JobPruner: A Machine Learning Assistant for Exploring Parameter Spaces in HPC Applications

Bruno Silva, Marco A. S. Netto, Renato L. F. Cunha
IBM Research

Abstract—High Performance Computing (HPC) applications are essential for scientists and engineers to create and understand models and their properties. These professionals depend on the execution of large sets of computational jobs that explore combinations of parameter values. Avoiding the execution of unnecessary jobs brings not only speed to these experiments, but also reductions in infrastructure usage—particularly important due to the shift of these applications to HPC cloud platforms. Our hypothesis is that data generated by these experiments can help users in identifying such jobs. To address this hypothesis we need to understand the similarity levels among multiple experiments necessary for job elimination decisions and the steps required to automate this process. In this paper we present a study and a machine learning-based tool called JobPruner to support parameter exploration in HPC experiments. The tool was evaluated with three real-world use cases from different domains including seismic analysis and agronomy. We observed the tool reduced 93% of jobs in a single experiment, while improving quality in most scenarios. In addition, reduction in job executions was possible even considering past experiments with low correlations.

Index Terms—HPC, Parametric Applications, Machine Learning, Search Space Exploration

1 Introduction

Big data and artificial intelligence are becoming crucial for the development of solutions in various fields including agronomy, health, aerospace, circuit design, astronomy, and oil & gas exploration. High Performance Computing (HPC) simulations used in these fields produce plenty of data that could be used to optimize new experiments. Such data is becoming even more pervasive due to the increasing efforts in improving reproducibility of computational experiments [1], [2], [3]. Natural questions are: what is the actual benefit of exploiting data from previously executed experiments and how to do that properly. This paper attempts to shed some light into answering these questions.

Ongoing efforts to reduce execution time of HPC experiments come mainly from (i) optimization of hardware and software layers that host user applications and (ii) use of Design of Experiments (DOE) [4] techniques to avoid running unnecessary jobs or prioritize jobs that provide the best "return of investment"—by doing so, users can run experiments in a more efficient manner with the same, or even a reduced, amount of computing power [5]. These reduction costs are particularly important due to the increasing efforts in moving HPC applications to cloud platforms [6], [7].

With the advances in the areas of big data and artificial intelligence, along with HPC platform improvements, mainly from the GPU arena, a complementary direction can be pursued to help users run their scientific experiments faster. Our goal here is to use data produced by previous experiments to help in new ones. We focus on the problem of reducing the number of jobs to explore parameters of user applications, which is a common process in scientific experiments when users have to calibrate models (e.g. crops, airplane wings, and circuit boards) or understand various what-if scenarios (e.g. soil properties, atmosphere conditions, and material properties). Users typically have to study a similar object/model but with different conditions/properties and use different strategies to make exploration vs exploitation decisions. One way to extract such a knowledge is via machine learning, where patterns can be found from previous executions that contain similar characteristics.

In this paper, we provide a step towards answering the question of how much knowledge can be leveraged from previous experiments to accelerate new ones. We analyze previous executions of real-world applications with similar search spaces to assess whether we can generate hints to users on which jobs can be prioritized or canceled on ongoing experiments. This analysis makes use of optimization and machine learning techniques. Our main contributions are therefore:

- JobPruner, a tool to help users explore parameter spaces in HPC experiments by building machine learning models from data obtained from previous executions (Section 3);
- Understanding of how reusing data from past executions affects results by evaluating three real HPC parametric applications from different fields, including seismic analysis and agriculture. The evaluation measures possible reductions in number of jobs when time for experimentation is limited and the impact on results considering different knowledge base sizes (Section 4).

Author preprint. Official Version at Future Generation Computer Systems (FGCS) DOI: 10.1016/j.future.2018.02.002 (Elsevier).

2 Background

Several research groups have conducted studies and developed technologies to assist users in their large scale HPC experiments. Related to our work are efforts from multiple communities, including computational steering, workflow management, design of experiment, optimization, and more recently big data and artificial intelligence. In this section, we describe some of these efforts and a formal description of the problem tackled in this paper.

2.1 Related Work

Computational steering [8], [9], [10], [11], [12] aims at providing users with tools for parameter reconfiguration of ongoing experiments. Parker and Johnson [13] introduced a system called SCIRun that uses a dataflow programming model and visual programming to simplify the tasks of creating, debugging, optimizing, and controlling complex scientific simulations. Van Wijk *et al.* [14] highlighted that the implementation of computational steering in practice is hard. To overcome this problem, they implemented an environment in which a data manager facilitates the interaction between user applications and steering components. Chin *et al.* [15] incorporated computational steering in mesoscale lattice Boltzmann simulations and showed the benefits of their work. They discussed that large scale simulations require not only computational resources but tools to manage these simulations and their produced results, what they called *simulation-analysis loop*. Netto *et al.* [16] introduced a scheduler system to automatically offer more resources to parametric application jobs based on the quality of their intermediate generated results so as users could get faster to their desired goal. More recently, Mattoso *et al.* [17] surveyed the use of steering in the context of HPC scientific workflows highlighting a tighter integration between the user and the underlying workflow execution system.

Workflow management systems [18], [19], [20], [21] are also relevant to help users in complex scientific experiments. Deelman *et al.* [19] developed a taxonomy of e-Science systems so scientists can assess the suitability of workflow systems to their experiments. Gil *et al.* [20] introduced Wings, a system that uses workflows to represent computational experiments and reasons about the experiment design space. In doing so, Wings can assist scientists in designing such experiments by tracking constraints and ruling out invalid designs.

Running complex experiments can always benefit from user input, therefore a community of researchers have been working on optimization assisted by humans [22], [23]. Meignan *et al.* [22] provided a detailed survey and taxonomy of efforts in interactive optimization applied to operations research. They explored the different roles a user can have in an optimization process, such as adjusting or adding new constraints and objective, helping on the optimization process itself, and guiding the optimization process by providing information related to decision variables. Meignan and Knust [23] proposed a system that employs user feedback on the optimization to use as long-term preferences for future runs. Nascimento and Eades [24] proposed a framework for humans to assist the optimization process via inserting domain knowledge, escaping from local minima, reducing the search space to be explored, and avoiding ambiguity for optimal multi-solutions. Abramson *et al.* [25] developed a system for users to parameterize arbitrary problems with the goal of optimizing a given objective function. Silva *et al.* [26] introduced a tool that incorporates user hints based on intermediate results to advise them about their strategies when running parametric applications tied to SLA constraints. Luszczek *et al.* [27] described a system to generate and prune search spaces for auto-tuning matrix multiplication kernels. Their solution is based on a declarative language to describe the search space alongside pruning constraints. Researchers have also explored visualization techniques to add humans to the optimization process [28], [29], [30], [31]. WorkWays [32], [33] is a science gateway with human-in-the-loop support for running and managing scientific workflows.

The area of Design of Experiments also has contributions to accelerate the evaluation of search spaces. Kleijnen *et al.* [4] developed a survey and a user guide on advances in this area until 2005, including benefits and drawbacks of various designs. Sanchez and Wan [5] also discussed the benefits of design of experiments to avoid brute-force approaches. They also presented a tutorial on how to apply experimental design to simulation experiments. Using the fractional factorial design technique, Abramson *et al.* [34] developed a system to facilitate parameter exploration and abstract the underlying computing platform.

The execution of large number of independent jobs may be complex to users, mainly when distributed resources are used to run them. Resources may have different computational power and availability and may have different mechanisms to be accessed. Therefore, over the years, several tools have been created such as Condor [35], XtremWeb [36], BOINC [37], Nimrod [38], and OurGrid [39] to facilitate user access to these computational platforms. These tools mainly focus on managing users jobs looking into the computational infrastructure. Our tool aims at looking into the user workload and finding patterns from past experiments, which allow users to drastically reduce their search spaces when performing experiments of similar nature.

With advances in artificial intelligence [40], [41] and big data [42], researchers are studying how to apply technologies from these areas to optimize the execution of experiments. For instance, in the data mining domain, Padillo *et al.* [43] proposed exhaustive search algorithms for subgroup discovery: identifying relations between a target variable and independent variables. Their algorithms can prune search spaces and find relationships in massive datasets. Weiss *et al.* [44] built upon the survey conducted by Pan and Yang [45] and investigated papers and software motived by the improvement of target predictive functions from one domain by using predictive functions learned for other domains (i.e. *transfer learning*).

Our research is based on existing efforts in optimization, design of experiments, and machine learning in order to build a tool for reducing the execution time of HPC experiments. Our tool is based on the hypothesis that it is possible to reuse knowledge from previously executed experiments to run new ones. To the best of our knowledge, there is no previous work that leverages previous HPC application experiments to prune search space of new ones.

2.2 Problem Description

Problems from several industries are tackled by users running complex computer simulations and optimizers that constitute a *software* with a set of *parameters*, where each parameter can receive different *values*. The optimization process for a parametric application is named *experiment*, and each individual software execution is a *job*. These jobs can be computational intensive, which may require High Performance Computing (HPC) platforms not only due to processing needs but also due to memory constraints. Although jobs run independently, which is a characteristic known in High Throughput Computing (HTC) applications, each job can run on a multi-core machine or even be distributed in a computing cluster as long as the instruction to do so is available to our tool.

We consider applications with n parameters and each parameter has a finite discrete domain $\mathcal{D}_i$ where $i \in \{1, \cdots, n\}$. A function $f : \mathcal{D}^n \rightarrow \mathbb{R}$ is adopted to evaluate the quality of parameter values and is specific for each parametric application. The function may also be subjected to constraints ($t_j \forall j \in \{1, \ldots, m\}$, $u_k \forall k \in \{1, \ldots, p\}$) depending on the problem characteristics. The considered optimization problem P is described as follows:

$$\underset{\vec{x} \in \mathcal{D}^n}{\text{maximize}} \quad f(\vec{x}) \tag{1}$$

$$\text{subject to} \quad t_j(\vec{x}) \leq 0, \quad j = 1, \ldots, m \tag{2}$$

$$u_k(\vec{x}) = 0, \quad k = 1, \ldots, p \tag{3}$$

We assume the user is able to generate $f(\vec{x})$ from the job output. For instance, if the output $g(\vec{x})$ of a simulation process is a real value that should be minimized, then $f(\vec{x})$ can be assigned as $f(\vec{x}) = -g(\vec{x})$ and the problem definition remains the same. If the parametric application a generates multiple outputs (e.g., $a : \mathcal{D}^n \rightarrow \mathcal{O}^m$), a wrapper function $h : \mathcal{O}^m \rightarrow \mathbb{R}$ should be employed to represent the behavior of f (i.e., $f(\vec{x}) = h(a(\vec{x}))$).

For instance, one of our case study applications (Section 4) predicts the production of multiple crops from a single region. We use a wrapper function that sums the crop yields to generate a single objective function. Multiple objective optimization is out of the scope of this paper.

Running all possible values for each parameter can be unfeasible—even with large HPC machines. Therefore, users need strategies to select a subset of values for each parameter that covers parts of the exploration space and gives them enough information to answer their questions. For instance, the user may want to know what is the best set of parameter values that provides a close-to-optimal solution. The strategies for parameter-value selections can be defined by Derivative Free Optimization (DFO) methods [46], which correspond to a subject of mathematical optimization where the derivative information is unavailable or impractical to be obtained.

An effective approach to speed up the solution finding in optimization problems corresponds to the reduction of the search space by pruning the range of parameter values or fixing a parameter with a given value. Often users use their background from previous experiments to determine the parameter ranges and/or prune the search space to get faster results. There has been little progress on employing machine learning techniques for reducing search space in HPC experiments.

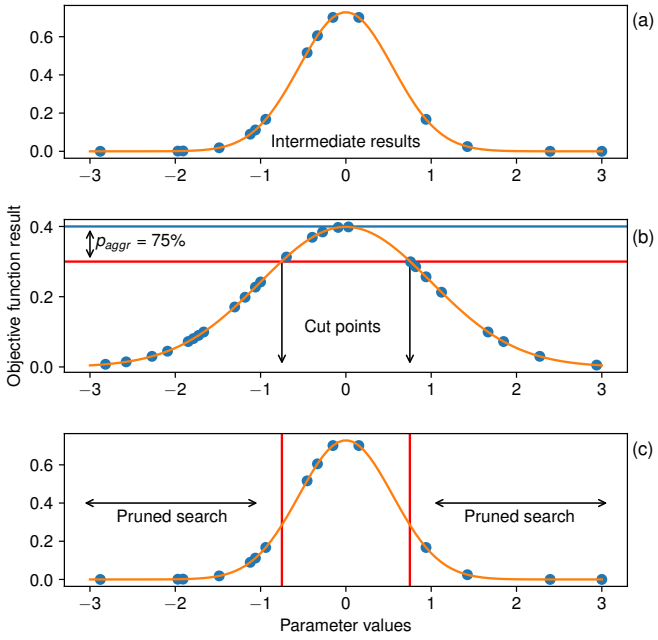


Fig. 1: Motivating example of search space reduction using previous experiments. In (a) we see a function that will be optimized. In (b) we see a function that is used as a template of previous evaluations. In (c) we see the function from (a) but with its space pruned based on information from (b).

3 System and Algorithms

This section describes JobPruner, the proposed solution for eliminating non-promising jobs of HPC experiments. JobPruner is part of a larger system, Copper [26], aimed at calibrating models and evaluating what-if scenarios by providing an integrated interface for executing experiments and using machine learning for providing users with insights. JobPruner focuses on improving the quality of experiments with restricted resources by using knowledge from already executed experiments to reduce the solution search space.

3.1 Motivating Example

Suppose one wants to maximize a function of a model. Assume the value output by the model depends only on a single independent variable instantiated in a given domain and also that the only way to obtain the output is by executing the model. Also assume this model has been executed for a number of times in different experiments and we are now given the task of evaluating this model in a new setting. For instance, if a crop model for one farm is under evaluation, we use data from previous evaluations on other farms to obtain insights about the current experiment. Figure 1 displays an example in which we want to maximize some function.

The pruning strategy works as follows: initially, the model is evaluated several times and a surrogate function

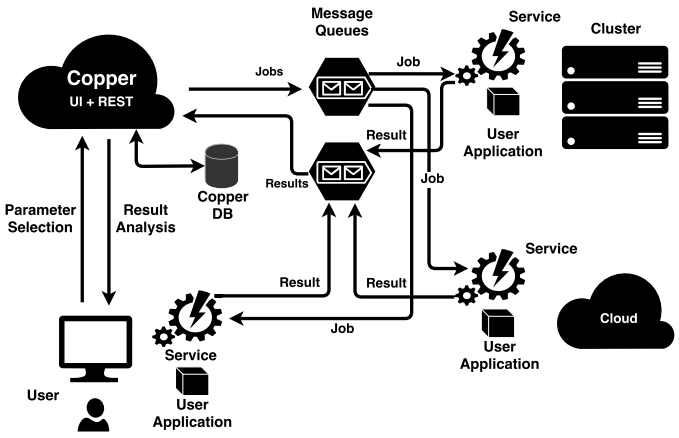


Fig. 2: Copper architecture

is fit to the obtained results (Figure 1a). Then, the newly-obtained surrogate is compared to all the previous surrogates found for this model in previous experiments. After the search, we select the previous experiment that has the surrogate closest to the one just obtained. That way, we can exploit the fact of having more evaluations of previous experiments to possibly avoid executing non-promising jobs.

In Figure 1b, we see such a domain. In our tool, we employ a pruning aggressiveness factor, p_{aggr}, that acts as a cutoff parameter: we discard points that generated values smaller than $p_{aggr} \times \max$, where $\max$ is the maximum value found in the previous experiments. In this example, we use $p_{aggr} = 75\%$ and the maximum value found in that experiment was 0.4. Therefore, points with value less than $75\% \times 0.4 = 0.3$ are discarded.

The new search space corresponds to parameter values that were not discarded, represented by the points above the red line in Figure 1b. By using a redefined search space, we may find better results faster for the current experiment, as no computational resources are applied in regions that may lead to non-promising results. Figure 1c shows the final region to be explored, where only the central region of the search space should be evaluated.

3.2 Copper

Figure 2 presents an overview of Copper, which is a cloud service for exploring search spaces in HPC parametric applications. By using Copper, users can prioritize or discard jobs to speedup the evaluation of their applications. The infrastructure is composed of a database (Copper DB) to store jobs results and their respective parameters, a backend (UI + REST), two message queues (Jobs and Results) and a Service for each user application.

The workflow for utilization of copper is the following. First, users select a batch of jobs to be executed, each job contains a single combination of parameters and may return a result (Section 2.2). Jobs are stored in Copper DB and sent to Jobs message queue (job id and respective parameters). Once user jobs are in the message queue, a service retrieves them for execution. Services can be located in different environments for job execution, including cluster, cloud or even the user personal machine. Once the job finishes, the results are returned to the users via Results message queue. For more details about Copper the reader is referred to [26].

3.3 JobPruner

Figure 3 shows the interaction between Copper and JobPruner and its components. The overall flow between Copper and JobPruner is the following:

1) **Problem description:** having an application registered in Copper, the user specifies a list of parameters to be evaluated, the range of variables each parameter can assume, constraints for time or number of jobs to complete an experiment, an application, and an optimization metric;
2) **Optimizer:** With the input provided by the user, the optimizer chooses a set of parameters to be used to evaluate a model input by the user. Copper utilizes DFO strategies such as generalized greedy randomized adaptive search (GRASP), general pattern search (GPS), particle swarm optimization (PSO), and simulated annealing (SA) [47], [46];
3) **User Application:** The set of parameters selected by the optimizer is then used for execution of the model. As new results are generated, the Optimizer selects a new batch of parameters to evaluate. Also, as intermediate results become available, the Optimizer communicates with JobPruner to potentially prune the search space;
4) **Stop:** The interaction between Optimizer and User Application continues until one stop criterion is met. Examples of criteria are: exceeding the maximum budget of jobs run, exceeding the time allocated for optimization, or successfully optimizing the model.

Inside JobPruner, the following steps are executed:

1) **Surrogate Generator:** Copper sends intermediate results to JobPruner to verify if the search space can be reduced. The entry point of JobPruner is the Surrogate Generator, which is responsible for creating a function based on available results of the ongoing experiment. This intermediate surrogate is stored in the Previous Experiments database and is refined each time the surrogate is updated with information from new jobs executed. JobPruner includes the following surrogate generation algorithms: linear, cubic, spline interpolations, and k-nearest neighbors regression. For this work we adopted k-nearest neighbors regression to estimate points that were not evaluated. Other machine learning approaches can be used without loss of generality (e.g., neural networks);
2) **Compare with Previous Experiments:** With a surrogate function, it is possible to search the previous experiments database (also known as *knowledge base*) for similar surrogates. If a similar surrogate is found, it can be used to reduce the ongoing experiment search space size.

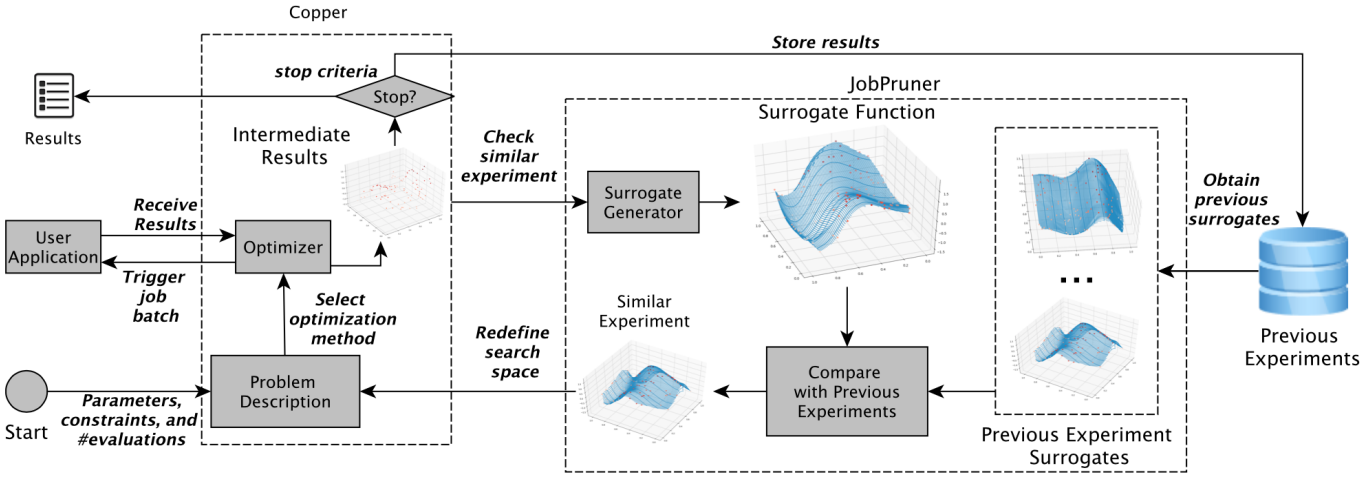


Fig. 3: Overview of the proposed pruning approach.

Next we describe how a previous experiment is selected for the pruning process which is the functionally of JobPruner.

3.4 Previous Experiment Selection

JobPruner utilizes the most similar surrogate from previous experiment database to prune the search space. In order to compare two surrogates, we employ normalized cross correlation [48], a technique from the pattern recognition and machine learning area, which is useful when surrogates with different output ranges are compared. This technique normalizes the outputs by subtracting each output point by the mean and dividing by the standard deviation. As the data is normalized, it is possible to evaluate experiments with different output amplitudes focusing the comparison on data shapes not in specific values. The normalized cross-correlation (n_{corr}) between two functions $f(\vec{x})$ and $p(\vec{x})$, with $\vec{x} \in \mathcal{D}^n$, is described as follows:

$$n_{corr} = \frac{1}{s} \sum_{\vec{x}_i} \frac{(f(\vec{x}_i) - \bar{f})(p(\vec{x}_i) - \bar{p})}{\sigma_f \sigma_g},$$

where s is the number of samples, $\bar{f}$ and $\bar{p}$ are the surrogate mean values of the current experiment and the previous experiment, respectively, σ_f and σ_g are the surrogate standard deviations for f and p. Then, we use the previous experiment with higher n_{corr} to prune the current experiment search space, if n_{corr} is higher than a user-define threshold.

3.5 Search Space Pruning Algorithm

Algorithm 1 is responsible for pruning the search space based on previous experiments. The algorithm receives the previous experiment results set ($\mathcal{P} \subseteq \mathcal{D}^n \rightarrow \mathbb{R}$), the current experiment search space ($\mathcal{C} \subseteq \mathcal{D}^n$), and the prune aggressiveness ($p_{aggr} \in \mathbb{R}$) as parameters. The model output corresponds to a potentially smaller new search space ($\mathcal{N} \subseteq \mathcal{C}$).

Initially, $\mathcal{N}$ receives the search space for the current experiment (Line 1). In Line 2, variable cut_val is created to establish a threshold for parameters that lead to non-promising results. Lines 3-13 loop over each parameter

Algorithm 1: *prune*, an algorithm for pruning the search space.

```
   Input  : P, C, p_aggr
   Output: N
1  N ← C;
2  cut_val ← p_aggr × max(P);
3  foreach D_i of N do
4  |  D_t ← D_i;
5  |  foreach val ∈ D_i do
6  |  |  jobs ← {j ∈ P | pvalue(j, i) = val};
7  |  |  r ← ⊤;
8  |  |  foreach job ∈ jobs do
9  |  |  |  if eval(job) ≥ cut_val then
10 |  |  |  |  r ← ⊥;
11 |  |  if r then
12 |  |  |  D_t ← D_t − {val};
13 |  D_i ← D_t;
14 return N;
```

$\mathcal{D}_i \in \mathcal{N}$ to remove parameter values that generate non-promising results. In Line 4, a temporary set $\mathcal{D}_t$ copies each parameter $\mathcal{D}_i$. For each parameter value $val \in \mathcal{D}_i$, we create a variable $jobs$ that corresponds to all jobs from previous experiment $\mathcal{P}$ that use parameter $\mathcal{D}_i$ with value val (Line 6). $\text{pvalue}(j \in \mathcal{P}, i \in \mathbb{N})$ returns the i-th parameter value for job j. If all values of $job \in jobs$ lead to results below the cut_val (Lines 7-12), val is removed from $\mathcal{D}_t$ (Line 12). Otherwise, $\mathcal{D}_t$ remains with the same values. Parameter $\mathcal{D}_i$ is updated with the new parameter values set in Line 13. Finally, the new search space is returned in Line 14. JobPruner executes Algorithm 1 after each job batch, and updates the current optimization method constraints to generate new jobs only in the newly-obtained search space region.

3.6 Automatic p_{aggr} Generation

JobPruner can automatically determine prune aggressiveness (p_{aggr}) values based on previous experiment data. Higher values of p_{aggr} result in larger cuts in the search space. Therefore, if p_{aggr} gets closer to 1, the probability

that an optimal solution will be excluded is higher. Whereas, if JobPruner uses lower p_{aggr} values, it cannot help users speed up their experiments by eliminating non-promising jobs.

Let $\mathcal{Z}$ be the set of previous experiment outputs (Section 3.3). We measure the spatial continuity of $\mathcal{Z}$ using experimental variograms [49]. Experimental variograms are defined as one-half of the average squared difference between each pair of points in $\mathcal{Z}$ with a distance h, also known as lag. Formally, we define the experimental variogram $v_r(h)$ as:

$$v_r(h) = \frac{1}{2n(h)} \sum_{i=1}^{n(h)} \left[z(x_i + h) - z(x_i)\right]^2,$$

where $z(x)$ corresponds to the data output taken at location x, $z(x+h)$ is the measurement taken at the point $x+h$, and $n(h)$ the number of data pairs that are h units apart from each other.

An important metric related to experimental variogram analysis corresponds to the variogram *nugget*, which is the value of $v_r(h=0)$. Theoretically, the nugget should be zero, however spatial sources of variation at distances smaller than the sampling interval leads to positive nuggets [49]. It can be used as an indicator of how overall data is spatially correlated for small distances [50]. Another metric is the *sill* that defines the variogram limit value which is given by $\lim_{h\to\infty} v_r(h) = variance(\mathcal{Z})$. If we divide the nugget by the sill, we obtain the proportion of data discontinuity for small lags, then the suggested prune aggressiveness (s_{aggr}) is given by:

$$s_{aggr} = 1 - v_r(0)/variance(\mathcal{Z})$$

which indicates how continuous the data is for small lags. In order to use variograms, $\mathcal{Z}$ should be normally distributed and presents spatially constant mean and variance. JobPruner checks these conditions to make s_{aggr} suggestions, if $\mathcal{Z}$ does not have these properties then a fixed value (e.g., 0.6) is proposed and the user is notified. We also adopt a user-defined superior limit for s_{aggr} to avoid a too aggressive prune.

4 Evaluation

In this section, we evaluate JobPruner to eliminate non-promising jobs in HPC experiments by using three real applications (Section 3). Our hypothesis is that we can use data from previous experiments to help on new ones, and we are interested in understanding the impact of pruning aggressiveness and knowledge base size on the quality of the produced results.

4.1 Experiment Setup

We split our evaluation in two sets of experiments using two DFO methods each (PSO and SA). The first considers the effects of the pruning aggressiveness on the search space reduction and the best solution found (Section 4.2). The second helps us understand the impact of the knowledge base size on the quality of the results (Section 4.3).

For all experiments, the limit of function evaluations corresponds to 10% of the search space size. The following metrics are analyzed: (i) difference between the best-found solution and the global optimum, (ii) search space pruning size. We performed all experiments using a 15-node Xeon E5-2680v2 cluster, each node with 20 cores and 128 GB of RAM.

Applications. We selected three applications from different domains and with different search space characteristics. Following is a short description of the applications with an overview of their input parameters and outputs users are interested in.

1. Seismic: A seismic image corresponds to a representation of sound waves through underground rock structures. By using these images, seismologists can estimate the shape and depth of gas and crude oil geological formations. To assist users in this task, we use an in-house application (Seismic) to analyze seismic images using machine learning. The tool has several parameters including a visual descriptor (e.g., local binary patterns) and a classifier (e.g., k-nearest neighbors) to estimate the composition of the different regions of a seismic image. Visual descriptors and classifiers have parameters to be tuned to get proper geological formation estimations. To evaluate the quality of a selection of feature extractor and classifier, the tool employs a database storing reservoir annotated images with a description of each image component (rock layers) for a specific field. Classification performance varies according to the selected feature extractor and classifier and is in the interval [0, 1], where 0 means the classifier gave wrong predictions for all images in validation set and 1 when all images were classified correctly. The user's goal is to select a visual descriptor from a set of three, a classifier also from a set of three, and their parameters in order to maximize the classification accuracy of a field seismic image. For each field evaluation (experiment), there are 5502 combinations of classifier and extractor configurations.

2. AgroAnalytics: AgroAnalytics is a tool to help farmers know how they should fertilize their crops. The tool is implemented on top of PCSE (Python Crop Simulation Environment)[1] which corresponds to a python implementation of WOrld FOod STudies (WOFOST) model [51]. As performance metric, we used the yield of winter wheat crops from farms located in the Anhui Province in China. The crops are divided in 50km $\times$ 50km grids and they have their own soil and weather characteristics. The objective of this case study is to maximize the production of the land by choosing how to fertilize the soil. Here the user can select one of four fertilizer configurations in five crop stages. Hence, we may have $4^5 = 1024$ fertilizing configurations for each crop.

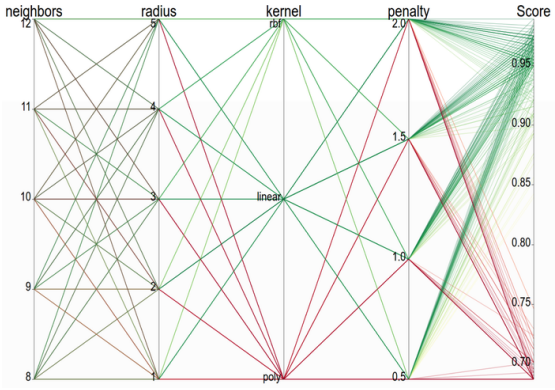
3. SchedSim: Scheduler Simulator (SchedSim) is a tool to assist system administrators to create management policies for HPC clusters. It accepts a variety of parameters including number of processors in the cluster, partitions, and scheduling algorithms. Tuning the scheduler and cluster properties to meet client business goals is not a trivial task and several scenarios must be executed to achieve that. For the evaluation, we had the following input: (i) a workload containing historical data of jobs submitted to a cluster and

1. PCSE: `http://pcse.readthedocs.io/en/stable/`

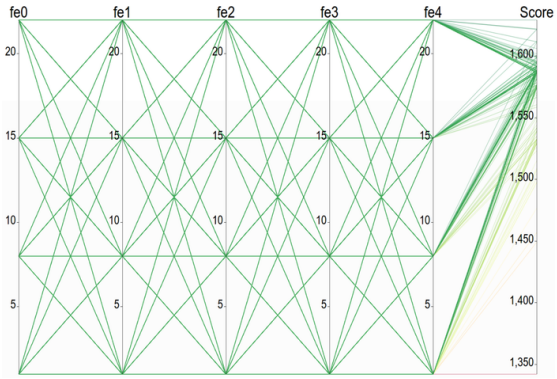
(ii) five queues, each with a configuration of user requested allocation time. For instance, a queue can only accept jobs with 15 minutes of requested time, whereas another can accept only jobs that take between 2 to 3 hours. We created an array of 13 possibilities of time configurations, thus generating 495 jobs for each experiment (that is, a workload analysis). Having five queues, we used four variables to determine the index in this array of possibilities to generate the ranges of requested times for the five queues. We used a fair share algorithm, given equal share to each queue to focus our analysis only on the parameters of the requested time per queue. The workloads came from a mix of 6-month portions of five clusters from the Parallel Workloads Archive [52] (HPC2N, SDSC-SP2, KTH-SP2, SDSC-BLUE, and SDSC-DS)[2]. The goal in this use case is to know what configuration each queue should have in order to minimize the average slowdown of all jobs in a given workload.

Figure 4 shows parallel coordinate graphs representing subsets of the search spaces of the applications. We used 20 evaluation subjects (i.e. an oil & gas field, a farm, and a cluster workload) for each application. We assessed the pruning strategy for each of the 20 subjects using the other 19 as part of the knowledge base. For instance, in the evaluation of crop number 15, we used subjects $[1, 14] \cup [16, 20]$ as previous experiments. For each application, we evaluated the absolute difference between the global optimum and the best-found solution using JobPruner or not. Moreover, the search space reduction is assessed for each pruned subject. We employed Particle Swarm Optimization (PSO) and Simulated Annealing (SA) [46] as optimization methods. Due to their stochastic nature, PSO and SA lead to different results depending on their initial conditions. We repeat the same experiment 200 times to find a 95% confidence interval of the studied metrics.

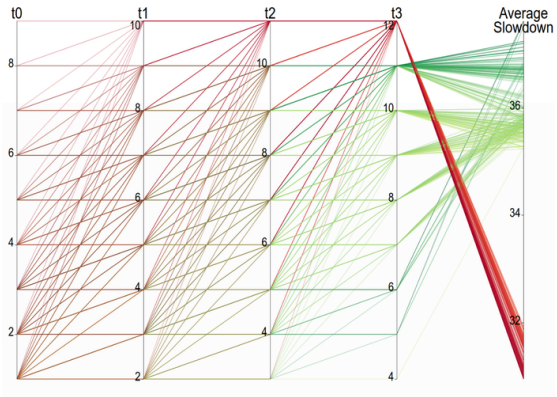
Heterogeneity of the experiments. For our analysis, it is relevant to understand how similar the experiments are among themselves. Within a use case, for each experiment we calculated its normalized cross-correlation (Section 3.4) against each of the other 19 experiments considering the evaluation of the entire search space. For each comparison with all other experiments, we selected the best correlation and saved it. The following values are the average of the best correlations: 0.8198, 0.8907, and 0.5756 for Seismic, AgroAnalytics, and SchedSim respectively. Therefore, we have three real world case studies with different levels of similarity.



(a) Seismic.



(b) AgroAnalytics.



(c) SchedSim.

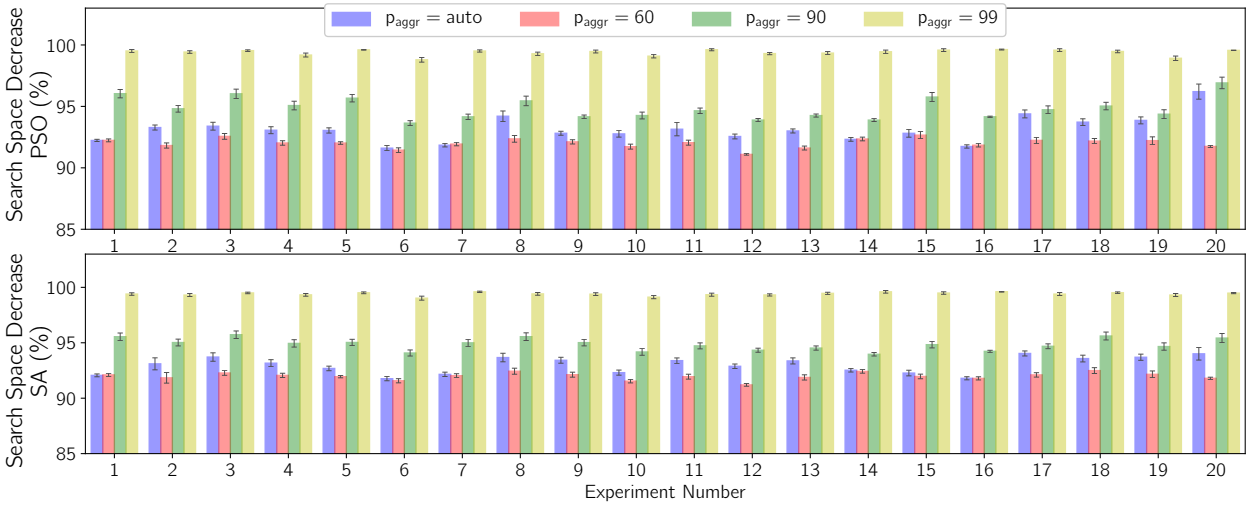
Fig. 4: Search space examples.

4.2 Results: Pruning Aggressiveness Analysis

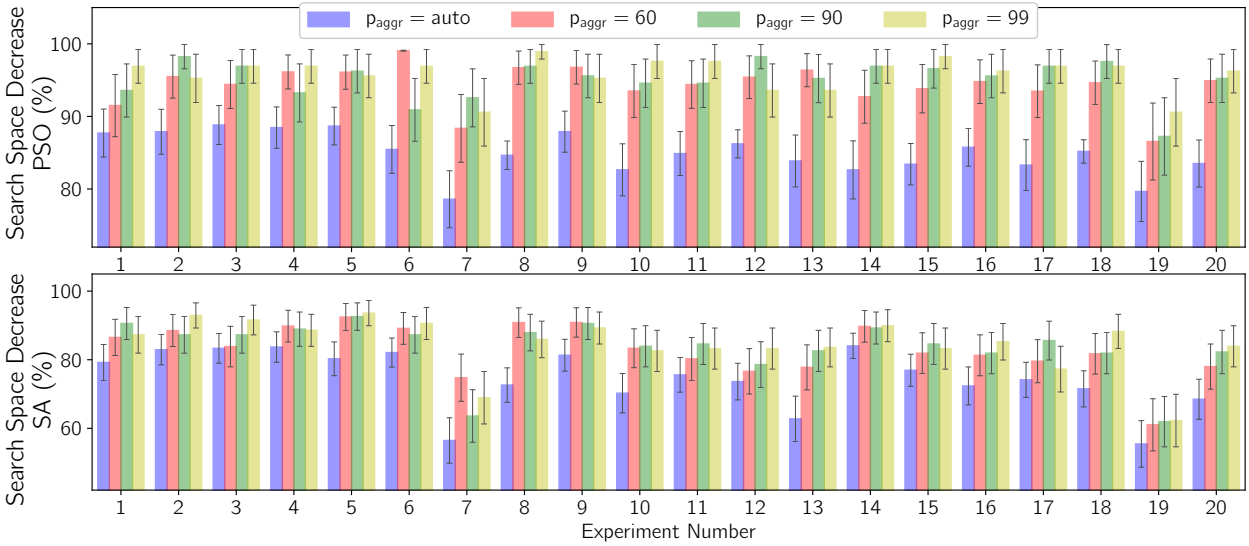
Before analyzing the impact of the pruning on the quality of the results, it is important to understand how much pruning was done depending on p_{aggr}. For each set of experiments, we used the following values for p_{aggr}: 60%, 90%, 99%, and auto, which means the p_{aggr} was provided automatically by JobPruner (Section 3.6). Figure 5 illustrates the search space reduction based on p_{aggr} for all use cases. The behaviour of the three use cases is different, which enables us to have a more comprehensive understanding of the pruning using past experiments. The impact of p_{aggr} on pruning among the experiments depends on the shape of the search space and the correlation between experiments. Seismic and AgroAnalytics have a more steady impact of pruning on theirs experiments, whereas SchedSim is more heterogeneous and presents less cuts on search space. The search space decrease is similar when comparing PSO and SA methods, which suggests that the optimization method has little impact on pruning.

Figures 6–8 present a comparison of the percentage difference between the optimal result for each application subject (experiment number) using the standard optimiza-

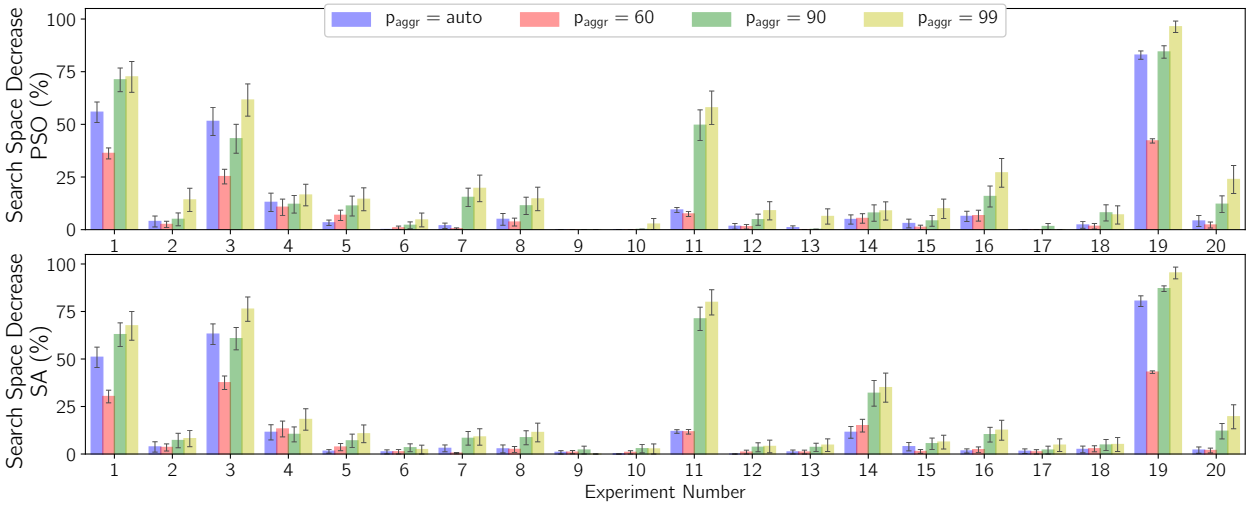
2. Parallel Workloads Archive: `http://www.cs.huji.ac.il/labs/parallel/workload`



(a) Seismic



(b) AgroAnalytics



(c) SchedSim

Fig. 5: Search space decrease based on pruning aggressiveness.

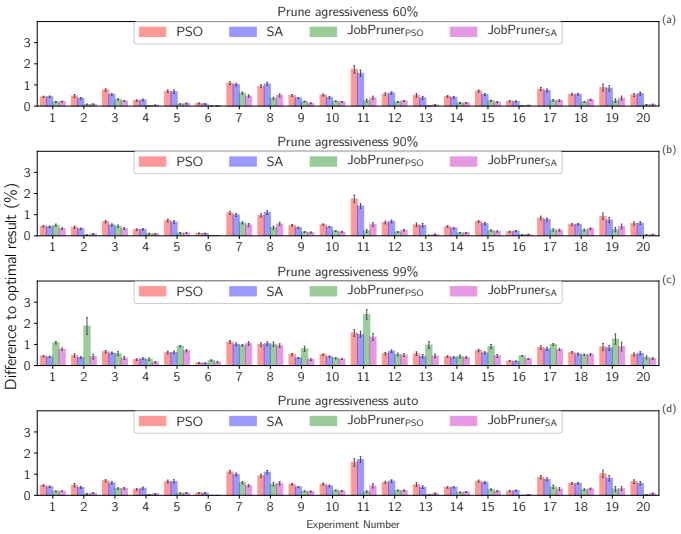


Fig. 6: Result quality as a function of pruning aggressiveness for Seismic.

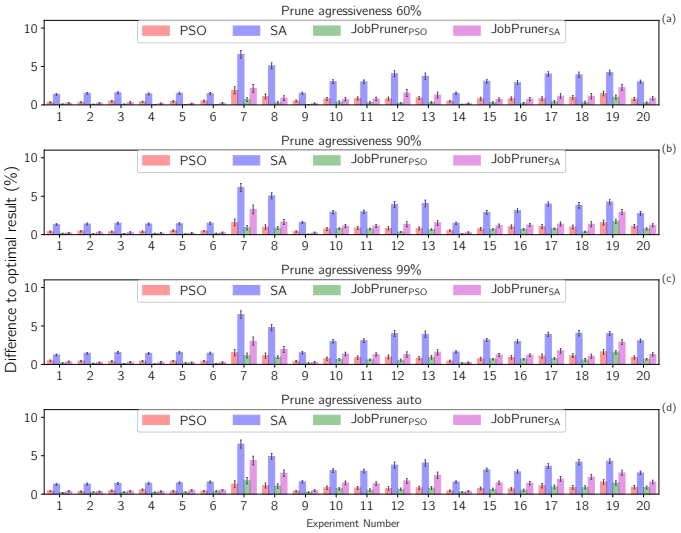


Fig. 7: Result quality as a function of pruning aggressiveness for AgroAnalytics.

tion methods PSO and SA and the same optimization methods with search space prunes (JobPruner$_{PSO}$ and JobPruner$_{SA}$). The big picture of the graphs is that the higher the pruning aggressiveness the closer to optimal the results are. In general, in the worst case, pruning gets similar results to PSO/SA without pruning. Higher positive impact of pruning comes when more jobs that are far from the optimal region of the search space are eliminated, thus saving resources to more promising jobs. The cases when PSO/SA and JobPruner are similar happen when search spaces have a spiky shape and it is not possible to create surrogates that can clearly identify jobs to be pruned. Rare cases of pruning producing worse results than pure PSO also happen. This is the scenario when regions containing optimal values are not identified by jobs that were executed to identify surrogates from the knowledge base. In this case, such regions may be eliminated if pruning is too aggressive.

To exemplify, for Seismic using PSO, experiments with

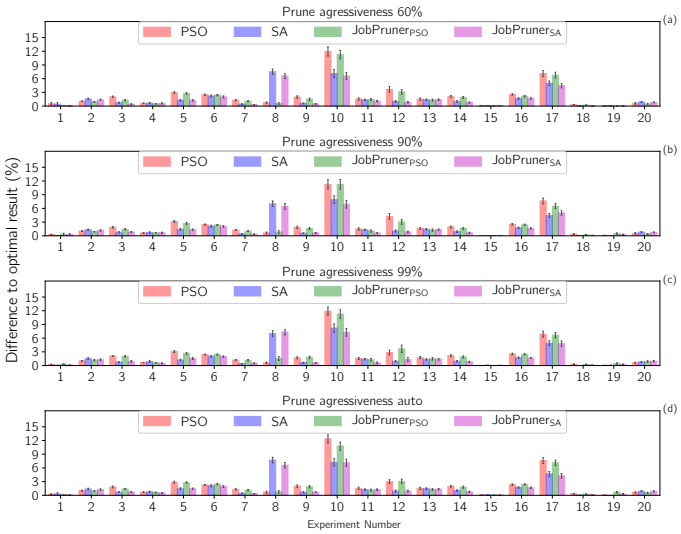


Fig. 8: Result quality as a function of pruning aggressiveness for SchedSim.

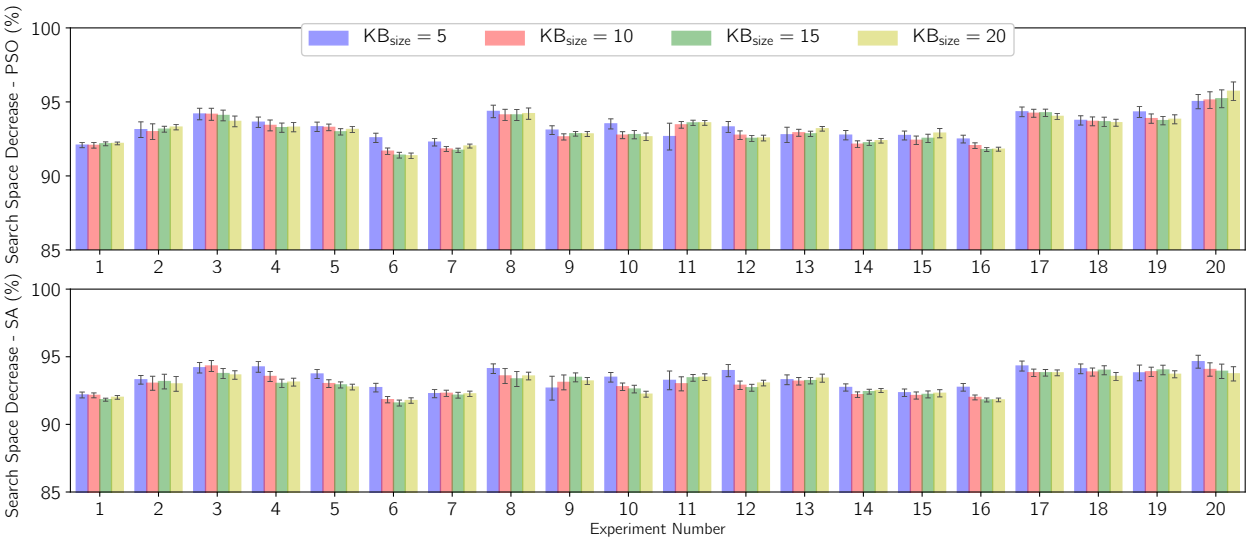
$p_{aggr} = 60\%$, $p_{aggr} = 90\%$ the difference between the best-found result and the global optimum decreases. However, for some experiments with $p_{aggr} = 99\%$ (e.g., 2, 11, and 13) the difference increases when compared to $p_{aggr} = 90\%$. This behaviour can be explained by analyzing the search space reduction in each experiment as illustrated in Figures 10–12. The search space reduces as long as the p_{aggr} values increase. For $p_{aggr} = 99\%$ the search space reduction gets close to 100% which may remove the global optimal from the search space. Note that the pruning process is based on previous results and not necessarily the optimal result of one model will match the optimal of the current model. Table 1 summarizes the results, observe that experiments with automatic generation of p_{aggr} provide results similar to the best selected value of p_{aggr}.

TABLE 1: Result quality (intervals) based on pruning aggressiveness.

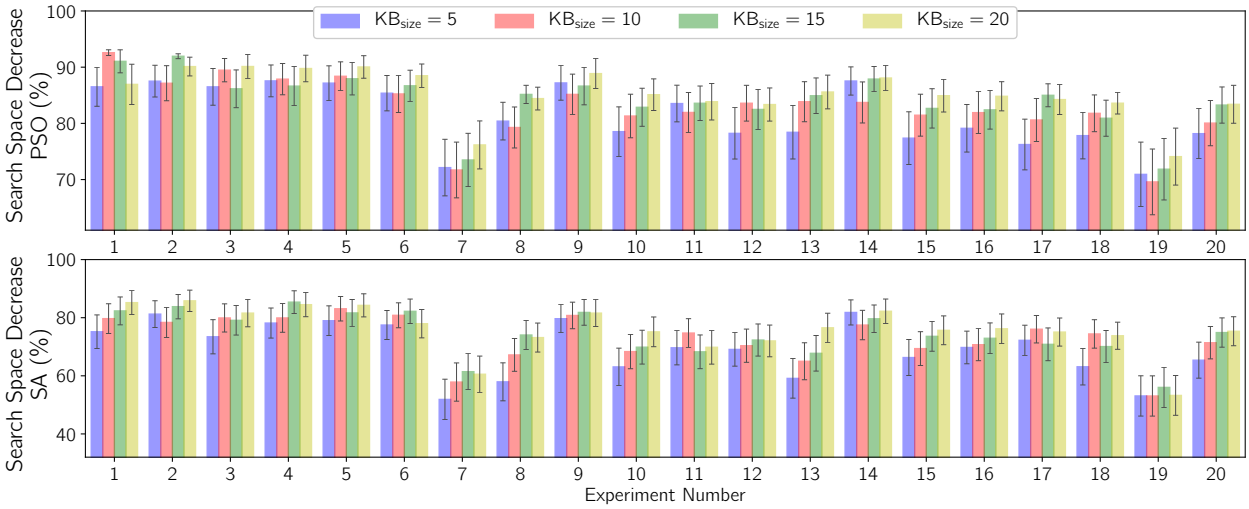
Opt	p_{aggr}	% diff (Seismic)	% diff (AgroAnalytics)	% diff (SchedSim)
PSO	0	[0.571, 0.705]	[0.580, 0.970]	[2.181, 2.212]
	60	[0.158, 0.219]	[0.145, 0.359]	[1.943, 1.979]
	90	[0.179, 0.257]	[0.395, 0.609]	[1.941, 2.030]
	99	[0.745, 0.947]	[0.415, 0.658]	[2.120, 2.224]
	auto	[0.168, 0.239]	[0.481, 0.793]	[2.006, 2.044]
SA	0	[0.530, 0.652]	[2.653, 3.148]	[1.675, 1.752]
	60	[0.168, 0.238]	[0.558, 1.024]	[1.478, 1.552]
	90	[0.198, 0.282]	[0.848, 1.279]	[1.517, 1.627]
	99	[0.484, 0.630]	[0.884, 1.320]	[1.605, 1.736]
	auto	[0.178, 0.256]	[1.198, 1.677]	[1.538, 1.628]

4.3 Results: Previous Experiments Database Size Analysis

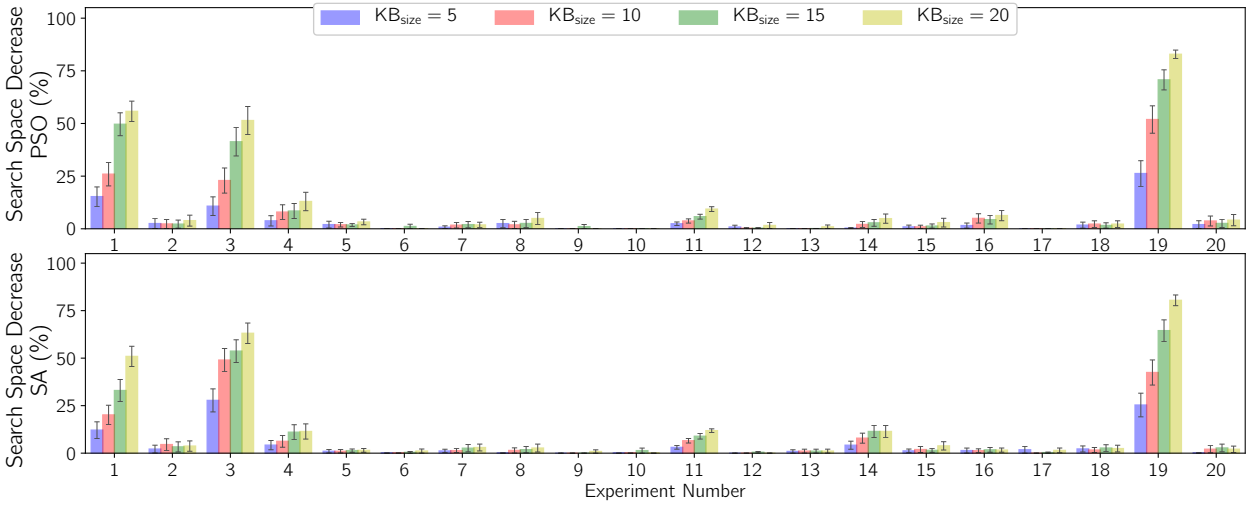
An important aspect to be evaluated is the impact of knowledge base size on the search space reduction. It is intuitive that the larger the knowledge base the more possibilities for pruning JobPruner has. However, as illustrated in Figure 9, use cases have different behaviour depending on the knowledge base size. In the following study we changed the



(a) Seismic



(b) AgroAnalytics



(c) SchedSim

Fig. 9: Search space decrease for different previous experiments knowledge base sizes.

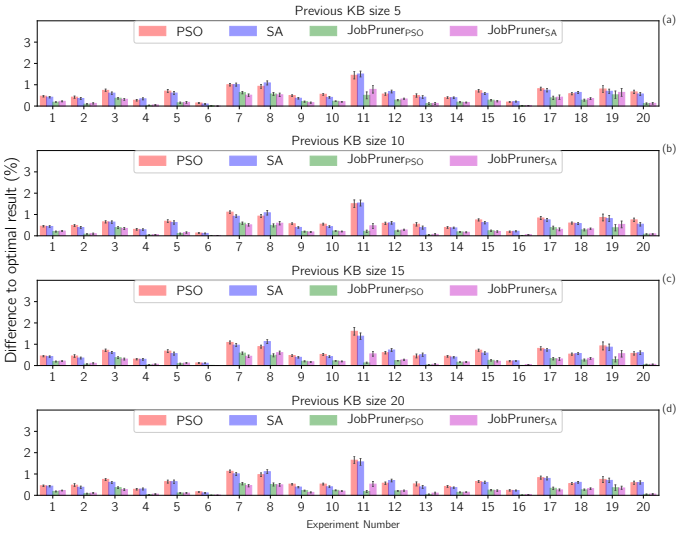


Fig. 10: Result quality as a function of the knowledge base size for Seismic.

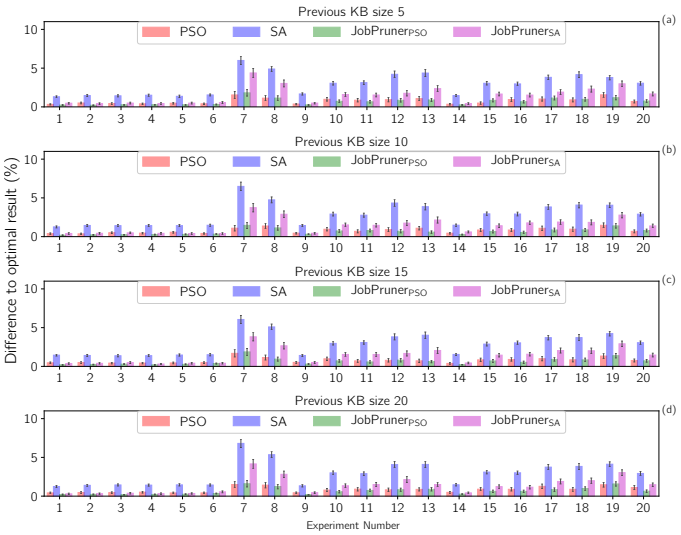


Fig. 11: Result quality as a function of the knowledge base size for AgroAnalytics.

knowledge base size of previous experiments and used automatic generation of prune aggressiveness ($p_{aggr} = auto$). For Seismic, we observe that influence is minimal due to the high similarities of the search space shapes among different experiments. AgroAnalytics and SchedSim show a more heterogeneous behaviour. The shapes of the search spaces vary considerably among different experiments, especially for SchedSim. The reduction in search spaces varies according to experiment number, but in general the cuts in search space increase with the knowledge base size.

Figures 10, 11, 12 present the quality of the results as a function of the knowledge base size for the three use cases. For Seismic we observe considerable improvements when compare knowledge base (KB) 5 to the others (10, 15, and 20). However, the experiments with size 10 to 20 present very similar results. This behaviour can be explained by the small variation in search size pruning due to high similarity between Seismic experiments.

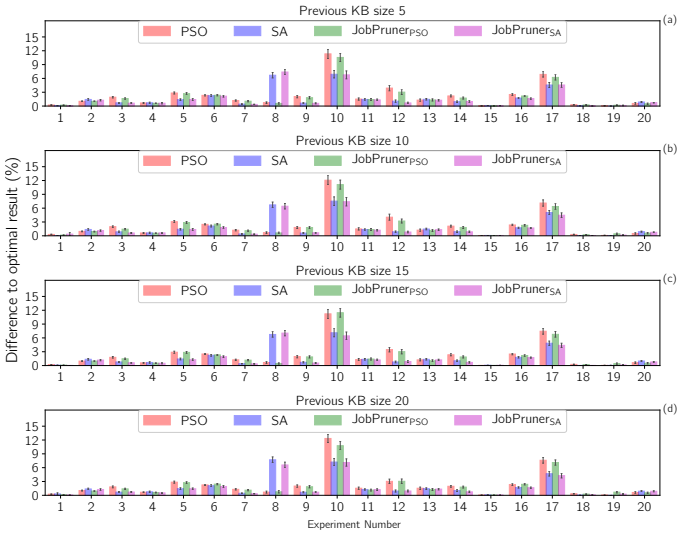


Fig. 12: Result quality as a function of the knowledge base size for SchedSim.

For AgroAnalytics, we also observe low variation of the results among different KB sizes. This happens due to the low variation of search space cuts (Figure 9b). For SchedSim, improvements due to DB size are not clearly observable. For this use case, even though there is a reduction in the search space due to a more comprehensive knowledge base (Figure 9c), this reduction is not significant enough to generate spare computing power to reach for better optimization results due to the great variability in the search space among the experiments of SchedSim. Table 2 summarizes the results. For all experiments, there is an improvement when comparing experiments with and without KB, and, for some, this improvement continues when the KB increases along all experiments (i.e., Seismic and AgroAnalytics with SA).

TABLE 2: Result quality (intervals) based on knowledge base size.

Optimizer	Previous Exp size	% diff (Seismic)	% diff (AgroAnalytics)	% diff (SchedSim)
PSO	0	[0.565, 0.696]	[0.587, 0.980]	[2.157, 2.188]
	5	[0.208, 0.305]	[0.523, 0.850]	[1.920, 1.972]
	10	[0.178, 0.254]	[0.472, 0.786]	[1.982, 2.034]
	15	[0.163, 0.231]	[0.472, 0.786]	[1.995, 2.033]
	20	[0.168, 0.238]	[0.494, 0.804]	[2.006, 2.044]
SA	0	[0.527, 0.652]	[2.636, 3.133]	[1.644, 1.720]
	5	[0.222, 0.326]	[1.280, 1.769]	[1.575, 1.651]
	10	[0.197, 0.287]	[1.175, 1.642]	[1.531, 1.646]
	15	[0.195, 0.279]	[1.171, 1.636]	[1.496, 1.573]
	20	[0.174, 0.250]	[1.135, 1.597]	[1.538, 1.628]

5 Conclusion

Scientists and engineers always look for ways to make their models more realistic to have insights about their subject of study. This usually imposes evaluation of a large number of computational jobs executed on HPC machines. In this paper we investigated the possibility and benefits of using data from past experiments to help identify unnecessary

jobs to be executed and speed up experiments of these professionals. We also introduced a machine learning-based tool, called JobPruner, to automate the process of identifying such jobs.

We executed a series of experiments using three real use case applications from different fields and were able to draw the following lessons:

- Pruning aggressiveness and search space sizes: Search space shapes have high influence on selecting jobs to be eliminated. Spiky search spaces tend to be more difficult to create surrogates that can easily identify jobs to be pruned;
- Knowledge base size: when experiments contain similar shapes of search spaces, expanding knowledge base size has no impact on pruning and quality of results. However, for experiments that are more heterogeneous, any additional experiments added to the knowledge base can bring new insights to prune jobs;
- Experiment similarities: even when experiments have low correlation, it is still possible learn portions of the search space that will not bring value to the user and the aggressiveness to make the proper prunes can be automatically identified based on surrogates generated from previous experiments.

This work is an indication that the reuse of experiments of the same kind of object/model is possible and we envision this is a rich research direction of exploring big data for optimizing HPC scientific experiments.

REFERENCES

[1] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, M. Taufer, Enhancing reproducibility for computational methods, Science 354 (6317) (2016) 1240–1241.

[2] K. Chard, J. Pruyne, B. Blaiszik, R. Ananthakrishnan, S. Tuecke, I. Foster, Globus data publication as a service: Lowering barriers to reproducible science, in: Proceedings of the 11th International Conference on e-Science, IEEE, 2015.

[3] I. Santana-Perez, R. F. da Silva, M. Rynge, E. Deelman, M. S. Pérez-Hernández, O. Corcho, Reproducibility of execution environments in computational science using semantics and clouds, Future Generation Computer Systems 67 (2017) 354–367.

[4] J. P. Kleijnen, S. M. Sanchez, T. W. Lucas, T. M. Cioppa, State-of-the-art review: a user's guide to the brave new world of designing simulation experiments, INFORMS Journal on Computing 17 (3) (2005) 263–289.

[5] S. M. Sanchez, H. Wan, Better than a petaflop: the power of efficient experimental design, in: Proceedings of the Winter Simulation Conference, IEEE, 2009.

[6] M. A. S. Netto, R. L. F. Cunha, N. Sultanum, Deciding when and how to move HPC jobs to the cloud, IEEE Computer 48 (11) (2015) 86–89.

[7] A. Gupta, L. V. Kale, F. Gioachin, V. March, C. H. Suen, B.-S. Lee, P. Faraboschi, R. Kaufmann, D. Milojicic, The Who, What, Why and How of High Performance Computing Applications in the Cloud, in: Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science, IEEE, 2013.

[8] J. D. Mulder, J. J. van Wijk, R. van Liere, A survey of computational steering environments, Future generation computer systems 15 (1) (1999) 119–129.

[9] H. Wright, R. H. Crompton, S. Kharche, P. Wenisch, Steering and visualization: Enabling technologies for computational science, Future Generation Computer Systems 26 (3) (2010) 506–513.

[10] M. García, J. Duque, P. Boulanger, P. Figueroa, Computational steering of CFD simulations using a grid computing environment, International Journal on Interactive Design and Manufacturing 9 (3) (2015) 235–245.

[11] B. K. Danani, B. D. D'Amora, Computational steering for high performance computing: applications on Blue Gene/Q system, in: Proceedings of the Symposium on High Performance Computing, Society for Computer Simulation International, 2015.

[12] E. Kail, P. Kacsuk, M. Kozlovszky, A novel approach to user-steering in scientific workflows, in: Proceedings of the International Symposium on Applied Computational Intelligence and Informatics, IEEE, 2015.

[13] S. G. Parker, C. R. Johnson, SCIRun: a scientific programming environment for computational steering, in: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, ACM/IEEE, 1995.

[14] J. J. Van Wijk, R. Van Liere, J. D. Mulder, Bringing computational steering to the user, in: Proceedings of the Scientific Visualization Conference, IEEE, 1997.

[15] J. Chin, J. Harting, S. Jha, P. V. Coveney, A. R. Porter, S. M. Pickles, Steering in computational science: mesoscale modelling and simulation, Contemporary Physics 44 (5) (2003) 417–434.

[16] M. A. S. Netto, A. Breda, O. de Souza, Scheduling complex computer simulations on heterogeneous non-dedicated machines: a case study in structural bioinformatics, in: Proceedings of the International Symposium on Cluster Computing and the Grid, IEEE, 2005.

[17] M. Mattoso, J. Dias, K. A. Ocaña, E. Ogasawara, F. Costa, F. Horta, V. Silva, D. de Oliveira, Dynamic steering of HPC scientific workflows: A survey, Future Generation Computer Systems 46 (2015) 100–113.

[18] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, Y. Zhao, Scientific workflow management and the Kepler system, Concurrency and Computation: Practice and Experience 18 (10) (2006) 1039–1065.

[19] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-science: An overview of workflow system features and capabilities, Future Generation Computer Systems 25 (5) (2009) 528–540.

[20] Y. Gil, V. Ratnakar, J. Kim, P. Gonzalez-Calero, P. Groth, J. Moody, E. Deelman, Wings: Intelligent workflow-based design of computational experiments, IEEE Intelligent Systems 26 (1) (2011) 62–72.

[21] E. Deelman, T. Peterka, I. Altintas, C. D. Carothers, K. K. van Dam, K. Moreland, M. Parashar, L. Ramakrishnan, M. Taufer, J. Vetter, The future of scientific workflows, The International Journal of High Performance Computing Applications 32 (1) (2018) 159–175.

[22] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, N. Gaud, A review and taxonomy of interactive optimization methods in operations research, ACM Transactions on Interactive Intelligent Systems 5 (3) (2015) 17.

[23] D. Meignan, S. Knust, Interactive optimization with long-term preferences inference on a shift scheduling problem, in: Proceedings of the 14th European Metaheuristics Workshop, Helmut-Schmidt-Univ., Faculty of Economics and Social Sciences, 2013.

[24] H. A. D. do Nascimento, P. Eades, User hints: a framework for interactive optimization, Future Generation Computer Systems 21 (7) (2005) 1177–1191.

[25] D. Abramson, A. Lewis, T. Peachey, C. Fletcher, An automatic design optimization tool and its application to computational fluid dynamics, in: Proceedings of the ACM/IEEE Conference on Supercomputing, ACM/IEEE, 2001.

[26] B. Silva, M. A. S. Netto, R. L. F. Cunha, SLA-aware Interactive Workflow Assistant for HPC Parameter Sweeping Experiments, in: Proceedings of the 11th Workshop on Workflows in Support of Large-Scale Science with The International Conference for High Performance Computing, Networking, Storage and Analysis, ACM/IEEE, 2016.

[27] P. Luszczek, M. Gates, J. Kurzak, A. Danalis, J. Dongarra, Search space generation and pruning system for autotuners, in: Proceedings of the IEEE International on Parallel and Distributed Processing Symposium Workshops, IEEE, 2016.

[28] G. W. Klau, N. Lesh, J. Marks, M. Mitzenmacher, Human-guided search, Journal of Heuristics 16 (3) (2010) 289–310.

[29] L. Colgan, R. Spence, P. Rankin, The cockpit metaphor, Behaviour & Information Technology 14 (4) (1995) 251–263.

[30] A. Endert, M. S. Hossain, N. Ramakrishnan, C. North, P. Fiaux, C. Andrews, The human is the loop: new directions for visual analytics, Journal of intelligent information systems 43 (3) (2014) 411–435.

[31] L. Tweedie, B. Spence, D. Williams, R. Bhogal, The attribute explorer, in: Proceedings of the Conference companion on Human factors in computing systems, ACM, 1994.

[32] H. A. Nguyen, D. Abramson, T. Kipouros, A. Janke, G. Galloway, WorkWays: interacting with scientific workflows, Concurrency and Computation: Practice and Experience 27 (16) (2015) 4377–4397.
[33] H. Nguyen, D. Abramson, WorkWays: Interactive workflow-based science gateways, in: Proceedings of the International Conference on E-Science, IEEE, 2012.
[34] D. Abramson, B. Bethwaite, C. Enticott, S. Garic, T. Peachey, Parameter exploration in science and engineering using many-task computing, IEEE Transactions on Parallel and Distributed Systems 22 (6) (2011) 960–973.
[35] M. J. Litzkow, M. Livny, M. W. Mutka, Condor - A hunter of idle workstations, in: Proceedings of the 8th International Conference on Distributed Computing Systems, IEEE, 1988.
[36] G. Fedak, C. Germain, V. Néri, F. Cappello, Xtremweb: A generic global computing system, in: Proceedings of the 1st International Symposium on Cluster Computing and the Grid, IEEE, 2001.
[37] D. P. Anderson, BOINC: A system for public-resource computing and storage, in: Proceedings of the 5th International Workshop on Grid Computing, IEEE, 2004.
[38] D. Abramson, J. Giddy, L. Kotler, High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, in: Proceedings of the 14th International Parallel & Distributed Processing Symposium, IEEE, 2000.
[39] N. Andrade, W. Cirne, F. Brasileiro, P. Roisenberg, OurGrid: An approach to easily assemble grids with equitable resource sharing, in: Proceeding of the Workshop on Job Scheduling Strategies for Parallel Processing, Springer, 2003.
[40] N. Segev, M. Harel, S. Mannor, K. Crammer, R. El-Yaniv, Learn on source, refine on target: a model transfer learning framework with random forests, IEEE transactions on pattern analysis and machine intelligence 39 (9) (2017) 1811–1824.
[41] L. Duan, D. Xu, I. W.-H. Tsang, Domain adaptation from multiple sources: A domain-dependent regularization approach, IEEE Transactions on Neural Networks and Learning Systems 23 (3) (2012) 504–518.
[42] J. Gao, W. Fan, J. Jiang, J. Han, Knowledge transfer via multiple model local structure mapping, in: Proceedings of the SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008.
[43] F. Padillo, J. Luna, S. Ventura, Subgroup discovery on big data: Pruning the search space on exhaustive search algorithms, in: Proceedings of the International Conference on Big Data, IEEE, 2016.
[44] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big Data 3 (1) (2016) 9.
[45] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2010) 1345–1359.
[46] A. R. Conn, K. Scheinberg, L. N. Vicente, Introduction to derivative-free optimization, Vol. 8, SIAM, 2009.
[47] T. A. Feo, M. G. Resende, Greedy randomized adaptive search procedures, Journal of global optimization 6 (2) (1995) 109–133.
[48] K. Briechle, U. D. Hanebeck, Template matching using fast normalized cross correlation, in: Aerospace/Defense Sensing, Simulation, and Controls, International Society for Optics and Photonics, 2001.
[49] A. J. Puppala, T. V. Bheemasetti, H. Zou, X. Yu, A. Pedarla, G. Cai, Spatial variability analysis of soil properties using geostatistics, Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering (2015) 195–226.
[50] P. K. Kitanidis, Introduction to geostatistics: applications in hydrogeology, Vol. 2, Cambridge University Press, 1997.
[51] H. Boogaard, A. De Wit, J. te Roller, C. Van Diepen, WOFOST Control Center 2.1: User's guide for WOFOST Control Center 2.1 and the crop growth simulation model WOFOST 7.1. 7, Wageningen, The Netherlands: Alterra.
[52] D. G. Feitelson, D. Tsafrir, D. Krakov, Experience with using the parallel workloads archive, Journal of Parallel and Distributed Computing 74 (10) (2014) 2967–2982.