Green Scheduling on the Edge

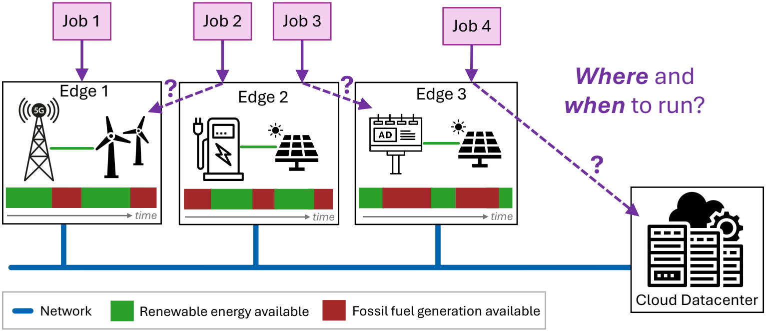
Anne Benoit[1] Joachim Cendrier[1] Andrew A. Chien[2]
Yves Robert[1] Frédéric Vivien[1] Rajini Wijayawardana[2]

[1]ENS Lyon, LIP, ROMA

[2]University of Chicago

July 9, 2025

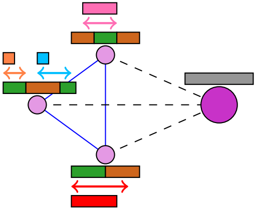
The problem



- Edge servers are connected to the energy grid and to renewable energy sources: *green* and *brown* energy intervals known in advance
- Jobs have **deadlines** to respect
- Possible execution on a big distant cloud server with large carbon cost (transfer + computation)
- Aim: complete all jobs before their deadlines while **minimizing the total carbon cost**

The model

- Set of n identical edge servers, and each edge e_i has *green* and *brown* intervals which respective carbon cost of 0 and k
- A CLOUD server with a higher carbon intensity and speed: $\frac{K}{s_{cloud}} \geq k$
- Set of m jobs, for each job J_j:
  - ℓ_j: execution time of job J_j on an edge
  - r_j: release date of job J_j
  - d_j: deadline of job J_j
  - o_j: arrival (and departure) edge of job J_j
  - f_j: communication volume of job J_j



Communications:

- Complete interconnection network
- Transfer time linear in the communication volume of the job: $\frac{f_j}{b_{trans}}$
- Carbon cost linear in the communication volume of the job: $f_j k_{trans}$

Assumptions and objective function

Assumptions:

- **Full knowledge of energy intervals** on all edges
- **Jobs arrive online**
- We can pause and resume (**freeze**) a job without any penalty (but neither preemption nor migration)

All jobs must be completed before their deadlines (potentially using the CLOUD)

Objective function: minimization of the total carbon cost:

$$\min\left(\sum_{0\leq j<m}\left(\left(\alpha_j k + \delta_j \frac{K}{s_{cloud}}\right)\ell_j + tr_j f_j k_{trans}\right)\right)$$

where α_j is the fraction of the job J_j executed using *brown* energy and δ_j indicates whether the job is executed on the CLOUD ($\delta_j \in \{0,1\}$ and $\alpha_j + \delta_j \leq 1$), tr_j is the number of transfers of job J_j.

Contents

Contents

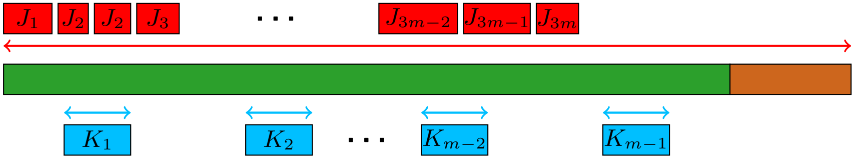
Complexity for the single edge, offline case

Assumptions:

- **One edge**
- **Offline** (release dates and deadlines are known)

Complexity:

- **Strongly NP-Complete** problem: proof by 3-partition



Algorithm for the single edge, offline case and ordered jobs

Assumptions:

- **One edge**
- **Offline** (release dates and deadlines are known)

Algorithm divided into two phases:

- **Ordering** of the jobs: e.g., Earliest Deadline First (EDF)
- Optimal linear algorithm, OFFLINEGREENEST, for job **scheduling**

Simplifying notations

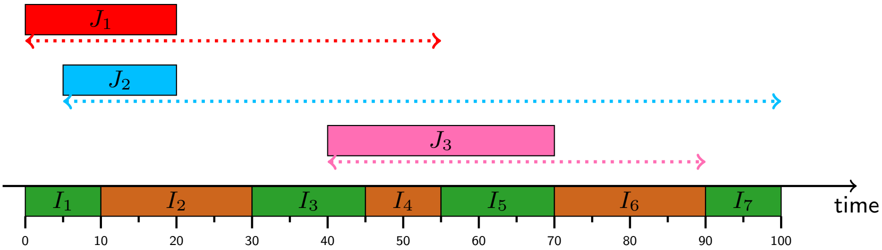
Jobs are **ordered**: $\forall i, j \in [1, m], i < j$ job J_i must complete before job J_j starts

- rr_j: earliest starting time for job J_j

$$rr_1 = r_1,$$
$$\forall j \in [2, m],\ rr_j = \max(rr_{j-1} + \ell_{j-1}, r_j)$$

- rd_j: latest completion time for job J_j

$$rd_m = d_m,$$
$$\forall j \in [1, m-1],\ rd_j = \min(rd_{j+1} - \ell_{j+1}, d_j)$$



Simplifying notations

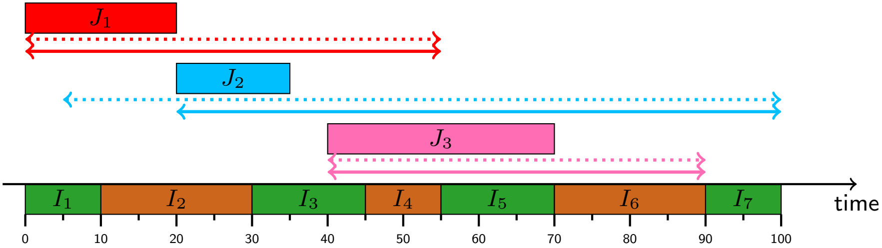
Jobs are **ordered**: $\forall i, j \in [1, m], i < j$ job J_i must complete before job J_j starts

- rr_j: earliest starting time for job J_j

$$rr_1 = r_1,$$
$$\forall j \in [2, m],\ rr_j = \max(rr_{j-1} + \ell_{j-1}, r_j)$$

- rd_j: latest completion time for job J_j

$$rd_m = d_m,$$
$$\forall j \in [1, m-1],\ rd_j = \min(rd_{j+1} - \ell_{j+1}, d_j)$$



Simplifying notations

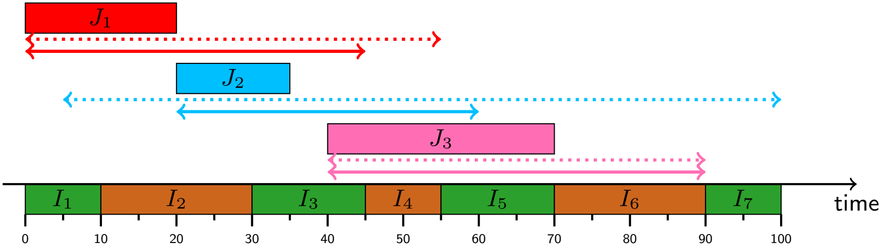
Jobs are **ordered**: $\forall i, j \in [1, m], i < j$ job J_i must complete before job J_j starts

- rr_j: earliest starting time for job J_j

$$rr_1 = r_1,$$
$$\forall j \in [2, m],\ rr_j = \max(rr_{j-1} + \ell_{j-1}, r_j)$$

- rd_j: latest completion time for job J_j

$$rd_m = d_m,$$
$$\forall j \in [1, m-1],\ rd_j = \min(rd_{j+1} - \ell_{j+1}, d_j)$$



Simplifying notations

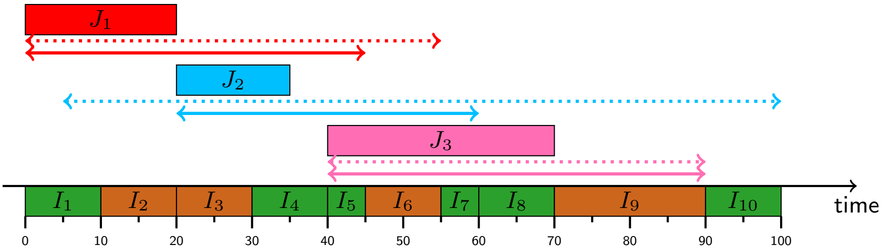
Jobs are **ordered**: $\forall i, j \in [1, m], i < j$ job J_i must complete before job J_j starts

- rr_j: earliest starting time for job J_j

$$rr_1 = r_1,$$
$$\forall j \in [2, m],\ rr_j = \max(rr_{j-1} + \ell_{j-1}, r_j)$$

- rd_j: latest completion time for job J_j

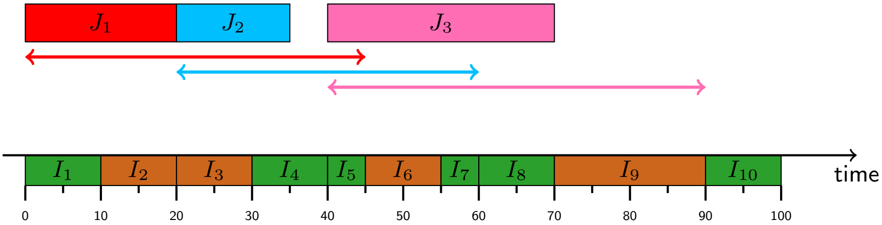
$$rd_m = d_m,$$
$$\forall j \in [1, m-1],\ rd_j = \min(rd_{j+1} - \ell_{j+1}, d_j)$$



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

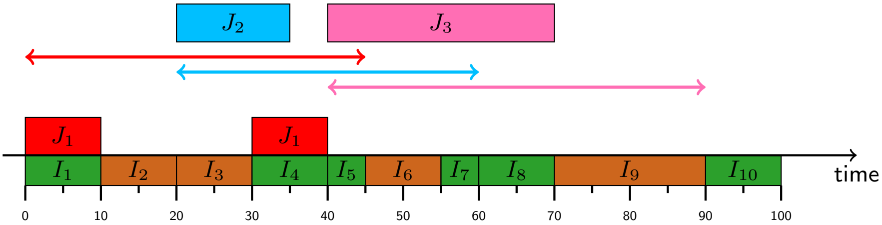
- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

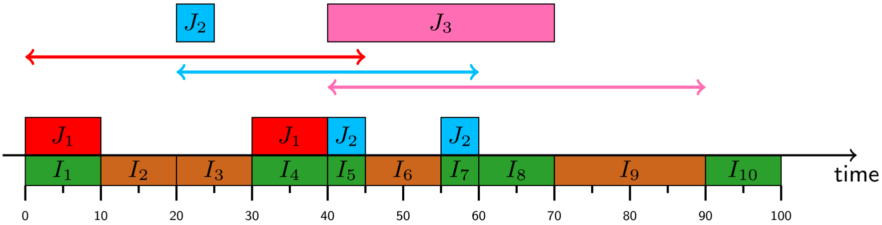
- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

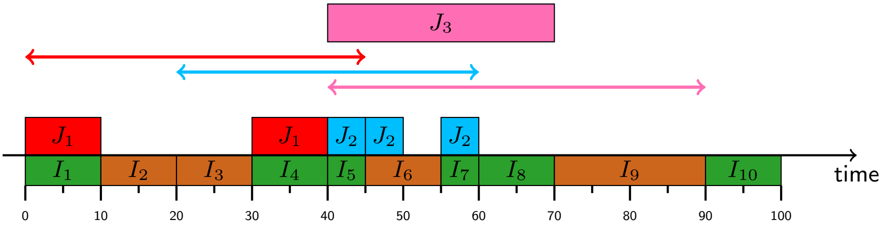
- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

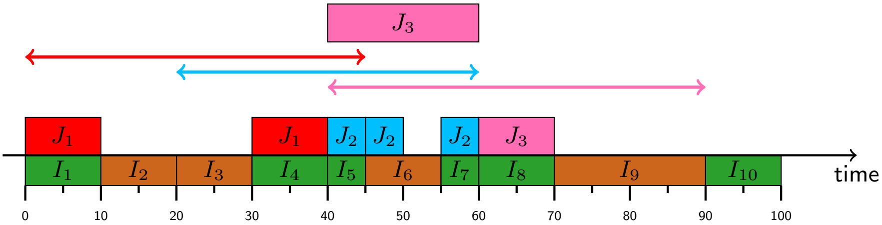
- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

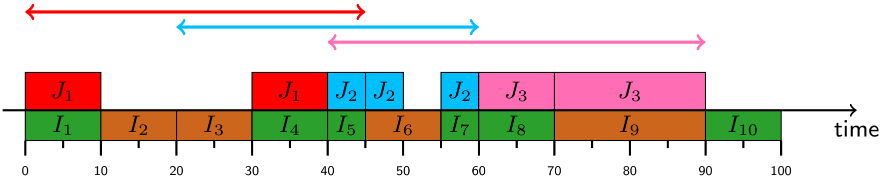
- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed



An optimal Greedy algorithm for the offline case

For each job J_j in the **order**:

- Browse from the completion time of the previous job to rd_j, allocating $green$ worktime while J_j not completed
- Browse from the completion time of the previous job to rd_j, allocating $brown$ worktime while J_j not completed

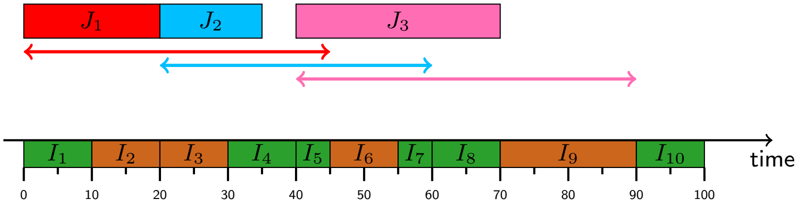


Carbon cost of the solution: $25k$; completion time: 90

Optimal for an offline problem but with a poor behavior for an online problem

OfflineGreenest

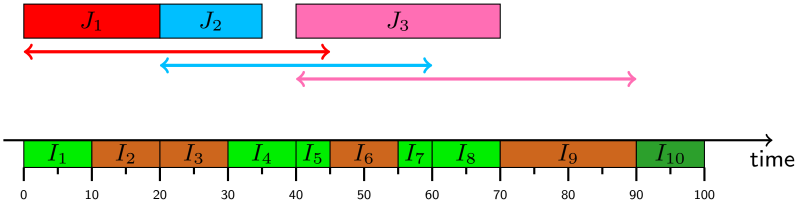
Two-rounds algorithm:



OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*



First round:

$$rr_j \rightarrow \mathrm{S}_i = \mathrm{S}_{i-1} + \ell_j$$

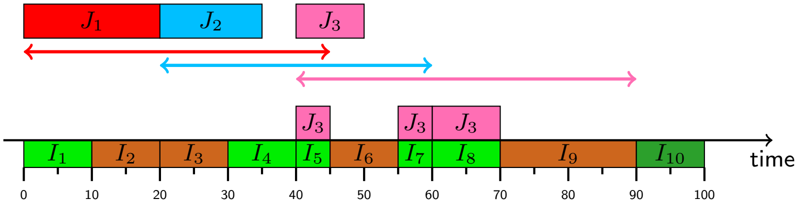
$$green\ I_i \rightarrow \mathrm{PF}_i = \mathrm{PF}_{i-1} + \min(|I_i|, \mathrm{S}_{i-1} - \mathrm{PF}_{i-1})$$

$$rd_j \rightarrow \begin{cases} \mathrm{S}_i & = \mathrm{S}_{i-1} - \ell_j \\ \mathrm{PF}_i & = \max(0, \mathrm{PF}_{i-1} - \ell_j) \end{cases}$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



First round:

$$
\begin{aligned}
rr_j &\rightarrow \mathrm{S}_i = \mathrm{S}_{i-1} + \ell_j \\
green\ I_i &\rightarrow \mathrm{PF}_i = \mathrm{PF}_{i-1} + \min(|I_i|, \mathrm{S}_{i-1} - \mathrm{PF}_{i-1}) \\
rd_j &\rightarrow \begin{cases} \mathrm{S}_i & = \mathrm{S}_{i-1} - \ell_j \\ \mathrm{PF}_i & = \max(0, \mathrm{PF}_{i-1} - \ell_j) \end{cases}
\end{aligned}
$$

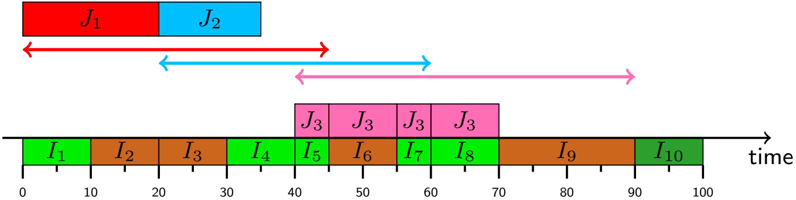
Second round:

$$
\begin{aligned}
rr_j &\rightarrow \begin{cases} missing_j & = \max(0, \ell_j - \mathrm{PB}_{i-1}) \\ \mathrm{PB}_i & = \max(0, \mathrm{PB}_{i-1} - \ell_j) \end{cases} \\
green\ I_i &\rightarrow \mathrm{PB}_i = \mathrm{PB}_{i-1} + |I_i|
\end{aligned}
$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



First round:

$$
\begin{aligned}
rr_j &\rightarrow \mathrm{S}_i = \mathrm{S}_{i-1} + \ell_j \\
green\ I_i &\rightarrow \mathrm{PF}_i = \mathrm{PF}_{i-1} + \min(|I_i|, \mathrm{S}_{i-1} - \mathrm{PF}_{i-1}) \\
rd_j &\rightarrow \begin{cases} \mathrm{S}_i & = \mathrm{S}_{i-1} - \ell_j \\ \mathrm{PF}_i & = \max(0, \mathrm{PF}_{i-1} - \ell_j) \end{cases}
\end{aligned}
$$

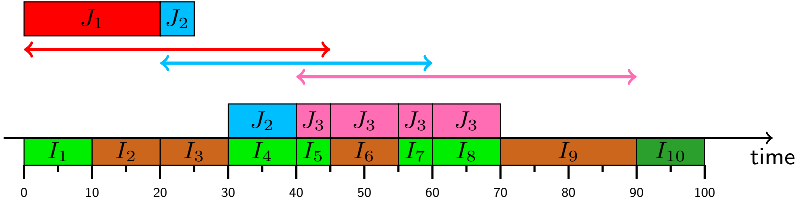
Second round:

$$
\begin{aligned}
rr_j &\rightarrow \begin{cases} missing_j & = \max(0, \ell_j - \mathrm{PB}_{i-1}) \\ \mathrm{PB}_i & = \max(0, \mathrm{PB}_{i-1} - \ell_j) \end{cases} \\
green\ I_i &\rightarrow \mathrm{PB}_i = \mathrm{PB}_{i-1} + |I_i|
\end{aligned}
$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



First round:

$$
\begin{array}{rl}
rr_j & \rightarrow \mathrm{S}_i = \mathrm{S}_{i-1} + \ell_j \\
green\ I_i & \rightarrow \mathrm{PF}_i = \mathrm{PF}_{i-1} + \min(|I_i|, \mathrm{S}_{i-1} - \mathrm{PF}_{i-1}) \\
rd_j & \rightarrow \begin{cases} \mathrm{S}_i & = \mathrm{S}_{i-1} - \ell_j \\ \mathrm{PF}_i & = \max(0, \mathrm{PF}_{i-1} - \ell_j) \end{cases}
\end{array}
$$

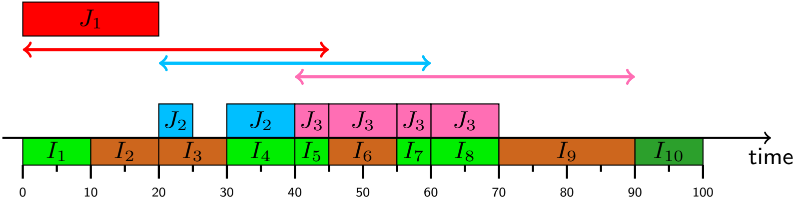
Second round:

$$
\begin{array}{rl}
rr_j & \rightarrow \begin{cases} missing_j & = \max(0, \ell_j - \mathrm{PB}_{i-1}) \\ \mathrm{PB}_i & = \max(0, \mathrm{PB}_{i-1} - \ell_j) \end{cases} \\
green\ I_i & \rightarrow \mathrm{PB}_i = \mathrm{PB}_{i-1} + |I_i|
\end{array}
$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



First round:

$$\begin{array}{rl} rr_j & \rightarrow \mathrm{S}_i = \mathrm{S}_{i-1} + \ell_j \\ green\ I_i & \rightarrow \mathrm{PF}_i = \mathrm{PF}_{i-1} + \min(|I_i|, \mathrm{S}_{i-1} - \mathrm{PF}_{i-1}) \\ rd_j & \rightarrow \begin{cases} \mathrm{S}_i & = \mathrm{S}_{i-1} - \ell_j \\ \mathrm{PF}_i & = \max(0, \mathrm{PF}_{i-1} - \ell_j) \end{cases} \end{array}$$

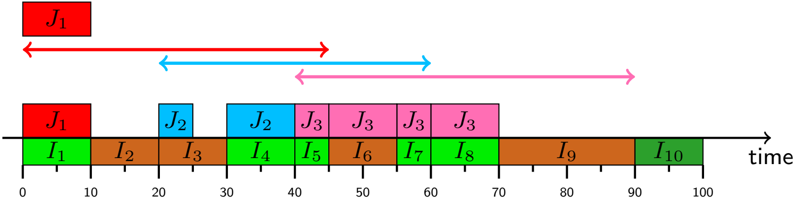
Second round:

$$\begin{array}{rl} rr_j & \rightarrow \begin{cases} missing_j & = \max(0, \ell_j - \mathrm{PB}_{i-1}) \\ \mathrm{PB}_i & = \max(0, \mathrm{PB}_{i-1} - \ell_j) \end{cases} \\ green\ I_i & \rightarrow \mathrm{PB}_i = \mathrm{PB}_{i-1} + |I_i| \end{array}$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



First round:

$$rr_j \rightarrow S_i = S_{i-1} + \ell_j$$
$$\textit{green } I_i \rightarrow PF_i = PF_{i-1} + \min(|I_i|, S_{i-1} - PF_{i-1})$$
$$rd_j \rightarrow \begin{cases} S_i & = S_{i-1} - \ell_j \\ PF_i & = \max(0, PF_{i-1} - \ell_j) \end{cases}$$

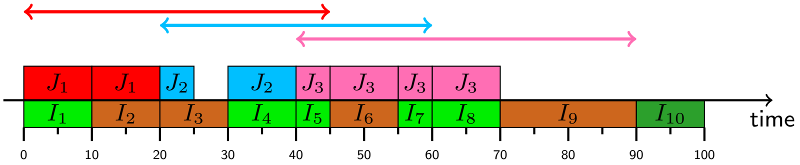
Second round:

$$rr_j \rightarrow \begin{cases} \textit{missing}_j & = \max(0, \ell_j - PB_{i-1}) \\ PB_i & = \max(0, PB_{i-1} - \ell_j) \end{cases}$$
$$\textit{green } I_i \rightarrow PB_i = PB_{i-1} + |I_i|$$

OfflineGreenest

Two-rounds algorithm:

- First round: Book *green*
- Second round: Allocate *green*, evaluate *missing* work and add *brown* if necessary



Carbon cost of the solution: $25k$; completion time 70

Better completion time! Still linear!

Contents

Greedy baseline heuristics

- ALLCLOUD: Sends and executes all jobs on the CLOUD server
- LOCAL: Schedules each job at the earliest on its edge
- ECT: Schedules each job at the earliest on any edge
- LOCALGREEN: Schedules each job at the earliest on its edge but only using *green* energy
- ECTGREEN: Schedules each job at the earliest on any edge but only using *green* energy

Algorithms built on OfflineGreenest

Three mapping strategies:

- LowCarb: Assign a job on the server that minimizes total carbon cost
- NoCarbComm: Assign a job on the server that minimizes total carbon cost, *while ignoring* transfer costs
- InPlace: Assign a job on its *origin* server; if not feasible use strategy LowCarb

Once a job is mapped on a server, schedule it using OfflineGreenest

Direct utilization defines three heuristics:

- GreedyLowCarb
- GreedyNoCarbComm
- GreedyInPlace

Algorithms built on OfflineGreenest with re-evaluation

At each job release time, mapping decisions for not yet started jobs, and scheduling decisions of started-but-not-completed jobs are re-considered

Two job priorities:

- Looseness: non-decreasing order of remaining time before deadline: $\frac{d_j - t}{\ell_j}$
- EDF: Earliest Deadline First

Choice of mapping strategy and job priority defines six heuristics

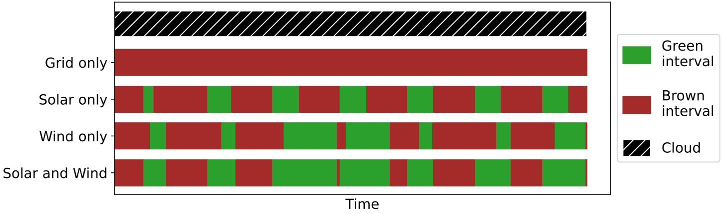
- ReallocInPlaceLooseness
- ReallocLowCarbLooseness
- ReallocNoCarbCommLooseness
- ReallocInPlaceEDF
- ReallocLowCarbEDF
- ReallocNoCarbCommEDF

Contents

Traces from real data

CAISO data

Green and brown intervals over 1 week, across the 4 on-site generation models



- Simulation length: $T = 30$ days
- 12 months
- 10 edge servers, with various on-site generation models:
  - Solar only: 4%-43% *green* intervals
  - Wind only: 27%-55% *green* intervals
  - Solar and wind: 45%-61% *green* intervals
  - Mix: 1 grid, 3 solar, 3 wind, 3 solar and wind
- $k = K/s_{cloud} = 180$ unit of carbon/s

Synthetics simulation parameters

- Job duration: between 20 seconds and 4 hours, with mean 1 hour
- Job data volume: uniformly distributed in $[2, 200]$ Gbit
- Load $\in \{0.1, 0.2, \ldots, 1\}$
- Looseness $= \frac{d_j - r_j}{\ell_j}$: $\{2, 4, 6\} \pm 10\%$
- 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
- b_{trans}: $\{10, 100, 500, 1000\}$ Mbit/s ; 250 Mbit/s for Cloud
- k_{trans}: $\{1, 10, 100, 1000\}$ unit of carbon/Mbit; 1000 unit of carbon/Mbit for Cloud

Run 20,000 experiments by randomly selecting a value for each parameter

Comparison to oracle

Oracle: for each instance knows which heuristic is best

- $RatioOracle = \frac{\text{Algorithm}}{\textsc{Oracle}}$

Algorithms	*Mean*	*SD*	*Best*	*10%*
AllCloud	18.058	3.608	0	0
LocalGreen	8.824	3.143	0	0
Local	4.940	3.247	1	3
ECTGreen	3.748	2.416	1	3
GreedyNoCarbComm	2.642	2.132	0	5
ReallocNoCarbCommLooseness	2.321	1.969	0	6
GreedyLowCarb	2.007	1.815	1	10
GreedyInPlace	1.883	1.683	1	8
ECT	1.816	1.644	0	3
ReallocLowCarbLooseness	1.617	1.493	1	13
ReallocNoCarbCommEDF	1.590	1.935	18	37
ReallocInPlaceLooseness	1.587	1.433	0	9
ReallocInPlaceEDF	1.118	1.256	48	70
ReallocLowCarbEDF	1.060	1.091	32	79

Table 1: Statistics on 20,000 random instances. Sorted by mean values.

- Very good performance for the ReallocLowCarbEDF algorithm

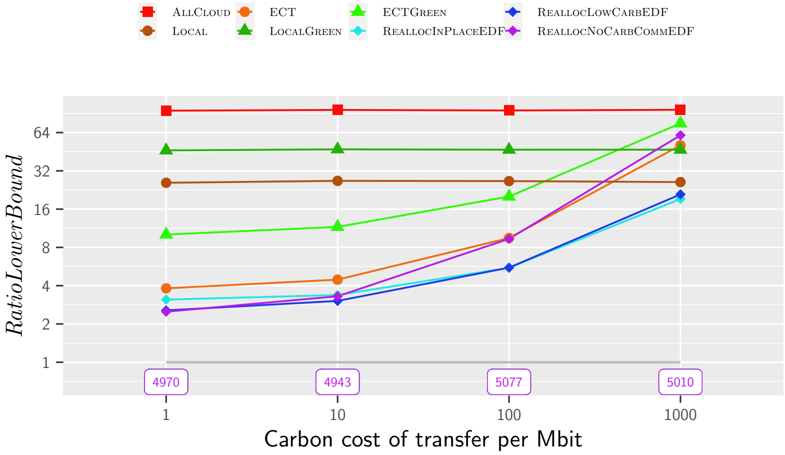
LowerBound

Simplifying assumptions:

- Communications are free (in time and carbon cost)
- Preemption and migration are allowed

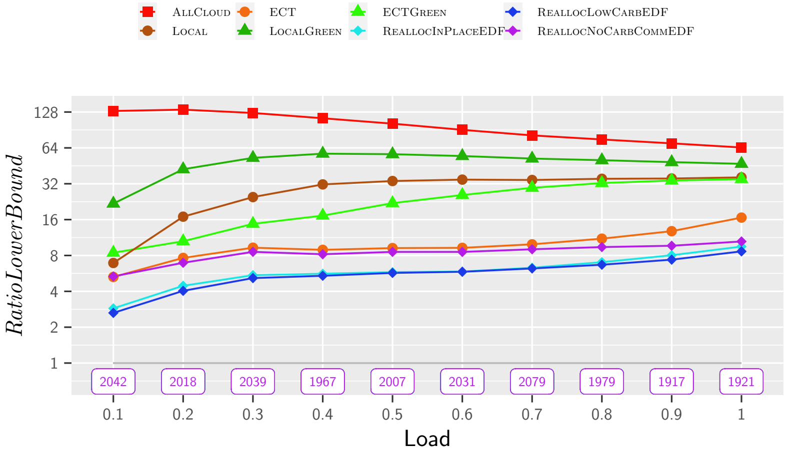
Comparison with $RatioLowerBound = \frac{\text{Algorithm}}{\text{LowerBound}}$

Impact of the carbon cost of transfer



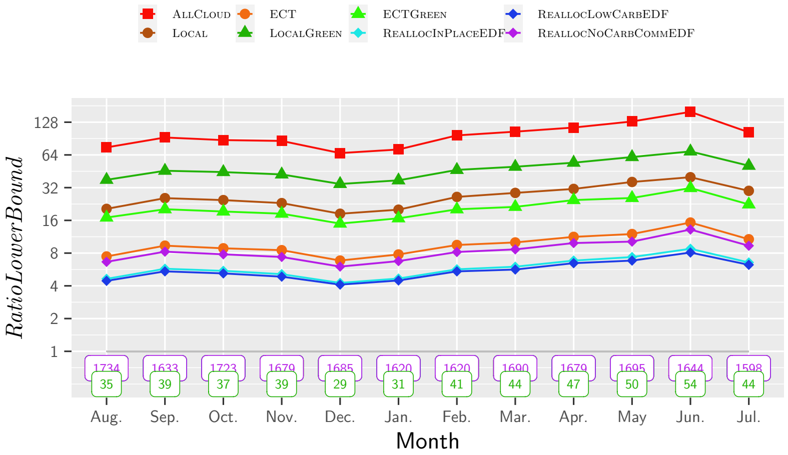
- Global algorithms get worse when carbon cost of transfer is high

Impact of the load



- Consistent performance from the ReallocLowCarbEDF algorithm

Impact of the month



- ReallocLowCarbEDF gets a saving of 79% compare to Local and a saving of 42% compare to ECT

Contents

Conclusion and future work

Conclusion:

- Modelisation of a complex edge scheduling problem
- Optimal linear algorithm to schedule ordered jobs, with good online properties
- A heuristic delivering robust performance and close to the lower bound

Future work:

- More than two carbon costs for energy
- Imperfect energy predictions