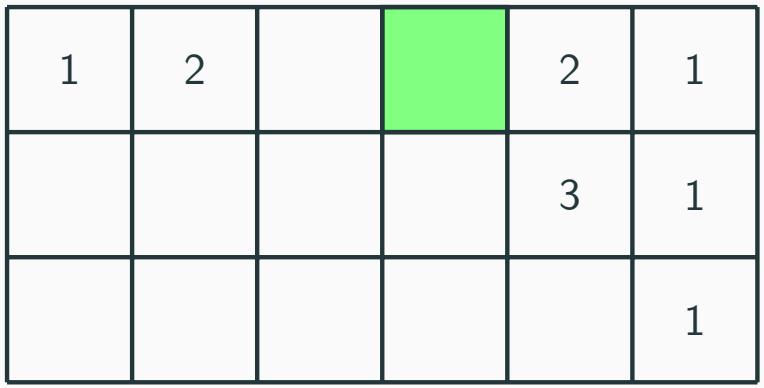


Flow accumulations

Study case : The flow accumulations

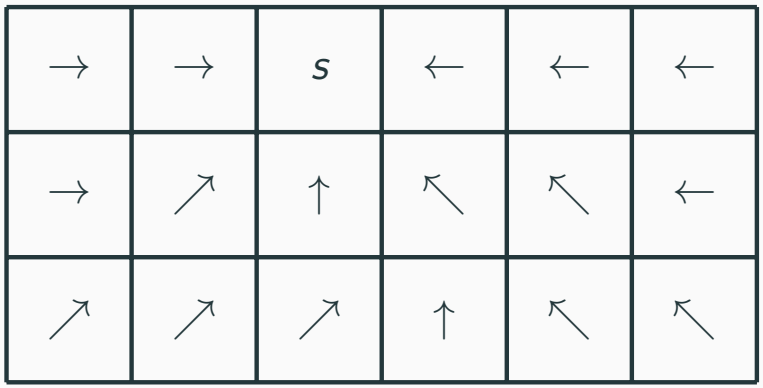


Flow directions

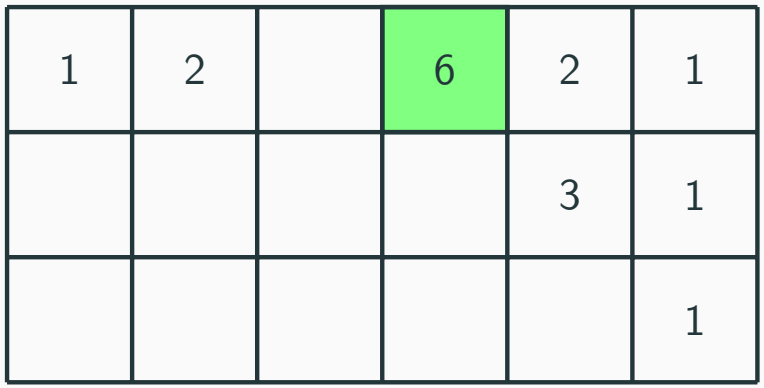


Flow accumulations

Study case : The flow accumulations



Flow directions



Flow accumulations

Study case : The flow accumulations

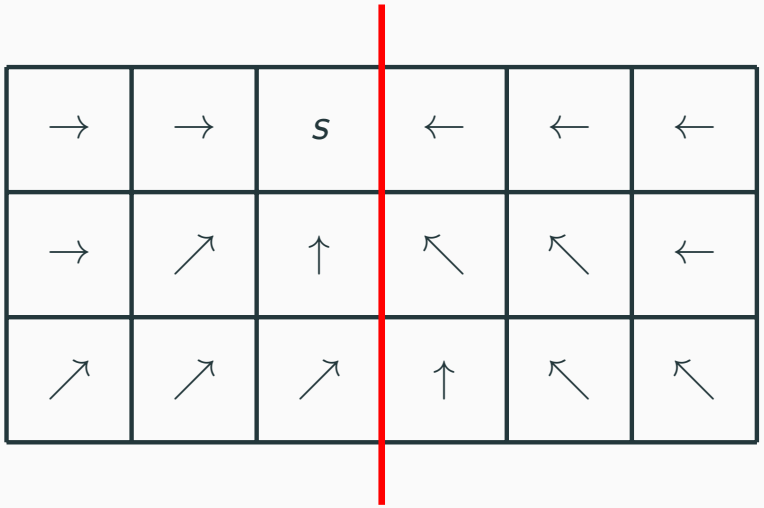
→	→	s	←	←	←
→	↗	↑	↖	↖	←
↗	↗	↗	↑	↖	↖

Flow directions

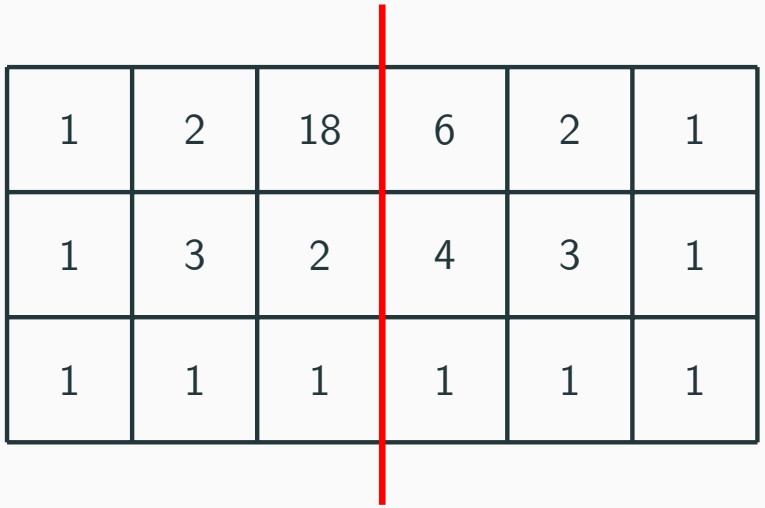
1	2	18	6	2	1
1	3	2	4	3	1
1	1	1	1	1	1

Flow accumulations

Study case : The flow accumulations

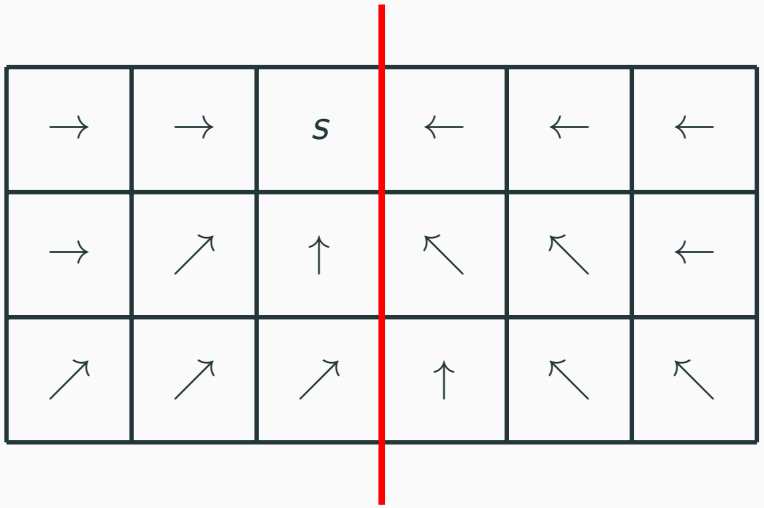


Flow directions

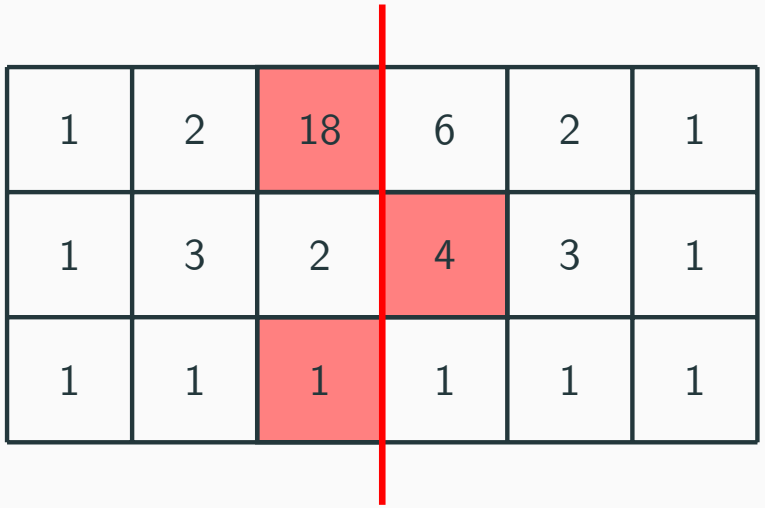


Flow accumulations

Study case : The flow accumulations



Flow directions



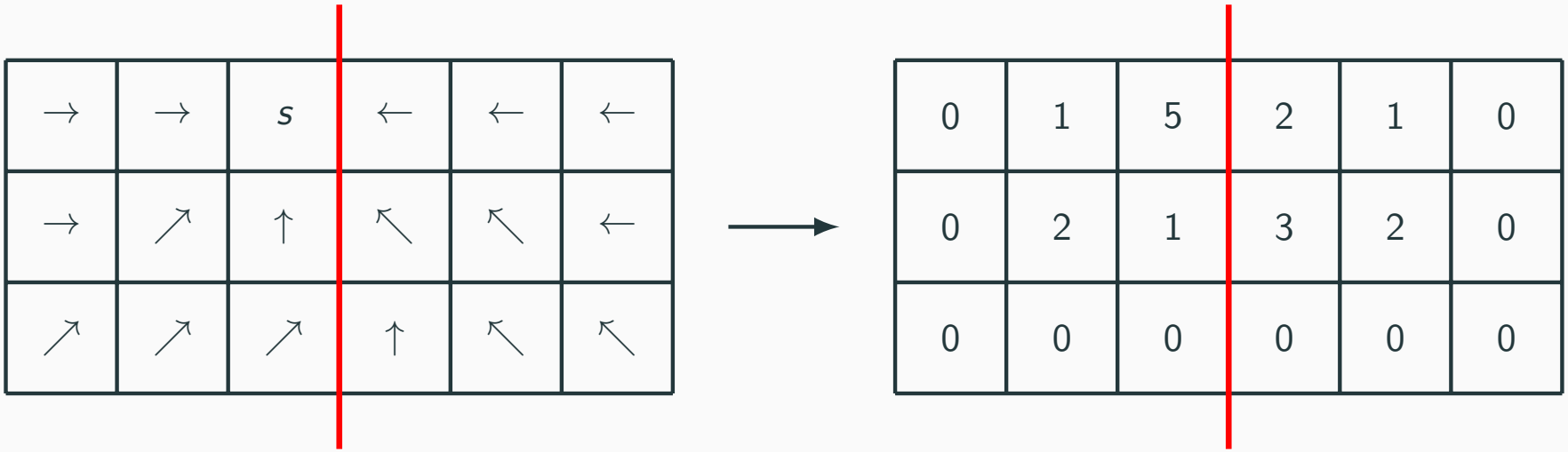
Flow accumulations

Parallel algorithm

1 Compute the raster of dependencies (stencil)

Parallel algorithm

1 Compute the raster of dependencies (stencil)



Flow directions

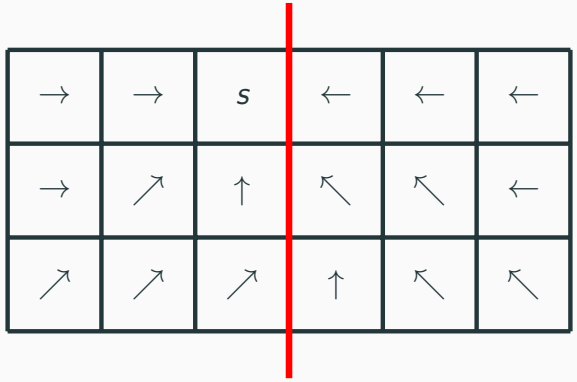
Dependencies

Parallel algorithm

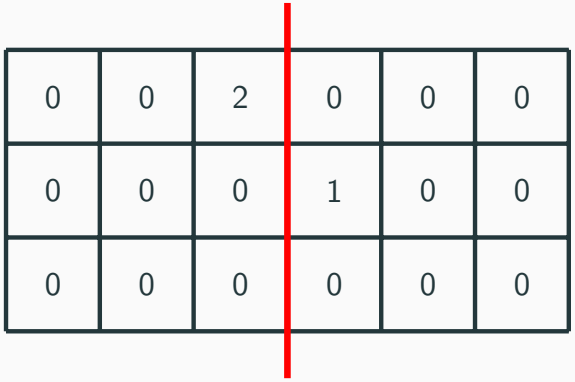
1. Compute the raster of dependencies (stencil)
2. Compute the flow accumulation of each cell without dependencies and update the dependencies

Parallel algorithm

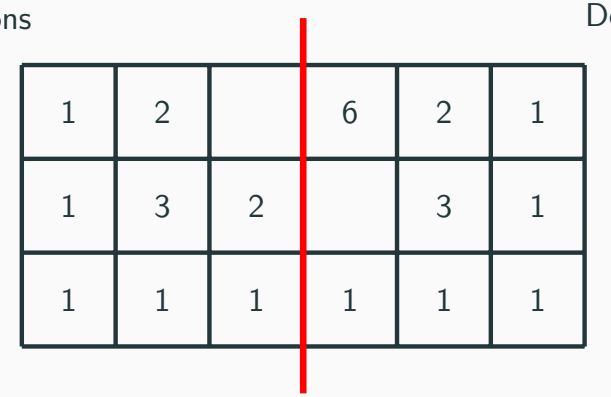
1. Compute the raster of dependencies (stencil)
2. Compute the flow accumulation of each cell without dependencies and update the dependencies



Flow directions



Dependencies



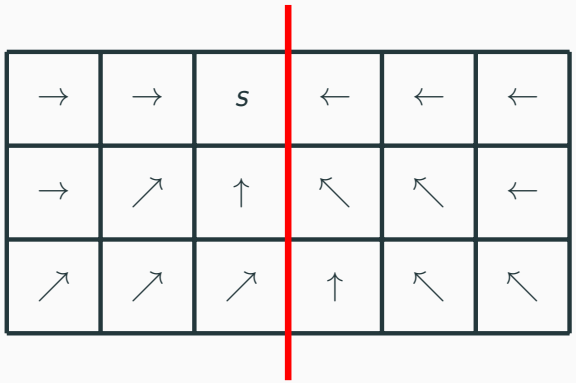
Flow accumulations

Parallel algorithm

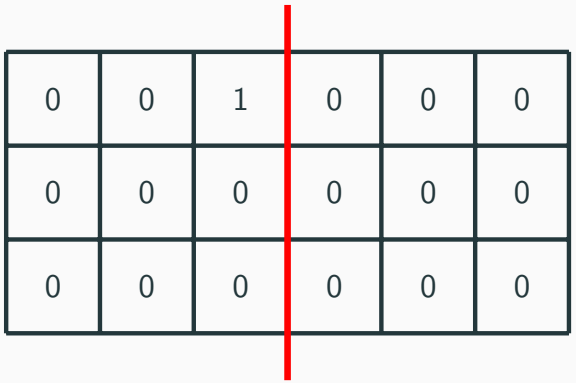
1. Compute the raster of dependencies (stencil)
2. Compute the flow accumulation of each cell without dependencies and update the dependencies
3. Exchange the ghost lines and update the dependencies

Parallel algorithm

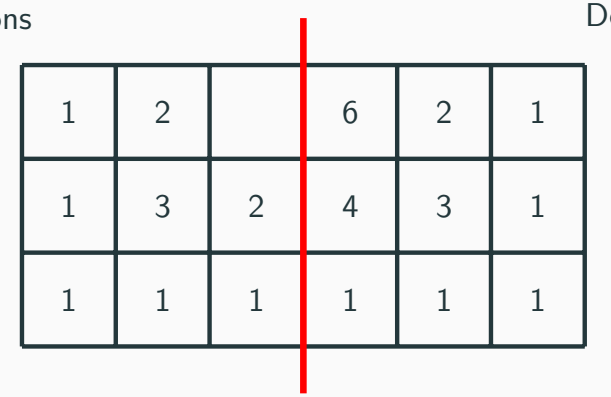
1. Compute the raster of dependencies (stencil)
2. Compute the flow accumulation of each cell without dependencies and update the dependencies
3. Exchange the ghost lines and update the dependencies



Flow directions



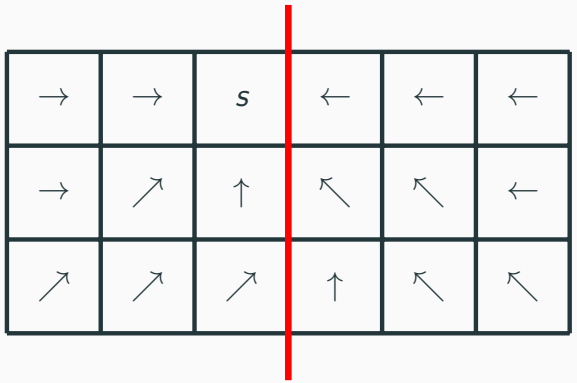
Dependencies



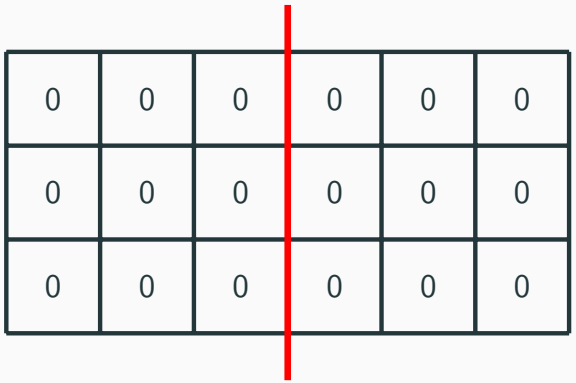
Flow accumulations

Parallel algorithm

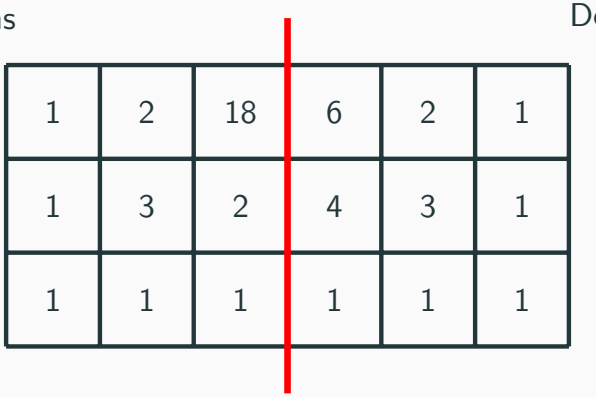
1. Compute the raster of dependencies (stencil)
2. Compute the flow accumulation of each cell without dependencies and update the dependencies
3. Exchange the ghost lines and update the dependencies



Flow directions



Dependencies



Flow accumulations

The PDD pattern

From the previous algorithm we can see a pattern emerge :

1. The stencil that computes the dependencies for which a relationship between the cells must be provide
2. The update step, for which each cell with a dependency of 0 is updated with an update function given by the user
3. A synchronization step to update the dependencies and the flow accumulations of the ghostlines.

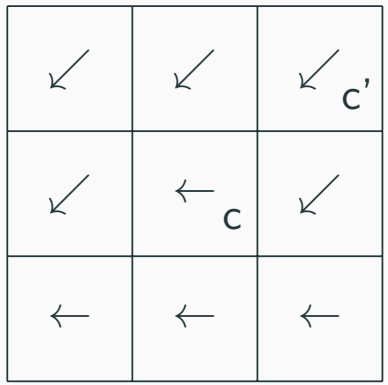
Thus, the user must provide two functions : a relation function and an update function.
Moreover, another version allows the user to memorize the cells passing order to be used later.

The relation function

The relation function must be defined as follows:

- a cell c
- a cell c'
- the raster r
- and returns a boolean

The relation indicates if c depends on c'.

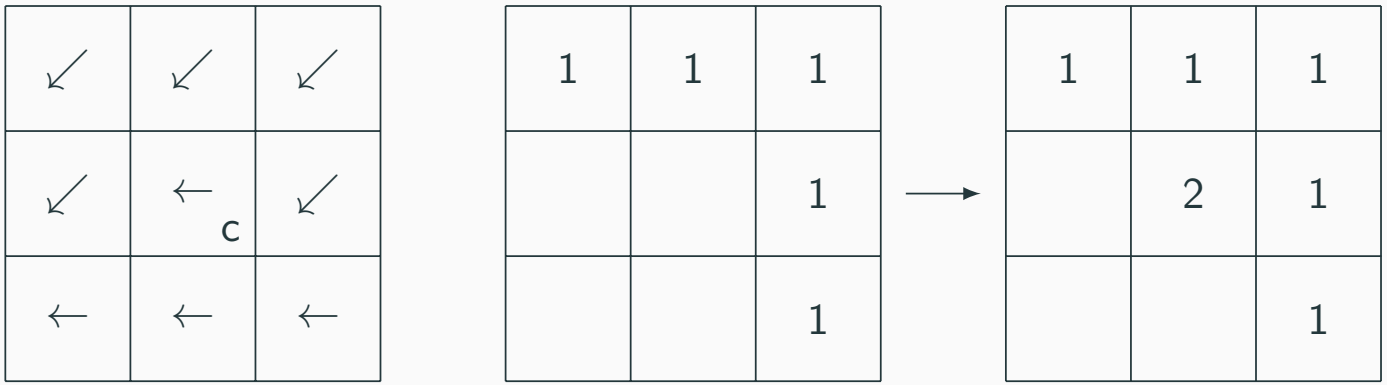


The update function

The update function must be defined as follows :

- a cell c
- the input raster to know the relation *in*
- the output raster *out*
- and returns the new value

The update function must return the new value of c depending on the value with which it is in relation.



Experimental results

PDD against handmade version

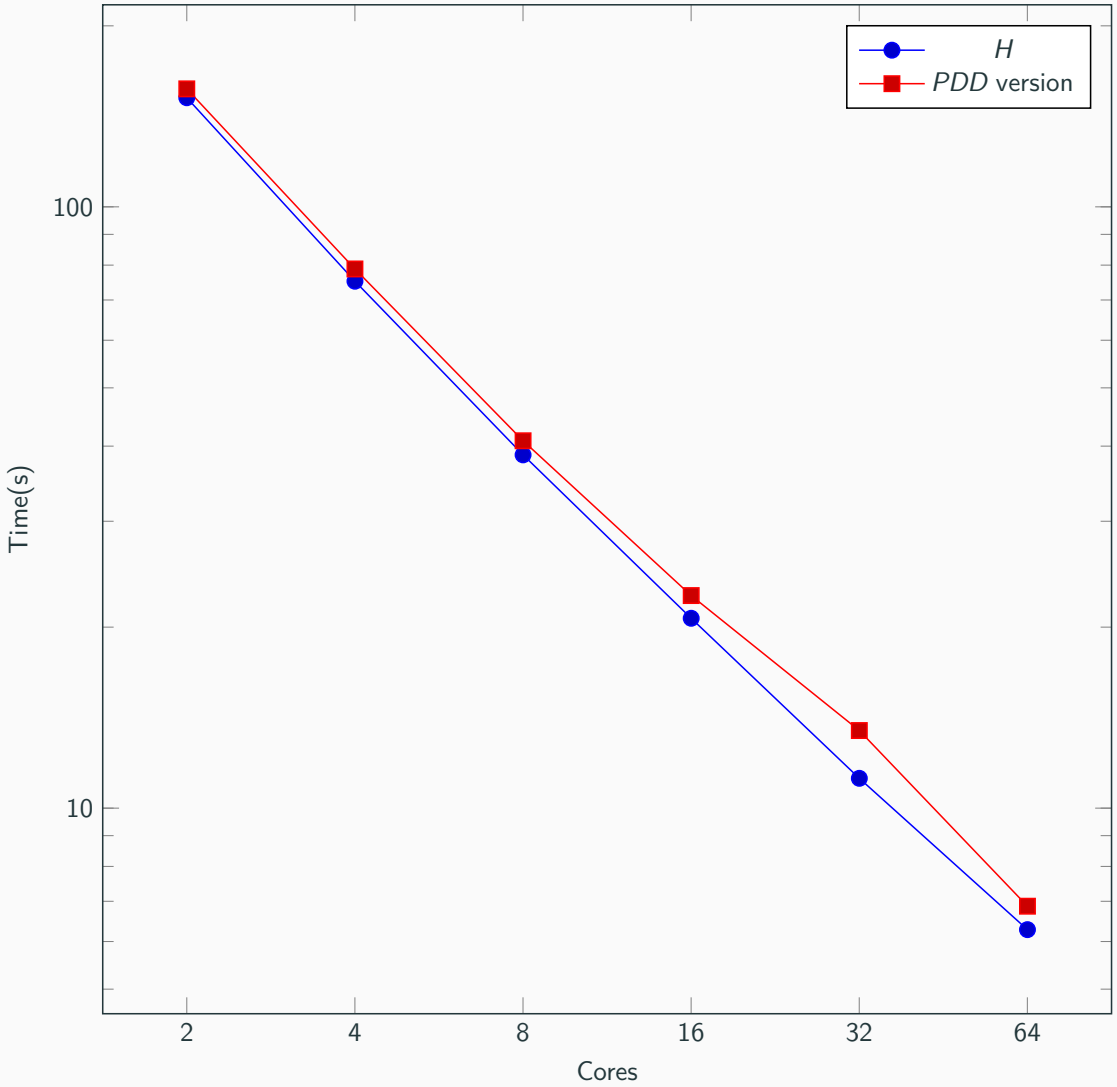


Figure 2: Experimental results for the PDD pattern against the handmade implementation of the flow accumulation algorithm

PDD against memorized order PDD

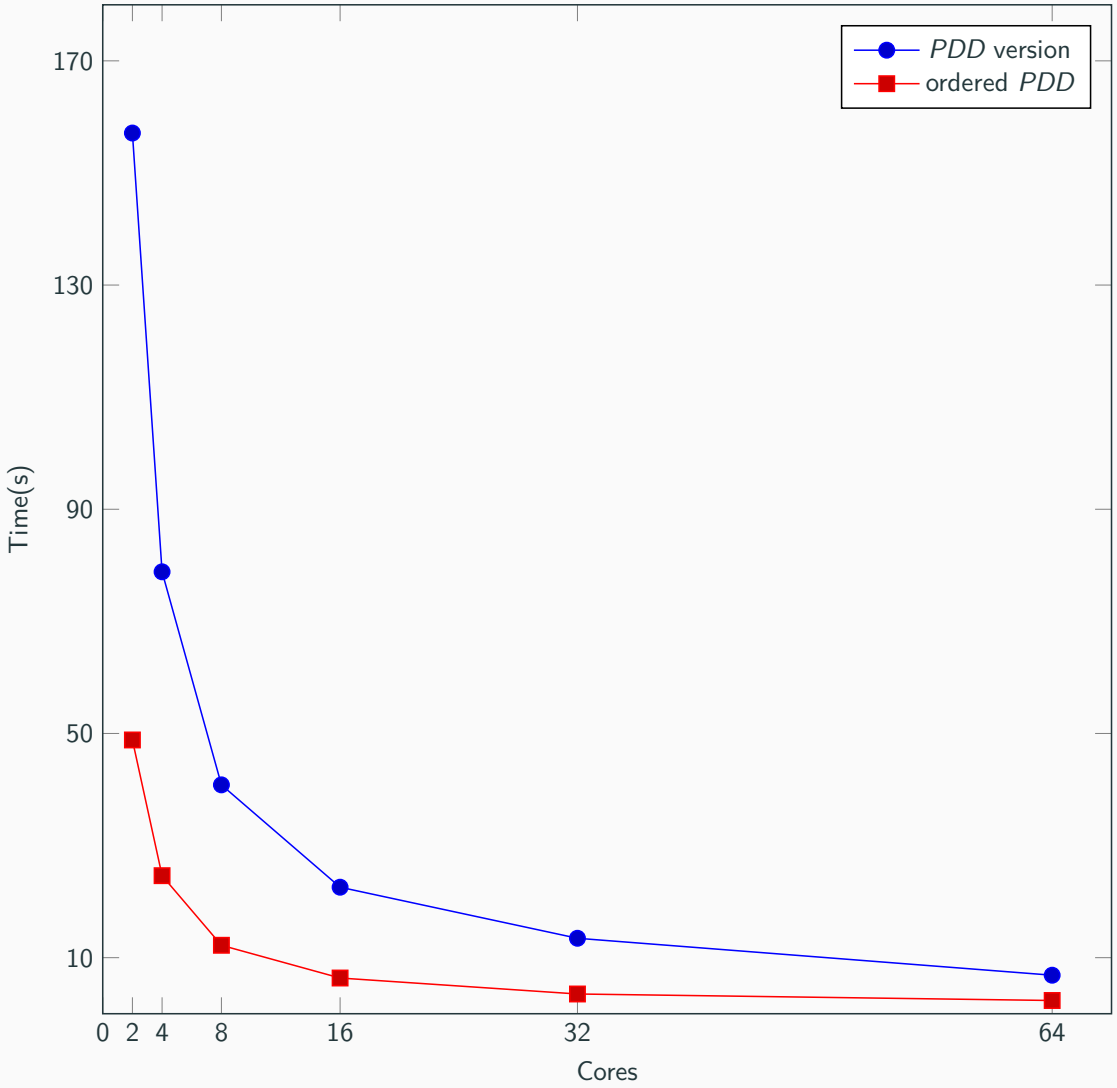


Figure 3: PDD pattern against the memorized order PDD version

PDD against memorized order PDD + save

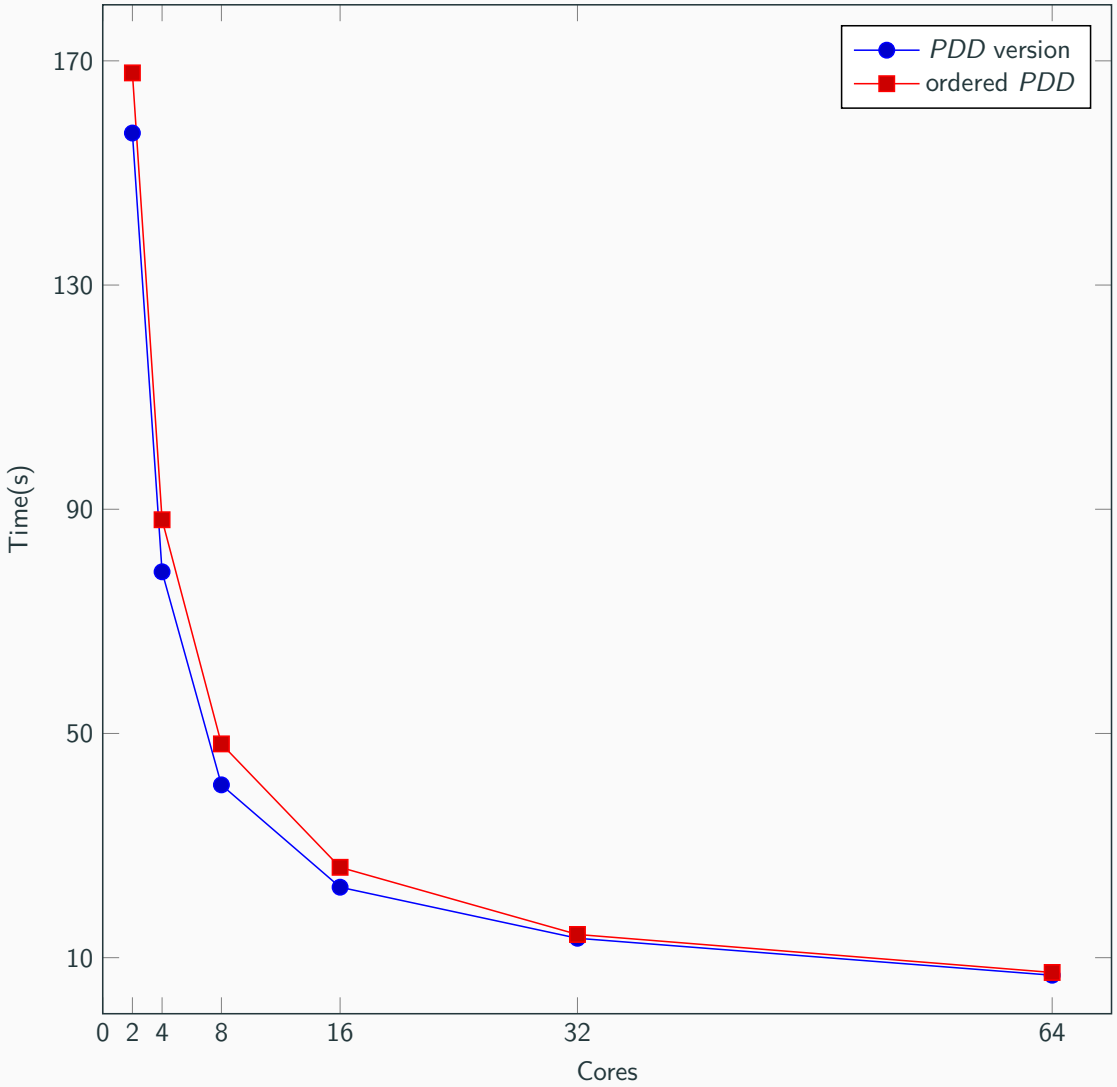


Figure 4: PDD pattern against the memorized order PDD version, including the time to memorize the order

Conclusion

Conclusion

Introduction to the PDD pattern :

- Easy to use : only two functions given by the user
- Adapted for irregular algorihms
- Limited overhead compared to a handmade implementation

Some possible improvments :

- The order is saved into a map, possibilty to use a tree
- With a tree the data could be redistributed to save communication time
- Allowing the user to use multiple input rasters

Perspectives

On the pattern layer :

- Add new patterns/skeletons
- Allowing the skeletons to be composed

On the DSL layer :

- Implement a python-like DSL (we already have a promising prototype)
- A user interface to select the desired patterns