A Supplementary for the Paper Titled ”Towards Designing Cost-Optimal Policies to Utilize IaaS Clouds with Online Learning”

Xiaohu Wu, Patrick Loiseau, and Esa Hyytiä

Abstract—This supplementary is used to help readers better understand the paper titled ”Towards Designing Cost-Optimal Policies to Utilize IaaS Clouds with Online Learning”. In this supplementary, all numeric indexes in parentheses correspond to the equation, inequality or other expressions in the supplemented paper.

A Proofs of Propositions

This section contains the proofs of the propositions in the Section 4 of the supplemented paper.

Proof of Proposition 4.1. Assume that a job j is allocated r_j self-owned instances in $[a_j, a_j + d_j - 1]$. At each of the first κ_0 allocations of j, the expected time of utilizing spot instances is $\beta \cdot Len$. If a job can be expected to be completed by the deadline by totally utilizing spot instances after the allocation of self-owned instances, we have that (i) it could be expected that the workload processed by self-owned instances plus the workload processed by spot instances at every allocation of j is no less than z_j, and (ii) after the allocation of self-owned instances, the allocation of spot and on-demand instances is always in the first phase as described in the Section 3.3 of the supplemented paper, i.e., the allocation is updated every hour where only spot instances are bid for.

Now, we analyze two cases. The first one is $d_j - \kappa_0 \cdot Len > \beta \cdot Len$. In this case, in the (κ_0+1)-th execution of j, the expected time of utilizing spot instances is $\beta \cdot Len$; then, it is expected that

$$r_j \cdot d_j + (\kappa_0 + 1) \cdot (\delta_j - r_j) \cdot Len \cdot \beta \geq z_j.$$

This leads to that $r_j \geq r_j'(\beta)$. The second case is $d_j - \kappa_0 \cdot Len \leq \beta \cdot Len$. In this case, in the (κ_0+1)-th execution of j, the expected time of utilizing spot instances is $\min\{\beta \cdot Len, d_j - \kappa_0 \cdot Len\} = d_j - \kappa_0 \cdot Len$; then, it is expected that

$$\begin{aligned} r_j \cdot d_j + \kappa_0 \cdot (\delta_j - r_j) \cdot Len \cdot \beta \\ + (d_j - \kappa_0 \cdot Len) \cdot (\delta_j - r_j) \geq z_j. \end{aligned}$$

This leads to that $r_j \geq r_j''(\beta)$. As a summary of our analysis of both cases, the proposition holds. ■

Proof of Proposition 4.2. When $x \in [0, \frac{d_j}{Len} - \kappa_0)$, $g_j(x) = \max\{r_j'(x), 0\}$; since $d_j \cdot \delta_j - z_j \geq 0$ and $(\kappa_0 + 1) \cdot Len > 0$, $r_j'(x)$ is a non-increasing function and so is $g_j(x)$. Similarly, when $x \in [\frac{d_j}{Len} - \kappa_0, 1)$, $g_j(x) = \max\{r_j''(x), 0\}$ is also non-increasing. In the rest of this proof, if suffices to show $g_j(x_1) \geq g_j(x_2)$ when $0 \leq x_1 < \frac{d_j}{Len} - \kappa_0 \leq x_2 < 1$. Given a job j, if $\kappa_0 = 0$, we have $g_j(x_1) \geq 0 = g_j(x_2)$. If $\kappa_0 \geq 1$ and $d_j \cdot \delta_j = z_j$, we have $g_j(x_1) = \delta_j = g_j(x_2)$. If $\kappa_0 \geq 1$ and $d_j \cdot \delta_j > z_j$, our analysis proceeds as follows. To prove $g_j(x_1) \geq g_j(x_2)$, it suffices to show $r_j''(x_2) \leq r_j'(x_1)$; the function $r_j''(x)$ itself is non-increasing when $x \in [0, 1)$, and we have $r_j''(x_2) \leq r_j''(x_1)$. Hence, to prove $r_j''(x_2) \leq r_j'(x_1)$, it suffices to prove $r_j''(x_1) \leq r_j'(x_1)$, which can be proved by showing $A = (1-x_1) \cdot \kappa_0 \cdot Len \leq d_j - (\kappa_0+1) \cdot Len \cdot x_1 = B$. Since $x_1 \in [0, \frac{d_j}{Len} - \kappa_0)$, we have

$$B - A = d_j - (\kappa_0 + x_1) \cdot Len > 0.$$

Finally, the proposition holds. ■

Proof of Proposition 4.4. Firstly, we prove by contradiction that the optimal value of $o_j^{\kappa_1}$ is 0. Assume that $\hat{o}_j^1, \cdots, \hat{o}_j^{\kappa_1}$ are an optimal solution to (8) where $\hat{o}_j^{\kappa} \geq 1$. The constraint (6) has no effect on the value of $o_j^{\kappa_1}$. We can reduce the value of $\hat{o}_j^{\kappa_1}$ to 0; such reduction can still guarantee that (7) is satisfied, and $\hat{o}_j^1, \cdots, \hat{o}_j^{\kappa_1-1}, o_j^{\kappa_1} = 0$ are a feasible solution to (8) under which (8) achieves a higher value, which contradicts that $\hat{o}_j^1, \cdots, \hat{o}_j^{\kappa_1}$ are an optimal solution to (8). Secondly, when $o_j^{\kappa_1} = 0$, the objective function (8) equals $(\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i) + \delta_j) \cdot Len \cdot \beta$. Under constraint (6), $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i) \leq \frac{d_j \cdot \delta_j - z_j}{Len \cdot (1-\beta)}$. Since $o_j^1, \cdots, o_j^{\kappa_1-1}$ are integers, the maximum possible value of $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i)$ is $\nu(z_j, d_j)$. On the other hand, since $\delta_j - o_j^i \leq \delta_j$, the constraint (5) indicates that $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i) \leq (\kappa_0 - 1) \cdot \delta_j$. Hence, the maximum possible value of $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i)$ is $\min\{\nu(z_j, d_j), (\kappa_0 - 1) \cdot \delta_j\}$. Now, we further show it is feasible. If $(\kappa_0 - 1) \cdot \delta_j \leq \nu(z_j, d_j)$, $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i) = (\kappa_0 - 1) \cdot \delta_j$ which leads to $\kappa_0 - 1 \leq \kappa_1 - 1$; to satisfy (5), we have $\kappa_1 = \kappa_0$. Then, constraint (7) holds trivially and constraint (6) is also satisfied. If $(\kappa_0 - 1) \cdot \delta_j > \nu(z_j, d_j)$, $\sum_{i=1}^{\kappa_1-1}(\delta_j - o_j^i) = \nu(z_j, d_j)$; in this case, we have $\kappa_1 - 1 \leq \kappa_0 - 1$. Furthermore, we also have $\nu(z_j, d_j) + \delta_j > \frac{d_j \cdot \delta_j - z_j}{Len \cdot (1-\beta)}$ and (7) is satisfied. Finally, the proposition holds. ■

Proof of Proposition 4.6. We can check that when the strategy of utilizing spot instances is as above, $o_j^1, \cdots, o_j^{\kappa_1}$ are of the form in Proposition 4.4; hence, it is optimal. ■

Proof of Proposition 4.7. Let us consider an arbitrary allocation of on-demand instances to process the remaining $z_j^{i_j+1}$ workload, denoted by $\mathcal{A}$, also illustrated in Fig. 1 (left). These workload will be processed on δ_j instances, and let x_h denote the total workload processed at the h-th instance where

$$\sum_{h=1}^{\delta_j} x_h \geq z_j^{i_j+1}, \qquad \text{(A)}$$

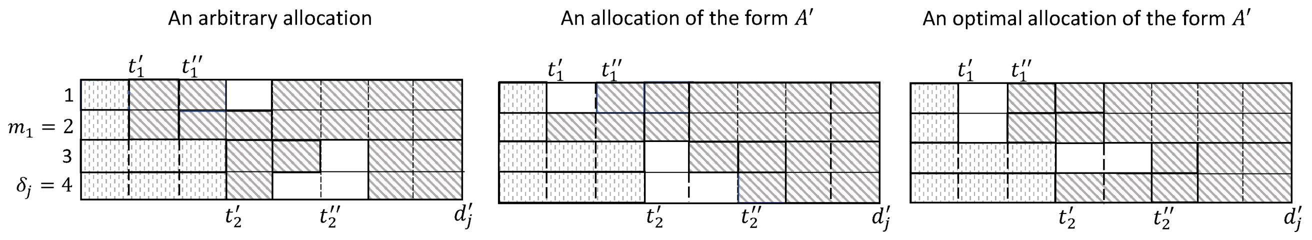

Fig. 1. Illustration for Proposition 4.7: the area of diagonal stripes denotes the allocation of on-demand instances to j.

$$\begin{aligned} &x_1, \cdots, x_{m_1} \in [0, d_j' - t_1' + 1], \\ &x_{m_1+1}, \cdots, x_{\delta_j} \in [0, d_j' - t_2' + 1]. \end{aligned} \quad \text{(B)}$$

The allocation $\mathcal{A}$ can be transformed into an allocation $\mathcal{A}'$ with the following form without increasing the total cost of utilizing instances: the x_h workload of the h-th instance is processed from the deadline d_j' towards earlier slots, i.e., in $[d_j' - x_h + 1, d_j']$, which is illustrated in Fig. 1 (middle). Hence, in the following, we only need to show the cost-optimal strategy of utilizing instances when the allocation is of the form $\mathcal{A}'$.

As illustrated in the Fig. 8 of the supplemented paper, let $\hat{\mathcal{I}}_1 = [t_1', d_j']$ and $\hat{\mathcal{I}}_2 = [t_2', d_j']$. From d_j' towards earlier slots in $\hat{\mathcal{I}}_1$ (resp. in $\hat{\mathcal{I}}_2$), let every Len slots constitute a time interval, i.e., $\mathcal{I}_i = [d_j' + 1 - i \cdot Len, d_j' - (i-1) \cdot Len]$; for $\hat{\mathcal{I}}_1$ the last interval is $\mathcal{I}_{\hat{\kappa}_1+1} = [t_1', t_1'' - 1]$ (resp. for $\hat{\mathcal{I}}_2$ the last is $\mathcal{I}_{\hat{\kappa}_2+1} = [t_2', t_2'' - 1]$). Now, we describe the cost structure when the allocation of j is of the form $\mathcal{A}'$. We use $x_{h,i}$ to denote the workload processed by the h-th instance in $\mathcal{I}_i$ where for all $h \in [1, m_1]$,

$$x_{h,1}, \cdots, x_{h,\hat{\kappa}_1} \in [0, Len], \;\; x_{h,\hat{\kappa}_1+1} \in [0, \phi_1], \quad \text{(C)}$$

and for all $h \in [m_1 + 1, \delta_j]$,

$$x_{h,1}, \cdots, x_{h,\hat{\kappa}_2} \in [0, Len], \;\; x_{h,\hat{\kappa}_2+1} \in [0, \phi_2]. \quad \text{(D)}$$

Let $\psi_h = \lceil \frac{x_h}{Len} \rceil$; under the allocation form of $\mathcal{A}'$, we have for all $h \in [1, \delta_j]$ that

$$\begin{aligned} &x_{h,1} = \cdots = x_{h,\psi_h - 1} = Len, \\ &x_{h,\psi_h} = x_h - (\psi_h - 1) \cdot Len, \\ &\text{the other } x_{h,i} = 0, \end{aligned} \quad \text{(E)}$$

and

$$\begin{aligned} &x_h = \sum\nolimits_{i=1}^{\hat{\kappa}_1+1} x_{h,i}, \text{ if } h \in [1, m_1] \\ &x_h = \sum\nolimits_{i=1}^{\hat{\kappa}_2+1} x_{h,i}, \text{ if } h \in [m_1 + 1, \delta_j] \end{aligned} \quad \text{(F)}$$

where $0 \leq x_{h,\psi_h} < Len$. We define the sign function $sgn(x)$: it equals 1 if $x > 0$ and 0 if $x = 0$. Let

$$y_{h,i} = sgn(x_{h,i}) \in \{0, 1\}, \quad \text{(G)}$$

and the price of utilizing the h-th instance is p times the sum of all $y_{h,i}$; here, by (E), the sum of all $y_{h,i}$ is ψ_h.

The cost minimization problem under the allocation form of $\mathcal{A}'$ is as follows, referred to as $\mathcal{Q}$**-I**:

$$\min \sum_{h=1}^{m_1} \sum_{i=1}^{\hat{\kappa}_1+1} p \cdot y_{h,i} + \sum_{h=m_1+1}^{\delta_j} \sum_{i=1}^{\hat{\kappa}_2+1} p \cdot y_{h,i} \quad \text{(H)}$$

subject to the constraints (A)-(G). $\mathcal{Q}$-I corresponds to another optimization problem: its objective function is also (H), subject to (A), (B), (F), (G), and for all $h \in [1, m_1]$

$$x_{h,1}, \cdots, x_{h,\hat{\kappa}_1} \in \{0, Len\}, \;\; x_{h,\hat{\kappa}_1+1} \in \{0, \phi_1\}, \quad \text{(I)}$$

and for all $h \in [m_1 + 1, \delta_j]$,

$$x_{h,1}, \cdots, x_{h,\hat{\kappa}_2} \in \{0, Len\}, \;\; x_{h,\hat{\kappa}_2+1} \in \{0, \phi_2\}. \quad \text{(J)}$$

The above mathematical problem is referred to as $\mathcal{Q}$**-II**. In the following, we prove that (i) any solution to $\mathcal{Q}$-I corresponds to a solution to $\mathcal{Q}$-II and their objective function (H) under these two solutions achieves the same value; then, (ii) an optimal solution to $\mathcal{Q}$-II corresponds to a solution to $\mathcal{Q}$-I, and their objective function under these two solutions also achieves the same value. The first point shows that the optimal value of $\mathcal{Q}$-II is a lower bound of the optimal value of $\mathcal{Q}$-I. The second point shows that there is a solution to $\mathcal{Q}$-I under which the value of (H) equals the optimal value of $\mathcal{Q}$-II; hence, this solution to $\mathcal{Q}$-I is optimal and we will give such an optimal solution while proving the two points above.

The decision variables of both $\mathcal{Q}$-I and $\mathcal{Q}$-II are the same, i.e., $\{y_{h,i} | h \in [1, m_1], i \in [1, \hat{\kappa}_1 + 1]\} \cup \{y_{h,i} | h \in [m_1 + 1, \delta_j], i \in [1, \hat{\kappa}_2 + 1]\}$. Given a solution to $\mathcal{Q}$-I denoted by Y, we set the decision variables of $\mathcal{Q}$-II to the same values. Now, we show Y is a feasible solution to $\mathcal{Q}$-II. Both in $\mathcal{Q}$-II and $\mathcal{Q}$-I, the same $x_{h,i}$ is set to non-zero and the others are set to zero by (G), and the non-zero's $x_{h,i}$ in $\mathcal{Q}$-II is $\geq$ the $x_{h,i}$ in $\mathcal{Q}$-I by (C), (D), (I), and (J). Since (A) holds in $\mathcal{Q}$-I where the value of x_h is defined in (F), we have (A) also holds in $\mathcal{Q}$-II. Hence, Y is feasible. Furthermore, $\mathcal{Q}$-I and $\mathcal{Q}$-II have the same objective function (H) that achieves the same value under the same Y. This finishes proving the first point above.

Now, we give an optimal solution to $\mathcal{Q}$-II. The physical meaning of $\mathcal{Q}$-II can be explained as follows. There are 3 types of items each with a weight p: (i) $\hat{\kappa}_1 \cdot m_1 + \hat{\kappa}_2 \cdot m_2$ items each with a size Len, (ii) m_1 items each with a size ϕ_1 ($< Len$), and (iii) m_2 items each with a size ϕ_2 ($< Len$); the objective is to select some items such that the total size of chosen items is $\geq z_j^{i_j+1}$ (satisfying (A)) while their total weight (i.e., (H)) is minimized. Since items have the same weight, an optimal solution is just to select the minimum number of items, e.g., the items with the largest sizes, to exactly satisfy the size requirement; correspondingly, an optimal solution to $\mathcal{Q}$-II is such that the value of $y_{h,i} \in \{0, 1\}$ satisfies

$$\begin{aligned} &y_0 = \sum_{h=1}^{m_1} \sum_{i=1}^{\hat{\kappa}_1} y_{h,i} + \sum_{h=m_1+1}^{\delta_j} \sum_{i=1}^{\hat{\kappa}_2} y_{h,i}, \\ &y_1 = \sum_{h=1}^{m_1} y_{h,\hat{\kappa}_1+1}, \;\; y_2 = \sum_{h=m_1+1}^{\delta_j} y_{h,\hat{\kappa}_2+1}, \end{aligned} \quad \text{(K)}$$

where y_0, y_1, y_2 are described in Proposition 4.7. We denote such a solution by OPT_2. Here, we set $x_{h,i}$ to non-zero if $y_{h,i} = 1$ and zero otherwise by (G); the particular value of $x_{h,i}$ depends on (I) and (J), and it determines the value of x_h by (F) that can satisfy (B); by (K), x_h can satisfy (A).

Next, we show OPT_2 corresponds to a solution OPT_1 to $\mathcal{Q}_1$-I, and their objective function (H) under OPT_1 and OPT_2 achieves the same value. In $\mathcal{Q}$-I, we set the value of x_h to the same value when the solution to $\mathcal{Q}$-II is OPT_2 where the constraints (A) and (B) in $\mathcal{Q}$-I are naturally satisfied; then, we use (E) to obtain feasible $x_{h,i}$ that will also satisfy (C) and (D); by (G), the value of $y_{h,i}$ in $\mathcal{Q}$-I can be set, deriving a feasible solution OPT_1 to $\mathcal{Q}$-I. In both $\mathcal{Q}$-I and $\mathcal{Q}$-II, we have the number of non-zero's $y_{h,i}$ is $\lceil x_h/Len \rceil$; hence, $\mathcal{Q}$-I under OPT_1 and $\mathcal{Q}$-II under OPT_2 achieve the same value. Finally, OPT is an optimal solution to $\mathcal{Q}$-I by the two points above.

In the proof of Proposition 4.7, we have given an optimal solution OPT_1 to $\mathcal{Q}$-I; it is a particular cost-optimal allocation of on-demand instances, which is also illustrated in Fig. 1 (right). ■

B The Online Learning Algorithm

In this section, we formally describe the online learning algorithm (TOLA) used in the supplemented paper to learn the most cost-effective parameters β_0, β, b.

The online learning algorithm that we adopt is the one in [1], presented as Algorithm 1, and is also a form of the classic weighted majority algorithm. There are a set of jobs $\mathcal{J}$ that arrive sequentially over time, indexed by $j = 1, 2, \cdots$, and a set of n parametric policies $\mathcal{P}$ each specified by $\{\beta_0, \beta, b\}$ and indexed by $\pi = 1, 2, \cdots$. Let $d = \max_{j \in \mathcal{J}}\{d_j\}$, i.e., the maximum relative deadline of all jobs. Let $\mathcal{J}_t \subseteq \mathcal{J}$ denote all jobs j that arrive at time slot t, i.e., $a_j = t$. There is a weight distribution w over n policies; initially, it is a discrete uniform distribution $\{1/n, \cdots, 1/n\}$ (lines 1-2 of Algorithm 1). The distribution w will be updated as time goes by (lines 3, 9-20) and it is used to choose a policy in $\mathcal{P}$ for each job (lines 4-8).

Time t goes from slot 1 to later slots (line 3). When a job $j \in \mathcal{J}_t$ arrives where $t = a_j$, the algorithm randomly picks a policy π_j from $\mathcal{P}$ according to the current w and bases the allocation of various instances to j on that policy (lines 4-8). Let the policy π_j be defined by $\{\beta_0^{(j)}, \beta^{(j)}, b^{(j)}\}$ and let the array N denote the number of self-owned instances unreserved/available at every slot after allocating self-owned instances to the previous jobs $1, \cdots, j-1$ via the policy (4) with $\beta_0^{(1)}, \cdots, \beta_0^{(j-1)}$ as the control parameters respectively; initially, if $j = 1$, we have $N(t) = R$ for all $t \in [1, T]$ where R is the total number of self-owned instances. As time t goes from slot a_j towards $a_j + d_j - 1$, the allocation of instances to j is taken by executing Algorithm 2, i.e., Dynalloc$\Big(a_j, d_j, z'_j, \delta_j, \beta_0^{(j)}, \beta^{(j)}, b^{(j)}, N, t\Big)$, at every slot $t \in [a_j, a_j + d_j - 1]$ until j is allocated enough instances to complete z_j workload. As modeled in the Section 3 of the supplemented paper, the cost of completing a job j is from the use of spot and on-demand instances alone, and denoted by $c_j(\pi_j)$.

On the other hand, when time goes to the beginning of slot $d+1$, the update of the weight distribution of policies begins (line 9). In particular, if $\mathcal{J}_{t-d} \neq \emptyset$, we sequentially consider every job j' in $\mathcal{J}_{t-d}$ (lines 10-12, 20). Let a virtual array N_{β_0} denote the number of self-owned instances unreserved/available at every slot if the allocation of self-owned instances to the previous jobs $1, \cdots, j'-1$ follows the policy (4) with the same control parameter β_0; initially, if $j' = 1$, we have $N_{\beta_0}(t) = R$ for all $t \in [1, T]$. For every policy $\pi \in \mathcal{P}$, it is defined by $\{\beta_0, \beta, b\}$; since the spot prices in $[t-d, t-1]$ have been revealed, we are able to compute the cost of completing the job j' in $[a_j, a_j + d_j - 1]$ under the policy π with N_{β_0} recording the available self-owned instances, denoted by $c_{j'}(\pi)$ (line 13). Subsequently, the weight of each policy (i.e., its probability) is updated so that the lower-cost (higher-cost) polices of this job are re-assigned the enlarged (resp. reduced) weights (lines 14-18).

Algorithm 1: OptiLearning

```
Input : a set $\mathcal{P}$ of $n$ policies, each $\pi$ parameterized for
        indexing so that $\pi \in \{1, 2, \cdots, n\}$; the set $\mathcal{J}_t$ of
        jobs that arrive at $t$;
1  $i \leftarrow 1$; // $i$ is used to track the number of times
     updating the weight distribution
2  initialize the weight vector of policies:
     $w_i = \{w_{i,1}, \cdots, w_{i,n}\} = \{1/n, \cdots, 1/n\}$;
3  for $t \leftarrow 1$ to $T$ do
       // time goes from slot 1 towards later slots
4      $\mathcal{J}'_t \leftarrow \mathcal{J}_t$;
5      while $\mathcal{J}'_t \neq \emptyset$ do
6          get a job $j$ from $\mathcal{J}'_t$ such that $j$ is the smallest;
7          pick a policy $\pi_j = \pi$ with a probability $w_{i,\pi}$,
             applied to $j$; // When time $t$ goes from $a_j$ to
             $a_j + d_j - 1$, the allocation of instances to
             $j$ is completed via the Algorithm 2 in
             the supplemented paper
8          $\mathcal{J}'_t \leftarrow \mathcal{J}'_t - \{j\}$;
9      if $t > d$ then
10         $\mathcal{J}''_{t-d} \leftarrow \mathcal{J}_{t-d}$;
11         while $\mathcal{J}''_{t-d} \neq \emptyset$ do
12             get a job $j'$ from $\mathcal{J}''_{t-d}$ such that $j$ is the
                 smallest;
13             compute the cost of completing $j'$ in the period
                 of $[a_{j'}, a_{j'} + d_{j'} - 1]$ under every policy
                 $\pi \in \mathcal{P}$, denoted by $c_{j'}(\pi)$;    // When $t'$
                 ranges from $a_{j'}$ to $a_{j'} + d_{j'} - 1$, the
                 allocation to $j'$ is completed via
                 Dynalloc$\Big(a_{j'}, d_{j'}, z'_{j'}, \delta_{j'}, \beta_0, \beta, b, N_{\beta_0}, t'\Big)$;
                 the cost is recorded accordingly
14             $\eta_t \leftarrow \sqrt{\frac{2\log n}{d(t-d)}}$;
15             for $\pi \leftarrow 1$ to $n$ do
16                 $w'_{i+1,\pi} \leftarrow w_{i,\pi} \exp^{-\eta_t c_{j'}(\pi)}$;
17             for $\pi \leftarrow 1$ to $n$ do
18                 $w_{i+1,\pi} \leftarrow \frac{w'_{i+1,\pi}}{\sum_{i=1}^{n} w'_{i+1,i}}$;
19             $i \leftarrow i + 1$;
20             $\mathcal{J}''_{t-d} \leftarrow \mathcal{J}''_{t-d} - \{j'\}$;
```

Let $N' = |\cup_{t=d+1}^{T} \mathcal{J}_t|$, i.e., the number of all jobs that arrive in $[d+1, T]$, and, as proved in [1], we have that

Proposition B.1. *For all $\delta \in (0, 1)$, it holds with a probability at least $1 - \delta$ over the random of online learning that*

$$\max_{\pi \in \mathcal{P}} \left\{ \sum_{t \in \cup_{t=d+1}^{T} \mathcal{J}_t} \frac{c_j(\pi_j) - c_j(\pi)}{N'} \right\} \leq 9\sqrt{\frac{2d\log(n/\delta)}{N'}}.$$

Proposition B.1 says that, as TOLA runs, the actual total cost of completing all jobs is close to the cost of completing all jobs under a policy $\pi^* \in \mathcal{P}$ that generates the lowest total cost. Recall that a policy is defined by a tuple of parameters from $\mathcal{P}$.

REFERENCES

[1] Ishai Menache, Ohad Shamir, Navendu Jain. ”On-demand, Spot, or Both: Dynamic Resource Allocation for Executing Batch Jobs in the Cloud.” *In 11th International Conference on Autonomic Computing (ICAC’14).* USENIX Association, 2014.