Learning to Dispatch Multi-Server Jobs in Bipartite Graphs with Unknown Service Rates

Hailiang Zhao, Shuiguang Deng, *Senior Member, IEEE,* Feiyi Chen, Jianwei Yin, Schahram Dustdar, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

Abstract—Multi-server jobs are imperative in modern cloud computing systems. A multi-server job has multiple components and requests multiple servers for being served. How to allocate restricted computing devices to jobs is a topic of great concern, which leads to the job scheduling and load balancing algorithms thriving. However, current job dispatching algorithms require the service rates to be changeless and knowable, which is difficult to realize in production systems. Besides, for multi-server jobs, the dispatching decision for each job component follows the All-or-Nothing property under service locality constraints and resource capacity limits, which is not well supported by mainstream algorithms. In this paper, we propose a dispatching algorithm for multi-server jobs that learns the unknown service rates and simultaneously maximizes the expected Accumulative Social Welfare (Asw). We formulate the Asw as the sum of utilities of jobs and servers achieved over each time slot. The utility of a job is proportional to the valuation for being served, which is mainly impacted by the fluctuating but unknown service rates. We maximize the Asw without knowing the exact valuations, but approximate them with exploration-exploitation. From this, we bring in several evolving statistics and maximize the statistical Asw with dynamic programming. The proposed algorithm is proved to have a polynomial complexity and a State-of-the-Art regret. We validate it with extensive simulations and the results show that the proposed algorithm outperforms several benchmark policies with improvements by up to 73%, 36%, and 28%, respectively.

Index Terms—Job dispatching, gang scheduling, regret bound, social welfare, bipartite graph.

1 Introduction

Today's computing clusters have plenty of multi-server jobs, e.g., the distributed training of deep neural networks [1], [2] and large-scale graph computations [3], [4]. A notable feature of multi-server jobs is that they usually have multiple associated components, and multiple computation devices are used during their serving time. From the published Google cluster trace dataset [5], more than 90% jobs request multiple CPU cores and nearly 20% jobs request CPU cores no less than 1000.

How to dispatch jobs or service requests to servers with service locality constraints and restricted resource capacities for optimizing some target (e.g., QoS, overall system efficiency, utility, revenue, etc.) has always been the focus of attention. Service locality is common in modern cloud computing systems, especially for *Machine Learning as a Service* [6] and Serverless computing [7], [8]. With service locality, a job may only be processed by a subset of servers where the runtime and dependencies exist. Typical problems are job scheduling and load balancing. Simplistically, job scheduling studies when to dispatch each job to which server. A majority of job scheduling algorithms have been proposed by formulating combinatorial optimization problems with scenario-oriented constraints [9], [10], [11], [12], [13], [14], [15]. To solve the combinatorial programs, algorithms are designed based on various theoretical approaches, including relaxed integer programming [10], online primal-dual approaches [11], heuristics [12], [13], deep reinforcement learning (DRL) [14], [15], etc. Correspondingly, load balancing studies how to dispatch the traffic of service requests to the backend servers with the purpose of minimizing the mean response time. Load balancing are usually combined with queuing models and stability theories, which leads to the job dispatching algorithms such as Join-the-Shortest-Queue (JSQ) [16], Power-of-d-Choices (Pod) [17], and Join-the-Fastest-of-the-Shortest-Queues (JFIQ) [18]. For exmaple, JSQ polls the queue length of each server and dispatches every newly arrived request to the shortest queue.

The fundamental decision-making in the above works is job dispatching with service locality constraints and restricted resource capacities. A prerequisite for executing the above algorithms is that the service rates of servers are either *knowable* or *predictable*. However, in production systems, a server is always in multi-tasking and the actual service rates experienced by jobs can be unknown, fluctuating over time, and highly variable. As a result, the algorithms described above may not be effective in production systems. More than that, job dispatching of multi-server jobs is more difficult. The biggest challenge is that, for a multi-server job, the dispatching of its components should follow the All-or-Nothing property [19]. Let us take the distributed training of a large-scale deep neural network as an example. All the components of it, including a Parameter Server (PS) and several workers[1], should be allocated computation resources, especially CPUs and GPUs, according to their requirements

- *H. Zhao, S. Deng, F. Chen, and J. Yin are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China. e-mail: {hliangzhao, dengsg, chenfeiyi, zjuyjw}@zju.edu.cn*
- *S. Dustdar is with the Distributed Systems Group, Technische Universität Wien, 1040 Vienna, Austria. e-mail: dustdar@dsg.tuwien.ac.at*
- *A. Y. Zomaya is with the School of Computer Science, University of Sydney, Sydney, NSW 2006, Australia. e-mail: albert.zomaya@sydney.edu.au*

1. For distributed model training, there are two mainstream averaging architectures, *Collecitve-All Reduce* and *PS-Worker*.

simultaneously before being served. Otherwise, the training could not start, and the resources allocated out could not be retrieved back until the training is forcibly terminated. This All-or-Nothing property is recognized as Gang scheduling [20]. However, existing scheduling algorithms are mainly designed for fine-grained short-lived jobs while leaving the Gang property untouched.

To present a theoretically guaranteed dispatching algorithm for multi-server jobs without knowing actual service rates, in this paper, we propose the algorithm ESDP (Efficient Sampling-based Dynamic Programming), which learns the distribution of the unknown service rates and simultaneously seeks to maximize the expected Accumulative Social Welfare (EASW). The EASW is formulated as the sum of the utilities of jobs and servers over each time slot. The utility of a job is defined as the valuation for being served minus the payment. From the game theoretical point of views, valuation is the *willingness-to-pay* of the job submitters [21]. Our contribution is built on the intuition that, for any kind of job, the valuation of being served is decided mainly by the actual service rates of the chosen server(s). In other words, the faster the job is served, the more the job submitter is willing to pay. As the antithesis, the utility of a server is the payment collected from all the jobs that it serves minus the operating and maintaining cost. We use a bipartite graph to model the service locality constraints. To maximize the EASW, since we cannot obtain exact service rates, we learn to dispatch multi-server jobs with sufficient *exploration-exploitation*. Based on the exploited patterns, we introduce several deterministic optimization problems with the expectation approximated by statistics. Then, we solve the deterministic optimization problems by dynamic programming within polynomial time. Our main contributions are summarized as follows.

- We formally formulate the dispatching problem for multi-server jobs in a bipartite graph with unknown service rates. The objective is to maximize the EASW from a long-term vision.
- We propose an algorithm, named ESDP, to maximize the EASW. We also prove that ESDP has a polynomial complexity and a State-of-the-Art regret.
- We validate the performance of ESDP with extensive simulations. Experimental results show that, in default settings, ESDP significantly outperforms several baseline policies with improvements by up to 73%, 36%, and 28%, respectively.

The rest of this paper is organized as follows. We establish the multi-server job dispatching model with bipartite graphs and formulate the EASW maximization problem in Sec. 2. We then present the design details of ESDP with theoretical analysis in Sec. 3. Numerical results are presented in Sec. 4. We discuss related works in Sec. 5 and close this paper in Sec. 6.

2 System Model

We consider a computing cluster of heterogeneous servers serving several types of multi-server jobs. Jobs of different types have different components and request different number of servers under service locality constraints.

2.1 Bipartite Graph Model under Service Localities

To model service locality and the Gang property simultaneously, we consider a bipartite graph $(\mathcal{L}, \mathcal{R}, \mathcal{E})$ where $\mathcal{L}$ and $\mathcal{R}$ are the set of left vertices and right vertices, respectively, and $\mathcal{E}$ is the set of edges between the two sets of vertices. The vertices in $\mathcal{L}$ are indexed by l and viewed as job types, while the vertices in $\mathcal{R}$ are indexed by r and represent servers. For a vertex $l \in \mathcal{L}$, we use $\mathcal{R}_l \subseteq \mathcal{R}$ to represent the set of right vertices it connects with. Similarly, we use $\mathcal{L}_r \subseteq \mathcal{L}$ to represent the set of left vertices for $r \in \mathcal{R}$. An edge $(l, r) \in \mathcal{E}$ exists *iff* the following two conditions hold:

1) Service locality is satisfied;
2) The type-l job could be served *solely* by server r, i.e., the total resource requirements of it could be satisfied by r.

We designate each vertex $l \in \mathcal{L}$ as *port* and each edge (l, r) as *channel*. The bipartite graph model is visualized in Fig. 1.

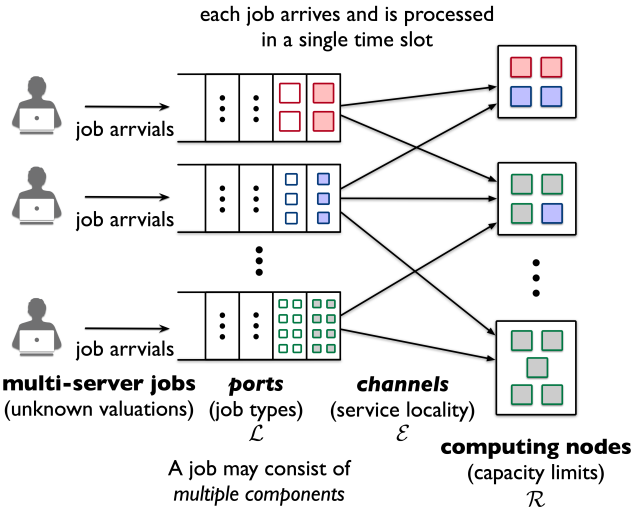

Fig. 1. The bipartite graph model for multi-server job dispatching.

2.2 Job Dispatching with Restricted Capacities

We consider a discrete time horizon. Time is slotted, and at each time $t \in \mathcal{T} \triangleq \{1, ..., T\}$, we assume that for each port, at most one job arrives. Concretely, at the beginning of time t, a job is yielded from port l with probability $\rho_l(t)$, and with probability $1 - \rho_l(t)$, there is no job. At time t, we use

$$\boldsymbol{x}(t) \triangleq \left[x_{(l,r)}(t)\right]^{\mathrm{T}}_{\forall (l,r) \in \mathcal{E}} \in \mathcal{X} \triangleq \{0,1\}^{|\mathcal{E}|}$$

to represent the dispatching decision. The time slot length is long enough to finish the most time-consuming job types. In this paper, all the vectors are column vectors if not specified. Note that if port l yields no job at t, denoted by $\mathbb{1}_l(t) = 0$, then $x_{(l,r)}(t) = 0$ for all $r \in \mathcal{R}_l$.

There are K types of computing devices in the considered cluster, e.g., CPUs, GPUs, NPUs, and FPGAs. For the type-k device, where $k \in \mathcal{K} \triangleq \{1, .., K\}$, the quantity of it that owned by the cluster is denoted by $c_k \in \mathbb{N}^+$. For each channel $(l, r) \in \mathcal{E}$, we use $a_k^{(l,r)} \in \mathbb{N}^+$ to denote the maximal requirements on the type-k computing device when all the components of the type-l job are dispatched to server r. In

other words, at least $a_k^{(l,r)}$ type-k computing devices are available on r. Therefore, we have the capacity constraints:

$$\sum_{(l,r)\in\mathcal{E}} a_k^{(l,r)} x_{(l,r)}(t) \le c_k, \forall k \in \mathcal{K}, t \in \mathcal{T}. \quad (1)$$

For the type-l job, all its components should be successfully dispatched[2]. Since $a_k^{(l,r)}$ is the maximal requirements on $r \in \mathcal{R}$, all the components can be dispatched onto r directly and be served *concurrently* (or partially parallel). Nevertheless, if the components of a job could be dispatched onto multiple servers and served *in parallelism*, the service rate could be improved. The cost is that all the resources required by the job will be locked and occupied during this time slot.

It is worth noting that the dispatching decision variable in our model, i.e., $x_{(l,r)}(t)$, is made for each channel, not for each job component. The reasons are as follows.

1) In practice, not all components of a job have to be executed. Take Alibaba ACK Pro as an exmaple, for configurating Gang scheduling jobs, it provides a field `min-available`[3] to represent the minimum number of components to be executed. The value of this field is decided by the job submitter and it is set flexibly according to the job types.
2) The main contribution of our paper is maximizing the EASW without knowing actual service rates. By changing the decision variable from channel to job component, the problem is much more complicated in mathematical form, which will make our work obscure. To illustrate this, let us re-define $\boldsymbol{x}(t)$ as

$$\boldsymbol{x}(t) \triangleq \left[x^l_{(q,r)}(t)\right]_{q\in\mathcal{Q}_l, r\in\mathcal{R}_l, l\in\mathcal{L}},$$

where $\mathcal{Q}_l$ stores the indices of components for the type-l job. Then, we have the following new constraints:

$$\begin{cases} \sum_{r\in\mathcal{R}_l}\sum_{q\in\mathcal{Q}_l} x^l_{(q,r)}(t) \ge m_l(t) & \forall l, t \\ \sum_{l\in\mathcal{L}}\sum_{q\in\mathcal{Q}_l} a^l_{(q,k)} x^l_{(q,r)}(t) \le c_{(k,r)} & \forall k, r, t, \end{cases}$$

where $m_l(t)$ is the minimum number of components to be executed, $a^l_{(q,k)}$ is the requirement of the type-k resource for the q-th component of the type-l job, and $c_{(k,r)} \in \mathbb{N}^+$ is the number of the type-k computing devices available to server r. The same to (1), the new constraint also has the form of $\mathbf{A}\boldsymbol{x} \le \boldsymbol{c}$. The new problem can be solved by a similar approach to the algorithms detailed in Sec. 3, but with much higher complexity. This could significantly reduce the readability of this paper.

2.3 Maximizing the EASW

The multi-server job dispatching is studied for maximizing the EASW, i.e., the sum of utilities of job submitters and the cluster providers. Our formulation is especially suitable for serverless computing because serverless comes with a *pay-for-value* billing model [8].

2. Actually, not all components of a job have to be executed. We will discuss it further on.

3. https://www.alibabacloud.com/help/doc-detail/178169.htm

At each time t, we assume that each arrived job of type-l has a *stochastic* quasi-linear utility $U_l(t)$, defined as

$$U_l(t) \triangleq \sum_{r\in\mathcal{R}_l} x_{(l,r)}(t) Z_{(l,r)}(t) - \mathbb{1}\Big\{\sum_{r\in\mathcal{R}_l} x_{(l,r)}(t) \ge 1\Big\}\pi_l(t),$$

where $Z_{(l,r)}(t)$ is the valuation of the type-l job being served through channel (l,r) and $\pi_l(t)$ is the payment. $\mathbb{1}\{p\}$ is the indicator function and it returns 1 if the predicate p is true, otherwise 0. $Z_{(l,r)}(t)$ is a random variable with an unknown mean $\upsilon_{(l,r)}$. As previously mentioned, we cannot know the exact value of $\upsilon_{(l,r)}$, but we try to approximate it with sufficient *exploration-exploitation*. Our fundamental assumption is that $\{\boldsymbol{Z}(t)\}_{t\in\mathcal{T}}$ follow some unknown distributions with the mean of $\boldsymbol{\upsilon} \triangleq [\upsilon_{(l,r)}]^{\mathrm{T}}_{\forall(l,r)\in\mathcal{E}}$. In addition, the valuation is *additive*, i.e., if a job is served through multiple channels in parallel, the final valuation is the sum of valuations obtained from all channels.

On the other side, the utility of the cluster for serving jobs is defined as

$$U_c(t) \triangleq \sum_{l\in\mathcal{L}} \mathbb{1}\Big\{\sum_{r\in\mathcal{R}_l} x_{(l,r)}(t) \ge 1\Big\}\pi_l(t) - \sum_{k\in\mathcal{K}}\sum_{(l,r)\in\mathcal{E}} f_k(a_k^{(l,r)}) x_{(l,r)}(t),$$

where $f_k(a_k^{(l,r)})$ is the supply cost for provisioning $a_k^{(l,r)}$ units of the type-k device for the type-l job at time t through the channel (l,r). In general, $\{f_k\}_{\forall k\in\mathcal{K}}$ models the operating, maintaining, and energy cost for serving jobs. Different from previous works [21], [22], [23], we make no assumptions on the convexity or differentiability of $\{f_k\}_{\forall k\in\mathcal{K}}$.

Our goal is to maximize the EASW, i.e., the expected sum of utilities of jobs and the cluster's in a long-term horizon. The problem is formulated as follows.

$$\mathcal{P}_1: \quad \max_{\forall t\in\mathcal{T}:\boldsymbol{x}(t)\in\mathcal{X}} \lim_{T\to\infty} \sum_{t=1}^{T} \mathbb{E}\Big[\mathrm{SW}\big(\boldsymbol{x}(t)\big)\Big]$$
$$s.t. \quad (1),$$
$$\sum_{r\in\mathcal{R}_l} x_{(l,r)}(t) = 0 \text{ if } \mathbb{1}_l(t) = 0, \forall l \in \mathcal{L}, t \in \mathcal{T}, \quad (2)$$

where $\mathrm{SW}\big(\boldsymbol{x}(t)\big)$ is the social welfare at time t:

$$\mathrm{SW}\big(\boldsymbol{x}(t)\big) \triangleq \sum_{l\in\mathcal{L}} U_l(t) + U_c(t). \quad (3)$$

With further transformation, we can get

$$\mathrm{SW}\big(\boldsymbol{x}(t)\big) = \sum_{(l,r)\in\mathcal{E}} x_{(l,r)}(t)\Big[Z_{(l,r)}(t) - \sum_{k\in\mathcal{K}} f_k(a_k^{(l,r)})\Big]. \quad (4)$$

3 Algorithm Design

In the following, we will detail the algorithm ESDP (Efficient Sampling-based Dynamic Programming) that solves $\mathcal{P}_1$ with polynomial complexity. ESDP is built on the ESCB algorithm [24], [25] and a recent derivative AESCB [26], for solving combinatorial semi-bandit problems. The overall steps of this section are as follows. Firstly, we introduce the regret minimization problem that corresponds to $\mathcal{P}_1$ and formulate several evolving statistics to approximate

the expected social welfare $\mathbb{E}\big[\textsc{Sw}\big(\boldsymbol{x}(t)\big)\big]$. Based on these statistics and a converge-to-zero sequence $\{\delta(t)\}_{t\in\mathcal{T}}$, we introduce a series of deterministic optimization problems. Then, we solve these deterministic problems with dynamic programming in polynomial time. Theoretical analysis for algorithm complexity and the upper bound of the regret are provided at the end.

3.1 Regret Minimizing with Evolving Statistics

$\mathcal{P}_1$ is an online *stochastic* optimization problem with random variables $\boldsymbol{Z}(t) = [Z_{(l,r)}(t)]^{\mathrm{T}}_{\forall(l,r)\in\mathcal{E}}$ not determined until the time t arrives. The EASW maximization problem $\mathcal{P}_1$ is equivalent to the regret minimization problem listed below:

$$\mathcal{P}_2: \min_{\forall t\in\mathcal{T}:\boldsymbol{x}(t)\in\mathcal{X}} \lim_{T\to\infty} \textsc{Re}(T) \triangleq \sum_{t=1}^{T}\mathbb{E}\Big[\Delta\big(\boldsymbol{x}(t)\big)\Big]$$
$$s.t. \quad (1), (2),$$

where the expected per-time slot gap $\mathbb{E}\big[\Delta\big(\boldsymbol{x}(t)\big)\big]$ is

$$\mathbb{E}\Big[\Delta\big(\boldsymbol{x}(t)\big)\Big] \triangleq \tilde{\boldsymbol{\upsilon}}^{\mathrm{T}}\boldsymbol{x}^*(t) - \mathbb{E}\Big[\textsc{Sw}\big(\boldsymbol{x}(t)\big)\Big] \tag{5}$$

and

$$\begin{cases} \tilde{\boldsymbol{\upsilon}} \triangleq \big[\upsilon_{(l,r)} - \sum_{k\in\mathcal{K}} f_k\big(a_k^{(l,r)}\big)\big]^{\mathrm{T}}_{\forall(l,r)\in\mathcal{E}} \in [0,1]^{|\mathcal{E}|} \\ \boldsymbol{x}^*(t) \triangleq \operatorname{argmax}_{\boldsymbol{x}(t)\in\Omega(t)} \big\{\tilde{\boldsymbol{\upsilon}}^{\mathrm{T}}\boldsymbol{x}(t)\big\} \\ \Omega(t) \triangleq \big\{\boldsymbol{x}(t)\in\mathcal{X} \mid (1)\ \&\ (2) \text{ hold at time } t\big\}. \end{cases} \tag{6}$$

The regret is the gap between the optimal social welfare achieved by an *omniscient* oracle who has the full knowledge on $\boldsymbol{\upsilon}$ and the social welfare achieved by the to-be-proposed algorithm. A good algorithm should achieve a smallest possible regret $\textsc{Re}(T)$ as T goes to infinity. For simplification, we use $\tilde{\boldsymbol{Z}}(t)$ to denote the column vector $[Z_{(l,r)}(t) - \sum_{k\in\mathcal{K}} f_k(a_k^{(l,r)})]^{\mathrm{T}}_{\forall(l,r)\in\mathcal{E}}$. W.O.L.G, $\tilde{\boldsymbol{Z}}(t)$ is normalized into $[0,1]^{|\mathcal{E}|}$. The non-negative property is widely accepted for utility functions [21], [23], [27], [28]. However, different from the above literature, we make no assumptions on the convexity or differentiability of $\{f_k\}_{\forall k\in\mathcal{K}}$.

Considering that $\mathcal{P}_2$ is still a stochastic optimization problem, based on the idea introduced by ESCB algorithm [24], ESDP introduces several statistics to approximate $\boldsymbol{\upsilon}$ based on explorated information. These statistics are used to supersede the random variables in $\mathcal{P}_2$. Specifically, at each time t, we define

$$n_{(l,r)}(t) \triangleq \sum_{t'=1}^{t} x_{(l,r)}(t') \tag{7}$$

as the *accumulative quantity* of channel $(l,r)\in\mathcal{E}$ been used up to time t. Based on it, we define the following statistics:

$$\hat{\upsilon}_{(l,r)}(t) \triangleq \begin{cases} \frac{\sum_{t'=1}^{t} x_{(l,r)}(t')\tilde{Z}_{(l,r)}(t')}{n_{(l,r)}(t)} & n_{(l,r)}(t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\hat{\sigma}^2_{(l,r)}(t) \triangleq \begin{cases} \frac{g(t)}{2n_{(l,r)}(t)} & n_{(l,r)}(t) > 0 \\ +\infty & \text{otherwise,} \end{cases} \tag{9}$$

where

$$g(t) \triangleq \ln t + 4\ln(\ln t + 1)\cdot \max_{t'\in\mathcal{T}}\Big\{\max_{\boldsymbol{x}\in\Omega(t')} \|\boldsymbol{x}\|_1\Big\}. \tag{10}$$

$\hat{\upsilon}_{(l,r)}(t)$ is a non-biased estimation based on historical noisy valuations for type-l job when served through channel (l,r). $\hat{\sigma}^2_{(l,r)}(t)$ is a metric proportional to the variance of the estimate $\hat{\upsilon}_{(l,r)}(t)$, proposed by [24]. We place a hat on the estimations to indicate that they are calculated and updated online. Inspired by the ESCB and AESCB algorithms, at time t, we introduce the following *deterministic* problem $\mathcal{P}_3(t)$:

$$\max_{\boldsymbol{x}(t)\in\Omega(t)} \tilde{\textsc{Sw}}(\boldsymbol{x}(t)) \triangleq \delta(t) + \hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t) + \sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}$$
$$s.t. \qquad (1),$$
$$\delta(t) > 0, \lim_{t\to\infty}\delta(t) = 0, \tag{11}$$

where

$$\begin{cases} \hat{\boldsymbol{\upsilon}}(t) \triangleq [\hat{\upsilon}_{(l,r)}(t)]^{\mathrm{T}}_{(l,r)\in\mathcal{E}} \\ \hat{\boldsymbol{\sigma}}^2(t) \triangleq [\hat{\sigma}^2_{(l,r)}(t)]^{\mathrm{T}}_{(l,r)\in\mathcal{E}} \end{cases}$$

are the corresponding column vectors.

In $\mathcal{P}_3(t)$, $\{\delta(t)\}_{t\in\mathcal{T}}$ could be any sequence converges to zero. For instance, $\delta(t) \triangleq \big(\ln(\ln t+1)+1\big)^{-1}$. The objective of $\mathcal{P}_3(t)$ is an approximated *statistical-based* social welfare for the multi-server job dispatching problem. From $\mathcal{P}_2$ to $\mathcal{P}_3(t)$, we remove the random variables $\boldsymbol{Z}(t)$ and transform the stochastic problem into a deterministic problem while keeping the solution space impervious. In most case, the following inequality should hold:

$$\Big|\big(\tilde{\boldsymbol{\upsilon}} - \hat{\boldsymbol{\upsilon}}(t)\big)^{\mathrm{T}}\boldsymbol{x}(t)\Big| \le \sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}. \tag{12}$$

By Chebyshev's Inequality, $\hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t) \pm \sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}$ covers nearly 60% population. To achieve a larger coverage, we can increase the numerical multiplier to the standard variance. For our multi-server job dispatching problem, setting the multiplier as 1 is enough to achieve the State-of-the-Art minimum regret upper bound. The analysis will be detailed in Sec. 3.3.

3.2 Polynomial-time Dynamic Programming

If the sequence $\{\delta(t)\}_{t\in\mathcal{T}}$ is removed from $\tilde{\textsc{Sw}}(\boldsymbol{x}(t))$ and (11) is dropped, $\mathcal{P}_3(t)$ is NP-hard [24], [25], i.e., it cannot be solved in polynomial time. Therefore, to solve it efficiently, inspired by the AESCB algorithm [26], ESDP resorts to solving several *relaxed* budgeted integer programming problems by adding the converge-to-zero sequence $\{\delta(t)\}_{t\in\mathcal{T}}$. Further, at each time t, based on $\delta(t)$, we define the following scale-up statistics for $\hat{\upsilon}_{(l,r)}(t)$ and $\hat{\sigma}^2_{(l,r)}(t)$ respectively:

$$\hat{\Upsilon}_{(l,r)}(t) \triangleq \Big\lceil \xi(t)\hat{\upsilon}_{(l,r)}(t)\Big\rceil \tag{13}$$

$$\hat{\Sigma}^2_{(l,r)}(t) \triangleq \Big\lceil \xi^2(t)\hat{\sigma}^2_{(l,r)}(t)\Big\rceil, \tag{14}$$

where

$$\xi(t) \triangleq \left\lceil \frac{\max_{t'\in\mathcal{T}}\big\{\max_{\boldsymbol{x}\in\Omega(t')}\|\boldsymbol{x}\|_1\big\}}{\delta(t)} \right\rceil \tag{15}$$

is the scaling size at time t. By the AESCB framework [26], at each time t, we introduce several budgeted integer

Algorithm 1: The ESDP Framework

Input: The bipartite graph $(\mathcal{L}, \mathcal{R}, \mathcal{E})$, requirements $\{a_k^{(l,r)}\}_{k\in\mathcal{K},(l,r)\in\mathcal{E}}$, capacities $\{c_k\}_{k\in\mathcal{K}}$, cost functions $\{f_k\}_{k\in\mathcal{K}}$, and the sequence $\{\delta(t)\}_{t\in\mathcal{T}}$

Output: Online solution to $\mathcal{P}_1$ (and $\mathcal{P}_2$) at time $t \in \mathcal{T}$

while $t = 1, ..., T$ **do**
  Observe the job arrival status from each port $l \in \mathcal{L}$
  Update $\hat{\mathbf{\Upsilon}}(t)$ and $\hat{\mathbf{\Sigma}}^2(t)$ with (13) and (14) based on $\delta(t)$, respectively
  /* Solve $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$ by Algorithm 2 */
  for *each* $s \in \mathcal{S}(t)$ **do**
    Solve $\mathcal{P}_4(s,t)$ and return $\boldsymbol{x}^*_{\mathcal{P}_4}(s,t)$
  end for
  $\boldsymbol{x}^*_{\mathcal{P}_4}(t) \leftarrow \boldsymbol{x}^*_{\mathcal{P}_4}(s^\star, t)$, where $s^\star$ staisfies (17)
  /* Satisfy constraint (2) of $\mathcal{P}_1$ */
  for *each* $l \in \mathcal{L}$ **do**
    if $\mathbb{1}_l(t) == 0$ **then**
      for *each* $r \in \mathcal{R}_l$ **do**
        Set the (l,r)-th element of $\boldsymbol{x}^*_{\mathcal{P}_4}(t)$ as 0
      end for
    end if
  end for
end while
return $\left\{\boldsymbol{x}^*_{\mathcal{P}_4}(t)\right\}_{t\in\mathcal{T}}$ *and* $\left\{\textsc{Sw}\left(\boldsymbol{x}^*_{\mathcal{P}_4}(t)\right)\right\}_{t\in\mathcal{T}}$

programming problems $\mathcal{P}_4(s,t)$ for each $s \in \mathcal{S}(t)$ where $\mathcal{S}(t) \triangleq \left\{0, 1, ..., \xi(t) \cdot \max_{t'\in\mathcal{T}} \max_{\boldsymbol{x}\in\Omega(t')} \|\boldsymbol{x}\|_1\right\}$ as follows:

$$\begin{aligned} \mathcal{P}_4(s,t): \quad & \max_{\boldsymbol{x}(t)\in\mathcal{X}} \hat{\mathbf{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t) \\ s.t. \quad & (1), (11), \\ & \hat{\mathbf{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t) \geq s. \end{aligned} \tag{16}$$

In $\mathcal{P}_4(s,t)$, $\hat{\mathbf{\Sigma}}^2(t)$ and $\hat{\mathbf{\Upsilon}}(t)$ are the corresponding column vectors for (13) and (14), respectively. Let us use $\boldsymbol{x}^*_{\mathcal{P}_4}(s,t)$ to denote the optimal solution for $\mathcal{P}_4(s,t)$. Then, the final solution to $\max\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$ at time t, denoted by $\boldsymbol{x}^*_{\mathcal{P}_4}(t)$, is set as some $\boldsymbol{x}^*_{\mathcal{P}_4}(s^\star, t)$ where $s^\star \in \mathcal{S}(t)$ staisfies

$$s^\star \in \operatorname*{argmax}_{s\in\mathcal{S}(t)} \left\{ s + \sqrt{\hat{\mathbf{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(s,t)} \right\}. \tag{17}$$

The main procedure of ESDP is summarized in **Algorithm 1**. The relations between $\mathcal{P}_3(t)$ and $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$, and how the solutions of $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$ affect the regret $\textsc{Re}(T)$ will be detailed in Sec. 3.3.

Now, the problem is how to solve $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$ optimally within polynomial time. ESDP solves it based on dynamic programming. Concretely, at each time t, corresponding to each $\mathcal{P}_4(s,t)$, we bring in the problem $\mathcal{P}_5(s,t,\boldsymbol{c},i)$ as follows.

$$\begin{aligned} \mathcal{P}_5(s,t,\boldsymbol{c},i): \quad & \max_{\boldsymbol{x}(t)\in\mathcal{X}} \hat{\mathbf{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t) \\ s.t. \quad & (1), (11), (16), \\ & \sum_{e=e_1}^{e_i} x_e(t) = 0, \end{aligned} \tag{18}$$

where $\boldsymbol{c} \triangleq [c_k]^{\mathrm{T}}_{k\in\mathcal{K}}$ is the capacity vector in (1), $e \triangleq (l,r) \in \mathcal{E}$ and e_i is the i-th edge (l,r) in $\mathcal{E}$. The new constraint (18) is used to set the first several dispatching decisions (until i) to 0 forcibly. Obviously, $\mathcal{P}_5(s,t,\boldsymbol{c},0)$ is equal to $\mathcal{P}_4(s,t)$ because (18) is not functioning when $i = 0$. The optimal solution of $\mathcal{P}_5(s,t,\boldsymbol{c},i)$ can be obtained by recursing over s, $\boldsymbol{c}$, and i. To do this, let us use $\boldsymbol{x}^*(s,t,\boldsymbol{c},i)$ to denote the optimal solution of $\mathcal{P}_5(s,t,\boldsymbol{c},i)$, and use $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i)$ to denote the corresponding objective. In the following, we demonstrate the recursing details.

Case I: If $x^*_{e_{i+1}}(s,t,\boldsymbol{c},i) = 0$, i.e., the $(i+1)$-element of $\boldsymbol{x}^*(s,t,\boldsymbol{c},i)$ is 0, then (18) is not violated for $\mathcal{P}_5(s,t,\boldsymbol{c},i+1)$. Thus, we have

$$\boldsymbol{x}^*(s,t,\boldsymbol{c},i+1) = \boldsymbol{x}^*(s,t,\boldsymbol{c},i) \tag{19}$$

and

$$V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i+1) = V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i). \tag{20}$$

The result means that $\boldsymbol{x}^*(s,t,\boldsymbol{c},i)$ is also the optimal solution to $\mathcal{P}_5(s,t,\boldsymbol{c},i+1)$.

Case II: If $x^*_{e_{i+1}}(s,t,\boldsymbol{c},i) = 1$, the optimal substructure is much more complicated. For simplification, we define matrix $\mathbf{A}$ by

$$\mathbf{A} = \left[a_k^{(l,r)}\right]^{K\times|\mathcal{E}|}.$$

Then we have

$$\mathbf{A}\Big(\boldsymbol{x}^*(s,t,\boldsymbol{c},i) - \boldsymbol{e}_{i+1}\Big) \leq \boldsymbol{c} - A_{:,i+1}, \tag{21}$$

where $\boldsymbol{e}_{i+1}$ is the $(i+1)$-th standard unit basis. Besides,

$$\hat{\mathbf{\Upsilon}}(t)^{\mathrm{T}}\Big(\boldsymbol{x}^*(s,t,\boldsymbol{c},i) - \boldsymbol{e}_{i+1}\Big) \geq s - \hat{\Upsilon}_{e_{i+1}}(t) \tag{22}$$

and

$$\hat{\mathbf{\Sigma}}^2(t)^{\mathrm{T}}\big(\boldsymbol{x}^*(s,t,\boldsymbol{c},i) - \boldsymbol{e}_{i+1}\big) = \hat{\mathbf{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*(s,t,\boldsymbol{c},i) - \hat{\Sigma}^2_{e_{i+1}}(t).$$

Combining the above formula with (21) and (22), we can get the following evolving optimal substructure:

$$\begin{aligned} V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i) = & V^*_{\mathcal{P}_5}\Big(\max\Big\{s - \hat{\Upsilon}_{e_{i+1}}(t), 0\Big\}, t, \\ & \max\{\boldsymbol{c} - A_{:,i+1}, 0\}, i+1\Big) + \hat{\Sigma}^2_{e_{i+1}}(t). \end{aligned} \tag{23}$$

Thus, for each possible s, $\boldsymbol{c}$, and i, we can update the solution to $\mathcal{P}_5(s,t,\boldsymbol{c},i)$ by

$$x^*_{e_{i+1}}(s,t,\boldsymbol{c},i) = \begin{cases} 0 & V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i) = V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c},i+1) \\ 1 & \text{otherwise.} \end{cases}$$

The recursion starts from condition $s = 0$, $\boldsymbol{c} = \mathbf{0}$, and $i = |\mathcal{E}|$. **Algorithm 2** summarizes the main procedure. It is used to substitute Step 5 $\sim$ Step 7 of ESDP. Obviously, **Algorithm 2** is of $\mathcal{O}\big(|\mathcal{E}| \cdot |\mathcal{S}(t)| \cdot \prod_{k\in\mathcal{K}} c_k\big)$-complexity, i.e., $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$ are solved in polynomial time. In the following analysis, we will show the relations between $\mathcal{P}_3(t)$ and $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$, and how the solution obtained by ESDP affects the social welfare regret $\textsc{Re}(T)$ defined in $\mathcal{P}_2$.

Algorithm 2: DP for solving $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$

Input: $\mathcal{S}(t)$, resource requirements $\{a_k^{(l,r)}\}_{k\in\mathcal{K},(l,r)\in\mathcal{E}}$, and scale-up statistics $\hat{\boldsymbol{\Upsilon}}(t)$ and $\hat{\boldsymbol{\Sigma}}(t)$

Output: Optimal solution to $\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$

1 $\forall i \in \{0,...,|\mathcal{E}|\}$, $\boldsymbol{x}^*(s,t,\boldsymbol{c},i) \leftarrow \mathbf{0}$ **for** s *from* 0 *to* $\xi(t)\cdot\max_{t'\in\mathcal{T}}\{\max_{\boldsymbol{x}\in\Omega(t')}\|\boldsymbol{x}\|_1\}$ **do**
2 **for** $\boldsymbol{c}'$ *from* $\mathbf{0}$ *to* $\boldsymbol{c}$ **do**
3 $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',|\mathcal{E}|)$ **is** 0 **if** $s==0$ **else** $-\infty$
4 **for** i *from* $|\mathcal{E}|-1$ *to* 0 **do**
5 **if** $\boldsymbol{c}'==\mathbf{0}$ **then**
6 $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i) \leftarrow V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i+1)$
7 **continue**
8 **end if**
9 $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i) \leftarrow \max\Big\{V^*_{\mathcal{P}_5}\Big(\max\{s-\hat{\Upsilon}_{e_{i+1}}(t),0\},t,\max\{\boldsymbol{c}'-A_{:,i+1},0\},i+1\Big)+\hat{\Sigma}^2_{e_{i+1}}(t), V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i+1)\Big\}$
10 **if** $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i) \neq V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i+1)$ **then**
11 $\boldsymbol{x}^*(s,t,\boldsymbol{c}',i) \leftarrow \boldsymbol{x}^*\Big(\max\{s-\hat{\Upsilon}_{e_{i+1}}(t),0\},t,\max\{\boldsymbol{c}'-A_{:,i+1},0\},i+1\Big)$
12 $x^*_{e_{i+1}}(s,t,\boldsymbol{c}',i) \leftarrow 1$ `// Update`
13 **if** $\mathbf{A}\boldsymbol{x}^*(s,t,\boldsymbol{c}',i) \le \boldsymbol{c}'$ *is violated* **then**
14 $V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i) \leftarrow V^*_{\mathcal{P}_5}(s,t,\boldsymbol{c}',i+1)$
15 $\boldsymbol{x}^*(s,t,\boldsymbol{c}',i) \leftarrow \boldsymbol{x}^*(s,t,\boldsymbol{c}',i+1)$
16 **end if**
17 **end if**
18 **end for**
19 **end for**
20 `/* Assign the solution of` $i=0$ `to` $\mathcal{P}_4(s,t)$ `*/`
21 $\boldsymbol{x}^*_{\mathcal{P}_4}(s,t) \leftarrow \boldsymbol{x}^*(s,t,\boldsymbol{c},0)$
22 **end for**
23 **return** $\{\boldsymbol{x}^*_{\mathcal{P}_4}(s,t)\}_{s\in\mathcal{S}(t)}$

3.3 Optimality and Regret Analysis

In this section, we will analyze the upper bound of $\mathrm{RE}(T)$ for ESDP when T goes to infinity. The result is based on the relations between the optimal solutions of several problems we defined above. The problems and their optimal solutions are summarized in Tab. 1 for quick reference. Our first result is that ESDP achieves the optimal statistical-based social welfare asymptotically with a certain probability.

Theorem 1. *(Probabilistic Asymptotical Optimality) By executing ESDP algorithm for problem $\mathcal{P}_3(t)$, $\lim_{t\to\infty}\mathrm{S\tilde{W}}(\boldsymbol{x}^*_{\mathcal{P}_4}(t))$ is at least*

$$\max_{\boldsymbol{x}(t)\in\Omega(t)}\left\{\hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)+\sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\right\} \tag{24}$$

with probability at most $\exp\left[-\frac{1}{3}\Big(|\mathcal{L}|-\sum_{l\in\mathcal{L}}\rho_l(t)\Big)^2\right]$.

Proof. Note that $\mathrm{S\tilde{W}}(\cdot)$ is the objective defined in $\mathcal{P}_3(t)$ and (24) is exactly the optimal objective of $\mathcal{P}_3(t)$ without the approximate parameter $\delta(t)$. Before our proof, we define the set

$$\Phi(t) \triangleq \{\boldsymbol{x}(t)\in\mathcal{X} \mid (1) \text{ holds at time } t\}, \tag{25}$$

Different from the set $\Omega(t)$, $\Phi(t)$ does not require constraint (2) to hold. Thus we have $\Omega(t)\subseteq\Phi(t)$. The following proof holds for every $t\in\mathcal{T}$.

TABLE 1
Probelms and Their Optimal Solutions.

Problems	Optimal Solutions
$\mathcal{P}_1$ (defined over $\mathcal{T}$)	$\{\boldsymbol{x}^*(t)\}_{t\in\mathcal{T}}$
$\mathcal{P}_2$ (defined over $\mathcal{T}$)	$\{\boldsymbol{x}^*(t)\}_{t\in\mathcal{T}}$, because $\mathcal{P}_1\equiv\mathcal{P}_2$
$\mathcal{P}_4(s,t)$	$\boldsymbol{x}^*_{\mathcal{P}_4}(s,t)$, also $\boldsymbol{x}^*(s,t,\boldsymbol{c},0)$
$\max\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$	$\boldsymbol{x}^*_{\mathcal{P}_4}(t)$ (by Step 8 of ESDP)
$\mathcal{P}_5(s,t,\boldsymbol{c},i)$	$\boldsymbol{x}^*(s,t,\boldsymbol{c},i)$ (by **Algorithm 2**)

By the definitions (8), (13), (14) and the fact $\tilde{\boldsymbol{Z}}\in[0,1]^{|\mathcal{E}|}$, we have

$$\hat{\boldsymbol{\upsilon}}(t) \le \frac{\hat{\boldsymbol{\Upsilon}}(t)}{\xi(t)} \le \frac{1}{\xi(t)}\mathbf{1}+\hat{\boldsymbol{\upsilon}}(t). \tag{26}$$

Thus,

$$\begin{aligned}&\max_{\boldsymbol{x}(t)\in\Omega(t)}\hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)+\sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\\ &\le\frac{1}{\xi(t)}\max_{\boldsymbol{x}(t)\in\Omega(t)}\hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}.\end{aligned} \tag{27}$$

Further, the RHS of (27) staisfies

$$\begin{aligned}&\max_{\boldsymbol{x}(t)\in\Omega(t)}\hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\\ &=\max_{s\in\mathcal{S}(t)}\max_{\hat{\boldsymbol{\Upsilon}}(t)(t)^{\mathrm{T}}\boldsymbol{x}(t)=s,\boldsymbol{x}(t)\in\Omega(t)}\left\{s+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\right\}\\ &\le\max_{s\in\mathcal{S}(t)}\max_{\hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)\ge s,\boldsymbol{x}\in\Omega(t)}\left\{s+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\right\}\\ &\le\max_{s\in\mathcal{S}(t)}\max_{\hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)\ge s,\boldsymbol{x}\in\Phi(t)}\left\{s+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)}\right\}.\end{aligned} \tag{28}$$

The RHS of (28) is exactly $\max\{\mathcal{P}_4(s,t)\}_{s\in\mathcal{S}(t)}$. The upper bound of it should be $s^\star+\sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t)}$ if no channel is shut down forcibly, i.e., Step 10 $\sim$ Step 16 are not executed by ESDP. To quantify the probability that no channels are forcibly shut down, we use the result of Chernoff Bounds. The upper tail of Chernoff Bounds is stated as follows.

If $X_1,...,X_n\in\{0,1\}$ are mutually independent, then $\forall x\ge\mu$, where $\mu\triangleq\mathbb{E}\big[\sum_i X_i\big]$, we have

$$\Pr\left[\sum_i X_i\ge x\right]\le e^{x-\mu}\left(\frac{\mu}{x}\right)^x.$$

Based on this conclusion, we can further derive that

$$\Pr\left[\sum_i X_i\ge(1+\varepsilon)\mu\right]\le\left(\frac{e^\varepsilon}{(1+\varepsilon)^{1+\varepsilon}}\right)^\mu, \tag{29}$$

where $\varepsilon\ge0$. With the Taylor-series expansion for $\ln(x+1)$ at $x=0$, we have

$$\frac{1}{\ln(1+\varepsilon)}\le\frac{1}{\varepsilon(1-\frac{1}{2}\varepsilon)}=\frac{1}{\varepsilon}+\frac{1}{2-\varepsilon}\le\frac{1}{\varepsilon}+\frac{1}{2}.$$

Applying the inequality to the RHS of (29), we can get

$$\Pr\left[\sum_i X_i \geq (1+\varepsilon)\mu\right] \leq \exp\left(-\frac{\varepsilon^2\mu}{3}\right). \tag{30}$$

Replacing X_i with $\mathbb{1}_l$ and $(1+\varepsilon)\mu$ with $|\mathcal{L}|$, (30) is transformed into

$$\Pr\left[\sum_{l\in\mathcal{L}} \mathbb{1}_l = |\mathcal{L}|\right] \leq \exp\left[-\frac{1}{3}\Big(|\mathcal{L}| - \sum_{l\in\mathcal{L}} \rho_l(t)\Big)^2\right], \tag{31}$$

which exactly quantifies the probability that every port yields at least one job. In this case, no channel $(l,r) \in \mathcal{E}$ is shut down forcibly. Thus, with this probability, the RHS of (28) staisfies

$$\begin{aligned}
&\max_{s(t)\in\mathcal{S}} \max_{\hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t)\geq s, \boldsymbol{x}(t)\in\Phi(t)} \left\{ s + \sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)} \right\} \\
&= s^\star + \sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t)} \quad \triangleright\ s^\star \textit{ staisfies } (17) \textit{ with prob. } (31) \\
&\leq \hat{\boldsymbol{\Upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t) + \sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t)} \qquad \triangleright\ (16) \\
&\leq \Big(\mathbf{1} + \xi(t)\hat{\boldsymbol{\upsilon}}(t)\Big)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t) + \sqrt{\hat{\boldsymbol{\Sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t)} \qquad \triangleright\ (26) \\
&\leq \xi(t)\left(\delta(t) + \hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t) + \sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}^*_{\mathcal{P}_4}(t)}\right). \qquad (32)
\end{aligned}$$

Combining the result of (27), (28), and (32), the following inequality holds for all the time $t \in \mathcal{T}$:

$$\max_{\boldsymbol{x}(t)\in\Omega(t)} \left\{ \hat{\boldsymbol{\upsilon}}(t)^{\mathrm{T}}\boldsymbol{x}(t) + \sqrt{\hat{\boldsymbol{\sigma}}^2(t)^{\mathrm{T}}\boldsymbol{x}(t)} \right\} \leq \tilde{\mathrm{SW}}\big(\boldsymbol{x}^*_{\mathcal{P}_4}(t)\big). \tag{33}$$

When $t \to \infty$ and $\delta(t) \to 0$, the result is tightly bounded. □

This theorem shows that ESDP can achieve approximately optimal statistical-based social welfare at each time slot with a certain probability. This optimality is important for minimizing the regret becasue it builds the upper bound of the optiaml social welfare $\tilde{\boldsymbol{\upsilon}}^{\mathrm{T}}\boldsymbol{x}^*(t)$ at each time t. The probabilistic regret upper bound is given by the following theorem.

Theorem 2. *(Regret Upper Bound under Certain Conditions) By executing the ESDP algorithm, as $T \to \infty$, $\mathrm{RE}(T)$ is upper bounded by*

$$\mathcal{O}\left(\ln T \cdot \frac{|\mathcal{E}| \cdot \big(\ln \boldsymbol{x}^*\big)^2}{\min_{t\in\mathcal{T}} \Delta\big(\boldsymbol{x}^*_{\mathcal{P}_4}(t)\big)}\right) \tag{34}$$

with probability at most $\exp\left(-\frac{1}{3}\big(|\mathcal{L}| - \sum_{l\in\mathcal{L}}\rho_l(t)\big)^2\right)$. In (34), $\Delta\big(\boldsymbol{x}^*_{\mathcal{P}_4}(t)\big)$ is introduced by (5), and $\boldsymbol{x}^*$ is defined as

$$\boldsymbol{x}^* \triangleq \underset{\forall t\in\mathcal{T}:\boldsymbol{x}^*(t)}{\operatorname{argmax}} \|\boldsymbol{x}^*(t)\|_1. \tag{35}$$

Proof. The result is immediate with **Theorem 1** and **Theorem 4.4** of [26]. Due to space limits, the derivation details are omitted. □

4 Numerical Results

In this section, we firstly verify the performance of ESDP against several handcrafted benchmarking policies on the accumulative social welfare. Then, we analyze the impact of several system and problematic parameters.

4.1 Experimental Setup

By default, the parameters are set as shown in Tab. 2. In this table, $\overline{\|\mathbf{A}\|_2}$ and $\underline{\|\mathbf{A}\|_2}$ are the upper bound and lower bound of $\{A_{ij}\}_{\forall i,j}$, respectively. Similarly, $\overline{\|\boldsymbol{c}\|_2}$ and $\underline{\|\boldsymbol{c}\|_2}$ are the upper bound and lower bound of $\|\boldsymbol{c}\|_2$, respectively. We use these bounds to limit the generations of $\mathbf{A}$ and $\boldsymbol{c}$. The bounds are carefully set such that (1) is not violated. Besides, $\rho_{(l,r)}$ is the probability that an edge exists between $l \in \mathcal{L}$ and $r \in \mathcal{R}$. $\forall (l,r) \in \mathcal{E}$, the true valuations, $v_{(l,r)}$, is generated by $\mathcal{N}\big(\mu_{(l,r)} \in [0.1, 1], \sigma_{(l,r)} = \frac{\mu_{(l,r)}}{2}\big)$.

For simplification, $\max_{t'\in\mathcal{T}}\big\{\max_{\boldsymbol{x}\in\Omega(t')}\|\boldsymbol{x}\|_1\big\}$ is calculated as $\alpha|\mathcal{E}|$, where $\alpha \in [0,1]$ is a coefficient by default to be 0.5. We set $\delta(t)$ and $g(t)$ as $\big(\ln(\ln(t+1)+1)+1\big)^{-1}$ and $\ln(t+1) + 4\ln\big(\ln(t+1)+1\big)\cdot\alpha|\mathcal{E}|$, respectively in default. Considering that those two sequences significantly affect the effectiveness of ESDP, we comprehensively discuss their variations in Sec. 4.3.

TABLE 2
Default parameter settings.

Param.	Value	Param.	Value
$\lvert\mathcal{L}\rvert$	8	$\lvert\mathcal{R}\rvert$	40
$\overline{\lVert\mathbf{A}\rVert_2}$	2	$\underline{\lVert\mathbf{A}\rVert_2}$	1
α	0.5	$\{\rho_l\}_{l\in\mathcal{L}}$	0.9
$\overline{\lVert\boldsymbol{c}\rVert_2}$	2	$\underline{\lVert\boldsymbol{c}\rVert_2}$	1
K	3	T	2000
$\{f_k\}_{k\in\mathcal{K}}$	$\sim\mathcal{N}(0.5, 0.1)$	$\{\rho_{(l,r)}\}_{(l,r)\in\mathcal{E}}$	0.1

ESDP is compared with the following handcrafted baselines.

- *The Highest Social Welfare First (HSWF)*: HSWF is different from ESDP in the following ways. At each time t, $\boldsymbol{Z}(t)$ is estimated as the average of historical observations. With the estimate, HSWF ranks each port in the descending order of $\sum_{r\in\mathcal{R}_l} x_{(l,r)}(t)\big(Z_{(l,r)}(t) - \sum_{k\in\mathcal{K}} f_k(a_k^{(l,r)})\big)$, and set the corresponding $x_{(l,r)}(t)$ as 1 in turn until (1) can not be satisfied.
- *The Lowest Cost First (LCF)*: Similar to HSWF, at each time t, $\boldsymbol{Z}(t)$ is estimated as the average of historical observations. Then, LCF ranks each job (non-empty port) in the ascending order of cost $\sum_{k\in\mathcal{K}} f_k(a_k^{(l,r)})$, and set the corresponding $x_{(l,r)}(t)$ as 1 in turn until (1) can not be satisfied.
- *The Longest Waiting Time First (LWTF)*: LWTF is different from ESDP in two ways. Firstly, $\boldsymbol{Z}(t)$ is estimated as the average of historical observations. Secondly, LWTF ranks each port in the descending order of the waiting time of jobs yield from that port, and set the corresponding $x_{(l,r)}(t)$ as 1 in turn until (1) can not be satisfied.

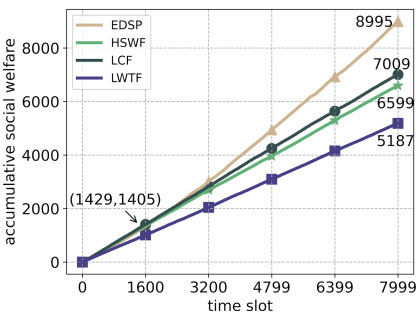

Fig. 2. The Asw vs. time slots.

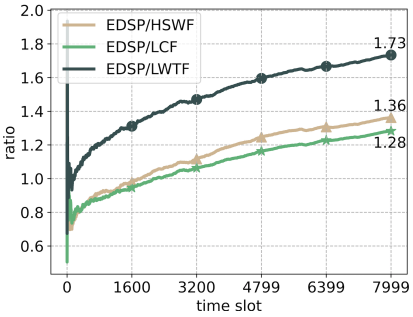

Fig. 3. The ratio between the Asws.

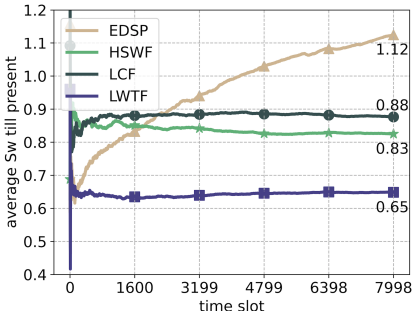

Fig. 4. The average Asw vs. time slots.

Note that we do not implement heuristics such as Genetic Algorithm (GA) for comparison. The reason is that, the EASW maximization problem is a stochastic optimization problem and traditional heuristics need to be revised carefully to match it. All of the three baselines use a similar method to estimate the historical valuation of each channel. With the estimate, the stochastic optimization problem is transformed into a deterministic one. Essentially, heuristics can be implemented by following a similar approach. However, a big problem that cannot be ignored is that heuristics are time-consuming with a non-polynomial complexity. Heuristics have to be called in every time slot, which could be very slow when the iteration number is large.

4.2 Performance Verification

We firstly analyze the accumulative social welfare (ASW) under different scale of time horizons. As Fig. 2 shows, ESDP outperforms the baselines by up to 73%, 36%, and 28%, respectively within 2000 time slots. In the beginning, HSWF performs better than EDSP, but as the time slots increase, ESDP gradually outperforms HSWF, and the gap between them keeps widening. EDSP is able to surpass HSWF becasue that, unlike HSWF, which does not adjust its strategy, EDSP constantly updates its strategy with the explorated valuation distributions. Besides, we also demonstrate the ratio between the ASW achieved by ESDP and the baselines in Fig. 3. We can conclude that, the performance of ESDP increases significantly when the time slots available to explore increase. The reason lies in that more time slots leads to more approximate estimate to $\{v_{(l,r)}\}_{(l,r)\in\mathcal{E}}$.

In Fig. 4, we calculate the average ASW in this way: for each time slot length T, the y-axis value is $\frac{1}{T}\sum_{t=1}^{T}\textsc{Sw}(\boldsymbol{x}(t))$. Different from the baselines, the average ASW welfare of ESDP increases steep and later flattens, which verifies that the ASW converges to an underlying upper bound (the Asw achieved by the oracle). The computation overhead of ESDP under different scales of the bipartite graph is shown in Fig. 5.

4.3 Sensitivity Analysis

In this section, we give a brief analysis on several important parameters. The first problematic parameter we test is the size of the solution space $\mathcal{X}$, which is tuned by $\mathbf{A}$ and $\boldsymbol{c}$. The x-axis of Fig. 6 is $\|\mathbf{A}^{-1}\boldsymbol{x}\|_2$. Without doubt, the ASW increases with the growth of $\mathcal{X}$ for all the algorithms becasue the can-be-processed jobs increase. Even so, ESDP has the highest growth in the ASW because it can fully exploit the estimated valuations.

The first algorithmic parameter we pay attention to is the sequence $\{\delta_t\}_{t\in\mathcal{T}}$, which is used to relax the NP-hard problem $\mathcal{P}_2$ to a polynomial one. The three $\{\delta_t\}_{t\in\mathcal{T}}$ shown in Fig. 7 are $\big(\ln(t+1)+1\big)^{-1}$, $\big(\ln(\ln(t+1)+1)\big)^{-1}$, and $\big(\ln((\ln(\ln(t+1)+1))+1)+1\big)^{-1}$, respectively. Fig. 7 demonstrates that different settings of the sequence has little effect on the performance of ESDP, but strong affect on the computation overhead. Another algorithmic parameter we are interested on is $\{g(t)\}_{t\in\mathcal{T}}$, which is used to estimate the variance (9). The three $\{g(t)\}_{t\in\mathcal{T}}$ demonstrated in Fig. 8 are $\ln(t+1)+4\ln(\ln(t+1)+1)\cdot\alpha|\mathcal{E}|$, $4\ln(\ln(t+1)+1)\cdot\alpha|\mathcal{E}|$, and $\ln(t+1)$, respectively. We can find that the third setting has an overwhelming advantage. The reason is that, theoretically, $g(t)$ acts as a balancer between exploration and exploitation. A smaller $g(t)$ leads to a higher tendency to exploitation.

Fig. 9 and Fig. 10 demonstrate the impact of job arrival rate ρ and the density of the bipartite graph. From Fig. 9 we can find that, with the increase of ρ, the ASW achieved by nearly all the algorithms also goes up. The result is obvious because more jobs can be processed within service capacities when ρ increases. Fig. 10 demonstrates the impact of the service locality constraint. When the number of edges increases in the bipartite graph, which means the service locality constraint is relaxed, the solution space $\mathcal{X}$ becomes larger. It significantly increases the difficulty of searching the optimal solution for ESDP.

5 Related Works

Job dispatching in scheduling problems. Job scheduling is extensive studied from both theoretical [9], [10], [11], [12], [13], [14], [15], [29] and system-level perspectives [30], [31], [32], [33], [34], [35]. The theoretical works usually formulate the job completion time (JCT) minimization problems as combinatorial, constrained optimizaiton problems and solves them with various approaches, especially the approximate algorithms. For Gang scheduling-based placement-sensitive Bulk Synchronous Parallel (BSP) jobs, Han et al propose an algorithm, named SPIN, with a rounding-based randomized approximation approch [10]. Their design is built on the relaxation of the Gang scheduling constraints and the JCT is minimized by linear programming algorithms. As for system-level schedulers, Tetris [30] was proposed for the multi-dimensional bin packing problem wherein task

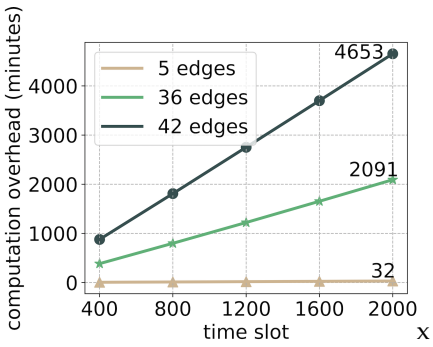

Fig. 5. Computation overheads.

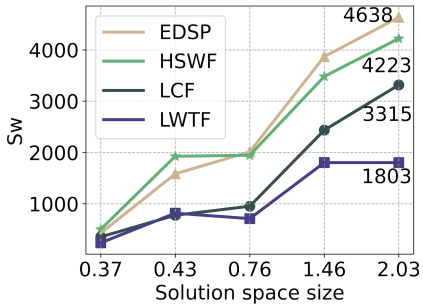

Fig. 6. Asw vs. $\mathcal{X}$.

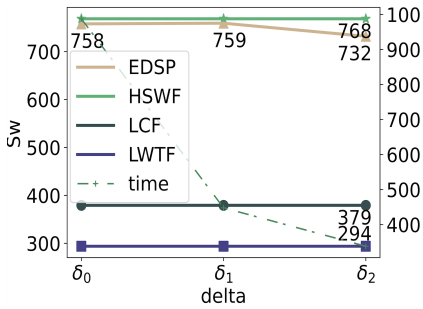

Fig. 7. Asw vs. $\{\delta(t)\}_{t\in\mathcal{T}}$.

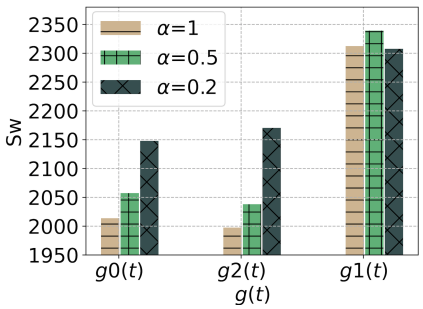

Fig. 8. Asw vs. $\{g(t)\}_{t\in\mathcal{T}}$.

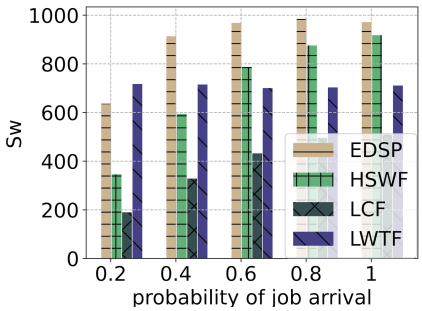

Fig. 9. Asw vs. ρ.

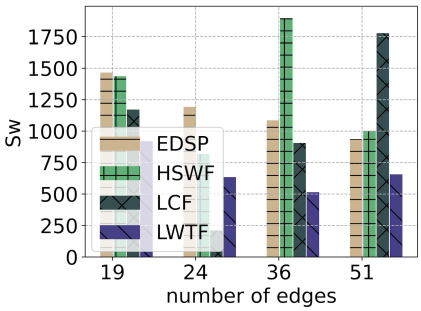

Fig. 10. Asw vs. $|\mathcal{E}|$.

arrivals and machine availability change constantly. Tetris improves the average JCT by preferentially serving jobs that have less remaining workload. Further, Graphene [31] and Decima [35] schedule jobs with Directed Acyclic Graph (DAG) task dependency. Decima allocates Spark workers to each stage of big-data analytic processing jobs with policy-based reinforcement learning algorithms [35]. The stages of a job formulates a DAG and the processing topology is based on the dependencies between them. The stages of a single job do not request Spark workers simultaneously. Except the recent work SPIN [10], none of works support Gang scheduling compulsorily.

Job dispatching in load balancing. Load balancing policies are usually designed based on Continuous-Time Markov Chains (CTMC) and Lyapunov Stability theories. They assume that jobs arrive according to Poisson process and service rates of computing instances are exponentially distributed [17], [18], [36], [37], [38]. As an example, the most classic policy JSQ [16] dispatches each new arrvied job to the shortest queue available. To simultaneously minimize the average job queuing delay and reduce the communication & memory cost in large-scale systems, Pod [17] was proposed and works by dispatching each arrived job to randomly selected d servers. The most recent policy is JFIQ (JFSQ) [18], which dispatches each job to the fastest of idle (shortest) server under service locality constraints. However, all the above works require the service rates to be known. The work that most similar to ours is [39]. This work proposes a job dispatching policy that learns the unknown service rates with the goal of minimizing the queuing delay of each job. By constrast, our work studies the *eAsw* maximization problem and supports Gang scheduling. In addition, the techniques adopted are totally different.

6 Conclusion

In this paper, we study the multi-server job dispatching problem with the purpose of maximizing the EAsw. To model the service localities and the Gang scheduling property of multi-server jobs, we introduce the bipartite graph model with restricted resource capacities. Considering that the service rates of jobs when dispatched to different servers are unknown, we alternatively learn the optimal job dispatching to maximize the statistical Asw based on several well-designed statistics. These statistics are formulated based on the information we have exploited. From this, we propose an efficient algorithm ESDP with a dynamic programming subroutine. ESDP solves several deterministic Asw maximization problems approximately in polynomial time and is proved to have a State-of-the-Art regret under certain probability. The performance of ESDP is also verified by extensive numerical results. Significantly, our model is also suitable for accumulated queuing and latency minimization. We leave this to our future work.

References

[1] S.-X. Zou, C.-Y. Chen, J.-L. Wu, C.-N. Chou, C.-C. Tsao, K.-C. Tung, T.-W. Lin, C.-L. Sung, and E. Y. Chang, "Distributed training large-scale deep architectures," in *International Conference on Advanced Data Mining and Applications*. Springer, 2017, pp. 18–32.

[2] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[3] D. Yan, J. Cheng, Y. Lu, and W. Ng, "Effective techniques for message reduction and load balancing in distributed graph computation," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1307–1317.

[4] W. Xiao, J. Xue, Y. Miao, Z. Li, C. Chen, M. Wu, W. Li, and L. Zhou, "Tux^2: Distributed graph computation for machine learning," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 669–682.

[5] M. Tirmazi, N. Deng, M.-E. Haque, Z.-G. Qin, S. Hand and A. Barker, "Google cluster-usage traces v3," https://github.com/google/cluster-data, 2019.

[6] M. Ribeiro, K. Grolinger, and M. A. Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[7] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski, and P. Suter, *Serverless Computing: Current Trends and Open Problems*. Singapore: Springer Singapore, 2017, pp. 1–20.

[8] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.

[9] J. V. Gautam, H. B. Prajapati, V. K. Dabhi, and S. Chaudhary, "A survey on job scheduling algorithms in big data processing," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2015, pp. 1–11.

[10] Z. Han, H. Tan, S. H.-C. Jiang, X. Fu, W. Cao, and F. C. Lau, "Scheduling placement-sensitive bsp jobs with inaccurate execution time estimation," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. IEEE Press, 2020, p. 1053–1062.

[11] Y. Bao, Y. Peng, C. Wu, and Z. Li, "Online job scheduling in distributed machine learning clusters," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 495–503.

[12] I. Attiya, M. Abd Elaziz, and S. Xiong, "Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm," *Computational intelligence and neuroscience*, vol. 2020, 2020.

[13] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling," *ieee transactions on cybernetics*, 2020.

[14] S. Liang, Z. Yang, F. Jin, and Y. Chen, "Data centers job scheduling with deep reinforcement learning," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2020, pp. 906–917.

[15] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, 2020, pp. 481–498.

[16] R. R. Weber, "On the optimal assignment of customers to parallel servers," *Journal of Applied Probability*, pp. 406–413, 1978.

[17] D. Mukherjee, S. C. Borst, J. S. Van Leeuwaarden, and P. A. Whiting, "Universality of power-of-d load balancing in many-server systems," *Stochastic Systems*, vol. 8, no. 4, pp. 265–292, 2018.

[18] W. Weng, X. Zhou, and R. Srikant, "Optimal load balancing in bipartite graphs," *arXiv preprint arXiv:2008.08830*, 2020.

[19] D. G. Feitelson and M. A. Jettee, "Improved utilization and responsiveness with gang scheduling," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 1997, pp. 238–261.

[20] H. D. Karatza, "Performance of gang scheduling strategies in a parallel system," *Simulation Modelling Practice and Theory*, vol. 17, no. 2, pp. 430–441, 2009.

[21] X. Tan, B. Sun, A. Leon-Garcia, Y. Wu, and D. H. Tsang, "Mechanism design for online resource allocation: A unified approach," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 2, Jun. 2020.

[22] H. Zhao, S. Deng, Z. Liu, Z. Xiang, J. Yin, S. Dustdar, and A. Zomaya, "Dpos: Decentralized, privacy-preserving, and low-complexity online slicing for multi-tenant networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[23] H. Zhao, S. Deng, Z. Xiang, and J. Yin, "Online social welfare maximization with spatio-temporal resource mesh for serverless," *CoRR*, vol. abs/2112.02456, 2021. [Online]. Available: https://arxiv.org/abs/2112.02456

[24] R. Combes, M. S. Talebi, A. Proutiere, and M. Lelarge, "Combinatorial bandits revisited," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15, 2015, p. 2116–2124.

[25] R. Degenne and V. Perchet, "Combinatorial semi-bandit with known covariance," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, 2016, p. 2972–2980.

[26] T. Cuvelier, R. Combes, and E. Gourdin, "Statistically efficient, polynomial-time algorithms for combinatorial semi-bandits," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, Feb. 2021.

[27] T. Roughgarden, "Algorithmic game theory," *Communications of the ACM*, vol. 53, no. 7, pp. 78–86, 2010.

[28] B. Sun, A. Zeynali, T. Li, M. Hajiesmaili, A. Wierman, and D. H. Tsang, "Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, Nov. 2020.

[29] S. Deng, H. Zhao, Z. Xiang, C. Zhang, R. Jiang, Y. Li, J. Yin, S. Dustdar, and A. Y. Zomaya, "Dependent function embedding for distributed serverless edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2346–2357, 2022.

[30] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 455–466.

[31] R. Grandl, S. Kandula, S. Rao, A. Akella, and J. Kulkarni, "GRAPHENE: Packing and dependency-aware scheduling for data-parallel clusters," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Nov. 2016, pp. 81–97.

[32] K. Mahajan, M. Chowdhury, A. Akella, and S. Chawla, "Dynamic query re-planning using QOOP," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, Oct. 2018, pp. 253–267.

[33] R. Grandl, M. Chowdhury, A. Akella, and G. Ananthanarayanan, "Altruistic scheduling in multi-resource clusters," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Nov. 2016, pp. 65–80.

[34] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: The next generation," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20, 2020.

[35] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19, 2019, p. 270–288.

[36] E. Anton, U. Ayesta, M. Jonckheere, and I. M. Verloop, "Improving the performance of heterogeneous data centers through redundancy," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, Nov. 2020.

[37] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-idle-queue: A novel load balancing algorithm for dynamically scalable web services," *Perform. Eval.*, vol. 68, no. 11, p. 1056–1071, Nov. 2011.

[38] D. Rutten and D. Mukherjee, "Load balancing under strict compatibility constraints," 2020.

[39] T. Choudhury, G. Joshi, W. Wang, and S. Shakkottai, "Job dispatching policies for queueing systems with unknown service rates," *arXiv preprint arXiv:2106.04707*, 2021.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation of China (NSFC) under Grants U20A20173 and 62125206. Schahram Dustdar's work is supported by the Zhejiang University Deqing Institute of Advanced technology and Industrilization (ZDATI).

Hailiang Zhao received the B.S. degree in 2019 from the school of computer science and technology, Wuhan University of Technology, Wuhan, China. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include cloud & edge computing, distributed systems and optimization algorithms. He has published several papers in flagship conferences and journals including IEEE ICWS 2019, IEEE TPDS, IEEE TMC, etc. He has been a recipient of the Best Student Paper Award of IEEE ICWS 2019. He is a reviewer for IEEE TSC and Internet of Things Journal.

Shuiguang Deng is currently a full professor at the College of Computer Science and Technology in Zhejiang University, China, where he received a BS and PhD degree both in Computer Science in 2002 and 2007, respectively. He previously worked at the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a visiting scholar. His research interests include Edge Computing, Service Computing, Cloud Computing, and Business Process Management. He serves for the journal IEEE Trans. on Services Computing, Knowledge and Information Systems, Computing, and IET Cyber-Physical Systems: Theory & Applications as an Associate Editor. Up to now, he has published more than 100 papers in journals and refereed conferences. In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a fellow of IET and a senior member of IEEE.

Feiyi Chen received the B.S. degree in 2021 from the school of computer science and engineering, Sun Yat-sen University (SYSU), Guangzhou, China. She is currently pursuing the master degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Her research interests include cloud computing, edge computing, and distributed systems.

Jianwei Yin received the Ph.D. degree in computer science from Zhejiang University (ZJU) in 2001. He was a Visiting Scholar with the Georgia Institute of Technology. He is currently a Full Professor with the College of Computer Science, ZJU. Up to now, he has published more than 100 papers in top international journals and conferences. His current research interests include service computing and business process management. He is an Associate Editor of the IEEE Transactions on Services Computing.

Schahram Dustdar is a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group at the TU Wien. He is founding co-Editor-in-Chief of ACM Transactions on Internet of Things (ACM TIoT) as well as Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Computing Surveys, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. Dustdar is recipient of multiple awards: TCI Distinguished Service Award (2021), IEEE TCSVC Outstanding Leadership Award (2018), IEEE TCSC Award for Excellence in Scalable Computing (2019), ACM Distinguished Scientist (2009), ACM Distinguished Speaker (2021), IBM Faculty Award (2012). He is an elected member of the Academia Europaea: The Academy of Europe, where he is chairman of the Informatics Section, as well as an IEEE Fellow (2016), an Asia-Pacific Artificial Intelligence Association (AAIA) President (2021) and Fellow (2021). He is an EAI Fellow (2021) and an I2CICC Fellow (2021). He is a Member of the 2022 IEEE Computer Society Fellow Evaluating Committee (2022).

Albert Y. Zomaya is the Peter Nicol Russell Chair Professor of Computer Science and Director of the Centre for Distributed and High-Performance Computing at the University of Sydney. To date, he has published > 600 scientific papers and articles and is (co-)author/editor of > 30 books. A sought-after speaker, he has delivered > 250 keynote addresses, invited seminars, and media briefings. His research interests span several areas in parallel and distributed computing and complex systems. He is currently the Editor in Chief of the ACM Computing Surveys and served in the past as Editor in Chief of the IEEE Transactions on Computers (2010-2014) and the IEEE Transactions on Sustainable Computing (2016-2020).

Professor Zomaya is a decorated scholar with numerous accolades including Fellowship of the IEEE, the American Association for the Advancement of Science, and the Institution of Engineering and Technology (UK). Also, he is an Elected Fellow of the Royal Society of New South Wales and an Elected Foreign Member of Academia Europaea. He is the recipient of the 1997 Edgeworth David Medal from the Royal Society of New South Wales for outstanding contributions to Australian Science, the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), IEEE Computer Society Technical Achievement Award (2014), ACM MSWIM Reginald A. Fessenden Award (2017), the New South Wales Premier's Prize of Excellence in Engineering and Information and Communications Technology (2019), and the Research Innovation Award, IEEE Technical Committee on Cloud Computing (2021).