Carbon-Aware Scheduling on Multiple Edge Servers

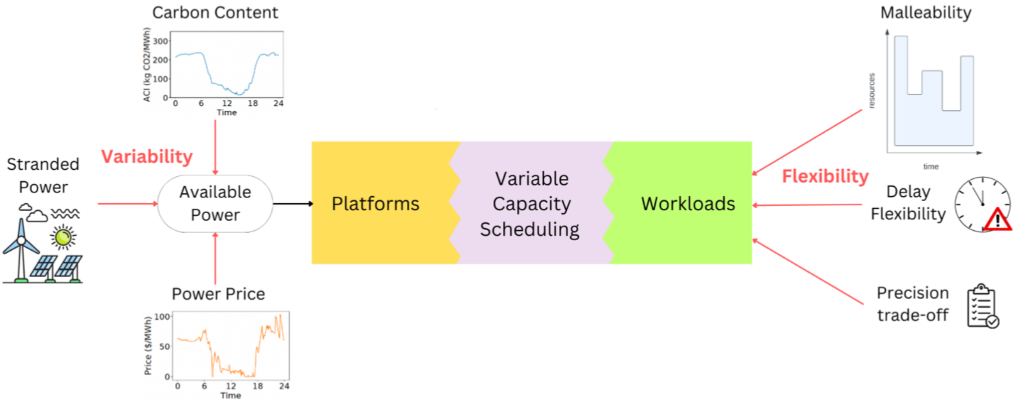
Anne Benoit
LIP, Ecole Normale Supérieure de Lyon
Institut Universitaire de France

Joint work with Y. Robert, J. Cendrier, F. Vivien (ENS Lyon)
and A. A. Chien, R. Wijayawardana (U. Chicago)

September 9, 2026 – CCDSC – La Maison des Contes

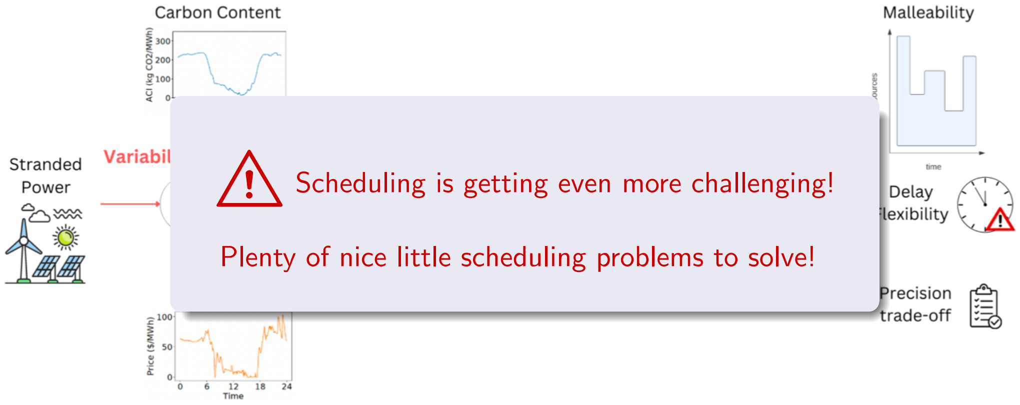
Where were we two years ago?

... Playing with variable capacity scheduling ☺



Where were we two years ago?

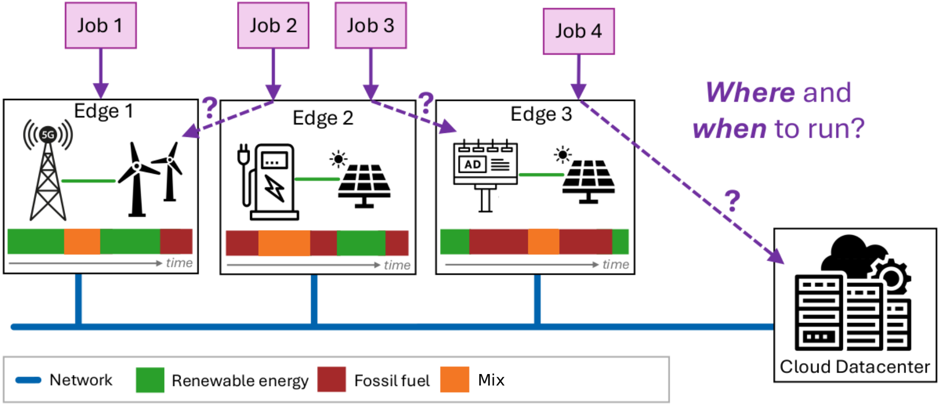
... Playing with variable capacity scheduling ☺



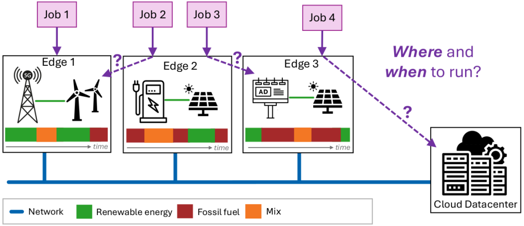
Scheduling is getting even more challenging!

Plenty of nice little scheduling problems to solve!

A nice little scheduling problem

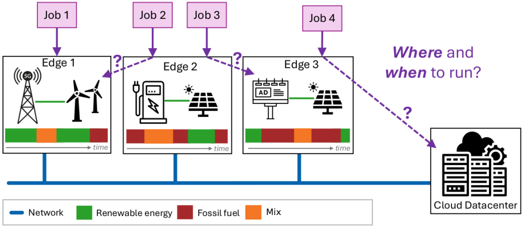


Platform



- Edge servers powered by renewable energy sources, with different levels of carbon intensity depending on time
- Migration possible between edge servers
- Powerful *cloud* server to avoid missed deadlines

Jobs



- Jobs arrive **online** with **deadlines** to respect
- Can be preempted and migrated during execution (cost for recovery/checkpoint)
- Possible execution on a big cloud server powered only by the grid
- Aim: complete all jobs before their deadlines while **minimizing the total carbon cost**

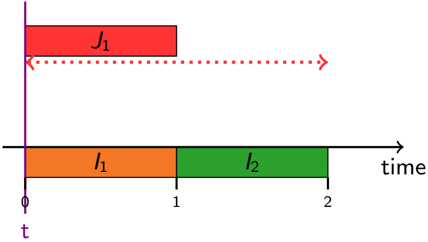
Contents

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



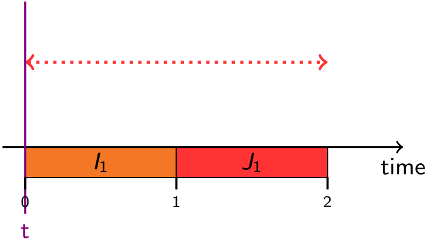
Two cases:

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

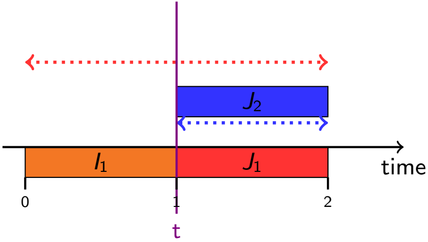
- k_2

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

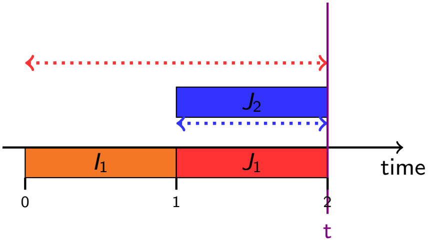
- $k_2 + K$

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

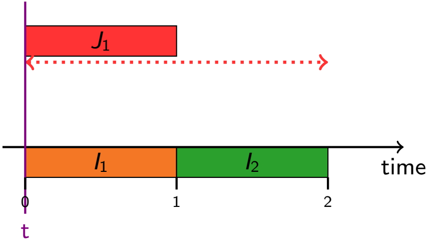
- $k_2 + K > k_1 + k_2$

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

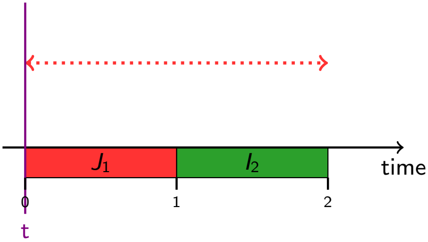
- $k_2 + K > k_1 + k_2$

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

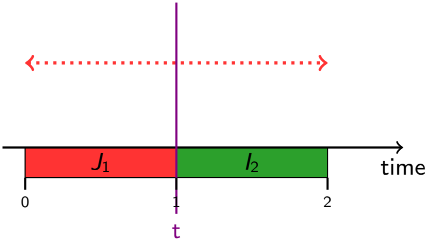
- $k_2 + K > k_1 + k_2$
- k_1

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

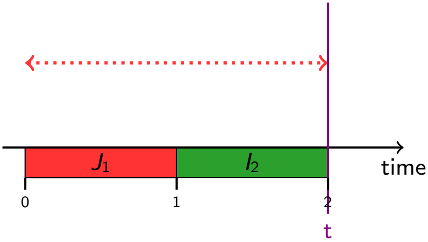
- $k_2 + K > k_1 + k_2$
- k_1

Online scheduling

Assumptions:

- **Online** (release dates and deadlines are unknown)
- One edge with total knowledge of energy intervals

Exemple with $k_1 > k_2$, respectively the carbon intensity of I_1 and I_2:



Two cases:

- $k_2 + K > k_1 + k_2$
- $k_1 > k_2$

Offline scheduling, one edge, $R + C > 0$

Assumptions:

- **Offline** (release dates and deadlines are known)
- One edge with total knowledge of energy intervals
- One job with $R + C > 0$

Complexity of the associated decision problem (carbon cost below K_{max}):

- **NP-hard** problem: proof by subset sum

Offline scheduling, one edge, $R + C = 0$

Assumptions:

- **Offline** (release dates and deadlines are known)
- One edge with total knowledge of energy intervals
- Multiple jobs with $R + C = 0$, which admit a valid schedule on the edge and available at time 0

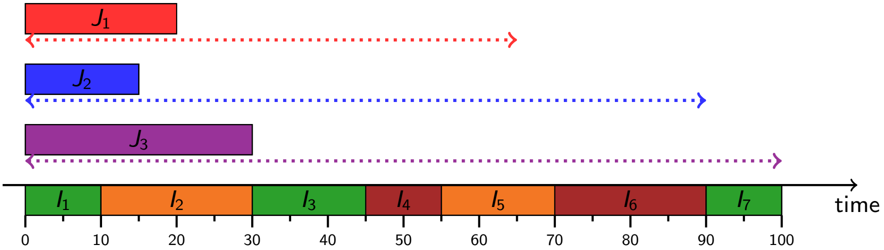
Optimal algorithm, OfflineGreenestPreemption, with **polynomial complexity** in: $O(u + mp + m\log(m))$

where u is the number of intervals in the edge server, m is the number of jobs and p the number of different carbon intensities

OfflineGreenestPreemption

One round per carbon type:

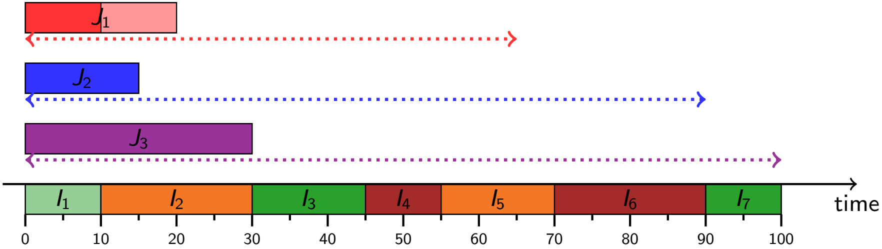
- Selection of the green intervals



OfflineGreenestPreemption

One round per carbon type:

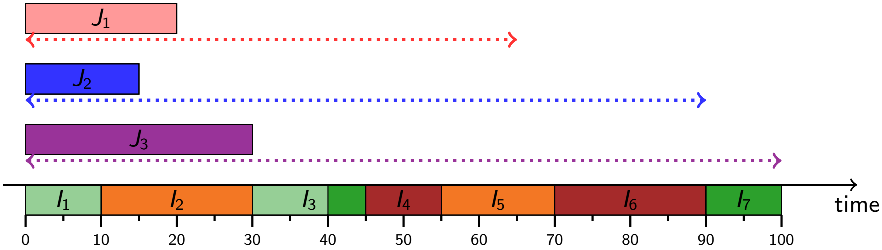
- Selection of the green intervals



OfflineGreenestPreemption

One round per carbon type:

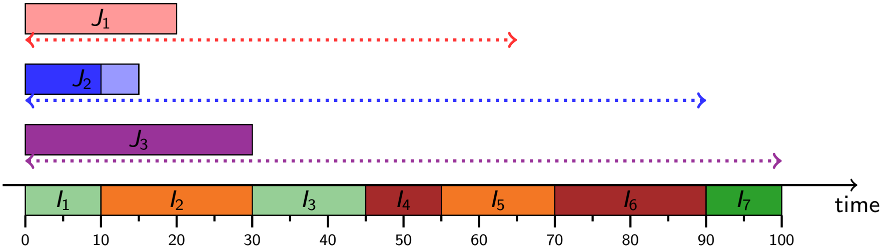
- Selection of the green intervals



OfflineGreenestPreemption

One round per carbon type:

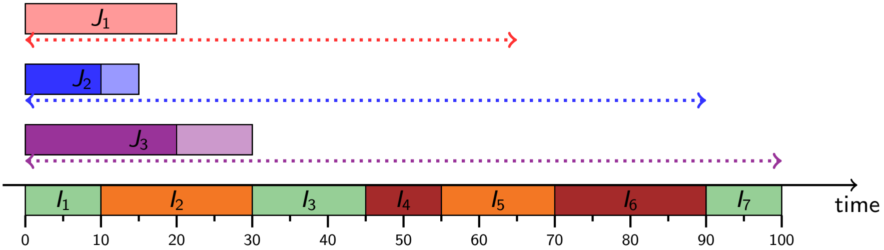
- Selection of the green intervals



OfflineGreenestPreemption

One round per carbon type:

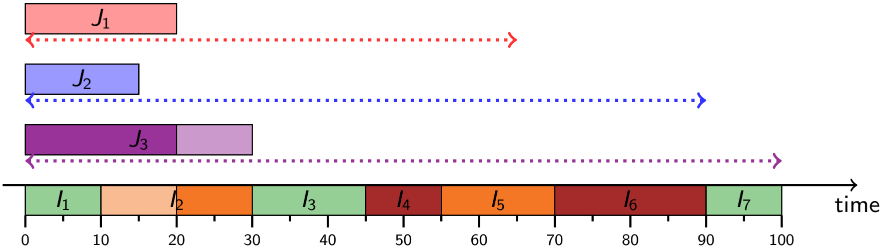
- Selection of the green intervals



OfflineGreenestPreemption

One round per carbon type:

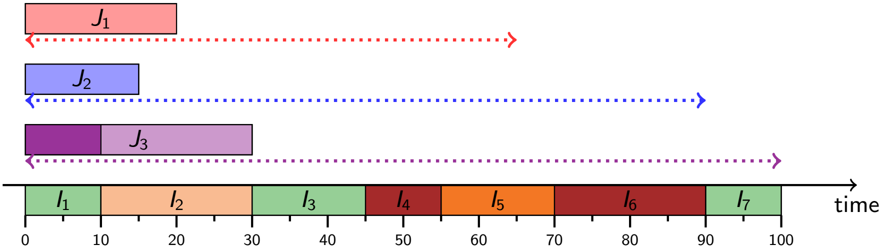
- Selection of the orange intervals



OfflineGreenestPreemption

One round per carbon type:

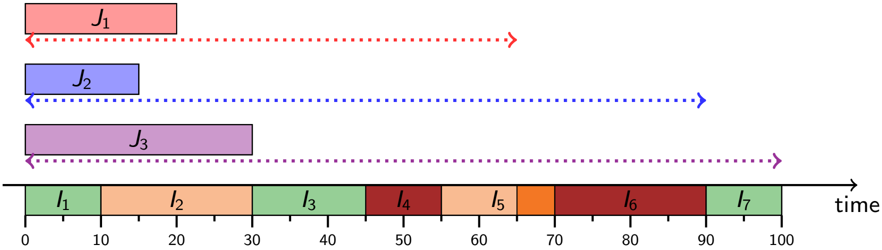
- Selection of the orange intervals



OfflineGreenestPreemption

One round per carbon type:

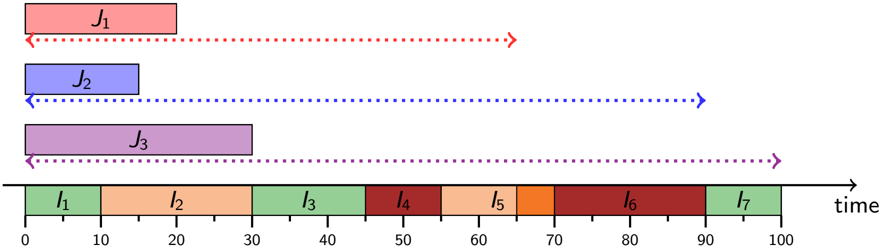
- Selection of the orange intervals



OfflineGreenestPreemption

One round per carbon type:

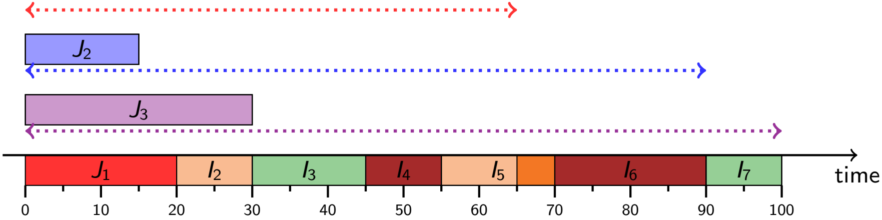
- Selection of the brown intervals



OfflineGreenestPreemption

One round per carbon type:

- Selection of the brown intervals

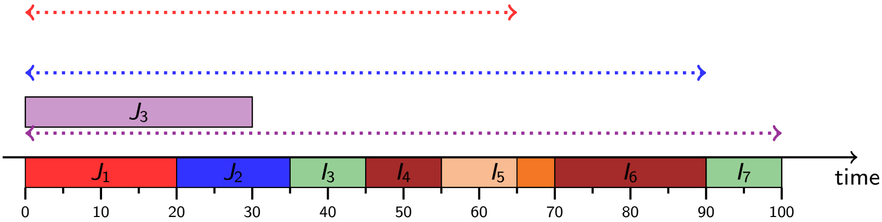


- Scheduling of jobs in the EDF order on the selected energy intervals

OfflineGreenestPreemption

One round per carbon type:

- Selection of the brown intervals

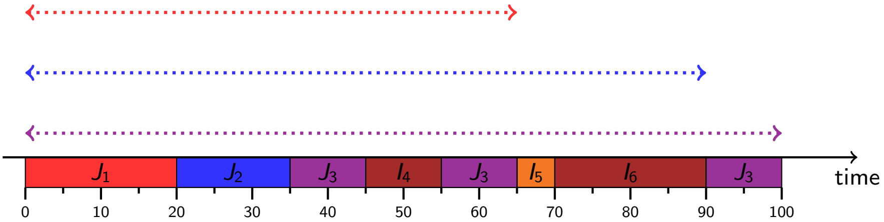


- Scheduling of jobs in the EDF order on the selected energy intervals

OfflineGreenestPreemption

One round per carbon type:

- Selection of the brown intervals



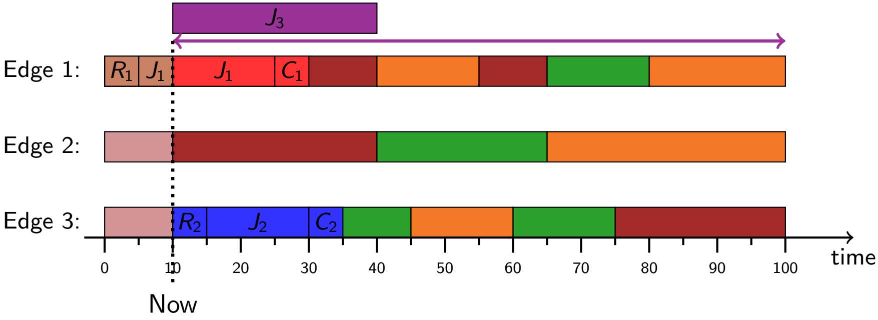
- Scheduling of jobs in the EDF order on the selected energy intervals

Contents

Baseline heuristics

Take decisions only when a new job is released, and only for this job:

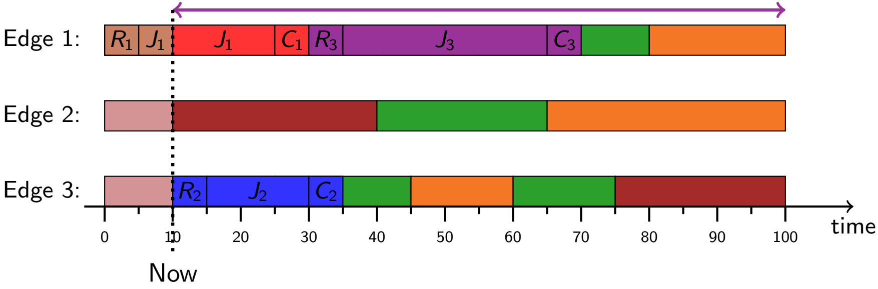
- LOCAL: Schedules each job at the earliest on its edge
- ECT: Schedules each job at the earliest on any edge



Baseline heuristics

Take decisions only when a new job is released, and only for this job:

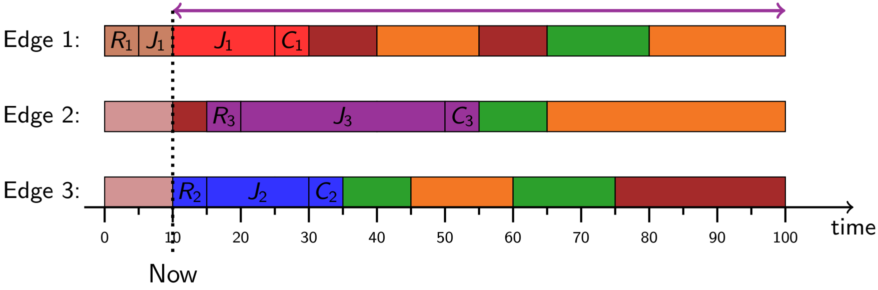
- LOCAL: **Schedules each job at the earliest on its edge**
- ECT: Schedules each job at the earliest on any edge



Baseline heuristics

Take decisions only when a new job is released, and only for this job:

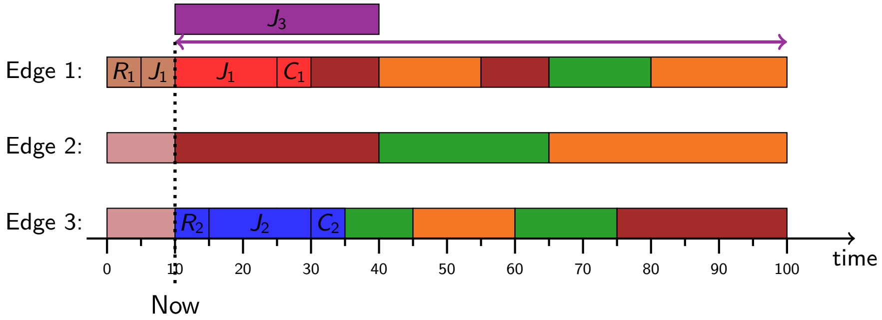
- LOCAL: Schedules each job at the earliest on its edge
- ECT: **Schedules each job at the earliest on any edge**



Green-aware heuristics

Take decisions only when a new job is released, and only for this job:

- LocalGreen: Schedules each job at the earliest on its edge but only using greenest energy interval
- EGT: Schedules each job at the earliest on any edge but only using greenest energy interval

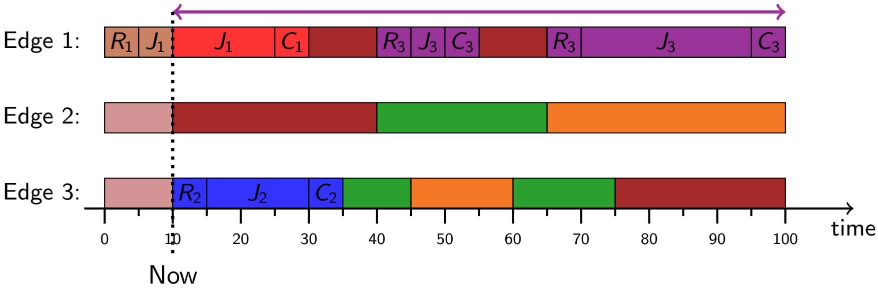


How can we define green energy? Should depend on the system load!

Green-aware heuristics

Take decisions only when a new job is released, and only for this job:

- LocalGreen: **Schedules each job at the earliest on its edge but only using greenest energy intervals**
- EGT: Schedules each job at the earliest on any edge but only using greenest energy interval

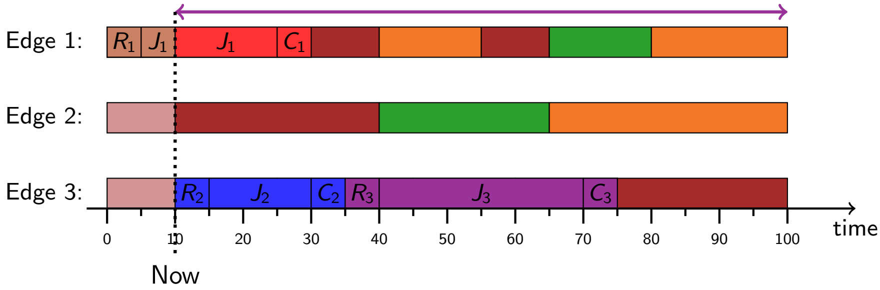


How can we define green energy? Should depend on the system load!

Green-aware heuristics

Take decisions only when a new job is released, and only for this job:

- LOCALGREEN: Schedules each job at the earliest on its edge but only using greenest energy interval
- EGT: **Schedules each job at the earliest on any edge but only using greenest energy intervals**



How can we define green energy? Should depend on the system load!

Re-evaluation heuristics

General idea:

- Each time a new job is submitted, cancels the current schedule and computes a new one, job by job (in a specific order, such as EDF)

Re-evaluation heuristics

General idea:

- Each time a new job is submitted, cancels the current schedule and computes a new one, job by job (in a specific order, such as EDF)

Two strategies:

- ReallocMigration: Try to shedule the job on each edge server (using OfflineGreenestPreemption with the already placed jobs) and choose the server with the lowest variation in carbon cost
- ReallocInPlace: Same than ReallocMigration, but when a job starts its execution on an edge, it will do all of its execution on this edge

Re-evaluation heuristics

General idea:

- Each time a new job is submitted, cancels the current schedule and computes a new one, job by job (in a specific order, such as EDF)

Two strategies:

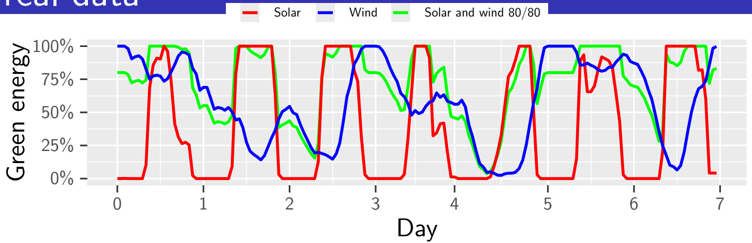
- ReallocMigration: Try to shedule the job on each edge server (using OfflineGreenestPreemption with the already placed jobs) and choose the server with the lowest variation in carbon cost
- ReallocInPlace: Same than ReallocMigration, but when a job starts its execution on an edge, it will do all of its execution on this edge

Predictions:

- Re-evaluation of the current schedule when there is an update to the predictions of edges' energy intervals

Contents

Traces from real data



- Simulation length: $T = 30$ days
- One year of ERCOT data (2024)
- 10 edge servers, with various on-site generation models and a **3-day forecast**:
  - Solar only, Wind only, Solar and wind, Mix: 3 solar, 3 wind, 4 solar and wind
- CLOUD carbon cost K ; carbon cost of transfer k^{trans}: depends on the month
- b^r: 2.15 Gbit/s ; 3.5 Gbit/s for Cloud
- b^w: 2.15 Gbit/s ; 7 Gbit/s for Cloud
- b^{trans}: 10 Gbit/s

Synthetics simulation parameters

- Job duration: between 1 hour and 24 hours, with various mean, depends on job skew
- Job skew: $[1.1, 1.15, 1.18, 1.2, 1.4, 1.6]$
- Job data volume:
  - proportional to the data volume of the job, in $[17, 409]$ Gbit
  - fixed at $[1, 10, 50, 100, 200, 400]$ Gbit
  - uniformly distributed in $[[1, 50], [1, 100], [1, 200], [1, 400], [50, 200], [200, 400]]$ Gbit
- Load $\in \{0.1, 0.2, \ldots, 1\}$
- Looseness $= \frac{d_j - s_j}{\ell_j}$: $\{1.15, 1.5, 2, 4, 6, 10, 20\} \pm 10\%$
- Job arrival models:
  - Uniform: the workload is distributed uniformly across all 10 edges
  - Clustered: 30% of the edges receive 90% of the workload
  - Event/Mall: one edge receives 80% of the workload

Run experiments by selecting default or outsider value for each parameter (at most one outsider value by experiment)

First results

Metrics:

- Ratio full knowledge: comparison of the total carbon cost with partial / total knowledge of the energy interval costs

Heuristics	*GeoMean ratio full knowledge*		
AllCloud	1.00		
Local	1.00		
LocalGreen	1.01		
ECT	1.00		
EGT	1.00		
ReallocMigration	1.09		
ReallocInPlace	1.07		

Table 1: Statistics on default instances. Sorted by ratio ECT values.

- Prediction does not degrade performance too much

First results

Metrics:

- Ratio full knowledge: comparison of the total carbon cost with partial / total knowledge of the energy interval costs
- Ratio ECT: comparison of the total carbon cost of a heuristic for an instance with the total carbon cost of the ECT heuristic

Heuristics	*GeoMean ratio full knowledge*	*GeoMean ratio ECT*	
AllCloud	1.00	2.02	
Local	1.00	1.23	
LocalGreen	1.01	1.18	
ECT	1.00	1.00	
EGT	1.00	0.88	
ReallocMigration	1.09	0.61	
ReallocInPlace	1.07	0.59	

Table 1: Statistics on default instances. Sorted by ratio ECT values.

- Prediction does not degrade performance too much
- Good performance for re-evaluation algorithm

First results

Metrics:

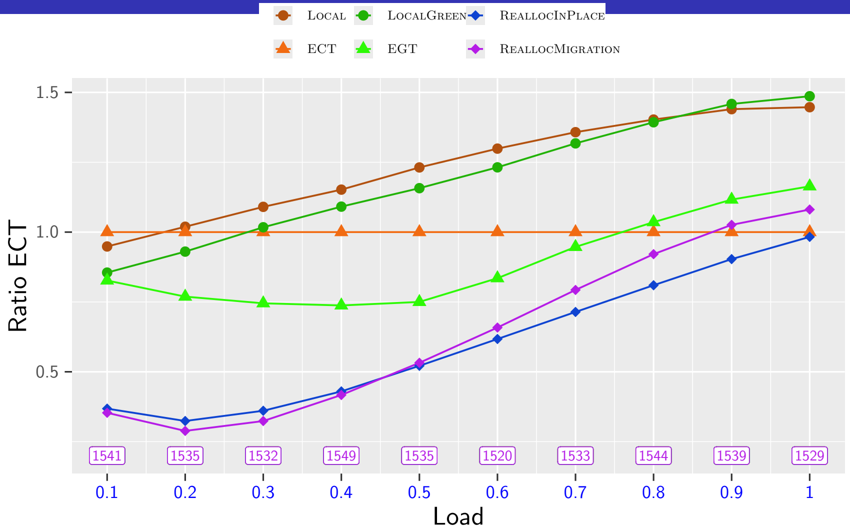
- Ratio full knowledge: comparison of the total carbon cost with partial / total knowledge of the energy interval costs
- Ratio ECT: comparison of the total carbon cost of a heuristic for an instance with the total carbon cost of the ECT heuristic
- Stretch: ratio between execution time and duration of jobs on edge servers

Heuristics	*GeoMean ratio full knowledge*	*GeoMean ratio ECT*	*GeoMean stretch*
AllCloud	1.00	2.02	1.00
Local	1.00	1.23	1.47
LocalGreen	1.01	1.18	1.99
ECT	1.00	1.00	1.07
EGT	1.00	0.88	2.21
ReallocMigration	1.09	0.61	2.57
ReallocInPlace	1.07	0.59	2.46

Table 1: Statistics on default instances. Sorted by ratio ECT values.

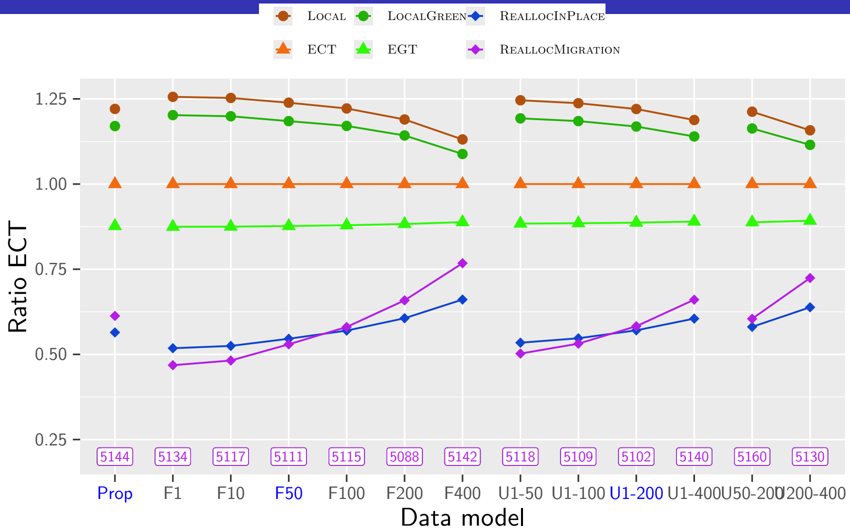
- Prediction does not degrade performance too much
- Good performance for re-evaluation algorithm at the expense of the stretch

Impact of the load on ratio ECT



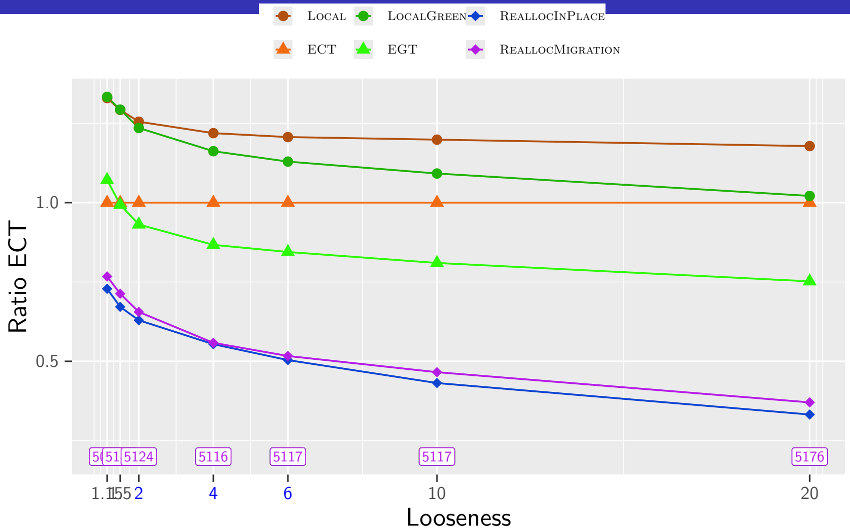
- Re-evaluation algorithms have very good gains at around 1/4 load

Impact of the data volume on ratio ECT



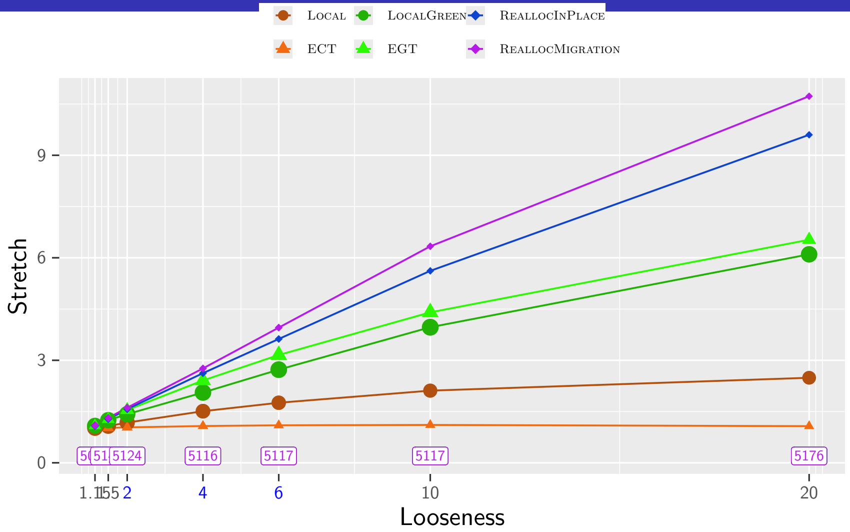
- Local-oriented algorithms are better when data volume is high

Impact of the looseness on ratio ECT



- Higher looseness implies better performance...

Impact of the looseness on stretch



- … but at the cost of the stretch

Conclusion

- Model of a complex edge scheduling problem
- Complexity results for the general problem and its subproblems
- Optimal polynomial algorithm to schedule jobs with free preemption
- 3-day prediction is enough to avoid significantly degrading performance
- Reevaluation heuristics offer robust performance