In search of efficient scheduling heuristics from simulations and Machine Learning

Lucas Rosa[1], **Alfredo Goldman**[1]

[1] Institute of Mathematics and Statistics – Department of Computer Science
University of São Paulo (USP) – São Paulo, SP – Brazil

roses.lucas@usp.br, gold@ime.usp.br

Abstract. *High Performance Computing (HPC) systems are used to solve a number of complex issues in different fields of knowledge. However, these platforms have been rapidly evolving in size and complexity; and ensuring efficiency in managing applications (jobs) has become a challenge. Typically, this management involves scheduling heuristics that consist of functions to order the jobs. In this work we evaluate the limits of regression methods for creating scheduling heuristics. Our results show that the simplest heuristic led to the most efficient scheduling, while the more complex heuristics showed instabilities due to multicollinearity.*

1. Introduction

Covering different fields of science and technology (climate, health, economics, artificial intelligence, *etc.*), High Performance Computing (HPC) has become essential to solve complex problems and process the immense amount of data generated every day. The 59th edition of the TOP500 [Jack Dongarra and Erich Strohmaier 2022] – a list of the 500 most powerful computer systems – revealed the Frontier system to be the first true exascale machine with an HPL score (a parallel implementation of the Linkpack Benchmark) of 1,102 Exaflop/s.

At this scale, resource management is a critical issue that needs to be solved efficiently since HPC platforms usually consume huge amounts of energy. Resource management involves assigning when and where HPC applications (hereafter referred to as jobs) will be processed in such platforms. Its users (research groups, companies, *etc.*) submit their jobs to be processed at any given point, and limited information about the jobs is available for decision-making. Both the applications and the platform resources are managed by a Resources and Jobs Management System (RJMS) that is responsible for the decision-making process, solving instances of a problem called *online parallel job scheduling*.

Far from theoretical advances, online parallel job scheduling is in practice solved with scheduling heuristics based on waiting queue sorting algorithms, with backfilling mechanisms [Mu'alem and Feitelson 2001] using the simple First-Come-First-Served (FCFS). A possible explanation for this phenomenon is the explainability of scheduling heuristics, given by their simplicity. Regarding FCFS, this high level of explainability, comes with the drawback of poor scheduling performances [Carastan-Santos et al. 2019a].

Related works [Carastan-Santos and de Camargo 2017, Legrand et al. 2019] employ Machine Learning (ML) techniques to exploit simulation and platform workload

data (logs recorded by RJMS) to create scheduling heuristics. With this regard, regression methods [Carastan-Santos and de Camargo 2017] appear as a promising approach, since we can express scheduling knowledge in the form of simple functional forms of the jobs' characteristics.

The main goal with this work was to enable the student to investigate the hypothesis of a trade-off between simplicity/performance in learning HPC job scheduling heuristics, starting from the premise that we may need to sacrifice simplicity in our models to obtain better scheduling performances. In this paper we restrict ourselves to only show the main results, since the full work was submitted to the *International Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD 2022).

Our contributions indicate that we don't need to sacrifice simplicity to obtain efficient scheduling heuristics. A functional form constituted by a linear combination (simple multilinear polynomial) of the jobs' characteristics presented the best scheduling performances. Whereas multicollinearity – a phenomenon intrinsic to regression methods – may drastically degrade the scheduling performance of obtained scheduling heuristics constituted by complex polynomials.

The remainder of this paper is organized as follows: in Section 2 we present some preliminary definitions of online parallel job scheduling. We present our methodology in Sections 3 and 4. Section 5 presents our experimental results. Lastly, in Section 6, we present our concluding remarks and future works.

2. Online Parallel Job Scheduling

Given a set of n identical machines (processors) $\mathcal{M}$ that need to process parallel jobs $j_1, j_2, \ldots$, the online parallel job scheduling problem consists of deciding when and at which machines the jobs will be processed, in a way to optimize a certain performance metric. We assume that these machines are homogeneous (i.e., have the same processing power), and are connected by any interconnection topology.

The RJMS has limited information about the jobs, and this information is only available after the job's submission. The main information available are notably (i) the estimated job's processing time ($\tilde{p}_j$), normally informed by the user, (ii) the number of processors required to process the job (q_j), and (iii) the time when the job was submitted in the platform (r_j, also known as release time). Another important piece of information that is only available after finishing a job j is its actual processing time (p_j).

Many online scheduling problems aim to minimize objective functions related to the time that the jobs spent in the system. On this basis, we adopt the average bounded slowdown (AVGbsld) as the scheduling performance metric. Given a job j, its bounded slowdown (bsld) value can be computed as follows

$$\text{bsld}_j = \max\left\{\frac{w_j + p_j}{\max(p_j, \tau)}, 1\right\} \tag{1}$$

where w_j is the time that a job waited for processing since its submission and τ is a constant that is typically set in the order of 10 seconds, that prevents small jobs from having excessively large slowdown values. The average bounded slowdown takes into

account the slowdown average for a set of jobs J and is defined as

$$\text{AVGbsld}(J) = \frac{1}{|J|} \sum_{j \in J} \max \left\{ \frac{w_j + p_j}{\max(p_j, \tau)}, 1 \right\}. \quad (2)$$

3. Parallel Job Scheduling Simulations

Let S and Q be two sets of jobs. We frame a proxy scheduling problem (similar as the target one) as the scheduling of the jobs in Q, in an HPC platform with m interconnected processors where the jobs in S are currently being processed. The job sets S and Q are generated in the following way: from a large enough job log file (also referred to as trace) N, we randomly select a subtrace $M \subset N$ with size $|S| + |Q|$. The first $|S|$ jobs from M belongs S and the remaining $|Q|$ jobs from M belongs Q.

For each pair (S, Q) – also referred to as *trials* of Q – we randomly sample permutations Q^* of the set Q and simulate the scheduling of the jobs (S, Q^*), following the order that the jobs are present in Q^*. With $\mathcal{P}$ being the set containing all sampled permutations, when the scheduling simulation of all trials end, we compute and assign a score for each job $j \in Q$:

$$score(j) = \frac{\sum_{Q_l^* \in \mathcal{P}(j_0 = j)} \text{AVGbsld}(Q_l^*)}{\sum_{Q_k^* \in \mathcal{P}} \text{AVGbsld}(Q_k^*)}. \quad (3)$$

The score represents the impact of scheduling a job $j \in Q$ first (represented by j_0 in Equation 3), in terms of the average bounded slowdown (Equation 2) of all jobs in Q. Jobs with a low score represent a positive impact on the total average slowdown when they are executed first. The set $\{score(j) \mid \forall j \in Q\}$ defines a score distribution of the jobs of Q given the initial state S.

The $score(j)$ along with the characteristics p_j, q_j and r_j are the central output of the simulations. We conjecture that with an initial state represented by the scheduling of the jobs in S, sorting the jobs in Q in increasing order of $score(j)$ results in an efficient schedule regarding the $AVGbsld(Q)$ (Equation 2).

We repeat the aforementioned simulation strategy with many samples of job set pairs (S, Q). The idea is that each distinct pair (S, Q) represents a certain scheduling situation, and the computed $\{score(j) \mid \forall j \in Q\}$ represents a good scheduling strategy for this scheduling situation.

4. Multiple Linear Regression

At the end of the simulation strategy, we have a data set with the information about the jobs' characteristics p_j, q_j and r_j, and their computed $score(j)$. Let J be all jobs present in the data set with their compute $score$ values, the problem consists in finding functions $f(p_j, q_j, r_j)$ that provide a good approximation to the $score$ values of all jobs $j \in J$. For that, we defined a function family $\mathcal{F}$, with parametrized functions of the form

$$f(\theta, \mathbf{x}) = \theta^T \mathbf{x} \quad (4)$$

where θ is a parameter vector, and $\mathbf{x}$ is a vector of functional forms of the jobs' characteristics p, q and r.

We create a function family containing four functions whose vectors $\mathbf{x}$ are presented by Table 1. Each element in the vectors (also referred to as basis functions) are polynomials of the jobs' characteristics, with degrees in the set $\{1, 2, 3, 4\}$.

Table 1. Functional forms of the four parametrized functions used in multiple linear regression.

Vector components	Vector $\mathbf{x}$			
	`Lin`	`Qdr`	`Cub`	`Qua`
$(1, p, q, r)$	✓	✓	✓	✓
(p^2, q^2, r^2, pq)		✓	✓	✓
$(p^3, q^3, r^3, p^2q, pq^2, (pq)^2)$			✓	✓
$(p^4, q^4, r^4, p^3q, pq^3, (pq)^3)$				✓

The function `Lin` is just a linear combination of the jobs' characteristics p, q and r. The others `Sq`, `Cub` and `Qua` are functions that progressively increase the degree of the basis functions, and with multiplicative factors related to pq, which is often referred in the literature [Carastan-Santos et al. 2019b] as the area of the jobs.

We employ a weighted multiple linear regression [Carroll and Ruppert 1988] procedure, which minimizes the weighted sum of squared loss function:

$$\Sigma_{wL} = \sum_{j \in J} \left[(p_j q_j) \cdot (f(\theta, \mathbf{x}) - score(j)) \right]^2 . \tag{5}$$

The weight $(p_j q_j)$ emphasizes that the approximation must perform a good estimation of the $score$ of large area jobs (i.e., jobs with p and q large), as they can end up blocking the execution of small jobs, degrading the overall scheduling performance.

After obtaining the coefficients $\hat{\theta}^f$ from the multiple linear regression approximation for all functions $f(\theta, \mathbf{x}) \in \mathcal{F}$, we can measure the approximation quality through the *Mean Absolute Error* function (MAE, Equation 6).

$$\text{MAE}(f) = \frac{1}{|J|} \sum_{j \in J} \| f(\hat{\theta}^{\mathbf{f}}, \mathbf{x}) - score(j) \| \tag{6}$$

With these approximated functions we perform a simulation experimental campaign to address how efficient these functions are when used as scheduling policies in an online parallel scheduling scenario.

5. Experiments and Results

Following the methodology, we adjusted the functions `Lin`, `Qdr`, `Cub` and `Qua` to the *score distribution*. The data set used in the multiple linear regression consists of 7168 jobs and their calculated $score(j)$ values. The simulations were performed using the SimGrid [Casanova et al. 2014] simulation framework, and we considered a synthetic homogeneous platform constituted by 256 processors. We defined the size of the sets S and Q as 16 and 32, respectively. The jobs' characteristics were obtained from a trace generated with the Lublin and Feitelson [Lublin and Feitelson 2003] workload model. Finally, we decided to keep 256 thousand trials, as it showed to provide accurate calculations of $score(j)$.

Table 2. Scheduling policies used for comparison. Details about WFP3, UNICEF and F2 policies can be found in [Tang et al. 2009], and [Carastan-Santos and de Camargo 2017]

Policy name	Function
FCFS	r_j
SPT	$\tilde{p}_j$
SAF	$\tilde{p}_j \cdot q_j$
WFP3	$-(w_j/\tilde{p}_j)^3 \cdot q_j$
UNICEF	$-w_j/(\log_2(q_j) \cdot \tilde{p}_j)$
F2	$\sqrt{\tilde{p}_j} \cdot q_j + 2.56 \times 10^4 \cdot \log_{10}(r_j)$

The full list of coefficients θ obtained by regression approximation is illustrated in Table 3. The functional forms showed instabilities in the coefficients between the four obtained functions. We observe this instability by the change in the sign (e.g., from positive to negative) of the coefficients of a specific functional form. The sign change, in turn, carries a meaning for the schedule. For instance, while `Lin` prioritizes jobs with small q, since `Lin` has a positive coefficient for q, function `Qdr` prioritizes jobs with large q, since `Qdr` has a negative coefficient for q.

Table 3. Functional forms of the four parametrized functions used in multiple linear regression, their adjusted coefficients, and their Variance Inflation Factor (VIF).

Vector x	Coefficients θ				VIF			
	`Lin`	`Qdr`	`Cub`	`Qua`	`Lin`	`Qdr`	`Cub`	`Qua`
1	$3.16 \cdot 10^{-2}$	$3.86 \cdot 10^{-2}$	$3.02 \cdot 10^{-2}$	$4.58 \cdot 10^{-2}$	–	–	–	–
p	$1.24 \cdot 10^{-7}$	$1.33 \cdot 10^{-7}$	$3.92 \cdot 10^{-7}$	$-2.08 \cdot 10^{-7}$	1.3	3.7	12.3	40.8
q	$3.10 \cdot 10^{-5}$	$-2.95 \cdot 10^{-5}$	$1.37 \cdot 10^{-4}$	$-2.64 \cdot 10^{-4}$	1.3	9.4	37.3	104.1
r	$-1.62 \cdot 10^{-7}$	$-3.63 \cdot 10^{-7}$	$-5.72 \cdot 10^{-7}$	$-7.56 \cdot 10^{-7}$	1.2	5.0	19.9	56.5
p^2	–	$-2.10 \cdot 10^{-12}$	$-5.03 \cdot 10^{-12}$	$-1.43 \cdot 10^{-11}$	–	2.5	46.0	430.4
q^2	–	$7.65 \cdot 10^{-8}$	$-1.04 \cdot 10^{-6}$	$1.65 \cdot 10^{-6}$	–	8.1	267.4	2841.9
r^2	–	$2.64 \cdot 10^{-12}$	$7.44 \cdot 10^{-12}$	$1.45 \cdot 10^{-11}$	–	4.0	72.8	623.1
pq	–	$9.98 \cdot 10^{-10}$	$4.49 \cdot 10^{-9}$	$2.20 \cdot 10^{-8}$	–	3.8	125.4	544.3
p^3	–	–	$4.17 \cdot 10^{-17}$	$2.56 \cdot 10^{-16}$	–	–	26.0	1301.6
q^3	–	–	$2.76 \cdot 10^{-9}$	$8.72 \cdot 10^{-10}$	–	–	152.4	10583.4
r^3	–	–	$-3.07 \cdot 10^{-17}$	$-1.26 \cdot 10^{-16}$	–	–	31.6	1693.6
p^2q	–	–	$-1.29 \cdot 10^{-13}$	$5.57 \cdot 10^{-14}$	–	–	67.8	863.8
pq^2	–	–	$-2.72 \cdot 10^{-11}$	$-1.81 \cdot 10^{-10}$	–	–	174.5	2148.2
$(pq)^2$	–	–	$7.64 \cdot 10^{-16}$	$1.27 \cdot 10^{-16}$	–	–	88.2	1365.6
p^4	–	–	–	$-9.20 \cdot 10^{-22}$	–	–	–	448.8
q^4	–	–	–	$-1.33 \cdot 10^{-11}$	–	–	–	3585.3
r^4	–	–	–	$4.15 \cdot 10^{-22}$	–	–	–	529.0
p^3q	–	–	–	$-1.75 \cdot 10^{-18}$	–	–	–	109.9
pq^3	–	–	–	$3.71 \cdot 10^{-13}$	–	–	–	519.4
$(pq)^3$	–	–	–	$2.17 \cdot 10^{-23}$	–	–	–	176.8

In order to evaluate the scheduling performance of the four candidate policies, online scheduling experiments were performed. Throughout this section, we refer to the term *online scheduling experiment* as being multiple simulations of the online scheduling of jobs, whose information are obtained from a certain workload log (trace) or model. For each experiment, we choose a function from Tables 1 and 2 and perform the job scheduling using it as a scheduling heuristic.

In the first experiment, the jobs were generated from the Lubin & Feitelson's synthetic workload model (considering an HPC platform constituted by 256 processors). Figure 1 shows the average bounded slowdown for fifty dynamic scheduling experiments executed for each heuristic with the above configuration. The results show that the `Qdr`,

Cub and Qua functions – the most complex – actually did not perform as well as scheduling policies, presenting the worst results. Although the MAE values remained very close for almost all functions (Lin = 4.48×10^{-3}, Qdr = 4.66×10^{-3}, Cub = 10.5×10^{-3}, and Qua = 6.42×10^{-3}), the performance of the most complex functions showed to be as inefficient as FCFS, thus reinforcing the negative effects of its coefficients. The Lin function showed better results, being comparable to SAF, which is known to be quite efficient.

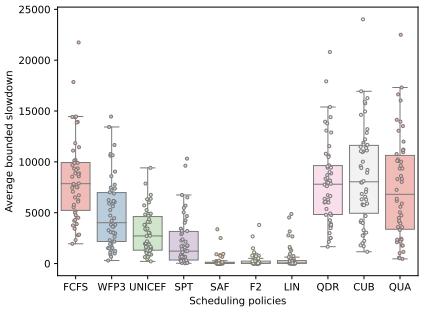

Figure 1. Scheduling performance comparison of each obtained function.

Table 4. Real workload traces used for evaluation of the scheduling policies.

Name	Year	# CPUs	# Jobs	Duration
Curie	2011	93,312	312,826	20 Months
ANL Interpid	2009	163,840	68,936	8 Months
SDSC Blue	2000	1,152	243,306	32 Months

This instability in the coefficients of Qdr, Cub and Qua indicates that these functions suffer from the phenomenon of multicollinearity [Alin 2010], that happens when the functional forms are highly correlated between each other. At this light, multicollinearity may be happening when we add derivative functional forms in conjunction to the jobs' characteristics p, q and r (e.g., q^2, q^3, q^4).

To verify this multicollinearity hypothesis, we calculated the Variance Inflation Factor [García et al. 2022] (VIF, see Table 3) for the functional forms present in the functions Lin, Qdr, Cub and Qua. The Qdr function displays VIF values greater than 5 (moderate correlation) as in q, r, q^2. Furthermore, the higher degree functions (Cub and Qua) presented very high values, reaching values greater than 10 thousand (extremely high correlation) as in the term q^3 of the function Qua. We conjecture therefore that adding more derivative functional forms will result to larger VIFs, and the multicollinearity effect will be present.

We then measure the ability of Lin (the function with the lowest level of multicollinearity) to generalize for different types of workloads, with jobs obtained from real workload traces of large scale HPC platforms, comparing it with the policies presented in Table 2. The used traces (see Table 4) are publicly available at Parallel Workloads Archive [Feitelson et al. 2014]. Moreover, we performed online scheduling experiments taking into account two situations, (i) scheduling using perfect information about the jobs processing time p, (ii) scheduling using jobs' processing time estimates $\tilde{p}$.

The results of the experiments are illustrated in Figure 2. The linear form presented low average bounded slowdown for the different workloads. In addition, the dis-

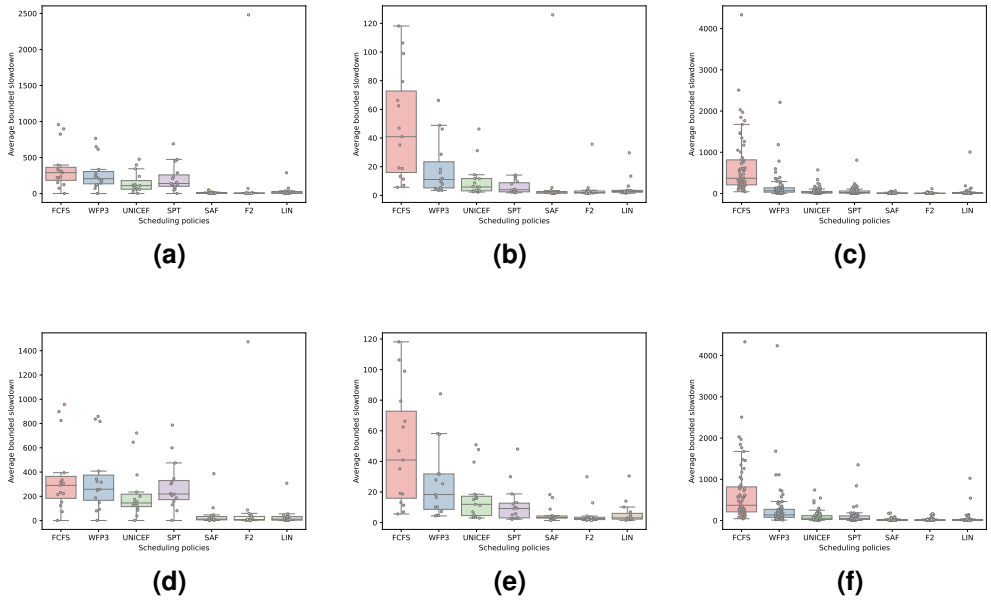

Figure 2. Figures (a), (b) and (c) illustrate the scheduling performance results using actual processing time for jobs from Curie, ANL Interpid and SDSC Blue platform workload logs, respectively. Following the same order, Figures (d), (e) and (f) illustrate the results using the estimated processing time.

persion of the data was small in almost all cases, ensuring greater stability and predictability. Unlike experiments (a), (b) and (c) – in which the actual processing time was used – experiments (d), (e) and (f) showed performance degradation. This is expected, since the estimated values are usually rough and inaccurate estimates. Finally, the `Lin` function showed one of the three best results for all experiments. Even though each workload trace was different, the linear function as a scheduling policy proved to be able to be generalized for different types of tasks.

6. Conclusions and Future Work

In this work, we explored a multiple regression method to understand the trade-offs between the simplicity and the performance of ML-obtained scheduling heuristics. We first created a data set of scheduling observations by performing scheduling simulations of different scheduling situations. We observed multicollinearity effect when using multiple linear regression that increased the instability in the coefficients of the obtained scheduling policies and resulted in poorly performing scheduling policies. We then evaluate the performance of the Lin policy and show that it presents good scheduling performances in the majority of the evaluated scenarios.

For future work, we intend to expand the scope of our investigations. First, we will look for more sophisticated methodologies to create scheduling heuristics, avoiding problems such as multicollinearity. Symbolic Regression, Neural Networks and Principal Component Analysis (PCA) are some candidates. Other possibility is to use platform-related features (e.g., platform utilization, remaining time of the processing jobs, etc.) and external factors (e.g., time of the day) in addition to p, q and r.

References

Alin, A. (2010). Multicollinearity: Multicollinearity. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(3):370–374.

Carastan-Santos, D. and de Camargo, R. Y. (2017). Obtaining dynamic scheduling policies with simulation and machine learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13, Denver Colorado. ACM.

Carastan-Santos, D., De Camargo, R. Y., Trystram, D., and Zrigui, S. (2019a). One Can Only Gain by Replacing EASY Backfilling: A Simple Scheduling Policies Case Study. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 1–10, Larnaca, Cyprus. IEEE.

Carastan-Santos, D., De Camargo, R. Y., Trystram, D., and Zrigui, S. (2019b). One can only gain by replacing easy backfilling: A simple scheduling policies case study. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 1–10.

Carroll, R. J. and Ruppert, D. (1988). *Transformation and weighting in regression*. Monographs on statistics and applied probability. Chapman and Hall, New York.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

Feitelson, D. G., Tsafrir, D., and Krakov, D. (2014). Experience with using the Parallel Workloads Archive. *Journal of Parallel and Distributed Computing*, 74(10):2967–2982.

García, C. G., Gómez, R. S., and Pérez, J. G. (2022). A review of ridge parameter selection: minimization of the mean squared error vs. mitigation of multicollinearity. *Communications in Statistics - Simulation and Computation*, 0(0):1–13.

Jack Dongarra and Erich Strohmaier (2022). TOP500 Supercomputer Sites.

Legrand, A., Trystram, D., and Zrigui, S. (2019). Adapting Batch Scheduling to Workload Characteristics: What Can We Expect From Online Learning? In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 686–695, Rio de Janeiro, Brazil. IEEE.

Lublin, U. and Feitelson, D. G. (2003). The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122.

Mu'alem, A. and Feitelson, D. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543.

Tang, W., Lan, Z., Desai, N., and Buettner, D. (2009). Fault-aware, utility-based job scheduling on BlueGene/P systems. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE.