Runtime Estimation and Scheduling on Parallel Processing Supercomputers via Instance-based Learning and Swarm Intelligence

Frank Po-Chen Lin and Frederick Kin Hing Phoa

Abstract—**Supercomputing has been indispensable in the unstoppable trend of high-speed computing evolution. This work aims at improving its running efficacy by introducing a new two-step scheduling approach. Based on the analysis of large historical data, we provide an accurate runtime estimation scheme using Instance-Based Learning (IBL) in the first step. Then a swarm intelligence based scheduling (SIBS) method is proposed to optimize the scheduling performance in terms of total runtime makespan and fair resource allocation. A method comparison on a dataset from the ALPS supercomputer, which consists of 804k workload data in 2016, shows that our proposed method outperforms the most commonly used strategy – Extensible Argonne Scheduling System (EASY).**

Index Terms—**Supercomputer, Scheduling, Swarm Intelligence, Instance-Based Learning, Runtime estimation.**

I. Introduction

With the advancement in technology, many high-speed computing techniques have emerged. Applications unimaginable in past few years has now become achievable. Owing to the development on Internet applications [1] and new computing schemes, scenarios such as cloud computing [2] and parallel computing [3] have come into play and resulted in dramatic improvements in high-speed computation [4-6]. Consequently, new fields of studies such as big data analysis and artificial intelligence have started to thrive. Supercomputer, a high-level performance computer, consists of tens of thousands of processors that are capable of performing billions to trillions of calculations per second and achieving massive computing power is, without doubt, the indispensable role in the unstoppable trend of high-speed computing evolution. Therefore, efficiently improving performance on a supercomputer would be without doubt a vital issue. Many institutions have started to increasingly add computing cores to achieve higher computation performance. However, [7] shows that simply expanding the number of processing nodes and leveraging technology scaling would not be an efficient way to improve the processing power of supercomputers, as power demand would increase unsustainably. To improve supercomputer's running efficacy, many researchers have devoted full effort into supercomputer scheduling [8-11], coming up with various scheduling schemes to enhance the overall performance of the supercomputer without the need of setting up additional hardware.

Before designing the scheduling scheme, an important factor in scheduling performances is the runtime estimation. It is an important attribute used by the schedulers in various scenarios. Its accuracy is proved to be highly correlated with scheduling performances by [12]. Researchers have been working thoroughly on this topic [13-16], trying to come up with different solutions to provide accurate estimates of runtime data. It would be important to have good domain knowledge and insight with their own runtime data to improve the accuracy of estimation. This work applies the data offered by the supercomputer, Advanced Large-scale Parallel Supercluster (ALPS) in National Center for High-Performance Computing (NCHC) in Taiwan.

As the need for large computation keeps increasing, large traffic workload has gradually become a burden for ALPS. To handle this issue, this work provides two major contributions. First, an accurate runtime estimation scheme based on the analysis of a large historical data from ALPS is proposed using Instance-Based Learning (IBL) [17]. Second, a new scheduling scheme for supercomputers on large traffic load using Swarm Intelligence is designed.

A scheduling scheme is a critical factor to the performance of a supercomputer. Many researchers have as well concentrated on the design of supercomputer scheduling trying to obtain a suitable approach in the optimization of various goals. Due to the attractiveness in simplicity, effectiveness, and fairness, the most common used strategy in supercomputer scheduling is FCFS (First-Come First Served) with backfilling, also known as the term EASY (Extensible Argonne Scheduling sYstem). Although easy to implement, job scheduling on supercomputers, however, can be complicated due to diverse demands of system administrators and may not be enough to be effectively approached by simply applying EASY. In fact, runtime efficiency and fairness are usually conflicting goals to be achieved. The inefficiency becomes evident especially when the workload is large. Therefore, to both consider the runtime efficiency and user fairness comprehensively while preserving the feature of simple implementation in EASY, a heterogeneous non-preemptible scheduling scheme to obtain a real-time scheduling on large traffic workload is proposed. This work designs a Swarm Intelligence Based Scheduling (SIBS) method to optimize the performance and achieve both efficiencies on total runtime makespan and fair resource allocation.

The rest of the paper is organized as follows. In Section II background knowledge on IBL and original SIB optimization are provided. Section III presents the design of IBL runtime estimation, based on data from ALPS, and the modified SIB

F.P.C. Lin was with the Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan. (e-mail: frank555076@gmail.com).

F. K. H. Phoa is with the Institute of Statistical Science, Academia Sinica, Taipei 11529, Taiwan (e-mail: fredphoa@stat.sinica.edu.tw).

for resource-constrained job scheduling. In Section IV, the simulation setup is described and the result of the proposal is evaluated. Finally, Section V concludes the paper and outlines the contribution of this work.

II. Background

To provide an efficient approach to improve the performance of a supercomputer, both runtime estimation, and job scheduling should not only be operative but also computationally effective. For runtime estimation, global parametric learning algorithms, such as neural networks, attempt to establish an input-output mapping via a single function with a global network view. However, this would neglect important properties of data partitions when the input is highly correlated to local data, which is often the case for runtime estimation. This work found IBL most suitable and perform good results of our estimates. For scheduling, classic optimization approaches such as nonlinear programming or dynamic programming can compute the exact solution and have better accuracy but are computationally time-consuming when the large-scale problem is considered. Therefore, this research designs a metaheuristic approaches SIB that gives near-optimal answers but is computationally efficient.

A. Instance-Based Learning

Runtime prediction of new input data is formed through past related experiences in the historical database. Experiences consist of several input features and one output result. Every input features depict the characteristics of the data while the output describes the runtime result corresponding to the conditions of these features. New input data consists of only input features whereas its runtime prediction is formed based on these features. Instead of querying the entire experiences in the database to form a prediction, only past experiences with high correlated input features are used as training sets to provide runtime estimation through similarity calculation. This allows an estimate to preserve useful local information and filter out unrelated information that would degrade the performance of accuracy.

IBL can be categorized into two major parts: similarity calculation and kernel regression. In similarity calculation, a distance function is defined as an indicator of similarity between two data according to the feature of the attributes. In kernel regression, a weighted-distance average of output is provided for final runtime prediction. The weights given to different runtimes are defined by the kernel function. The kernel function determines the weights on a given runtime data according to the measured similarity between input and historical experiences.

1) Distance Function

The distance function for similarity measure is defined as

$$d_{ij} = d(i,j) = \frac{\sum_{f=1}^{n} w_f \delta_{ij}^f d_{ij}^f}{\sum_{f=1}^{n} w_f \delta_{ij}^f}, \qquad (1)$$

where f is the feature, w_f is its weight,

$$\delta_{ij}^f = \begin{cases} 1, & \text{if feature } f \text{ exists in both data} \\ 0, & \text{otherwise} \end{cases}, \qquad (1\text{-}1)$$

$$d_{ij}^f = \begin{cases} overlap_f(i,j), & \text{if nominal} \\ avediff_f(i,j), & \text{if numerical} \end{cases}, \qquad (1\text{-}2)$$

where

$$overlap_f(i,j) = \begin{cases} 0, & i_f = j_f \\ 1, & \text{otherwise} \end{cases}$$

$$avediff_f(i,j) = \frac{|i_f - j_f|}{\max_f - \min_f}$$

2) Kernel Function

Kernel function provides the result of predicted runtime estimates E_R through similarities obtained from distance function and is formulated as

$$E_R(i) = \frac{\sum_i K(d(i,j)) \cdot R_j}{\sum_i K(d(i,j))}, \qquad (2)$$

where R_j is the actual runtime of related experience j and $K(d)$ is the exponential kernel function used to derive the weight for runtime R_j shown below.

$$K(d) = \exp(-d)^2. \qquad (3)$$

B. Swarm Intelligence Optimization

The concept of Swarm Intelligence has been applied to many different applications [18]. The idea of the SIB algorithm in depicted in Figure 1.

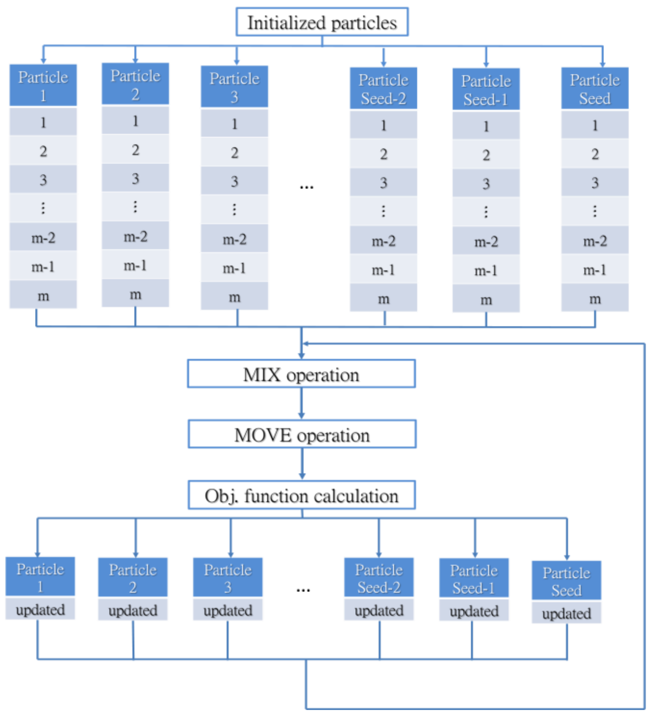


Fig. 1. SIB Algorithm

In the step of initialization, possible solutions are generated as initial particles and the objective values for these particles are evaluated. Through evaluation, each particle perceives its own location of initial optimum in the search space called Local Best (LB) particles. All particles share information by comparing its LB with other to obtain the overall optimum called Global Best (GB) particle. For particles to collectively arrive at the perceived overall optimum solution, they go through the steps of MIX and MOVE operations iteratively after initialization. In the MIX operation, particle X exchanges elements with LB and GB particles to form new particles mixed LB and mixed GB respectively. In the MOVE operation, the objective value of mixed GB, mixed LB, and particle X are evaluated. A particle with better objective value is chosen to replace particle X. However, if both mixed GB and mixed LB do not make particle X move toward a better location in the search space, elements in particle X would be replaced with any random particle as a prevention of being trapped in a locally optimal solution. GB and LBs are updated if any better solutions are found. LB particles and GB particle are updated continuously in every iteration until the stopping criterion is fulfilled. The stopping criteria can be the reach of either the pre-specified maximum number of iterations or a known optimal value of the GB particle.

III. Runtime Estimation and Job Scheduling on Supercomputers

This section introduces the method of runtime estimation on user workloads using IBL and describes a newly designed SIB scheduling algorithm for supercomputers.

A. *Job Runtime Estimation*

This work evaluates the prediction technique using data from the ALPS supercomputer system. Characteristics of execution jobs in ALPS have shown in Table I.

TABLE I: Feature of Workloads

Input Features		
Feature	**Feature**	**Feature**
User ID	User ID	User ID
Queue Name	Queue Name	Queue Name
Job Name	Job Name	Job Name
Number of CPU Cores	Number of CPU Cores	Number of CPU Cores
Submit Time	Submit Time	Submit Time
Output Feature		
Feature	**Feature**	**Feature**
Runtime	Runtime	Runtime

Through correlation analysis, a strong degree of dependency between jobs summited by users and the runtime feature can be found. As a result, the search space of every new input data is separated into various partitions according to different users. For instance, if user 1 submits a new job to the system to perform IBL prediction, the system only considers user 1's historical experiences as a relevant dataset for runtime estimation. This not only preserves data locality but also decreases the search space to perform similarity computation, which would cause huge computation burden when the entire dataset is considerably large.

After deciding the relevant dataset of user 1, the distance function between input data and all experiences in the dataset are calculated. All distance metrics are now available for the next step. Finally, K nearest neighbors with the lowest values of distances are chosen. The runtime prediction of the newly submitted job is determined by these K nearest neighbors using the kernel function.

The estimation procedure can be generalized into four major steps upon receiving a new job request:

1) *Dataset Determination*

The identity of job submitter is first determined. Afterward, the submitter's past experiences are chosen as the relevant dataset to perform IBL.

2) *Similarity computation*

The similarity metric between features from the new input and its corresponding experiences in the relevant dataset is computed with the distance function.

3) *K-Nearest Neighbors*

After acquiring all similarity metrics, K experiences with the lowest similarity values are selected as the final dataset to perform runtime prediction. Through simulations, the results show that the estimation provides great accuracy when only three nearest experiences (K=3) are selected as the final dataset. This decision of the parameter K narrows down the estimation complexity without compromising the overall accuracy.

4) *Runtime Estimate*

The kernel function takes the experiences in the final dataset as input and comes out with the runtime estimation result for the input data.

B. *Swarm Intelligence based Scheduling*

In previous researches, like most of the evolutionary algorithms, SIB optimization focused on unconstrained and non-ordering problems. In supercomputer scheduling, however, resource constraints on remaining supercomputer cores needs consideration. Every scheduling decision should concern the availability of cores in every time slot in order to make full use of the resources and result in an efficient job schedule that reduces the total operating time.

Under busy traffic conditions, job arrivals are large and have to wait in the queue for processing. The scheduler reschedules the arrival jobs in every t_{update} time units. During t_{update}, the scheduler awaits for new arrivals while the supercomputer simultaneously provides operations on previously arrived jobs. To provide scheduling efficiency, the setting of t_{update} has to satisfy two conditions. First, it has to be larger than the total makespan $t_{makespan}$ of previous assigned jobs in the system so the supercomputer would not be waiting without serving any jobs. Second, it should be large enough for a certain amount of new jobs to arrive so that the scheduling result would be influential enough to upgrade system's performance. To satisfy the above-mentioned criteria, the relationship of t_{update} and $t_{makespan}$ satisfies

$$\frac{\lambda \cdot t_{update}}{\mu} \geq t_{makespan}, \tag{4}$$

where λ is the average arrival rate of jobs and μ is the average amount of finishing workloads in the one-time unit. It is clear that Equation 4 would be easily satisfied under busy traffic conditions when λ is larger than μ.

In this research, the objective of a scheduling decision is to minimize total waiting time of all user-submitted jobs. This objective ensures fairness on waiting time for different jobs is considered while improving system's performance on total operation time. While being processed, the job i requires r_i^c processing cores and d_j units of processing time with job starting time t_i. The total amount of available cores in the supercomputer is denoted as R_c. The optimization problem can be formulated as follows.

$$\min \sum_i (t_i + d_i + \delta t_{update}),\ i \in S(\tilde{t});\ \delta \in \{0,1\}.$$
$$s.t. \begin{cases} \sum_{i \in A(\tilde{t})} r_i^c(t) \leq R_c, \forall t \geq 0. \\ t_i, d_i, r_i^c \geq 0,\ i \in S(\tilde{t}). \end{cases} \tag{5}$$

where $S(\tilde{t})$ is the set of all submitted jobs during the time interval $\tilde{t}$.

Jobs being assigned after t_{update} would only be scheduled on the supercomputer in the next scheduling session. Therefore, to enhance the fairness of the waiting time of all jobs. δt_{update} is added as a penalty factor to the objective value for assigning any jobs after t_{update}.

In order to solve the constrained optimization problem, this work designs a new particle formation and a novel MIX operation for SIB scheduling. In SIB scheduling, every particle is designed to represent a list of scheduling priority and every element in a particle represents a job ID. Every particle's performance is evaluated by decoding the priority list into job schedules through the parallel scheduling generation scheme. In this research, the idea of particle transformation is applied due to its merit that the scheduling result decoded from particles after MIX and MOVE is still feasible to construct a resource feasible project schedule.

1) Parallel Scheduling Generation Scheme

A parallel scheduling generation scheme iterates over the time stamp of projects and adds jobs that are eligible to the schedule. Scheduling starts at the time point $t_g = 0$ and schedules jobs before the time pointer is increased. It selects a job at each decision point t_g from the eligible job set E_g and assigns a scheduling sequence of these eligible jobs according to the priority list in each particle. The pseudo-code of the scheme is as follows:

Algorithm 1 Parallel Schedule Generation Scheme

```
Initialize: $g = 0$, $t_g = 0$, $A_0 = \{0\}$, $\tilde{R}_c(0) = R_c$
while $|A_g \cup C_g| \leq n$ do
    $g = g + 1$
    $t_g = \min_{j \in A_g}\{F_j\}$
    Calculate: $C_g$, $A_g$, $\tilde{R}_c(t_g)$, $E_g$
    while $|E_g| > 0$ do
        Select the first job $i \in E_g$
        $F_i = t_g + d_i$
        Calculate: $A_g$, $\tilde{R}_c(t_g)$, $E_g$
    end while
end while
$F_{n+1} = \max_{h \in P_{n+1}}\{F_h\}$.
```

where the complete set C_g contains all jobs that have been scheduled and completed before t_g and the active set A_g contains all scheduled but unfinished jobs. F_i corresponds to the finishing time of the job i, where $\tilde{R}_c(t_g)$ is the remaining core available at the time t_g so that $E_g = \{i \in S \backslash C_g \cup A_g | r_{i,c} \leq \tilde{R}_c(t_g)\}$.

2) MIX Operation for SIB Scheduling

Instead of exchanging particle elements with LB and GB, new particles are formed through the idea of imitation. During MIX operation, every particle X imitates the priority order of a good particle (LB or GB) starting from the first element. In every iteration, a position and job ID of the element v of the good particle is perceived. Particle X searches its own elements to find the index that contains the value v. After finding the corresponding index, particle X swaps the job ID in that index with the initially perceived position in the good particle. The pseudocode of the MIX is shown below. The $BinarySearch(v, ParticleX)$ returns the index of $ParticleX$ that contains the value v.

Algorithm 2 MIX Operation in SIB Scheduling

```
Initialize: $id = 0$
while $|id| \leq swapNum$ do
    $v = goodParticle[id]$
    $temp = ParticleX[id]$
    $idx = BinarySearch(v, ParticleX)$
    $ParticleX[id] = v$
    $Particle[idx] = temp$
    $id = id + 1$
end while
```

IV. Simulation

The simulation performed in this section is carried out by a SIB program written in R. This section evaluates the performance of the proposed runtime prediction technique and the efficiency of SIB supercomputer scheduling using data from the ALPS supercomputer. Workloads from January to December 2016 with the total amount of 804697 data are used to evaluate the designs in this work.

A. Runtime Prediction Evaluation

In the simulation, the training workload consists of 563287 experiences and the testing workload contains 241410 inserting data. The performance of the proposed runtime prediction is evaluated using Root-Mean-Squared Error (RMSE) indicator as shown in Table II.

The runtime prediction scheme provides an average estimation error of 20.73 minutes with a standard deviation of 33.82 minutes. The best estimation error can be achieved below 1 minute, while the error in the worst case is bounded by 151.58 minutes. The result shows that the prediction provides good accuracy on runtime estimation.

TABLE II: RMSE OF RUNTIME PREDICTION

RMSE Estimation (minutes)			
Mean RMSE	SD RMSE	Worst RMSE	Best RMSE
20.73 (m)	33.82 (m)	151.58 (m)	0.22 (m)

B. SIB Scheduling Evaluation

The performance of the SIB scheduling design is compared with the EASY scheduling scheme. New inserting jobs are sampled from the 241410 testing data workloads. The scheduling schemes use the estimated time from IBL runtime prediction to determine their schedules. The total makespan of the actual execution time of the works between two different designs is evaluated as the indicator of scheduling performance. In order to examine the efficacy of the scheduling design under every condition of busy traffic, the worst case, that is $t_{update} = t_{makespan}$, is considered in the simulation.

The performance between different amounts of sampled workloads and total makespans for two different scheduling schemes is demonstrated. The impact of the number of initial particles on the SIB scheduling performance is also shown in Figure 2ab. The performance on the makespan of job arrivals of SIB scheduling outperforms the EASY scheduling scheme. Moreover, as the number of the initial particles (seed) increases, the scheduling performance of the SIB upgrades to a higher level. The performance of SIB scheduling on a supercomputer is proved to be efficient.

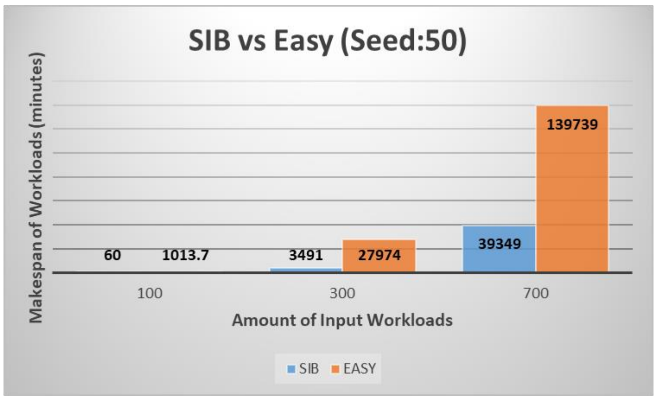


Fig 2(a). SIB versus EASY at 50 seeds.

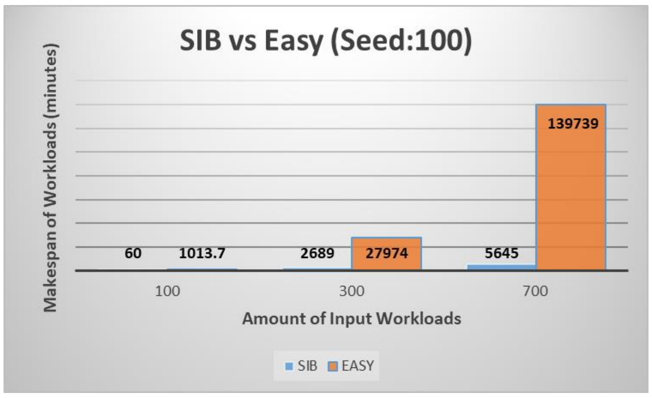


Fig. 2(b). SIB versus EASY at 100 seeds.

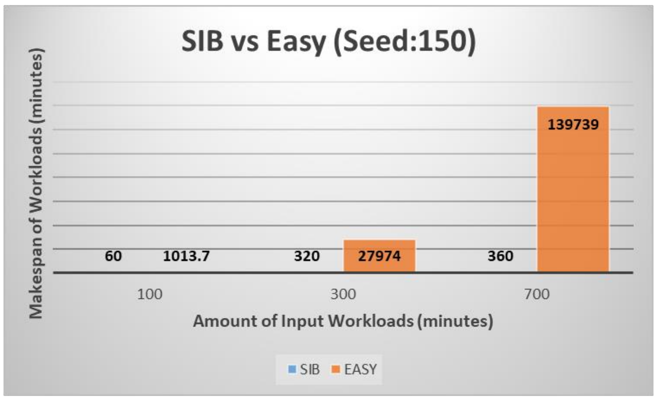


Fig. 2(c). SIB versus EASY at 150 seeds.

As shown in Fig. 2(a) to Fig. 2(c), the performance on the makespan of job arrivals of SIB scheduling outperforms the EASY scheduling scheme. Moreover, as the number of the initial particles (seeds) increases, the scheduling performance of the SIB upgrades to a higher level. The performance of SIB scheduling on a supercomputer is proved to be efficient.

V. CONCLUSIONS

To improve supercomputer's running efficiency, it is important to both upgrade the accuracy of runtime prediction and enhances the efficiency of the scheduling algorithm, while maintaining the system to work below a certain complexity level. This work designs a runtime prediction scheme and a novel scheduling algorithm via instance-based learning and Swarm Intelligence. Both designs require little computation efforts compared to classic neural network learning and convex optimizations.

The instance-based learning runtime estimation scheme is proposed based on the characteristics of the data in the ALPS supercomputer to improve the accuracy of prediction while the new swarm intelligence scheduling algorithm is designed to optimize the performance and achieve both efficiency on runtime makespan and fair resource allocation on supercomputers under busy traffic conditions.

VI. ACKNOWLEDGEMENT

This work was supported in part by the Career Development Award of Academia Sinica (Taiwan) Grant Number 103-CDA-M04 and the Ministry of Science and Technology (Taiwan) Grant Number 105-2118-M-001-007-MY2.

REFERENCES

[1] S. H. Wang, F. P. C. Lin, and C. P. Li, "Secure channel estimation method in TDD OFDM systems," in *2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2016, pp. 1-4.

[2] A. Goyal and S. Dadizadeh, "A survey on cloud computing," *University of British Columbia Technical Report for CS,* vol. 508, pp. 55-58, 2009.

[3] F. P.-C. Lin and F. K. H. Phoa, "A Performance Study of Parallel Programming via CPU and GPU on Swarm Intelligence Based Evolutionary Algorithm," presented at the Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence, Hong Kong, Hong Kong, 2017.

[4] Y. A. Basallo, V. E. Senti, and N. M. Sanchez, "Artificial intelligence techniques for information security risk assessment," *IEEE Latin America Transactions,* vol. 16, no. 3, pp. 897-901, 2018.

[5] A. v. d. Mei and J. P. Doomernik, "Artificial intelligence potential in power distribution system planning," *CIRED - Open Access Proceedings Journal,* vol. 2017, no. 1, pp. 2115-2117, 2017.

[6] S. Liu, Y. Wang, M. Fardad, and P. K. Varshney, "A Memristor-Based Optimization Framework for Artificial Intelligence Applications," *IEEE Circuits and Systems Magazine,* vol. 18, no. 1, pp. 29-44, 2018.

[7] F. Fraternali, A. Bartolini, C. Cavazzoni, and L. Benini, "Quantifying the Impact of Variability and Heterogeneity on the Energy Efficiency for a Next-Generation Ultra-Green Supercomputer," *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, no. 7, pp. 1575-1588, 2018.

[8] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling Using System-Generated Predictions Rather than User Runtime Estimates," *IEEE Transactions on Parallel and Distributed Systems,* vol. 18, no. 6, pp. 789-803, 2007.

[9] W. Tang, D. Ren, Z. Lan, and N. Desai, "Toward balanced and sustainable job scheduling for production supercomputers," *Parallel Computing,* vol. 39, no. 12, pp. 753-768, 2013/12/01/ 2013.

[10] D. Tsafrir and ד. צפריר, *Modeling, evaluating, and improving the performance of supercomputer scheduling*. Hebrew University, 2006.

[11] M. F. Tompkins, "Optimization techniques for task allocation and scheduling in distributed multi-agent operations," Massachusetts Institute of Technology, 2003.

[12] D. Tsafrir and D. G. Feitelson, "The Dynamics of Backfilling: Solving the Mystery of Why Increased Inaccuracy May Help," in *2006 IEEE International Symposium on Workload Characterization*, 2006, pp. 131-141.

[13] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Modeling User Runtime Estimates," in *Job Scheduling Strategies for Parallel Processing*, Berlin, Heidelberg, 2005, pp. 1-35: Springer Berlin Heidelberg.

[14] S. Krishnaswamy, S. W. Loke, and A. Zaslavsky, "Estimating computation times of data-intensive applications," *IEEE Distributed Systems Online,* vol. 5, no. 4, p. 1, 2004.

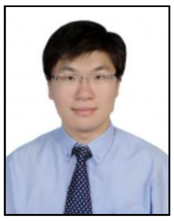
[15] W. Tang, N. Desai, D. Buettner, and Z. Lan, "Job scheduling with adjusted runtime estimates on production supercomputers," *Journal of Parallel and Distributed Computing,* vol. 73, no. 7, pp. 926-938, 2013/07/01/ 2013.

[16] M. A. Iverson, F. Ozguner, and L. C. Potter, "Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment," in *Heterogeneous Computing Workshop, 1999. (HCW '99) Proceedings. Eighth*, 1999, pp. 99-111.

[17] W. Smith, "Prediction Services for Distributed Computing," in *2007 IEEE International Parallel and Distributed Processing Symposium*, 2007, pp. 1-10.

[18] F. P. C. Lin and F. K. H. Phoa, "An Efficient Construction of Confidence Regions via Swarm Intelligence and Its Application in Target Localization," *IEEE Access,* vol. 6, pp. 8610-8618, 2018.

Frank Po-Chen Lin received the B.S. degree in electrical engineering from National Sun Yat-sen University, Taiwan, in 2016 and the M.S. degree from National Taiwan University, Taiwan in 2018. He is a Research Assistant with the Institute of Statistical Science, Academia Sinica. His current research interests include software defined networking, scheduling algorithm designs, parallel computing and supercomputing. He was a recipient of the College Student Research Creativity Award in the Ministry of Science and Technology, Taiwan, in 2016.



Frederick Kin Hing Phoa received the Ph.D. degrees in statistics from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 2009. From 2009 to 2013, he was an Assistant Research Fellow with the Institute of Statistical Science, Academia Sinica, Taiwan, where he was promoted to an Associate Research Fellow, in 2013. He is an author of over 50 scientific articles, a speaker of over 90 invited talks in the international conferences and 60 seminar talks in the universities around the world. His research interests include design and analysis of physical, computer and network experiments, analysis of internet and social media data, network data analysis, nature-inspired metaheuristics optimization, big data analysis, stochastic control in large-scale systems, semiparametric methods to the data with missing covariates, deep learning and neural network modeling. Dr. Phoa was a recipient of the Career Development Award in 2014, the Ta-You Wu Memorial Award (the Young Researcher Award) in 2014, and the Best Paper Award in the World Congress of Engineering in 2015. He was a recipient of the Special Talent Researcher Award from 2012 to 2017. He received the Excellent Young Researcher Project from 2013 to 2016 supported by the Ministry of Science and Technology (MOST), Taiwan, and the International Cost-Share Exchanges Scheme Project from 2016 to 2018 between the MOST and the Royal Society of U.K.