Scheduling under memory constraint

Maxime GONTHIER

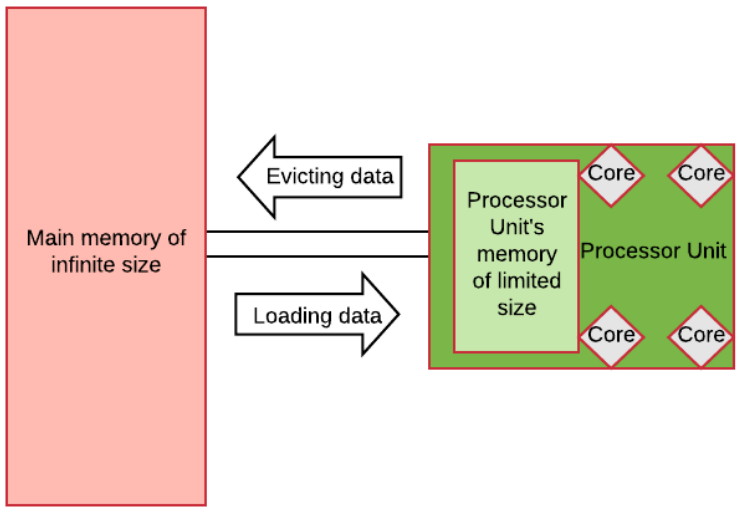
ENS Lyon - ROMA - Université de Bordeaux - LaBRI - Inria
Supervised by Samuel THIBAULT and Loris MARCHAL

December 2020 - Plenary SOLHARIS +
HPC-SCALABLE-ECOSYSTEM

Interest of the research internship

- Scientific computing requires a large amount of data
- GPUs calculate quickly but their memory are too small for scientific computing
- Let's develop an efficient way to access data that are not stored on the GPU's memory

⇒ Development of 6 algorithms regrouped under the name AATO (Affinity Aware Task Ordering)



Summary

Formalization of the problem

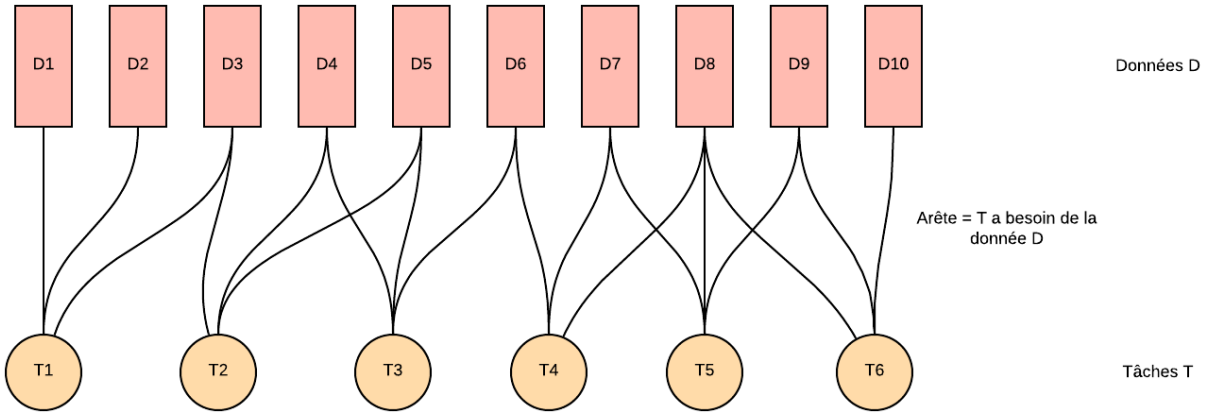


Figure: Bipartite graph representing the links between data and tasks

- D the data. We only consider the input data. Outputs are ignored
- T the tasks
- Our intuition is to group together tasks that share a lot of data in order to re-use them for multiple tasks

$\Rightarrow$ We partition the tasks into packages noted P_i

$W(P_i)$: weight of the data needed to compute the set of tasks P_i.

First step: minimize the number of packages

- Constraint on the size of a package ($W(P_i)$): $\forall i\ W(P_i) \leq B$
- Partition m tasks in l packages $P_1, P_2, ..., P_l$ such as l is minimal
- $B = MEM_{GPU}/2$ to allow a prefetch of the data in parallel with the computation

Minimize the number of packages $\rightarrow$ grouping together tasks that share the most data

Second step: minimizing the number of data transfers

- Optimize the processing order of packages
- Optimize the processing order of tasks within packages

Application case

We use GEMM. Multiplication of matrices by blocks such as:

- 1 task is 1 a computation of a block of C
- 1 data is a row of A or a column of B

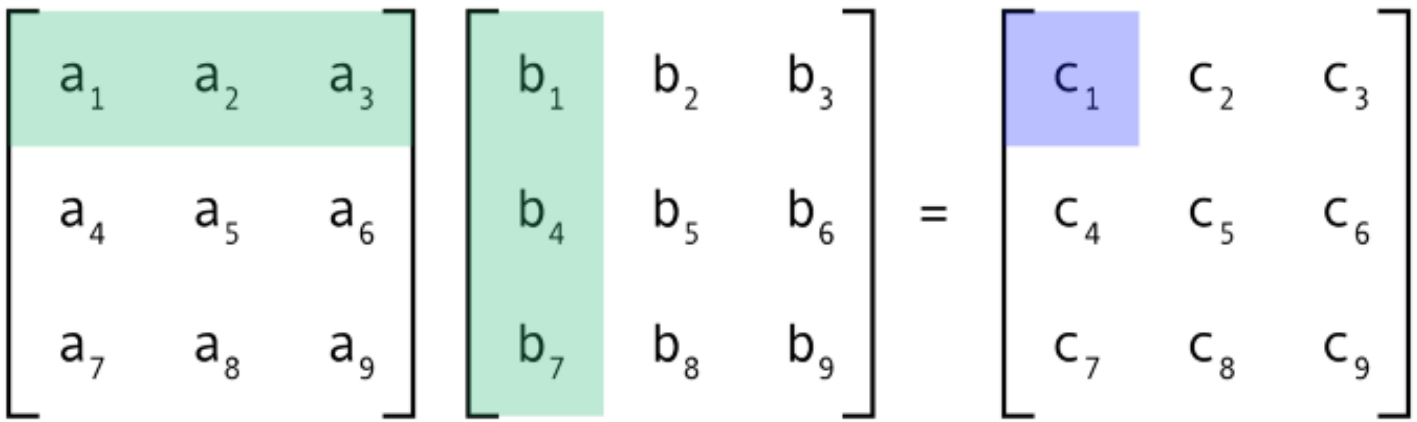


Figure: Computation of a block of C with a row of A and a column of B

It is a simplified version, but it already allow us to study locality problems.

Can we partition $\mathbb{T}$ in at most L packages such as $\forall i\ W(P_i) \leq B$?

Proof of strong NP-Completeness

Let's use 3-partition
$RP_1 \rightarrow$ 3-partition by contradiction
3-partition $\rightarrow RP_1$ because we have n triplet of same size

Proposed algorithms

Heavy Edge Matching

- From "Graph Partitioning" of Charles-Edmond Bichot
- Objective: find a maximum coupling of a graph that minimize the cut

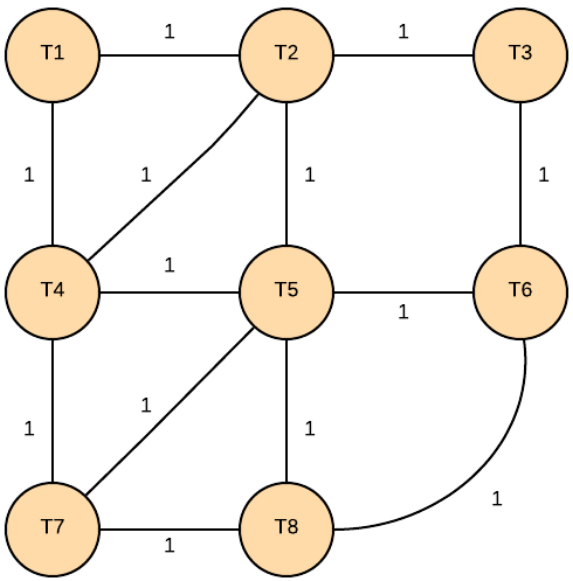


Figure: Initial graph

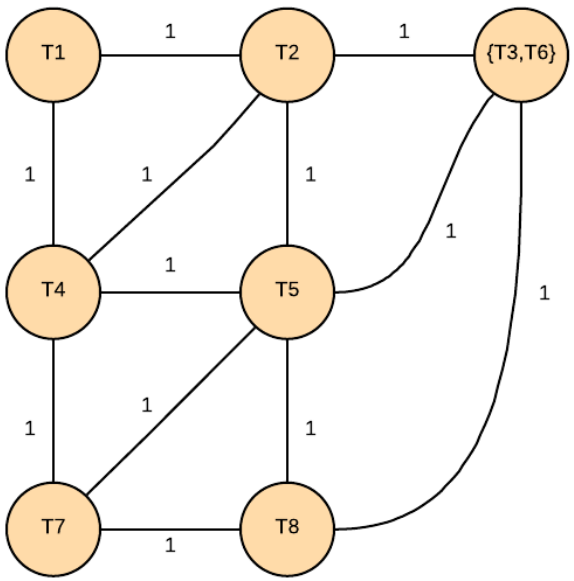


Figure: Randomly chosen vertex 3, edge 3-6 removed

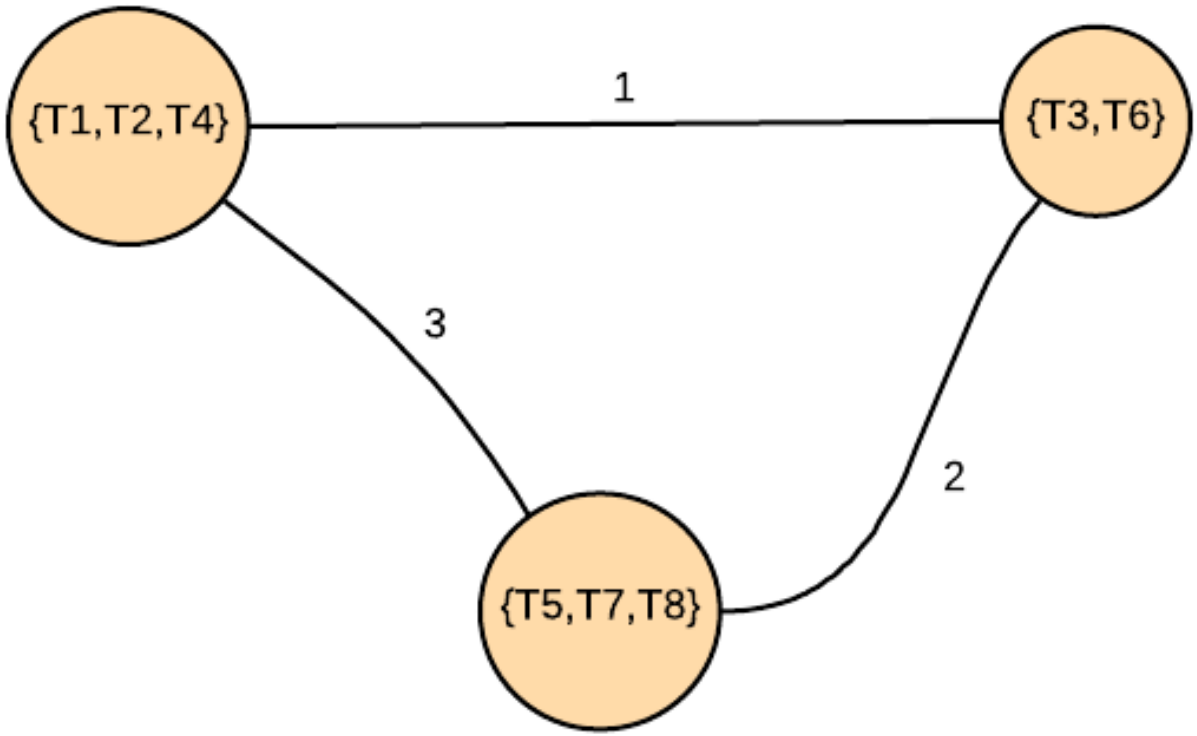


Figure: Final graph

Homogeneous packing

The tasks or packages that share the most data are grouped together. If there is an equality with several packages, the package with the lowest index is chosen.

Description step by step

- 0. Consider that each task form an elementary package
- 1. Compute the maximum weight of common data on all pairs of packages
- 2. Merge couples (P_i, P_j) which reach this maximum value and with $\forall i\ W(P_i) \leq B$
- 3. Repeat from step 1 until no more possible merge

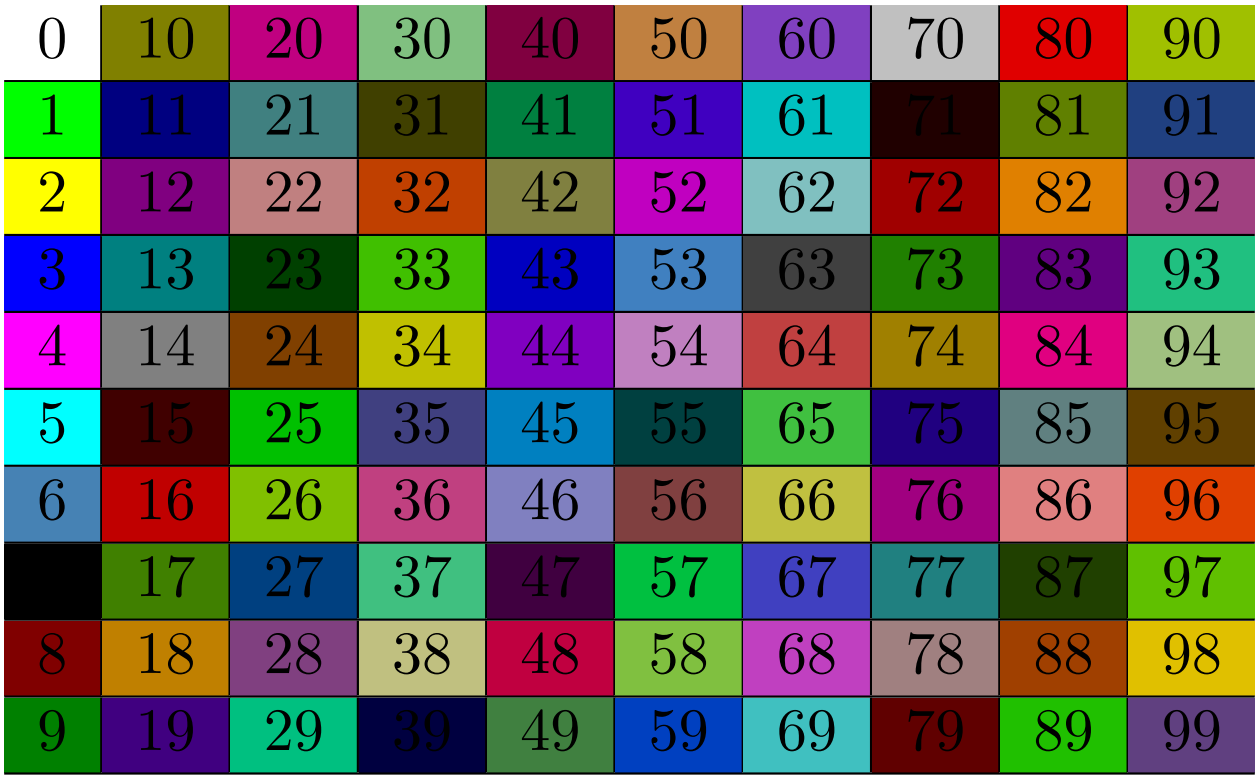


Figure: Initial packing of Homogeneous packing on a 10x10 matrix

0	10	20	30	40	50	60	70	80	90
0	10	20	30	40	50	60	70	80	90
2	12	22	32	42	52	62	72	82	92
2	12	22	32	42	52	62	72	82	92
4	14	24	34	44	54	64	74	84	94
4	14	24	34	44	54	64	74	84	94
6	16	26	36	46	56	66	76	86	96
6	16	26	36	46	56	66	76	86	96
8	18	28	38	48	58	68	78	88	98
8	18	28	38	48	58	68	78	88	98

Figure: First iteration of Homogeneous packing on a 10x10 matrix

0	0	20	20	40	40	60	60	80	80
0	0	20	20	40	40	60	60	80	80
2	2	22	22	42	42	62	62	82	82
2	2	22	22	42	42	62	62	82	82
4	4	24	24	44	44	64	64	84	84
4	4	24	24	44	44	64	64	84	84
6	6	26	26	46	46	66	66	86	86
6	6	26	26	46	46	66	66	86	86
8	8	28	28	48	48	68	68	88	88
8	8	28	28	48	48	68	68	88	88

Figure: Second iteration of Homogeneous packing on a 10x10 matrix

0	0	0	0	40	40	40	40	80	80
0	0	0	0	40	40	40	40	80	80
0	0	0	0	40	40	40	40	80	80
0	0	0	0	40	40	40	40	80	80
4	4	4	4	44	44	44	44	4	4
4	4	4	4	44	44	44	44	4	4
4	4	4	4	44	44	44	44	4	4
4	4	4	4	44	44	44	44	4	4
0	0	0	0	40	40	40	40	80	80
0	0	0	0	40	40	40	40	80	80

Figure: Sixth iteration of Homogeneous packing on a 10x10 matrix

Greedy packing

Only two packages are merged at each step, the ones sharing the most data.

0	6	12	18	24	30
0		13	19	25	31
2	8	14	20	26	32
3	9	15	21	27	33
4	10	16	22	28	34
5	11	17	23	29	35

Figure: First iteration

0	0	0	0	0	27
0	0	0	0	0	27
0	0	0	0	0	27
0	0	0	0	27	27
0	0	0	0	27	27
0	0	0	0	27	27

Figure: Iteration 35

Fair packing

Only the packages with fewer tasks can merge.

0	6	12	18	24	30
0	6	12	18	24	30
2	8	14	20	26	32
2	8	14	20	26	32
4	10	16	22	28	34
4	10	16	22	28	34

Figure: Iteration 1

0	0	12	12	24	24
0	0	12	12	24	24
0	0	12	12	24	24
0	0	12	12	24	24
4	4	4	4	28	28
4	4	4	4	28	28

Figure: Iteration 3

0	0	12	12	24	24
0	0	12	12	24	24
0	0	12	12	24	24
0	0	12	12	24	24
28	28	28	28	28	28
28	28	28	28	28	28

Figure: Iteration 4

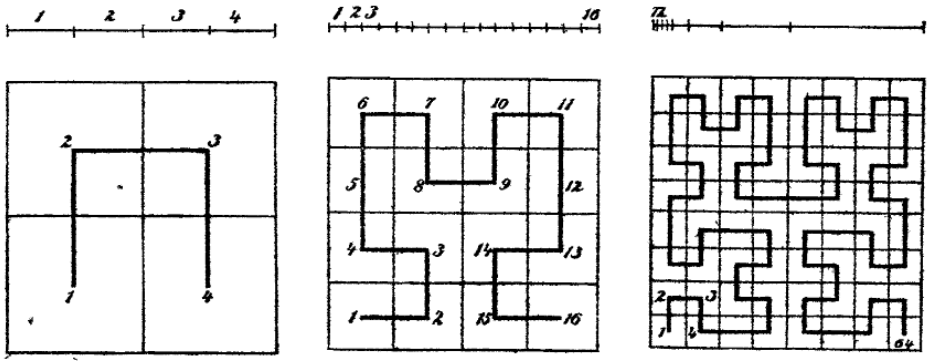
24	24	24	24	24	24
24	24	24	24	24	24
24	24	24	24	24	24
24	24	24	24	24	24
28	28	28	28	28	28
28	28	28	28	28	28

Figure: Iteration 6

Hierarchical algorithms

- 1. Same method as Homogeneous or Fair packing at first
- 2. Removal of the limit B
- 3. Repeat the same steps as Homogeneous or Fair packing as long as there is more than 1 package

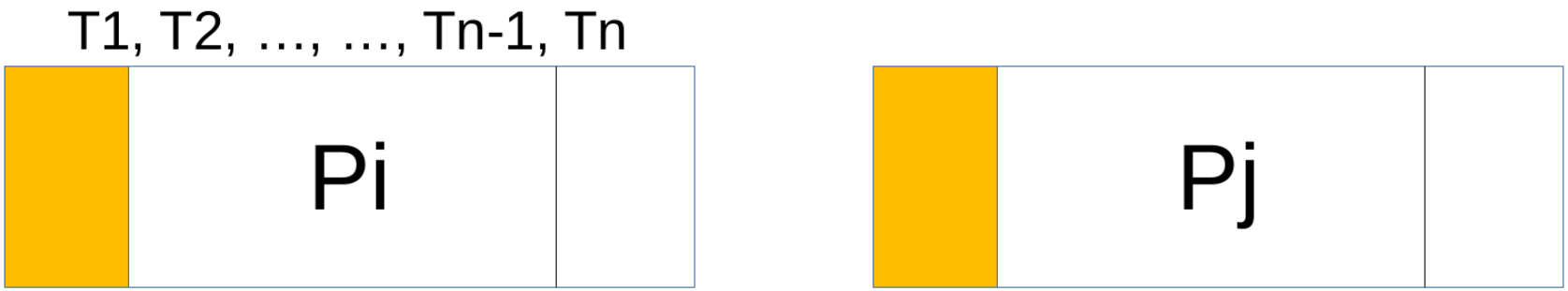
Order U



Goals and hypothesis

- A "U" shape to order tasks improves locality
- Improving the locality can reduce the number of transfers

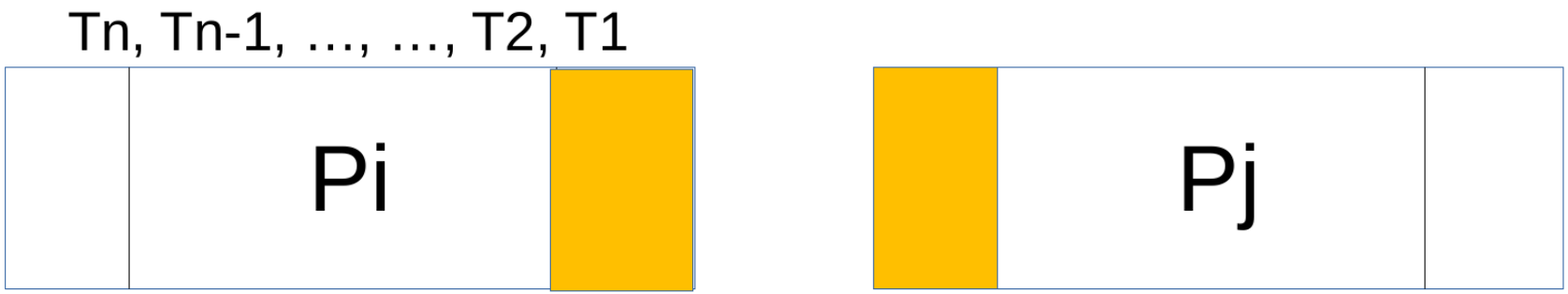
Order U



Goals and hypothesis

- A "U" shape to order tasks improves locality
- Improving the locality can reduce the number of transfers

Order U

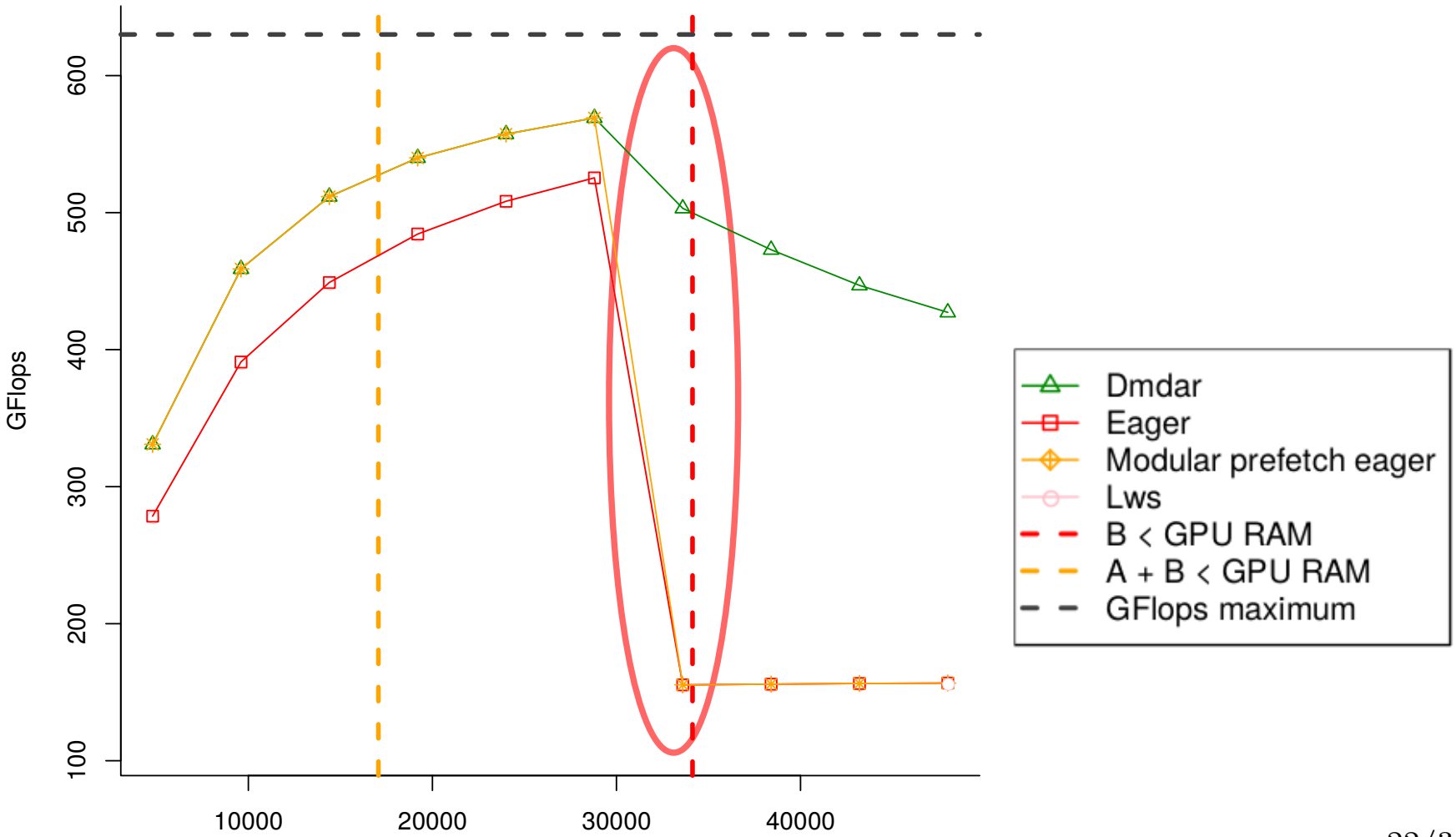


Goals and hypothesis

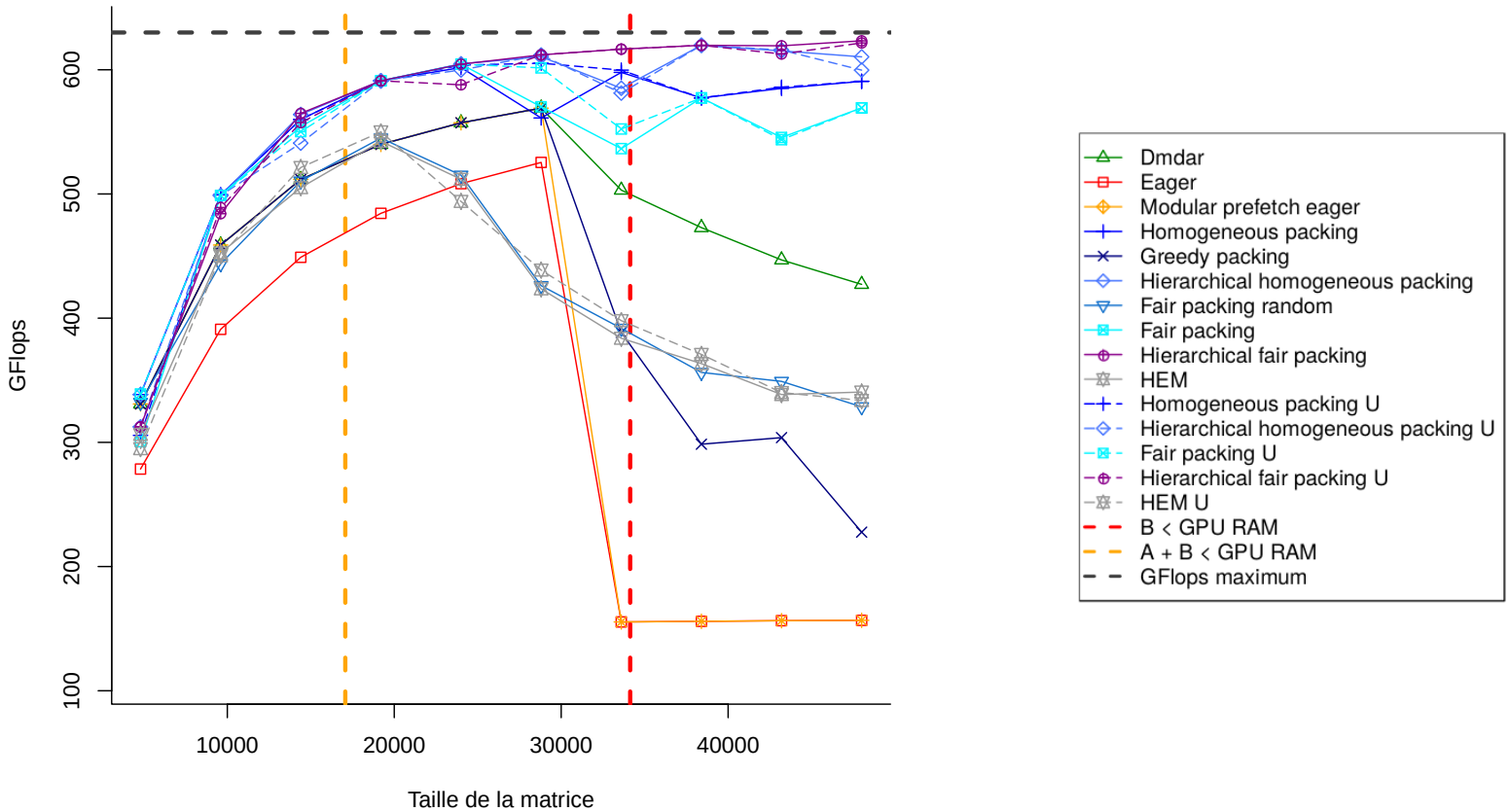
- A "U" shape to order tasks improves locality
- Improving the locality can reduce the number of transfers

StarPU's schedulers

Performances of StarPU's schedulers



Experimental evaluation of proposed algorithms



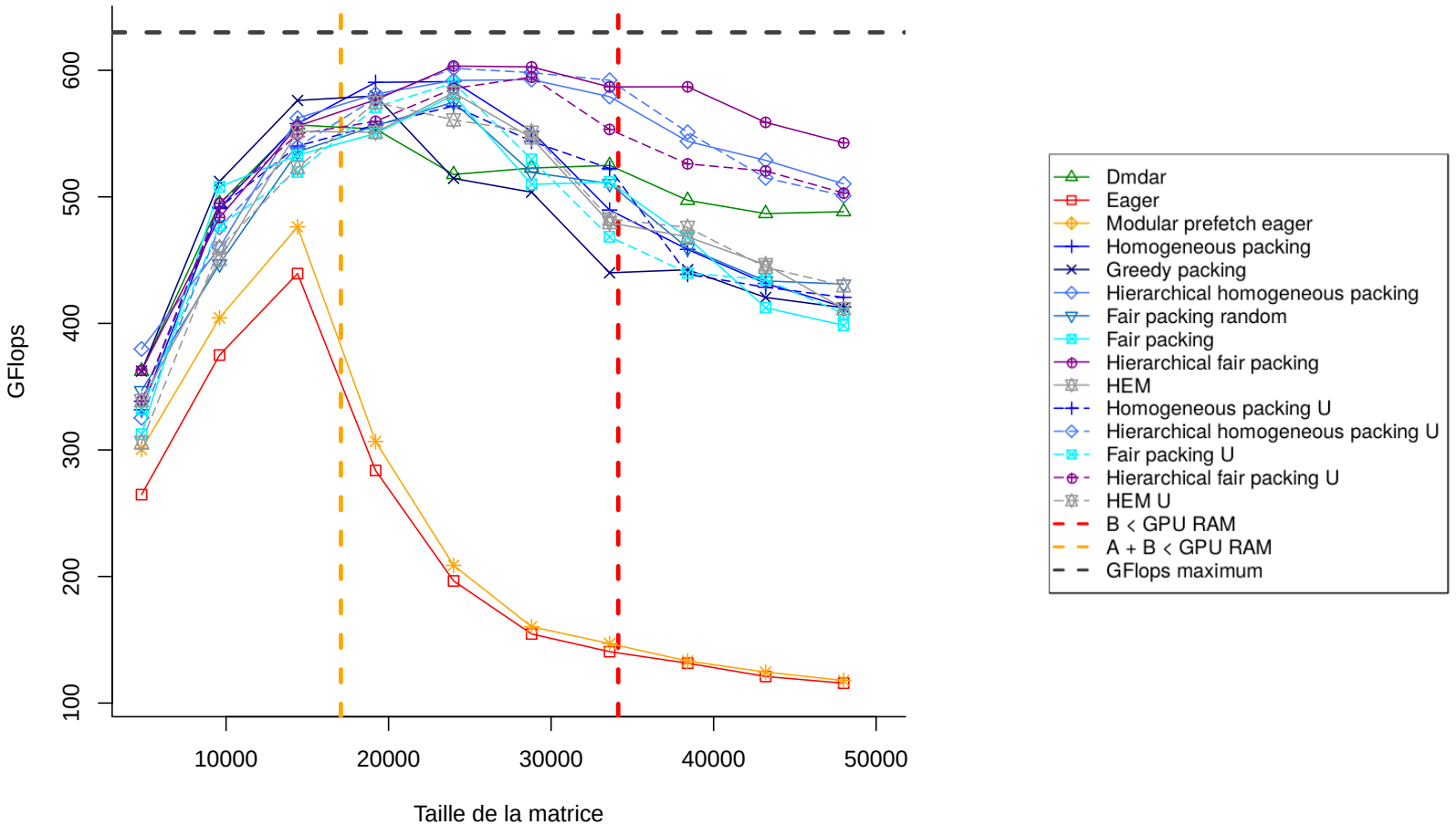
- There is a pathological matrix size corresponding to a little less than the size of the GPU RAM
- Hierarchical algorithms are the most efficient
- U-order does not improve performances

0	2	8	10	41	43	49	51	**18**
1	3	9	11	42	44	50	52	**19**
4	6	12	14	45	47	53	55	59
5	7	13	15	46	48	54	56	60
21	23	29	31	61	63	69	71	39
22	24	30	32	62	64	70	72	40
25	27	33	35	65	67	73	75	79
26	28	34	36	66	68	74	76	80
16	**17**	37	38	57	58	77	78	**20**

Figure: Processing order of C with Homogeneous packing

0	2	8	10	32	34	40	42	72
1	3	9	11	33	35	41	43	73
4	6	12	14	36	38	44	46	74
5	7	13	15	37	39	45	47	75
16	**18**	24	26	48	50	56	58	76
17	**19**	25	27	49	51	57	59	77
20	22	28	30	52	54	60	62	78
21	23	29	31	53	55	61	63	79
64	65	66	67	68	69	70	71	80

Figure: Processing order of C with Hierarchical Homogeneous packing



- Eager has worse performances
- Hierarchical algorithms are on average 9% better than Dmdar

Other contributions

Order of arrival of tasks in Z

Improves locality
Improves Eager, Dmdar and Greedy packing

Improved performance of Dmdar

Better performances with random dependencies
Avoids the pathological case of LRU

Conclusion

Goal: minimize data transfer time when multiplying matrix whose data does not fit into memory

AATO's algorithms results

- Best algorithms: the Hierarchical algorithms which, after initial packaging, optimize the order of processing of packets
- 17 % better than Dmdar on average
- 9 % better than Dmdar on average with random dependencies
- The order U does not add anything

StarPU's schedulers

- There is a pathological matrix size
- Ddmar suffers less from this pathological case

Work in progress

Motivations

- Our first goal was building packages of tasks to increase data locality
- What if we relieve this artificial constraint of packets ?
- Focus on finding the order that minimize data transfer time

First problem

For a given set of tasks $\mathbb{T}$ sharing a set of data $\mathbb{D}$ and a limited processor unit memory M, what is the schedule σ and the eviction policy $\mathbb{V}$ that minimize the number of loads ?

Second problem

How to minimize the computation time ? $\Rightarrow$ Maximize the overlap of computations and transfers.

Complexity of the first problem

The NP-completeness proof consists in a reduction from the cutwidth minimization problem

Belady's eviction policy is optimal. Consist in evicting data that will be used the latest.

Heuristics for the first problem

- Maximum Spanning Tree
- Cuthill–McKee algorithm
- Hierarchical fair packing

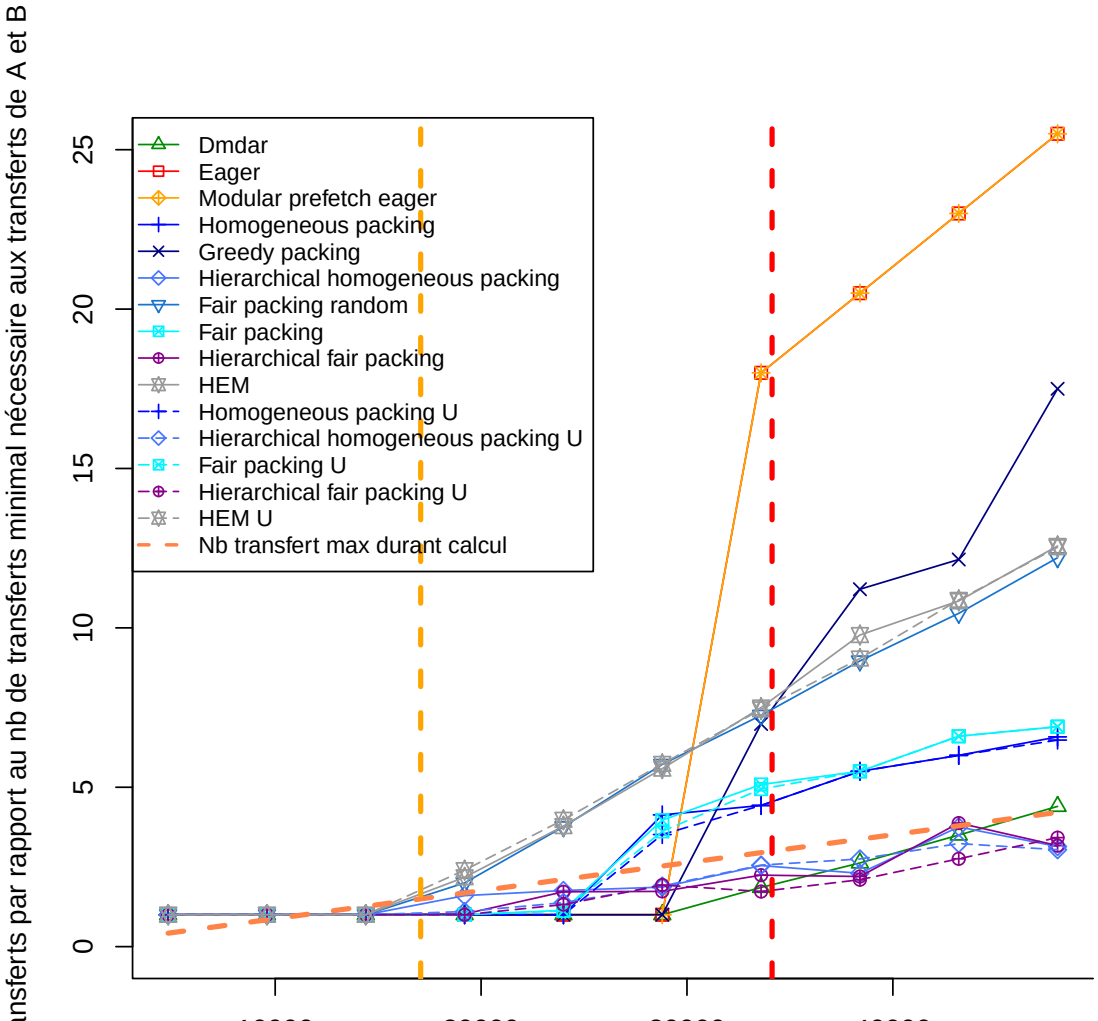
Future works

- Adapt our algorithms to situations with a "hot start"
- Work on the second problem

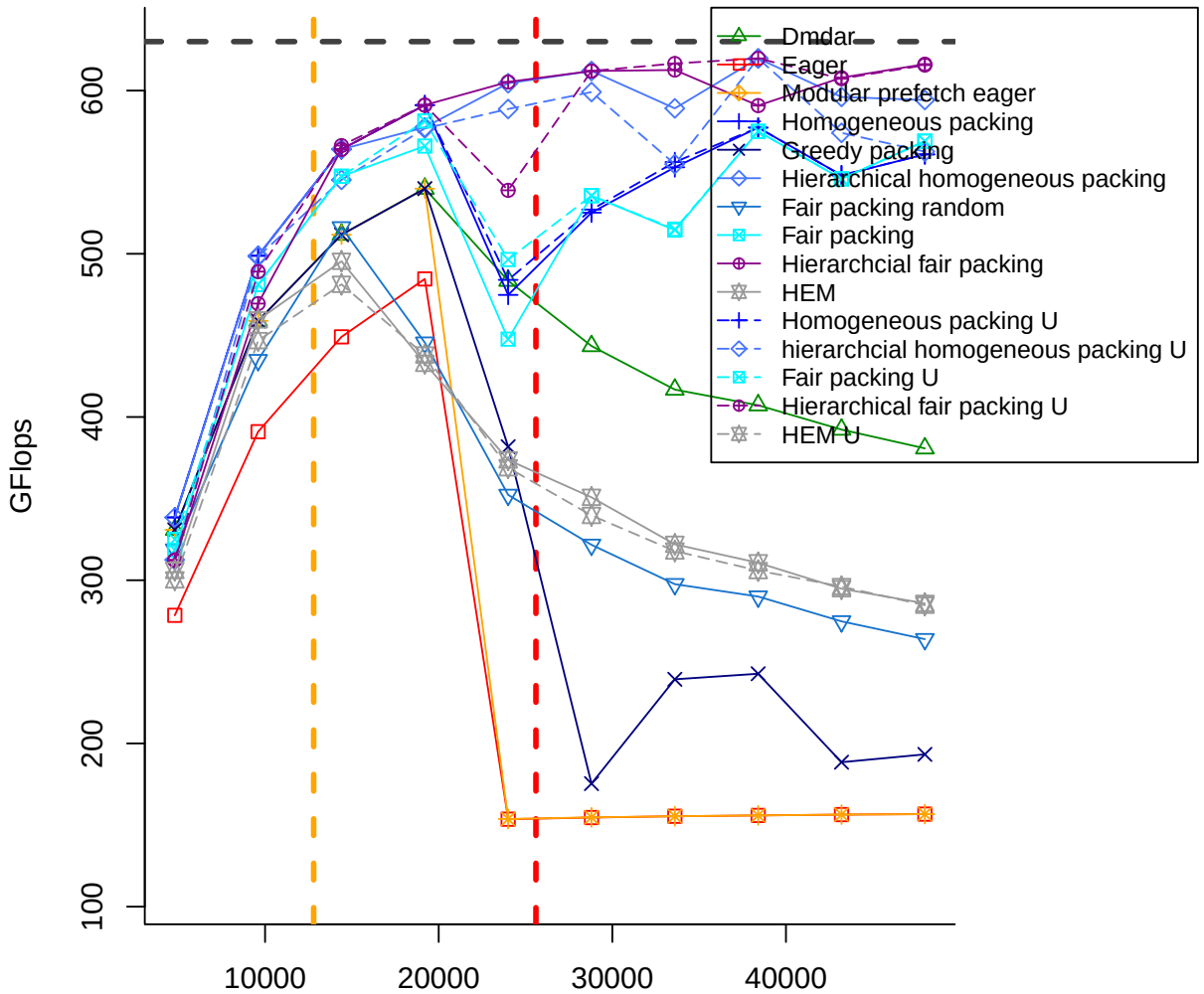
Thank you

Joker slides

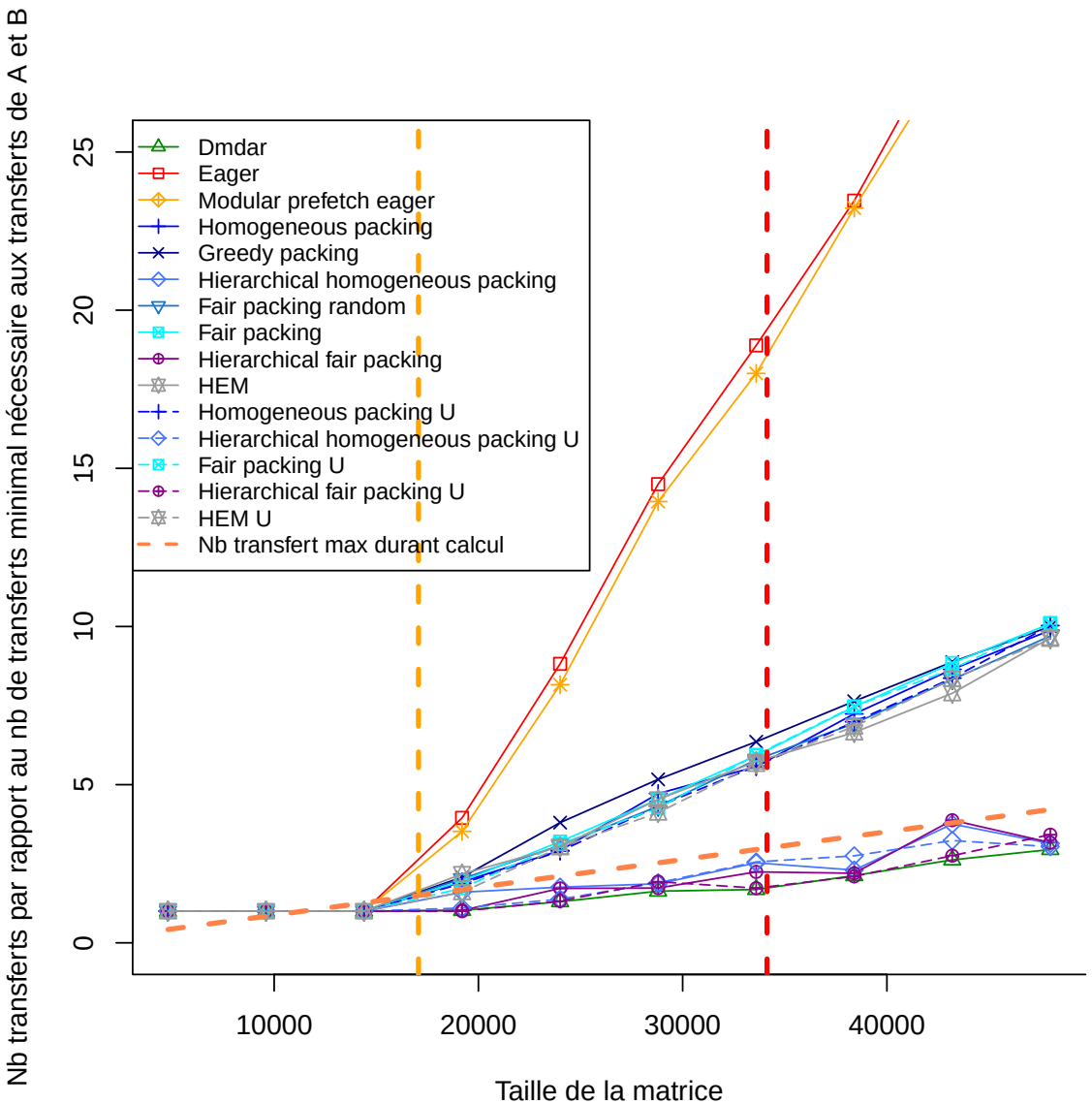
Number of transfers normalized according to the size of the matrix



Performances with a smaller GPU's memory



Number of transfers with random dependencies

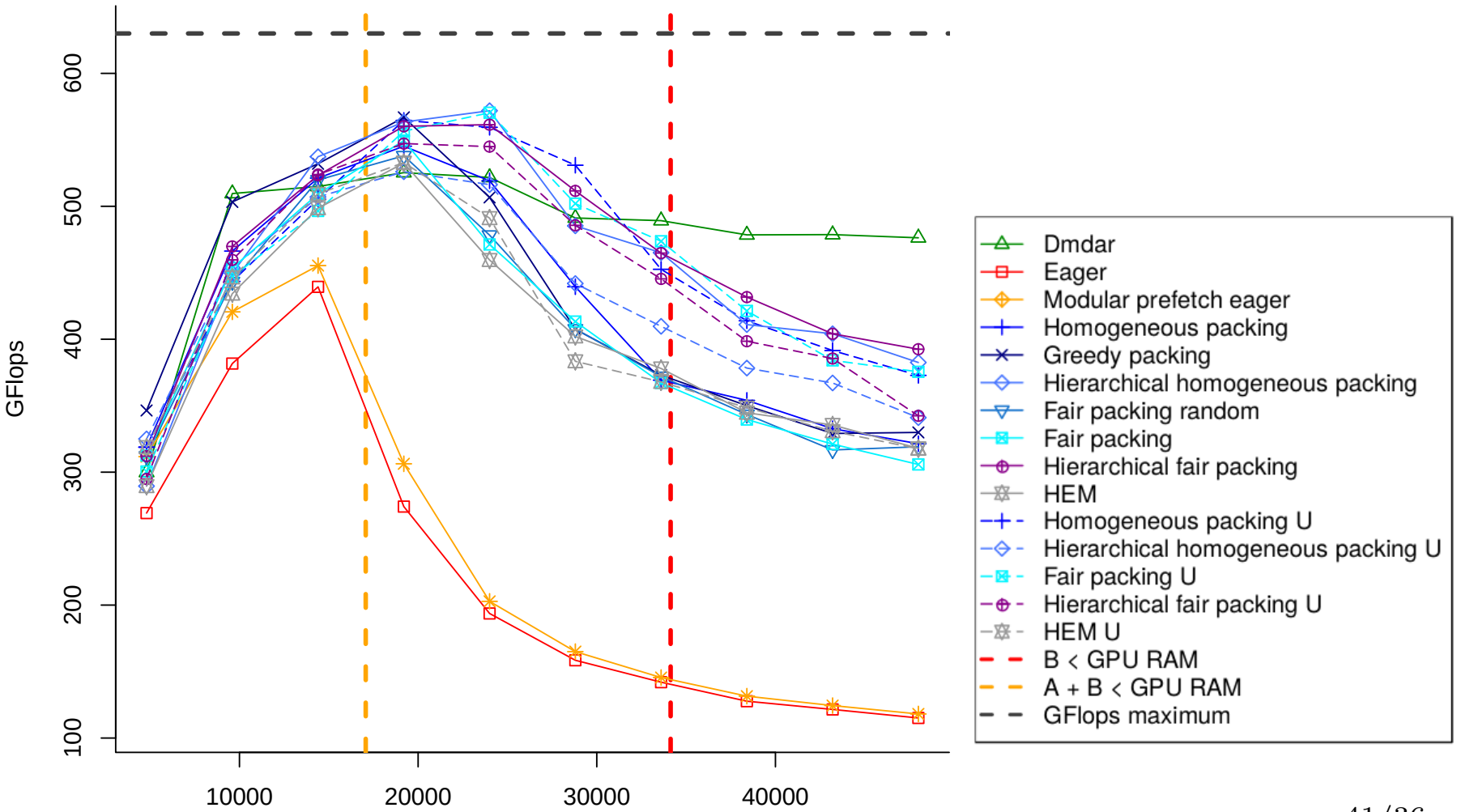


Computation order with Homogeneous packing with order U

2	5	17	14	43	46	58	55	19
3	4	16	15	44	45	57	56	18
9	6	10	13	50	47	51	54	60
8	7	11	12	49	48	52	53	59
38	35	23	26	78	75	63	66	40
37	36	24	25	77	76	64	65	39
31	34	30	27	71	74	70	67	80
32	33	29	28	72	73	69	68	79
0	1	22	21	41	42	62	61	20

Figure: Processing order of C with Homogeneous packing and with order U

Random order of arrival of tasks

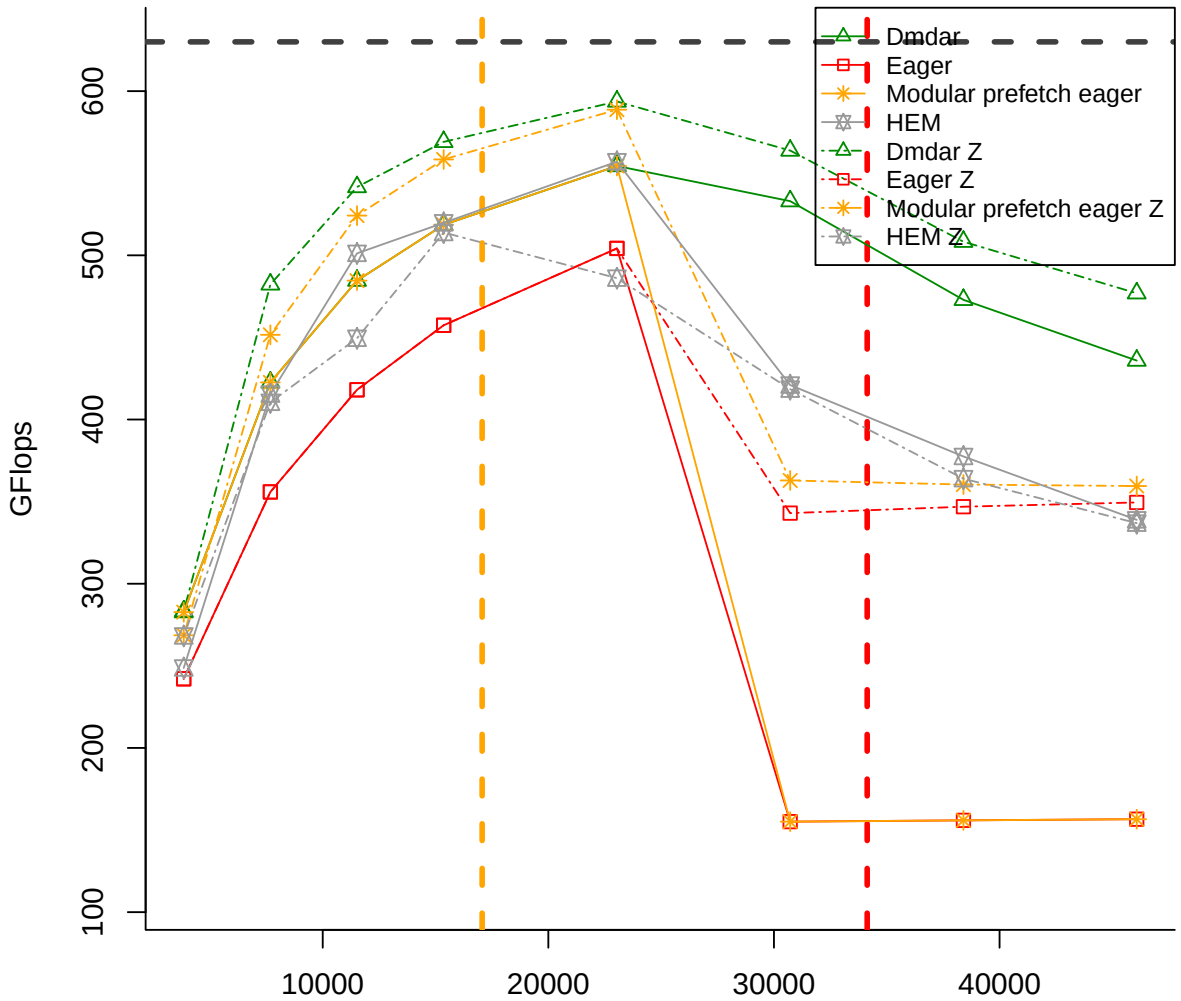


Packing example with random task arrival order

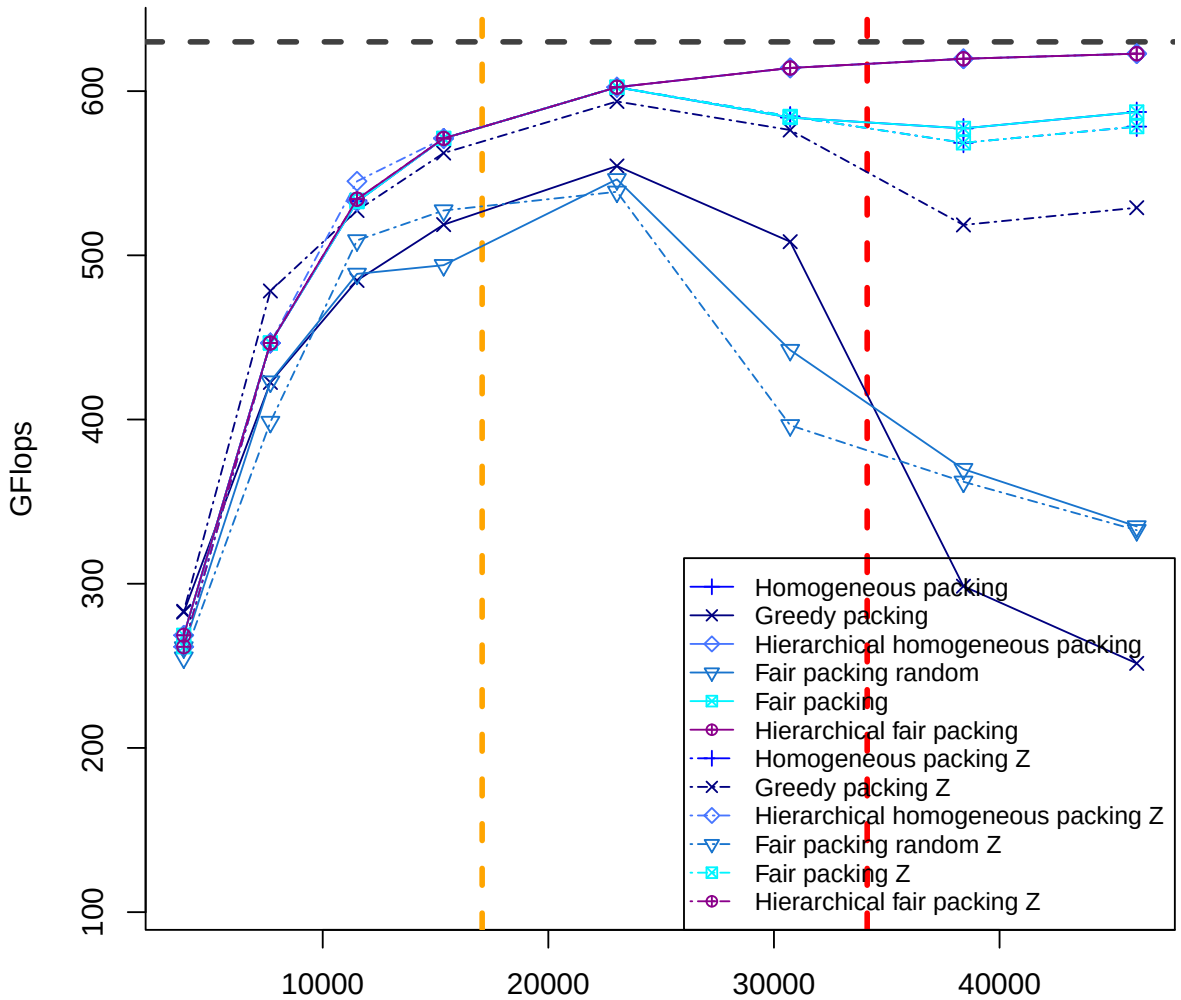
0	1	1	0	1	1	6	6	12	1
6	0	6	0	12	6	6	0	12	6
29	29	1	0	1	0	29	1	29	29
29	1	12	0	12	1	62	29	12	1
6	29	6	6	29	6	6	6	12	6
29	1	1	6	1	6	6	1	29	29
6	12	1	6	1	1	6	1	12	29
0	12	12	0	12	12	62	0	12	0
0	1	1	0	1	0	62	0	12	1
0	29	12	0	29	0	29	29	12	29

Figure: Homogeneous packing

StarPU's schedulers and HEM with order Z



AATO's algorithms with order Z



Greedy packing without order Z

0	0	15	25	48	48	63	73	96	96	111	121
0	0	25	25	48	48	73	73	96	96	121	121
0	0	25	25	48	48	73	73	96	96	121	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121
0	15	15	25	48	63	63	73	96	111	111	121

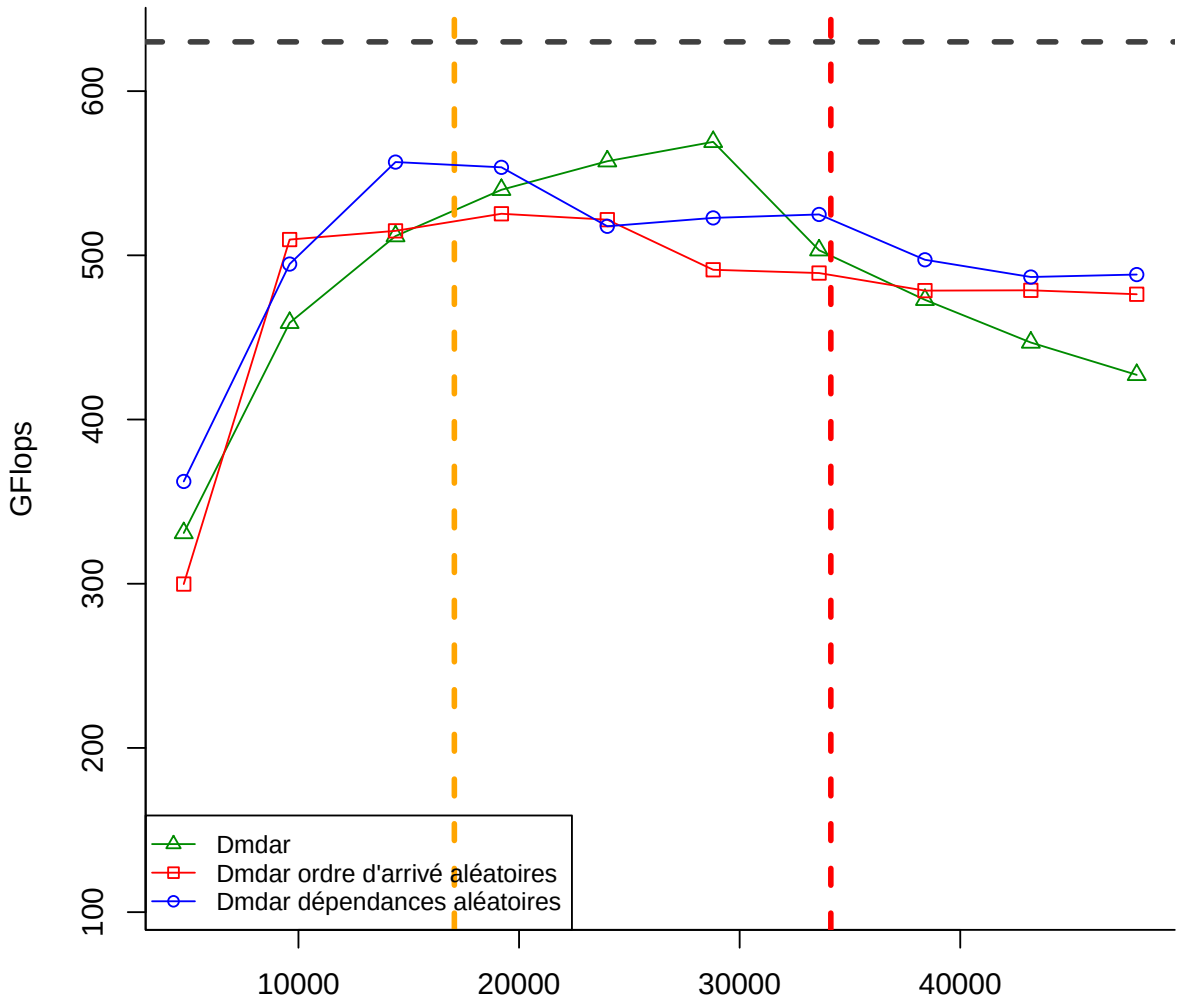
Figure: Packing of Greedy packing on a 12x12 matrix without order Z

Greedy packing with order Z

0	0	0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	0	0	25	25	25	25	25
0	0	0	0	0	0	25	25	25	25	25	25
48	48	48	48	48	48	25	25	25	25	25	25
48	48	48	48	48	48	48	75	75	75	75	75
48	48	48	48	48	48	48	75	75	75	75	75
48	48	48	48	48	48	48	75	75	75	75	75
75	75	99	99	99	99	99	75	75	75	75	75
75	99	99	99	99	99	99	123	123	123	123	123
99	99	99	99	99	99	99	123	123	123	123	123
99	99	99	99	99	99	99	123	123	123	123	123

Figure: Packing of Greedy packing on a 12x12 matrix with order Z

Improved performance of Dmdar



Complexity of Homogeneous packing

Best case: complete pairing at each step

$$\Delta \times m^2 + \frac{\Delta \times m^2}{2^1} + \frac{\Delta \times m^2}{2^2} + ... + \frac{\Delta \times m^2}{2^i} < O(\Delta \times m^2).$$

In the worst case

$$O(m^3 \times \Delta^2)$$

Eager

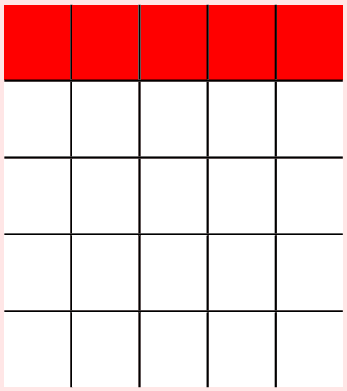


Figure: Matrix A

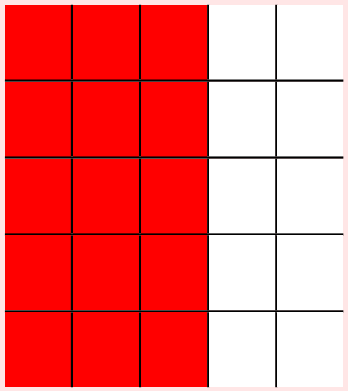


Figure: Matrix B

1	**2**	**3**	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure: Matrix C

Dmdar

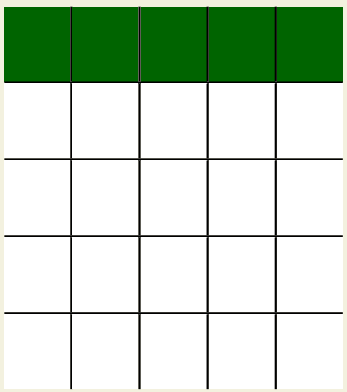


Figure: Matrix A

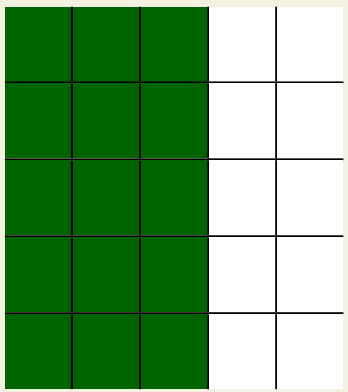


Figure: Matrix B

1	**2**	**3**	4	5
9	10	6	7	8
11	12	13	14	15
19	20	16	17	18
21	22	23	24	25

Figure: Matrix C

Eager

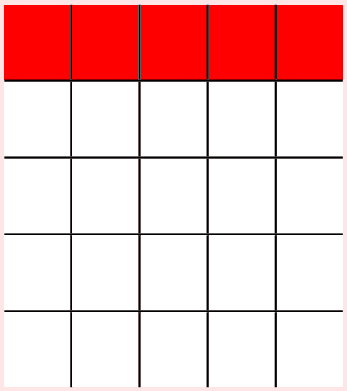


Figure: Matrix A

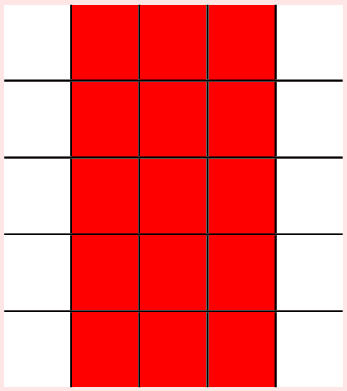


Figure: Matrix B

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure: Matrix C

Dmdar

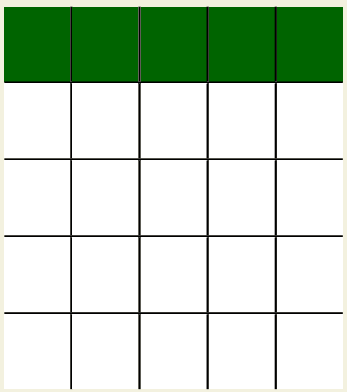


Figure: Matrix A

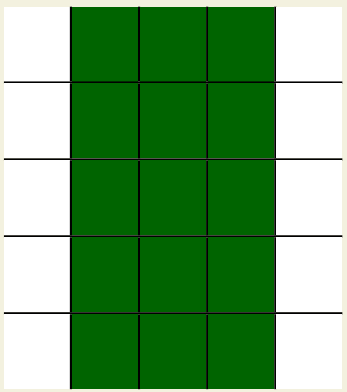


Figure: Matrix B

1	2	3	4	5
9	10	6	7	8
11	12	13	14	15
19	20	16	17	18
21	22	23	24	25

Figure: Matrix C

Eager

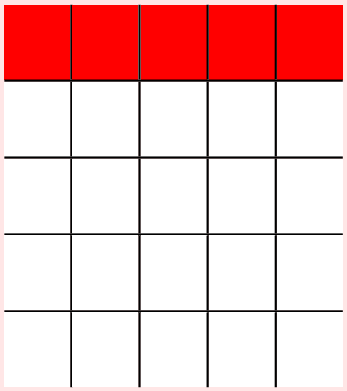


Figure: Matrix A

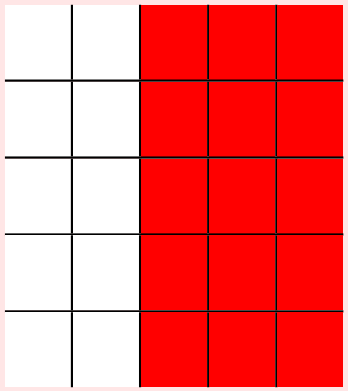


Figure: Matrix B

1	2	3	4	**5**
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure: Matrix C

Dmdar

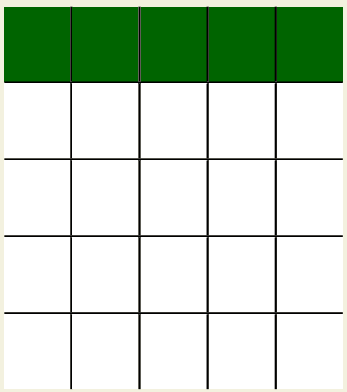


Figure: Matrix A

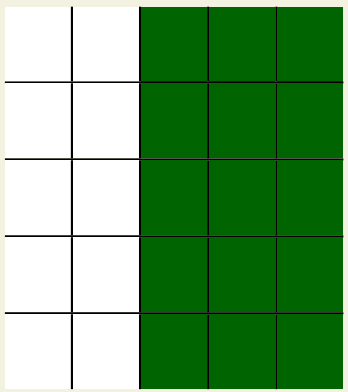


Figure: Matrix B

1	2	3	4	**5**
9	10	6	7	8
11	12	13	14	15
19	20	16	17	18
21	22	23	24	25

Figure: Matrix C

Eager

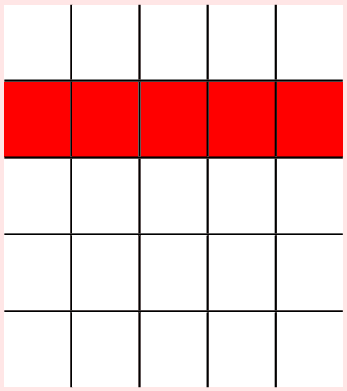


Figure: Matrix A

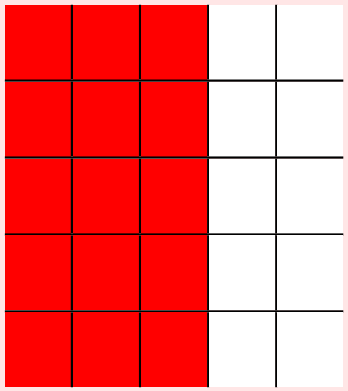


Figure: Matrix B

1	2	3	4	5
6	**7**	**8**	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure: Matrix C

Dmdar

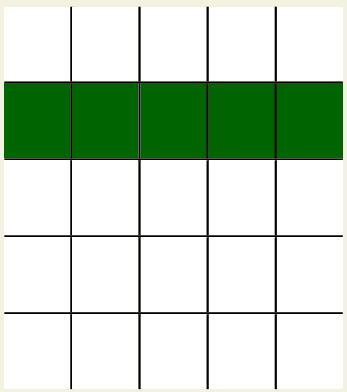


Figure: Matrix A

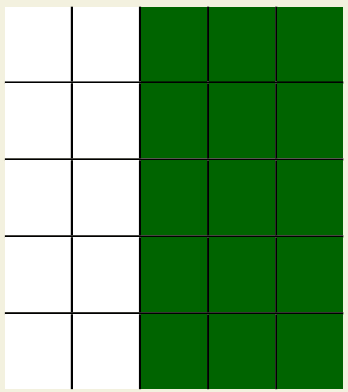


Figure: Matrix B

1	2	3	4	5
9	10	**6**	**7**	**8**
11	12	13	14	15
19	20	16	17	18
21	22	23	24	25

Figure: Matrix C

Eager

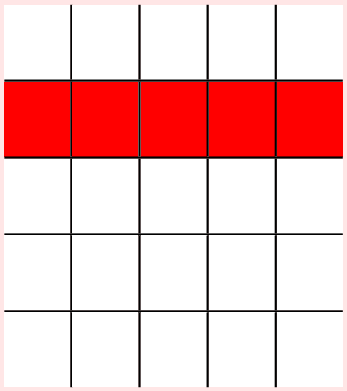


Figure: Matrix A

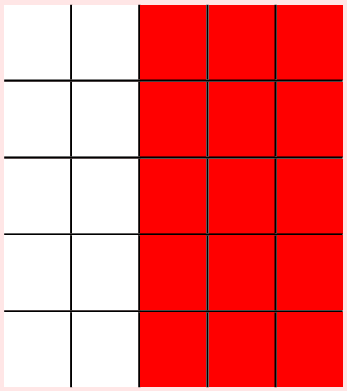


Figure: Matrix B

1	2	3	4	5
6	7	8	**9**	**10**
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure: Matrix C

Dmdar

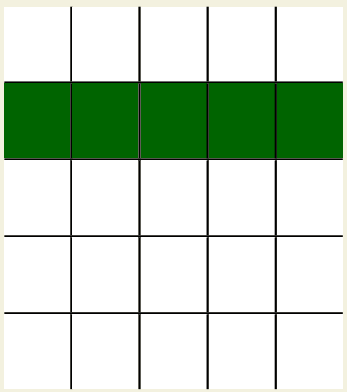


Figure: Matrix A

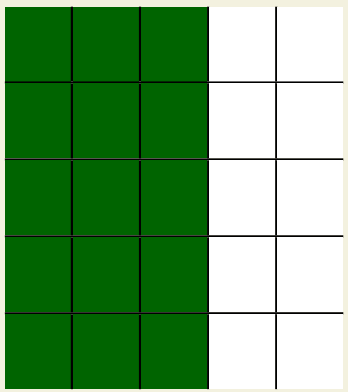


Figure: Matrix B

1	2	3	4	5
9	**10**	6	7	8
11	12	13	14	15
19	20	16	17	18
21	22	23	24	25

Figure: Matrix C

Performances of StarPU's schedulers

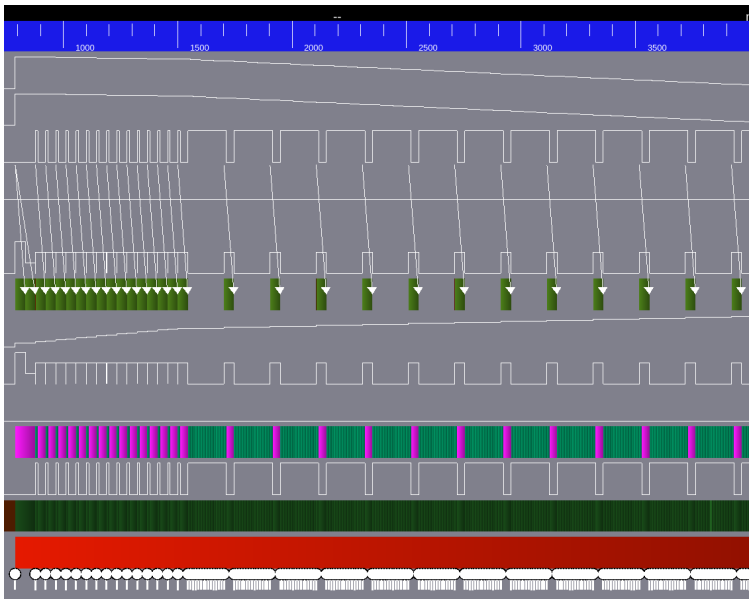


Figure: Eager's trace for a 15x15 matrix

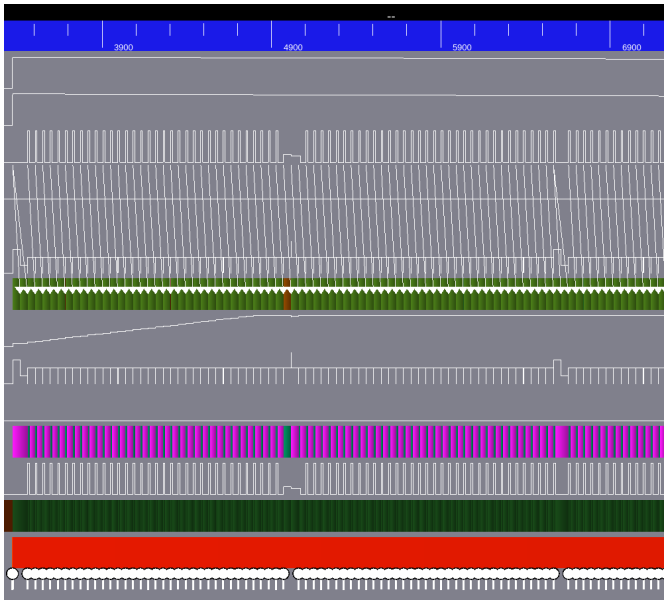


Figure: Eager's trace for a 35x35 matrix