NANYANG TECHNOLOGICAL UNIVERSITY

Generalized Skyline Interval Coloring and Dynamic Geometric Bin Packing Problems

Runtian Ren and Xueyan Tang

School of Computer Science and Engineering
Nanyang Technological University
Singapore

ICPP 2021

Background: Cloud Computing

- During the past decade, renting cloud servers for processing computational jobs has become a popular way for many companies to run the business.
- The most famous cloud service providers include Amazon Elastic Compute Cloud (EC2), Google Cloud and Microsoft Azure.
- The most salient feature of cloud computing is the “pay-as-you-go” billing mechanism.

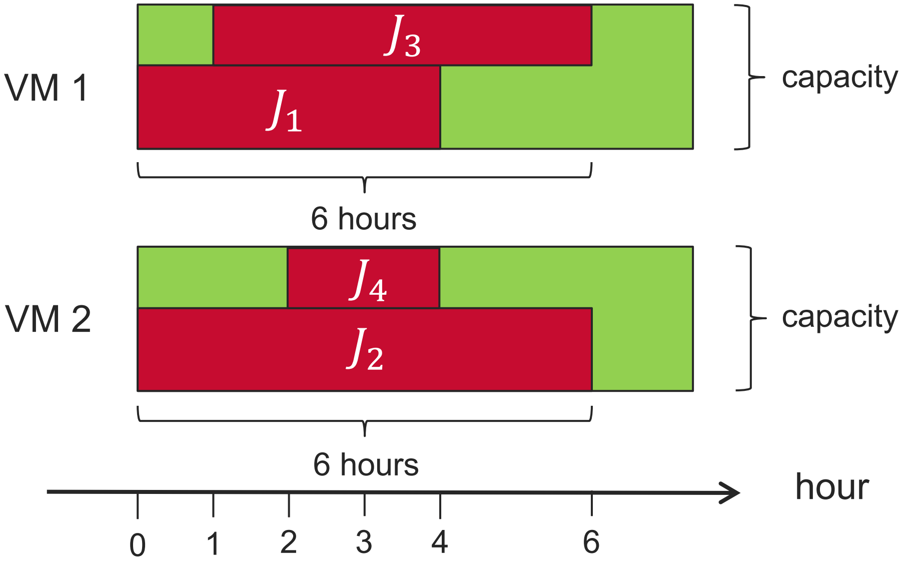
amazon web services™ EC2

Google Cloud Platform

Microsoft Azure

Background: Cloud Computing

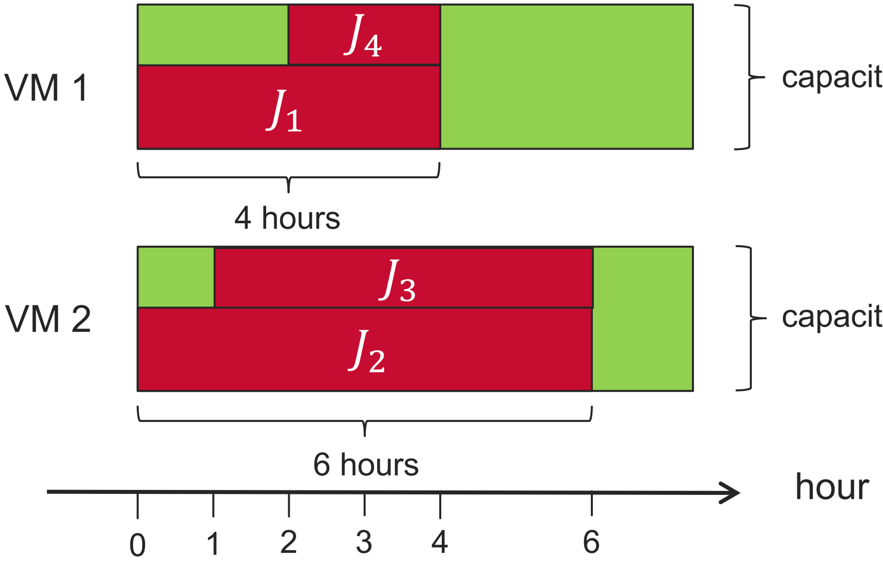
- Due to the “pay-as-you-go” billing mechanism, a natural issue faced by many cloud users is to dispatch jobs onto the cloud servers (VMs) for minimizing the renting cost of the servers.



the total cost = 12 hours × renting cost rate

Background: Cloud Computing

- Due to the “pay-as-you-go” billing mechanism, a natural issue faced by many cloud users is to dispatch jobs onto the cloud servers (VMs) for minimizing the renting cost of the servers.



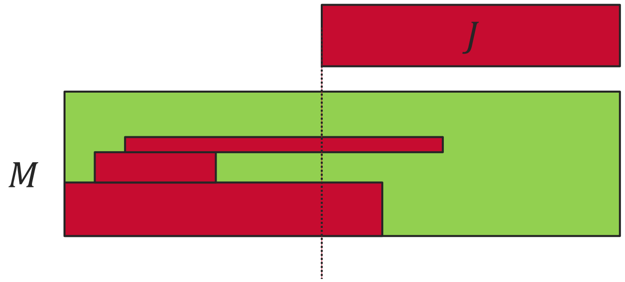
the total cost = 10 hours × renting cost rate

Previous Work: MinUsageTime DBP

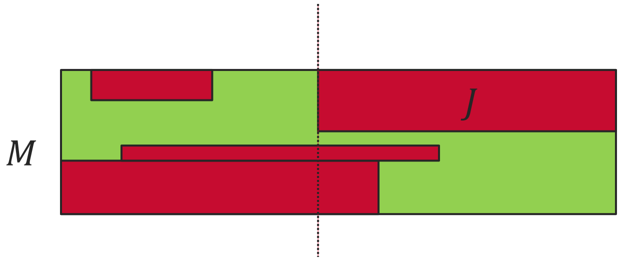
- The MinUsageTime Dynamic Bin Packing (DBP) problem [SPAA ’14, ’16, ’17] is used to model such cloud computing issue.
- Formally, the input is a set of interval jobs, each specified by an active interval and a size.
- The machines with the same capacity are used to run the jobs.
- Once a job arrives, it has to be immediately assigned to a machine with enough unused capacity.
- The target is to minimize the total busy time of the machine used for running all the jobs.
  - Here, a machine is called busy at time t, if at least one job is running on it at this moment.

New Restricted Assumption

- In a stricter scenario, each job must be assigned a continuous memory of its size for its execution when running on a machine.
- No two jobs can share the same memory when running concurrently on a machine at any time.
  - Formally, for each job J (with size $s(J)$), a vertical interval $[\alpha(J), \alpha(J) + s(J)]$ needs to be allocated to J.
  - If two overlapping jobs J and J’ (i.e., $I(J) \cap I(J') \neq \emptyset$) are placed onto the same machine, then their vertical intervals cannot overlap, i.e.,
$$[\alpha(J), \alpha(J) + s(J)] \cap [\alpha(J'), \alpha(J') + s(J')] \neq \emptyset$$
  - Such a stricter assumption is not required in MinUsageTime DBP: as long as a machine M has enough unused capacity for accommodating a job, this job can be assigned to M.



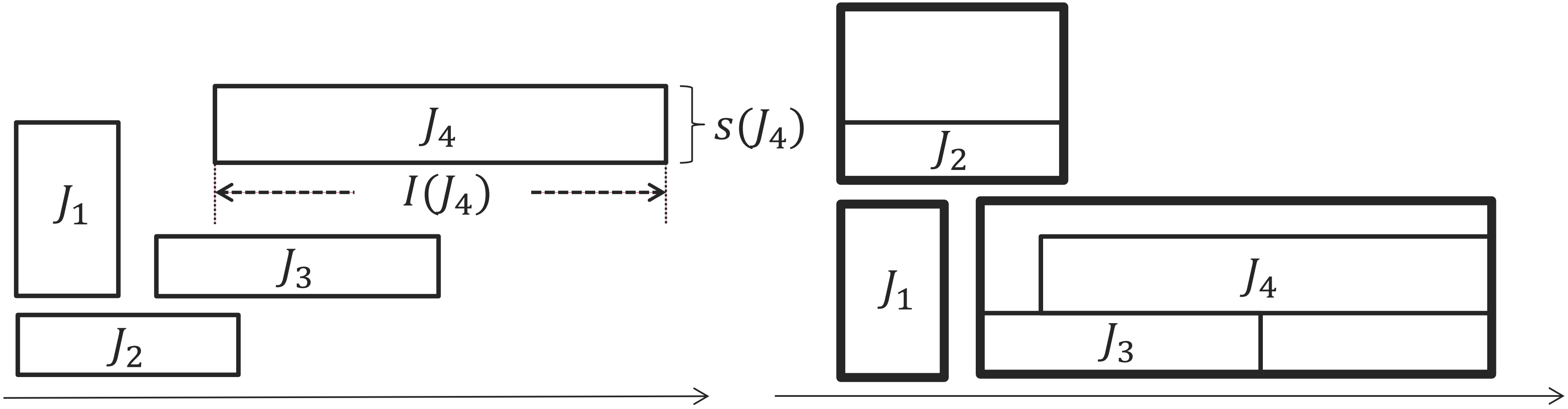
J cannot be assigned to M because there is no continuous vertical interval of J’s size in M



J can be assigned to M because there exists no continuous vertical interval of J’s size in M

DGBP & GSIC

- We name such a problem as Dynamic Geometric Bin Packing (DGBP).
  - Note that each feasible solution can be demonstrated in a geometrical manner.
- MinUsageTime DBP can thus be seen as a generalized version of DGBP.
- Fact: given a set of jobs as input, a DGBP solution must be a feasible solution to MinUsageTime DBP, but a MinUsageTime DBP solution is not necessary a feasible solution to DGBP.

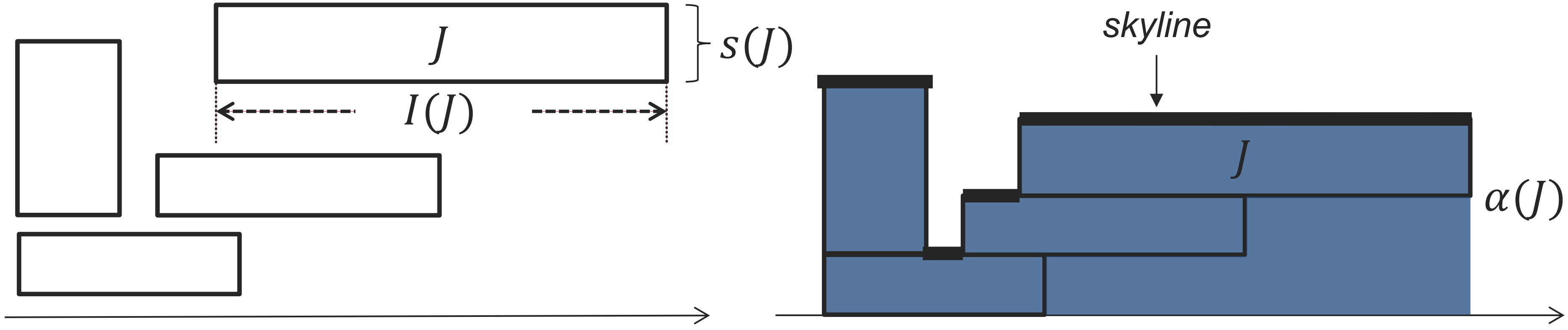


DGBP & GSIC

- We also consider another variant problem called Generalized Skyline Interval Scheduling (GSIC):
  - The input is the same as the input of MinUsageTime DBP or DGBP.
  - Each job J needs to be assigned a vertical interval $[\alpha(J), \alpha(J) + s(J)]$.
  - For any two overlapping jobs J and J’, their vertical intervals cannot overlap.
  - At any time t, the cost of the allocation α is defined to be the height of the highest point allocated to the jobs active at this moment, i.e.,

$$\alpha(\boldsymbol{J}, t) = \max_{J \in \boldsymbol{J} : t \in I(J)} (\alpha(J) + s(J))$$

  - The target is to find an allocation such that the accumulated cost produced, i.e., $\int \alpha(\boldsymbol{J}, t)\, dt$, is minimized.

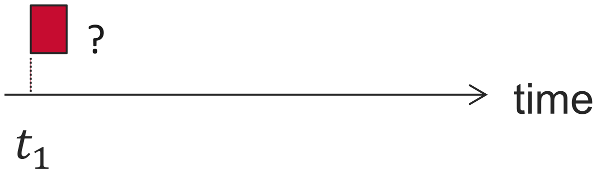


Offline & Online Settings

- We consider GSIC and DGBP in both offline and online settings.
- In the offline setting, all the input jobs are known before the scheduling process:
  - Since both problems are NP-hard, we propose approximation algorithms for GSIC and DGBP.
- In the online setting, there are two different flavours: non-clairvoyant online setting and clairvoyant online setting.

Non-Clairvoyant Online Setting

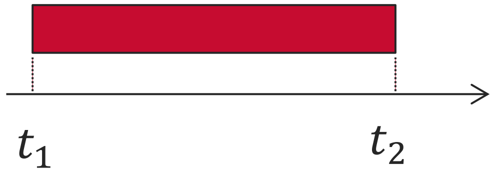
- In the non-clairvoyant online setting, the jobs are released in the order of their arrival times.
- Once a job arrives, its size is known and it needs to be scheduled immediately with no information of the future jobs.
- Furthermore, the departure time of a job is only revealed when this job finishes its execution.



at time t_1, we don't know when the job departs until its departure time.

Clairvoyant Online Setting

- In the clairvoyant online setting, the jobs are released in an arbitrary order one by one.
- Once a job is released, its active interval and its size are all known.
- Each job needs to be scheduled immediately when it is released with no information of the jobs that will be released in the future.



When this job is released, we know that it has its left-endpoint and right-endpoint being t_1 and t_2, i.e., it has a length of $t_2 - t_1$.

Previous Work

- For MinUsageTime DBP:
  - In the offline setting, a Dual Coloring algorithm achieves an approximation ratio of 4.
  - In the non-clairvoyant online setting, the First Fit algorithm [SPAA ’14] achieves a competitive ratio of $\mu + 3$ (μ is the maximum ratio of job lengths), which is asymptotically optimal.
  - In the clairvoyant online setting, the Hybrid algorithm [SPAA ’17] achieves a competitive ratio of O($\sqrt{\log \mu}$), which is asymptotically optimal.
- Attention: the instances to establish the lower bound on competitiveness of MinUsageTime DBP can be also used for DGBP, which suggests:
  - In the non-clairvoyant online setting, no deterministic online algorithm achieves a competitive ratio less than μ.
  - In the clairvoyant online setting, no deterministic online algorithm achieves a competitive ratio less than $\Omega(\sqrt{\log \mu})$.

Previous Work

- Skyline Interval Scheduling (SIC) [COCOA ’17] is a special case of GSIC (where each interval job has the same size 1); for such problem:
  - In the offline setting, 2-approximation algorithms [ESA ’03, FSTTCS ’05] exist.
  - In the clairvoyant online setting, a O$(\log \mu)$-competitive online algorithm is proposed (based on classifying the jobs according to their lengths) [COCOA ’17].
  - An instance is used to establish a lower bound of $\Omega(\log \mu)$ on the competitive ratio of any deterministic online algorithm for SIC.
- Attention: this instance also proves that no deterministic online algorithm achieves a competitive ratio less than $\Omega(\log \mu)$ for GSIC.

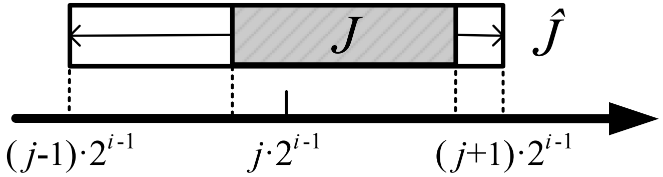
Our Contributions

- In the offline setting, we apply an aligning technique to design O(1)-approximation algorithms for GSIC and DGBP.
- Such an aligning technique is also applied in the clairvoyant online setting to design asymptotically optimal online algorithms for both problems.
- In the non-clairvoyant online setting, we apply the guess-and-double technique to design O(μ)-competitive online algorithms for GSIC and DGBP, which are asymptotically optimal.

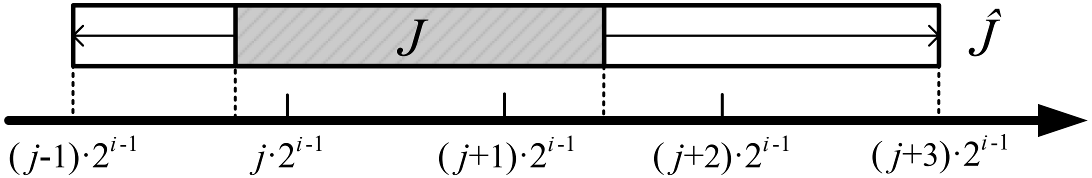
	GSIC	DGBP
Offline	O(1)	O(1)
Non-Clairvoyant Online	O(μ)	O(μ)
Clairvoyant Online	O($\log \mu$)	O($\sqrt{\log \mu}$)

An Aligning Technique

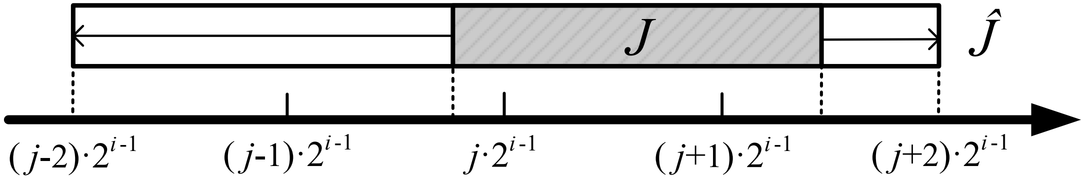
- Given any job J, suppose J satisfies $\mathrm{len}(I(J)) \in (2^{i-1}, 2^{i}]$ and $I(J)^{-} \in [(j-1)\cdot 2^{i-1}, j \cdot 2^{i-1})$.
- Then, the aligned job J' for this job J is determined in the three cases below depending on $I(J)^{+}$.


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$

Case 1: $I(J)^{+} \leq (j+1)\cdot 2^{i-1}$


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$
$(j\text{+}3)\cdot 2^{i\text{-}1}$

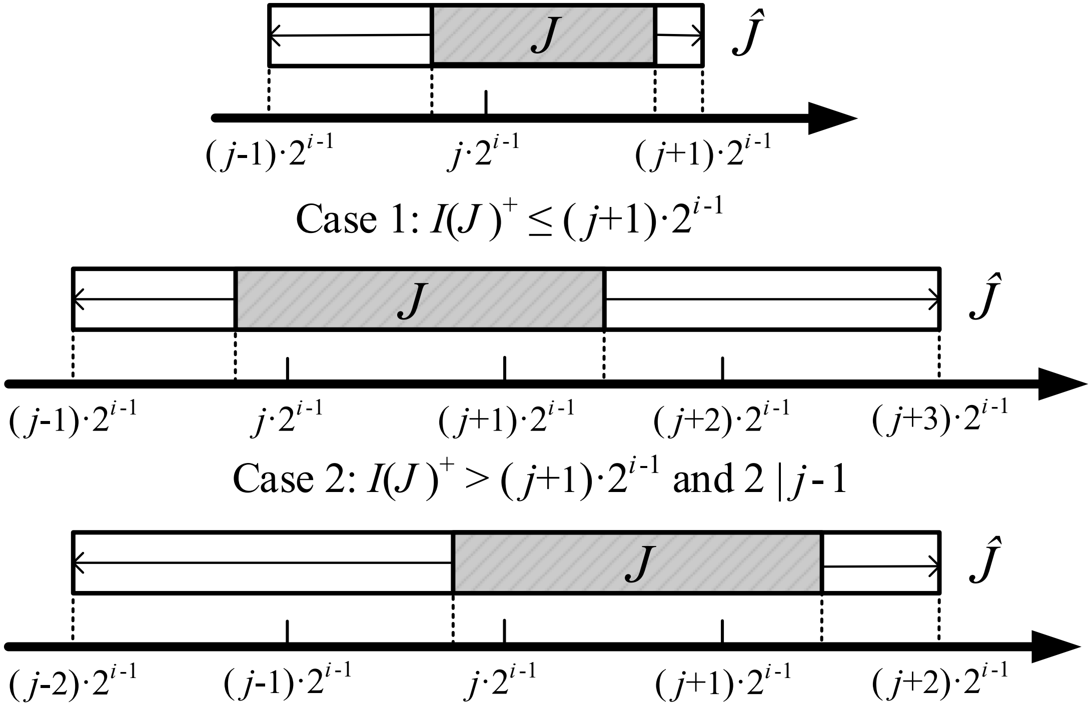
Case 2: $I(J)^{+} > (j+1)\cdot 2^{i-1}$ and $2 \mid j-1$


J
$\hat{J}$
$(j\text{-}2)\cdot 2^{i\text{-}1}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$

Case 3: $I(J)^{+} > (j+1)\cdot 2^{i-1}$ and $2 \mid j$

An Aligning Technique

- The aligned instance $\boldsymbol{J}'$ can be constructed by such a method given any instance $\boldsymbol{J}$, such that
  - Any feasible solution for the aligned instance is also a feasible solution for the original instance.
  - $\sum_{J' \in \boldsymbol{J}'} s(J') \cdot \mathrm{len}(I(J')) \le 4 \cdot \sum_{J \in \boldsymbol{J}} s(J) \cdot \mathrm{len}\big(I(J)\big)$.



J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$
$(j\text{+}3)\cdot 2^{i\text{-}1}$
J
$\hat{J}$
$(j\text{-}2)\cdot 2^{i\text{-}1}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$

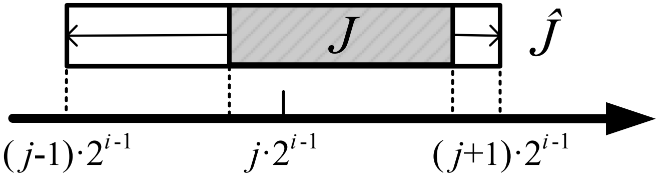
Case 1: $I(J)^+ \le (j\text{+}1)\cdot 2^{i\text{-}1}$

Case 2: $I(J)^+ > (j\text{+}1)\cdot 2^{i\text{-}1}$ and $2 \mid j - 1$

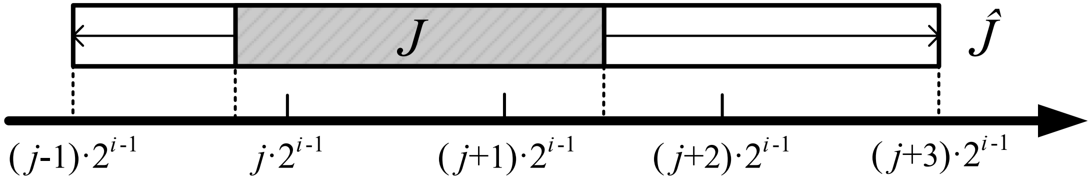
Case 3: $I(J)^+ > (j\text{+}1)\cdot 2^{i\text{-}1}$ and $2 \mid j$

An Aligning Technique

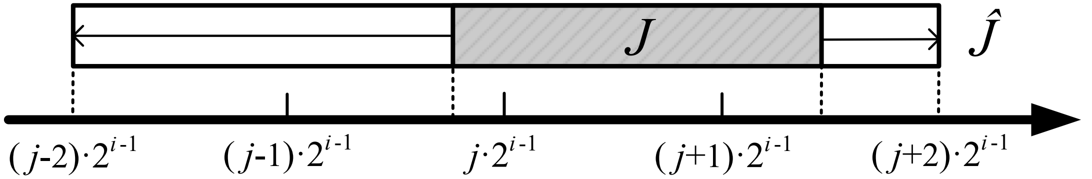
- Besides, given any aligned job J' with $\mathrm{len}(I(J')) = 2^i$, $I(J')^-$ must be a multiple of 2^{i-1}. Under such a case:
  - All the aligned jobs can be classified into two categories.
  - if $I(J')^-$ is a multiple of 2^i, then J' is called a *regular* job;
  - otherwise, J' is called an *irregular* job.


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$

Case 1: $I(J)^+ \leq (j\text{+}1)\cdot 2^{i\text{-}1}$


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$
$(j\text{+}3)\cdot 2^{i\text{-}1}$

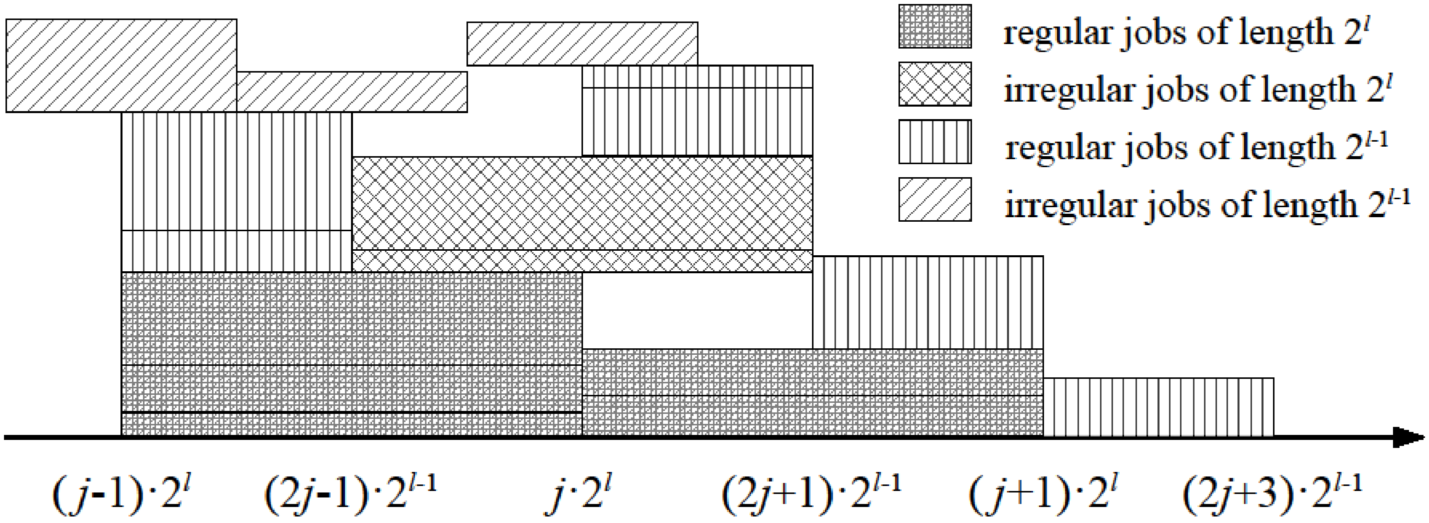
Case 2: $I(J)^+ > (j\text{+}1)\cdot 2^{i\text{-}1}$ and $2 \mid j\text{-}1$


J
$\hat{J}$
$(j\text{-}2)\cdot 2^{i\text{-}1}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$

Case 3: $I(J)^+ > (j\text{+}1)\cdot 2^{i\text{-}1}$ and $2 \mid j$

Offline GSIC

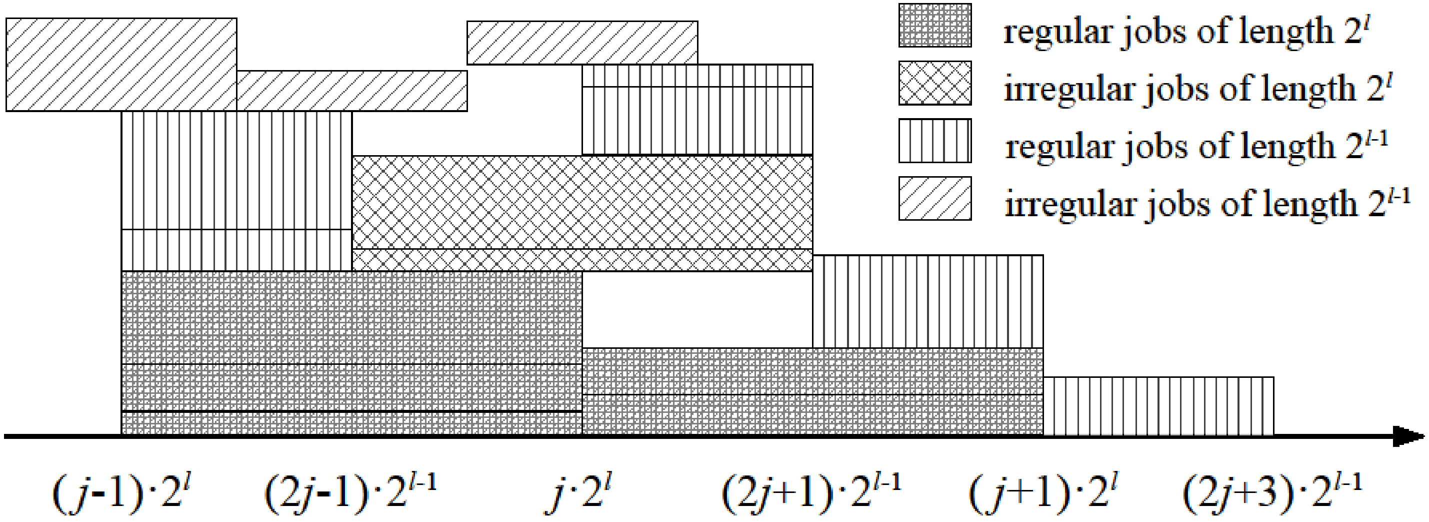
- Given the aligned instance J', an O(1)-approximation algorithm can be designed for Offline GSIC:
  - Sort the jobs J' in the decreasing order of their lengths, without loss of generality, suppose the aligned jobs have lengths of 1, 2, …, 2^l.
  - In each iteration i, the jobs of length 2^{l-i+1} are "piled up": the regular jobs are "piled up" first, then the irregular jobs are "piled up".



regular jobs of length 2^l
irregular jobs of length 2^l
regular jobs of length 2^{l-1}
irregular jobs of length 2^{l-1}
$(j-1)\cdot 2^l$
$(2j-1)\cdot 2^{l-1}$
$j\cdot 2^l$
$(2j+1)\cdot 2^{l-1}$
$(j+1)\cdot 2^l$
$(2j+3)\cdot 2^{l-1}$

Offline GSIC

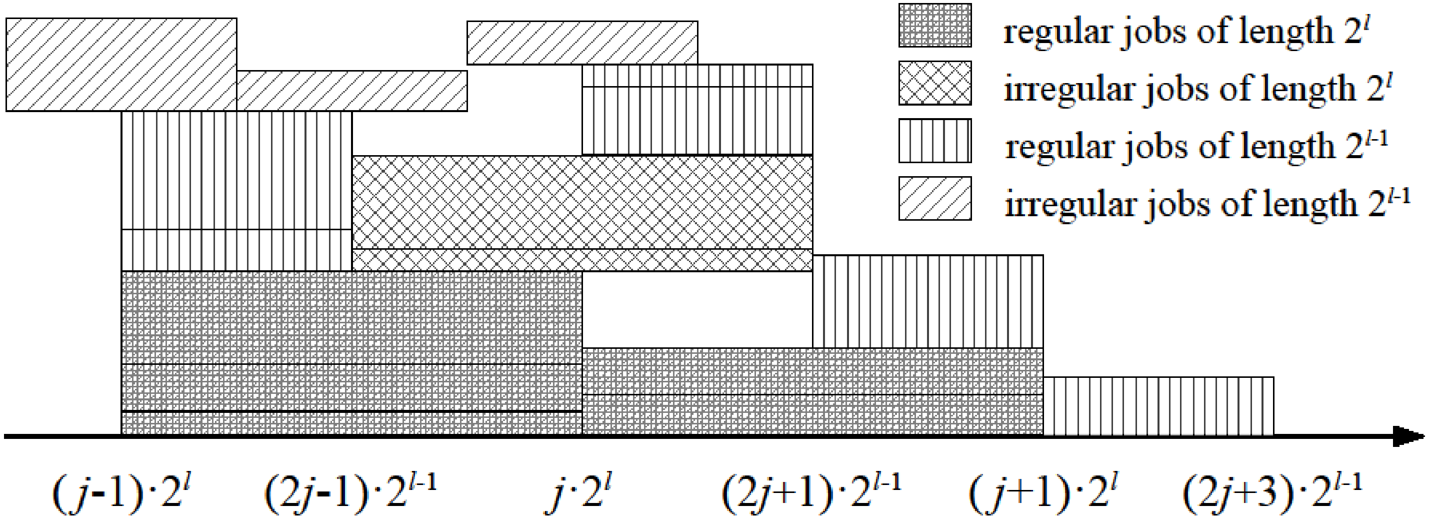
- Given the aligned instance $\boldsymbol{J}'$, an O(1)-approximation algorithm can be designed for Offline GSIC:
  - Sort the jobs $\boldsymbol{J}'$ in the decreasing order of their lengths, without loss of generality, suppose the aligned jobs have lengths of 1, 2, …, 2^l.
  - In each iteration i, the jobs of length 2^{l-i+1} are “piled up”: the regular jobs are “piled up” first, then the irregular jobs are “piled up”.


regular jobs of length 2^l
irregular jobs of length 2^l
regular jobs of length 2^{l-1}
irregular jobs of length 2^{l-1}
$(j-1)\cdot 2^l$
$(2j-1)\cdot 2^{l-1}$
$j\cdot 2^l$
$(2j+1)\cdot 2^{l-1}$
$(j+1)\cdot 2^l$
$(2j+3)\cdot 2^{l-1}$

This “pile-up” method only causes waste of area, which can be bounded by O(1) times the area of all the jobs!

Offline GSIC

- Given the aligned instance $\boldsymbol{J}'$, an O(1)-approximation algorithm can be designed for Offline GSIC:
  - Sort the jobs $\boldsymbol{J}'$ in the decreasing order of their lengths, without loss of generality, suppose the aligned jobs have lengths of 1, 2, …, 2^l.
  - In each iteration i, the jobs of length 2^{l-i+1} are "piled up": the regular jobs are "piled up" first, then the irregular jobs are "piled up".

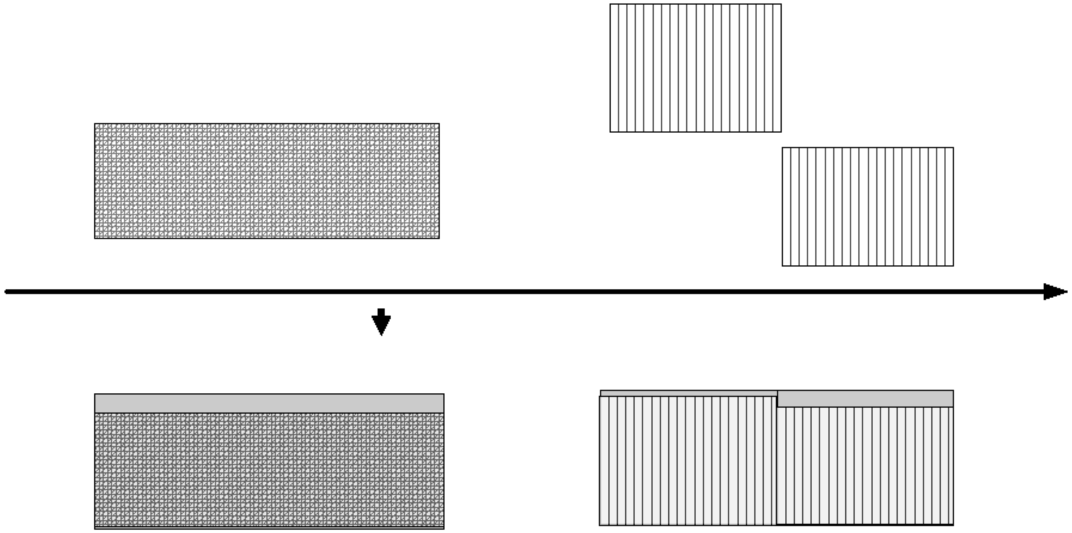


regular jobs of length 2^l
irregular jobs of length 2^l
regular jobs of length 2^{l-1}
irregular jobs of length 2^{l-1}
$(j-1)\cdot 2^l$
$(2j-1)\cdot 2^{l-1}$
$j\cdot 2^l$
$(2j+1)\cdot 2^{l-1}$
$(j+1)\cdot 2^l$
$(2j+3)\cdot 2^{l-1}$

Thus, such an offline algorithm achieves an approximation ratio of O(1).

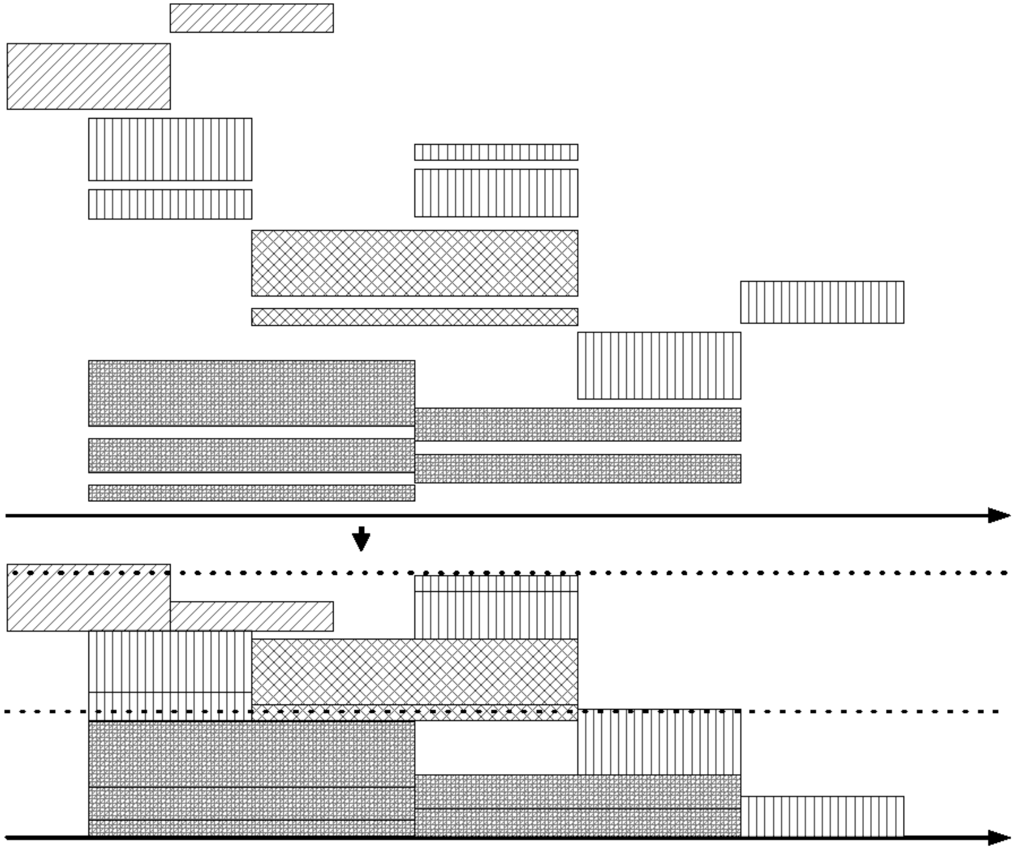
Offline DGBP

- Based on the approximation algorithm for Offline GSIC, an O(1)-approximation algorithm can also be designed for Offline DGBP:
    - Each large job (with size at least 0.5) is directly placed onto a separate machine.



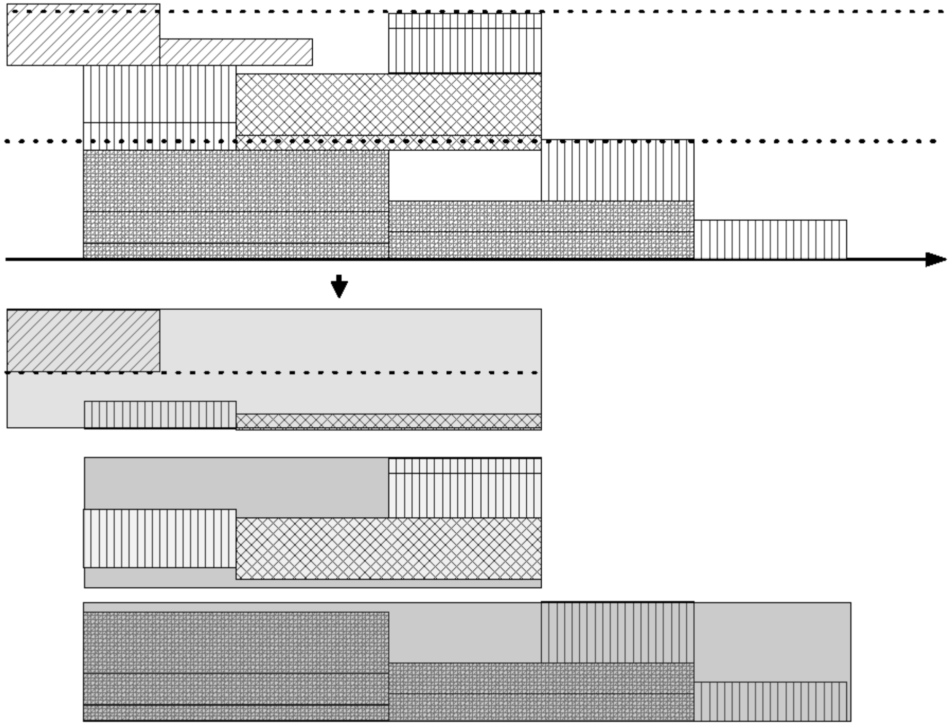
Offline DGBP

- Based on the approximation algorithm for Offline GSIC, an O(1)-approximation algorithm can also be designed for Offline DGBP:
  - All the small jobs (with size no more than 0.5) are selected.
  - An aligned instance is constructed for these small jobs and a GSIC solution is produced for this aligned instance based on the previous algorithm.



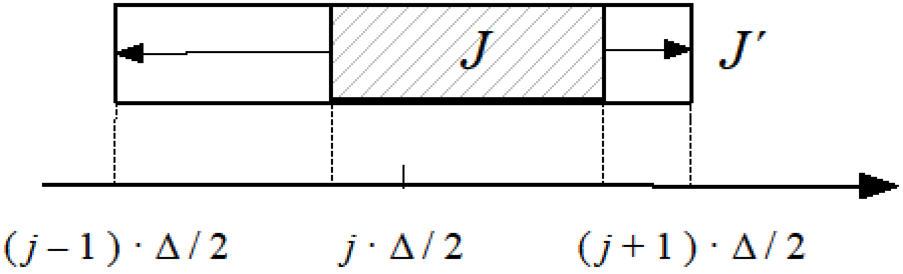
Offline DGBP

- Based on the approximation algorithm for Offline GSIC, an O(1)-approximation algorithm can also be designed for Offline DGBP:
  - Then, a “slicing” technique is applied on the GSIC solution to produce a feasible DGBP solution:
    - the GSIC solution is sliced into strips of height 1 (the machine capacity);
    - for jobs fully inside one strip, one machine is used to schedule these jobs;
    - for jobs crossing two strips, half a machine is used to schedule these jobs.



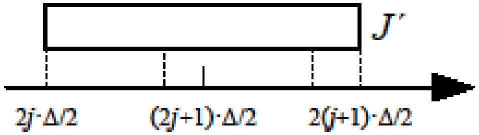
Non-Clairvoyant Online Setting

- In the non-clairvoyant online setting, we focus on introducing how to deal with DGBP (GSIC can be dealt with in a similar manner).
- One special case of DGBP (when maximum job length Δ is known a priori) is easy:
  - Given a job J with its arrival time $I(J)^- \in [(j-1)\cdot\Delta/2, j\cdot\Delta/2)$, an aligned job J' with an active interval $[(j-1)\cdot\Delta/2, (j+1)\cdot\Delta/2)$ is constructed under the non-clairvoyant online setting.

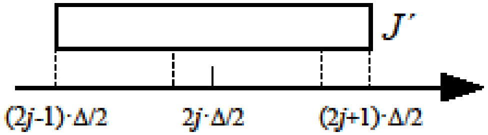


Non-Clairvoyant Online Setting

- In the non-clairvoyant online setting, we focus on introducing how to deal with DGBP (GSIC can be dealt with in a similar manner).
- One special case of DGBP (when maximum job length Δ is known a priori) is easy:
  - In this way, the aligned instance can be classified into two categories again:
    - if $I(J')^{-}$ is $j \cdot \Delta/2$ (j is even), then J' is called a *regular* job;
    - if $I(J')^{-}$ is $j \cdot \Delta/2$ (j is odd), then J' is called an *irregular* job.



J′ is regular



J′ is irregular

Non-Clairvoyant Online Setting

- In the non-clairvoyant online setting, we focus on introducing how to deal with DGBP (GSIC can be dealt with in a similar manner).
- One special case of DGBP (when maximum job length Δ is known a priori) is easy:
  - For each job category, the First-Fit rule is applied to schedule jobs onto machines (which guarantees a feasible allocation for each job).
  - Such an algorithm is O(μ)-competitive.

Non-Clairvoyant Online Setting

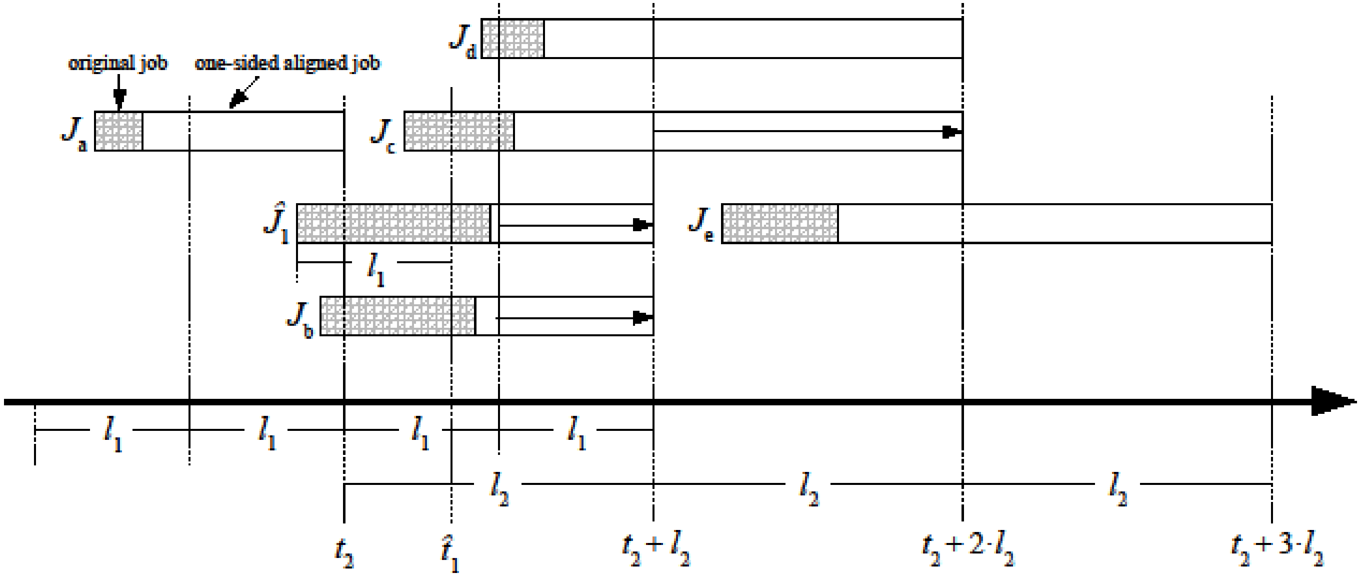
- In the general case, we can apply a “guess-and-double” technique to construct the aligned instance:
  - At the beginning (in the first phase), the maximum job length is assumed to be the length of the first job l_1 (note that this is only revealed when the first job finishes its execution).
  - The aligned instance can be “temporarily” constructed (using l_1 as Δ) and scheduled by the same method introduced before.

Non-Clairvoyant Online Setting

- In the general case, we can apply a “guess-and-double” technique to construct the aligned instance:
  - At the beginning (in the first phase), the maximum job length is assumed to be the length of the first job l_1 (note that this is only revealed when the first job finishes its execution).
  - The aligned instance can be “temporarily” constructed (using l_1 as Δ) and scheduled by the same method introduced before.
  - Note that if no job has a length exceeding l_1, then the above method is already enough to produce a feasible solution for all the jobs.

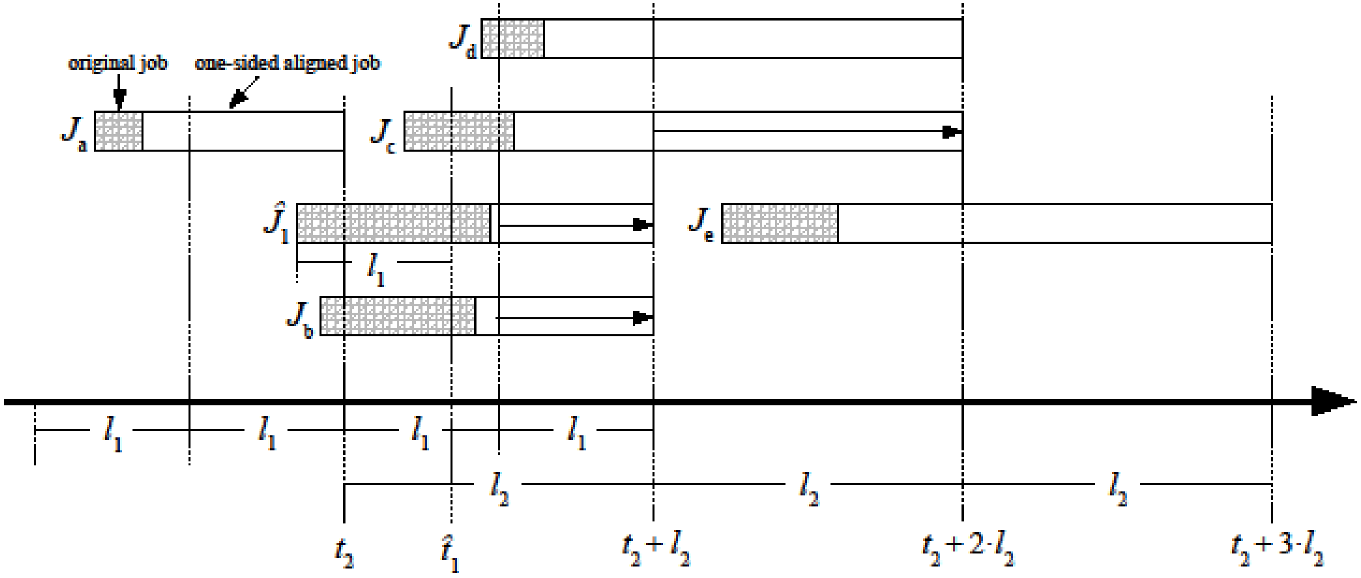
Non-Clairvoyant Online Setting

- In the general case, we can apply a “guess-and-double” technique to construct the aligned instance:
  - During the scheduling process (as time passes), if the algorithm discovers that one job has its length exceeding l_1, then
    - the current phase terminates and the next phase starts;
    - the maximum job length is doubled (i.e., use $l_2 = 2l_1$ as Δ);
    - as a result, each unfinished job’s active interval is updated, such that the First-Fit rule is still used to deal with incoming jobs.



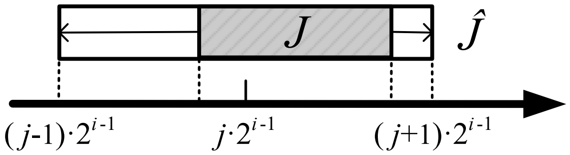
Non-Clairvoyant Online Setting

- In the general case, we can apply a "guess-and-double" technique to construct the aligned instance:
  - In general, once the algorithm discovers a job's length exceeding the current "guessed" $\Delta = l_i$, then
    - phase i terminates immediately and phase $i+1$ starts;
    - the "guessed" maximum job length Δ is updated to $l_{i+1} = 2l_i$;
    - each currently running job's active interval is also updated;
    - in this way, the incoming jobs can be scheduled similar to previous phases without compromising the feasibility of the online solution.



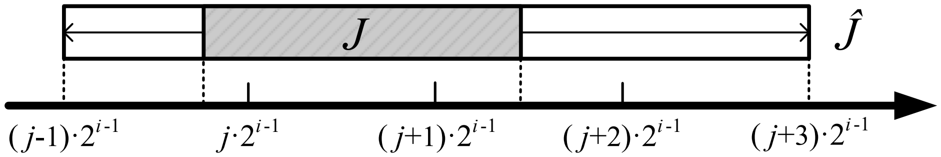
Clairvoyant Online Setting

- In the clairvoyant online setting, the offline aligning technique can also be applied.
  - For DGBP, the Hybrid algorithm introduced by Azar and Vainstein [SPAA ’17] can be used to deal with the aligned instance in an online manner.
  - Such an online strategy achieves a competitive ratio of O($\sqrt{\log \mu}$), which is asymptotically optimal.



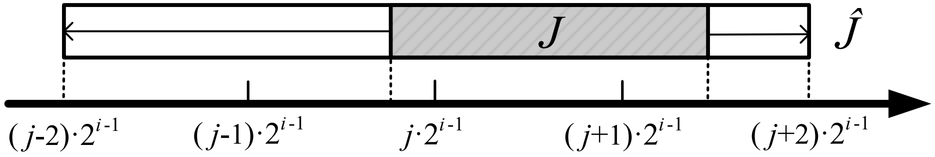
J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$

Case 1: $I(J)^{+} \leq (j+1)\cdot 2^{i-1}$



J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$
$(j\text{+}3)\cdot 2^{i\text{-}1}$

Case 2: $I(J)^{+} > (j+1)\cdot 2^{i-1}$ and $2 \mid j - 1$

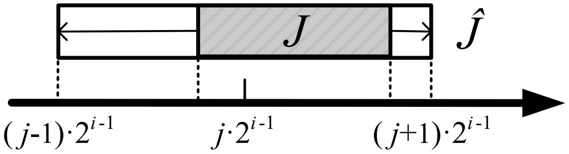


J
$\hat{J}$
$(j\text{-}2)\cdot 2^{i\text{-}1}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j\text{+}1)\cdot 2^{i\text{-}1}$
$(j\text{+}2)\cdot 2^{i\text{-}1}$

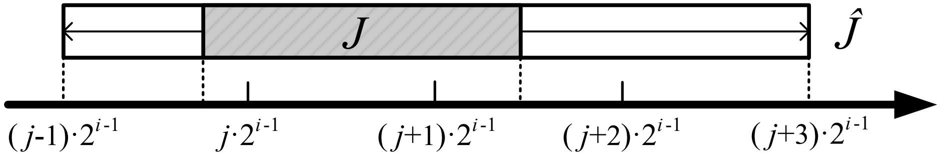
Case 3: $I(J)^{+} > (j+1)\cdot 2^{i-1}$ and $2 \mid j$

Clairvoyant Online Setting

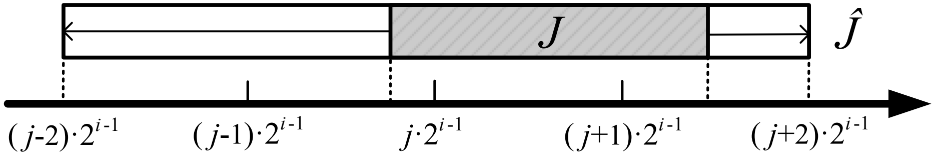
- In the clairvoyant online setting, the offline aligning technique can also be applied.
  - For GSIC, the whole vertical extent is sliced into blocks, with $O(\log \mu)$ blocks of size 2^i for each i; once an aligned job needs to be scheduled, it is placed into the particular blocks of its length by the First-Fit rule.
  - Such an online algorithm achieves a competitive ratio of $O(\log \mu)$, which is asymptotically optimal.


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j+1)\cdot 2^{i\text{-}1}$

Case 1: $I(J)^+ \leq (j+1)\cdot 2^{i\text{-}1}$


J
$\hat{J}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j+1)\cdot 2^{i\text{-}1}$
$(j+2)\cdot 2^{i\text{-}1}$
$(j+3)\cdot 2^{i\text{-}1}$

Case 2: $I(J)^+ > (j+1)\cdot 2^{i\text{-}1}$ and $2 \mid j - 1$


J
$\hat{J}$
$(j\text{-}2)\cdot 2^{i\text{-}1}$
$(j\text{-}1)\cdot 2^{i\text{-}1}$
$j\cdot 2^{i\text{-}1}$
$(j+1)\cdot 2^{i\text{-}1}$
$(j+2)\cdot 2^{i\text{-}1}$

Case 3: $I(J)^+ > (j+1)\cdot 2^{i\text{-}1}$ and $2 \mid j$

Open Problem

- Throughput Maximization with Bounded Busy Time:
  - The input is a set of interval jobs $\boldsymbol{J}$ and a budget of machine busy time T, each job J having an active interval $I(J)$, a size $s(J)$ and a weight $w(J)$.
  - A subset of jobs $\boldsymbol{J}'$ is to be selected from $\boldsymbol{J}$ and scheduled onto machines of the same capacity.
  - The total busy time of the machines used cannot exceed T.
  - The target is to maximize the total weight of the selected jobs $\boldsymbol{J}'$.
- Such a maximization problem can be seen as a dual variant of MinUsageTime DBP.
- Even for the special cases when $w(J) = 1$ or $w(J) = s(J) \cdot \mathrm{len}(I(J))$ for each J, this maximization problem is not trivial at all.

Thanks