*Proceedings of the 2017 Winter Simulation Conference*
*W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.*

# OPTIMAL DESIGN OF MASTER-WORKER ARCHITECTURE FOR PARALLELIZED SIMULATION OPTIMIZATION

Haobin Li
Xiuju Fu
Xiao Feng Yin

Department of Computing Science
Institute of High Performance Computing, A*STAR
1 Fusionopolis Way, #16-16 Connexis
138632 SINGAPORE

Giulia Pedrielli

School of Computing, Informatics, &
Decision Systems Engineering
Arizona State University
669 S Mill Ave
Tempe, AZ 85281, USA

Loo Hay Lee

Department of Industrial Systems Engineering & Management
National University of Singapore
1 Engineering Drive 2
117576 SINGAPORE

## ABSTRACT

This study formulates and solves the design problem for a master-worker architecture dedicated to the implementation of a parallelized simulation optimization algorithm. Such a formulation does not assume any specific characteristic of the optimization problem being solved, but the way the algorithm is parallelized. In particular, we refer to the master-worker paradigm, where the master makes sampling decisions while the workers receive solutions to evaluate. We identify two metrics to be optimized: the throughput of the workers in terms of the number of evaluations per time unit, and the lack of synchronization between the master and the workers. We identify several design parameters: number of workers ($n$), the buffer size for each worker and for the master and the sample size $m$, i.e., the number of solutions used by the master for sampling decisions at each iteration. Numerical experiments show optimal designs over randomly generated simulation optimization algorithm instances.

## 1 INTRODUCTION

For large-scale and complex industrial systems, simulation-based optimization (sim-opt) is an effective way to identify the optimal design or configurations. In a typical sim-opt framework, candidate designs (i.e., solutions) are generated by the optimization algorithm (i.e., the optimizer), and passed to the simulation model (i.e., the simulator) for evaluating their performance. The evaluated performance is returned to the optimizer, for guiding the generation of the next sample of solutions.

Due to the computational complexity of executing the simulation model, it is time-consuming to deploy a sim-opt solver in a real-time. As a result, sim-opt has been largely applied in off-line environments.

Nevertheless, the development of high-performance computing infrastructures gives an opportunity for parallelization of sim-opt procedures opening the possibility to use this technique in real-time. The parallel computing infrastructure we refer to includes a multi-core workstation, computer cluster/grid, and cloud computing. The performance of this architecture varies based on the number of nodes, the execution rate of each node, and communication time between the nodes.

Several contributions have been proposed to exploit the advantage of parallel architectures. Mühlenbein (1989), Pierreval and Paris (2000), propose a genetic algorithm where each processing node keeps a local population that evolves with limited communication with other nodes. In such an approach, the computational effort of the optimization algorithm is distributed to the multiple processors. The approach is effective, but the way the parallelization is implemented highly depends upon the specific optimizer (e.g., genetic algorithms with the concept of local population).

Algorithms implemented following the general master-worker parallelization have been proposed as well (Figure 1). Different from decentralized approaches, the optimization algorithm is dedicated to the master node that is in charge of generating the candidate designs to be evaluated, whereas the worker nodes are responsible for executing the simulation (evaluation task) and returning the results to the master. As an example, Luo et al. (2000) illustrates the parallel implementation of a ranking & selection procedure that identifies an optimal solution with optimal computing budget allocation (OCBA) to choose the allocation of simulation replications; Laganá et al. (2006) illustrates an example of applying grid computing to solve a large-scale sim-opt problem with simulated annealing and Rinott's procedure. (Luo, Hong, Nelson, and Wu 2015) proposes a fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. Ni et al. (2017) provides fundamental results while considering the statistical issues in implementing such parallelization.

The objective of this paper is to formulate the design of the architecture in Figure 1 as a stochastic multi-objective simulation optimization problem, regardless of the specific optimization problem (e.g., continuous/discrete, ordered/categorical) algorithm or simulation model.

## 2 THE MASTER-WORKER ARCHITECTURE

In this section, we present the adopted architecture in order to highlight the parameters, decision variables and the objective function for the optimization of the execution of a parallelized simulation-optimization algorithm in a master-worker environment.

In this paper, similar to the Ranking & Selection literature such as Nelson et al. (2001), the number of measurements is one of the main consideration, as it is assumed that the simulation evaluation is the most computational expensive. Nevertheless, as in the distributed algorithms literature, we also consider that the time required by the optimizer to sample new solutions is not negligible. This can be particularly true when the number of workers becomes large. The resulting infrastructure is illustrated in Figure 1. The *master* is dedicated to the optimizer, and *workers* are for simulation purposes. Figure 2 illustrates several deployments of the master-worker architecture, both in stand-alone machines such as multi-core PCs and workstations, and through an intranet and the Internet.

We consider the following factors as having effect over the performance of the parallel algorithm, some of which can be controlled:

- Uncontrollable Factors:
  - **Sampling Time** $t_s$: the time required by the optimizer to sample a new batch of solutions. It is a random variable with moments as functions of the size of the sampled solutions $m$;
  - **Simulation Time** $t_e$: the time required for completing a single run simulation. It is a random variable and it depends on the complexity of the solution being sampled (Ni et al. 2017) and the processor;
  - **Transmission Time** $t_\tau$: the time required to exchange design and performance values between master and workers for simulation. This is a random variable depends on the computing infrastructure, e.g., network condition;
- Controllable Factors:

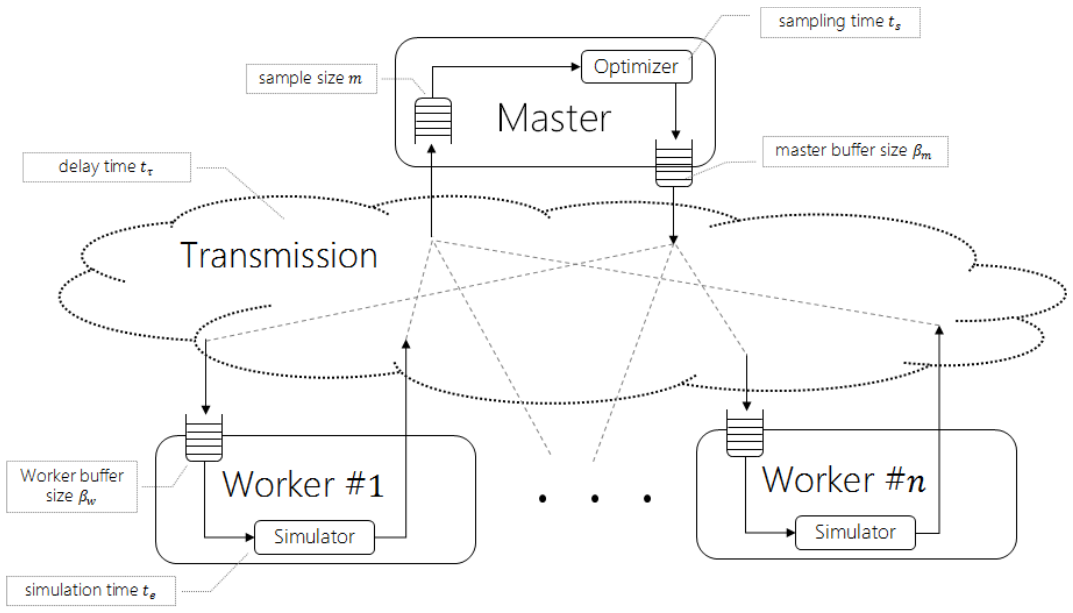


Figure 1: The Master-Worker Architecture for Parallelized Simulation based Optimization.

- **Number of Workers** $n$: the number of worker nodes for simulation evaluation in parallel, i.e., the number of cores;
- **Buffer Size** $\beta_m, \beta_w$: the number of designs sampled but not yet processed. It occurs at master level ($\beta_m$) to store designs sampled but yet to be transmitted to *workers*; and at worker level ($\beta_w$), where designs are received but yet to be evaluated.
- **Sample Size** $m$: the number of evaluated designs required to generate a sampling decision at each iteration. Note that according to the sampling algorithms of the optimizer, $m$ could be a function of search iteration. But in this paper, we assume it remains as a constant throughout the search, i.e., the master need to collect the same number of evaluations before triggering the optimizer to sample for the next batch.

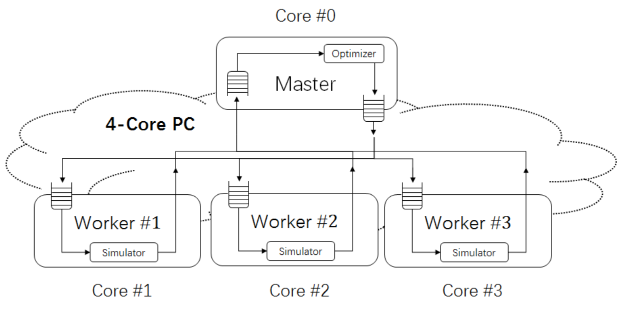
These factors affect the performance of the whole parallel sim-opt procedure. In particular, in this paper, we consider two performance measures: the (1) **Throughput of Evaluation** ($f_1$), as the number of simulation replications performed at each iteration of the master; the (2) **Age of Information** ($f_2$) as the number of simulation replications that have been sampled and assigned to the workers, but not evaluated at each iteration.

Intuitively, $f_1$ is positively affected by the number of nodes and the buffer size, but negatively affected by the variation of simulation time, and it is bounded by the sampling speed. It can be argued that a low $f_2$ is generally preferred (the algorithm is working with updated information) and it is positively affected by the number of workers, and the buffer size.

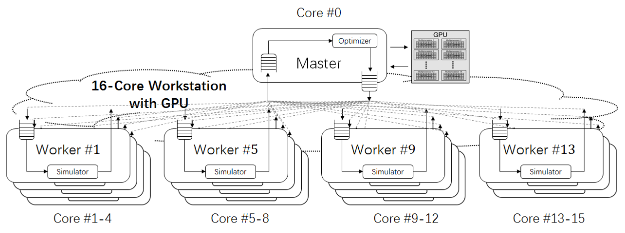
In this work, we analyze the relationship between the controllable factors and provide empirical support for how to design a high-efficiency master/worker framework with general (simulator, optimizer) pair. The problem becomes:

$$\min_{n,\beta_m,\beta_w,m} \{-f_1(n,\beta_m,\beta_w,m), f_2(n,\beta_m,\beta_w,m)\}. \quad (1)$$

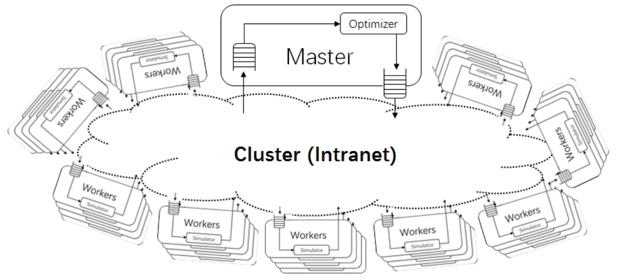
Where both functions can be estimated with noise by running a simulation model which emulates the behavior of the optimization algorithm. It is apparent that this is a stochastic simulation optimization problem with multiple objectives. In order to clarify the proposed approach, we describe how we derived the emulator of the algorithm in Section 3.



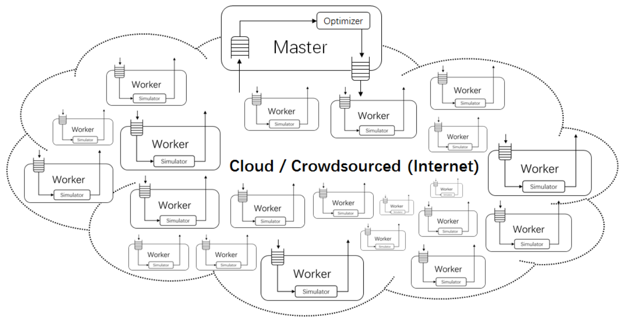
(a) 4-Core PC



(b) 16-Core Workstation with GPU



(c) Cluster connected by an intranet



(d) Cloud / crowdsourced computers via Internet

Figure 2: Illustration of Master/Worker Sim-Opt framework on different computing infrastructure.

## 3 A SIM-OPT METHOD FOR OPTIMAL ARCHITECTURE DESIGN

In this section, we illustrate the simulation optimization algorithm designed to solve problem (1). Section 3.1 illustrates the simulation model for the optimization architecture, while section 3.5 presents the search algorithm to optimize the parameters identified in section 2 against the objectives $f_1$ and $f_2$.

### 3.1 Simulating the Parallel Framework

The O$^2$DES framework (Li et al. 2015) is adopted for constructing the simulation model. In addition, complying with the DEVS (discrete event system specification) formalism (Zeigler 1987, Zeigler et al. 2000), the proposed system is modularized as a coupling of three atomic components, i.e., the *master*, *worker*, and *transmission*.

Note that in a primitive O$^2$DES simulation, the model is defined by three elements, i.e., the static properties that describe the configuration of the system to be simulated, the dynamic properties which provide the snapshot of the system state in runtime, and the events that describe the change of the state at discrete times. With the modularization suggested by DEVS, both static and dynamic properties are defined in the hierarchy, i.e., the properties of the coupled component consist of the properties from the subs. Only at the atomic level, events can be specified in the way to update the dynamic properties or trigger another event to be scheduled in future time. At the level of coupled system, only connections among input and output events are to be formed.

For the proposed *master-worker* Sim-Opt structure, the coupled system is specified in Figure 3. Both static and dynamic properties at each component level, as well the coupling relationships are reported in the figure. Note that, the static properties are label as green, whereas the dynamic properties are in orange. The red labels indicates both input events that trigger the state change of components, and the output events that are triggered by the components. Zooming to the component level, Figure 4 shows the event diagram at atomic levels, which connects from the input to the output events for each of the components.

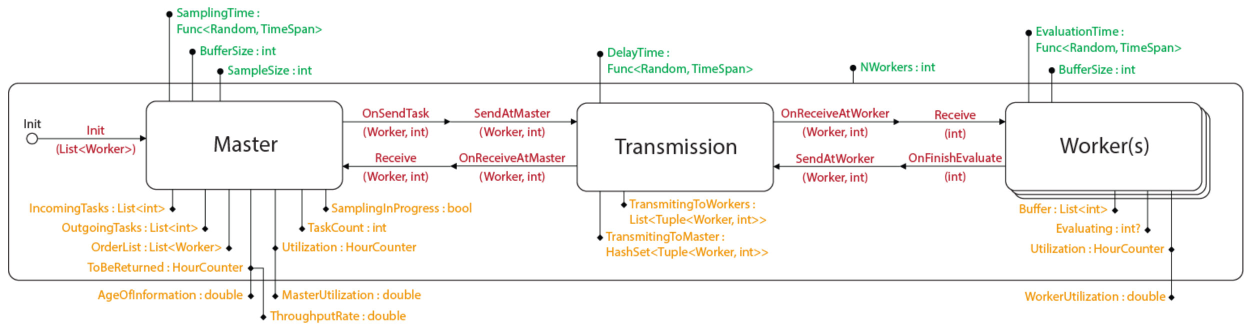
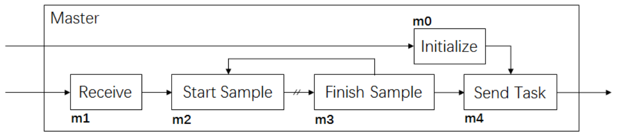
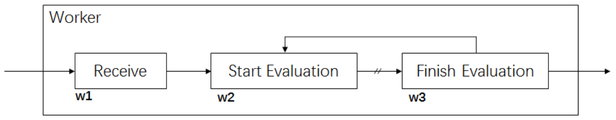

SamplingTime : Func<Random, TimeSpan>
BufferSize : int
SampleSize : int
DelayTime : Func<Random, TimeSpan>
NWorkers : int
EvaluationTime : Func<Random, TimeSpan>
BufferSize : int
Init
Init (List<Worker>)
Master
OnSendTask (Worker, int)
SendAtMaster (Worker, int)
Receive (Worker, int)
OnReceiveAtMaster (Worker, int)
Transmission
OnReceiveAtWorker (Worker, int)
Receive (int)
SendAtWorker (Worker, int)
OnFinishEvaluate (int)
Worker(s)
IncomingTasks : List<int>
OutgoingTasks : List<int>
OrderList : List<Worker>
ToBeReturned : HourCounter
SamplingInProgress : bool
TaskCount : int
Utilization : HourCounter
TransmitingToWorkers : List<Tuple<Worker, int>>
TransmitingToMaster : HashSet<Tuple<Worker, int>>
Buffer : List<int>
Evaluating : int?
Utilization : HourCounter
AgeOfInformation : double
MasterUtilization : double
ThroughputRate : double
WorkerUtilization : double

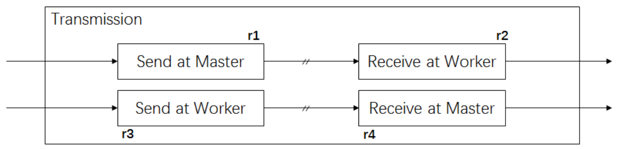
Figure 3: The O$^2$DES model with coupling of three atomic components.



(a) Master



(b) Worker



(c) Transmission

Figure 4: Event diagrams at the level atomic components.

In the following, we provide a precise definition for all static, dynamic properties and events for each atomic component. Note that across all description, $i$ is the index that denotes a *worker*, and $k$ refers to the evaluation task.

## 3.2 Master

**Statics** The static properties of the *master* define the settings of the *master* node, namely:

$m$ - sample size. We assume the optimizer receives $m$ evaluations before performing any sampling decision.

$\beta_m$ - buffer size. The maximum number of solutions to be sampled and that the optimizer can store waiting to be assigned to workers.

$T_s(m)$ - the random variable that describes the time required for the *master* to sample a new batch of tasks. It is reasonable to assume that the random variable is a function of the sample size $m$, and the expected time increases slower-than-linear, due to the scale of economy in the implementation of the sampling algorithm.

**Dynamics** The following variables denote the dynamic properties that define the runtime state of the *master*.

$M$ - the set of indices for incoming solutions, ordered by the time of receiving. It contains all tasks that have been received at the *master*, but have not been passed to the optimizer for updating and sampling, $\|M\| \leq m$.

$A$ - the set of indices for outgoing solutions, ordered by the time of sampling. It contains all tasks that have been sampled by the optimizer, but not been sent to workers for evaluation. Constrained by the specified buffer size, at any time $\|A\| \leq \beta_m$.

$O$ - the "backlog" list for *workers*, ordered by the time of logging. It contains the indices of the *workers* which have returned and are currently waiting for further candidates. The set allows duplicated items (i.e., a *worker* is waiting for multiple tasks to be sent to).

$P$ - a binary variable, indicates if the optimizer is sampling (i.e., 1 for sampling, 0 for idle).

$C$ - the total number of tasks that have been sampled.

$R$ - the number of tasks to be returned. It is the difference between $C$ and the total number of tasks that have been returned back to the optimizer. This variable is used mainly for calculating the *age of information*.

**Events** The following describes how dynamic properties are updated at each event. Refer to Figure 4(a) for the general triggering relationships.

$E_{m0}(\{1,\ldots,w\})$ - *Initialization event*, given a set of index of *workers* where $w$ indicates the number *workers*. The event sets $M \leftarrow \emptyset$ and $O \leftarrow \emptyset$, and proceeds with the immediate execution of $E_{m4}(i,k)$ for all workers $i$ in the following set:

$$i \in \{1,\ldots,w\} \text{ and } k \in \left\{ \sum_{i'=1}^{i-1} \beta_{i'} + i, \ldots, \sum_{i'=1}^{i-1} \beta_{i'} + i + \beta_i \right\}, \tag{2}$$

which sends just-sufficient tasks to fill all *workers* with their respective buffers. Subsequently, it sets $A \leftarrow \{\sum_{i=1}^{w} \beta_i + 1, \ldots, \sum_{i=1}^{w} \beta_i + \beta_m\}$, that fills the outgoing buffer; $R \leftarrow \max_{i \in A} i$, $C \leftarrow R+1$, and $P \leftarrow 0$.

$E_{m1}(i,k)$ - the event of receiving task $k$ from *worker i*. It firstly updates $M \leftarrow M \bigcup \{k\}$, and triggers an immediate execution of $E_{m2}$. If $\|A\| > 0$, remove the first element from $A$ as $k'$, and execute $E_{m4}(i,k')$; otherwise, set $O \leftarrow O \bigcup \{i\}$.

$E_{m2}$ - the event that attempts to start the sampling. The event proceeds only when $\|M\| \geq m$ and $P = 0$, otherwise no updating is incurred. It removes the first $m$ elements from $M$, set $P \leftarrow 1$, $R \leftarrow R - m$, and schedule a new event $E_{m3}$ after time $t_s$ sampled from $t_s \sim T_s(m)$.

$E_{m3}$ - the event to finish the sampling. Let $\delta = \min(m, \|O\|)$. Then for the first $\delta$ elements $i \in O$, iteratively execute $E_{m4}(i,C)$, update $C \leftarrow C+1$, and remove $i$ from $O$. In the case where $\delta < m$, set $A \leftarrow A \bigcup \{C, \ldots, C+m-\delta\}$. Then, execute $E_{m2}$ to attempt starting a new sampling.

$E_{m4}(i,k)$ - the event of sending the task $k$ to *worker i*. It is an output event which does not incur any updating at the atomic component level.

## 3.3 Worker

**Statics** The static properties of a *worker i* includes the following.

$T_e(i)$ - the random variable that describes the time required for the *worker i* to evaluate a given task. It is assumed that the evaluation time is independent of the tasks, where it can differ from other *workers*.

$\beta_w(i)$ - buffer size. The maximum number of tasks allowed to be received and temporarily store at the *worker i* before evaluation. The buffer is used to mitigate the time delay for the transmission and sampling.

**Dynamics** And the dynamic properties of *worker i* include

$B(i)$ - the set of indices for incoming tasks received by *worker i* and waiting to be evaluated, ordered by the time of receiving.

$q(i)$ - the index of the task currently being evaluated at *worker i*. It is set to 0 if no task is being evaluated.

**Events** For the events related to *worker i*, we have

$E_{w1}(i,k)$ - receive the task $k$ at the *worker i*. It update $B(i) \leftarrow B(i) \bigcup \{k\}$, following with execution of $E_{w2}(i)$.

$E_{w2}(i)$ - attempt to start evaluation at *worker i*. It proceeds only if $q(i) = 0$ and $\|B(i)\| > 0$. Remove the first element from $B(i)$ as $k'$, set $q(i) \leftarrow k'$, and schedule a new event $E_{w3}$ after $t_e$ sampled from $t_e \sim T_e(i)$.

$E_{w3}(i)$ - finish evaluating current task. It reset $q(i) \leftarrow 0$ and triggers an execution of $E_{w2}(i)$. This also serves as an output event for the *worker* component.

## 3.4 Transmission

**Statics** There is only one static property to define the transmission between the *master* and *servers*.

$T_\tau$ - the random variable that describes the time delay in the transmission. It is assumed to he homogeneous independent of *workers*, tasks, and the direction.

**Dynamics** Correspondingly, two dynamic properties are required.

$S_1$ - set of tasks transmitting from the *master* to *servers*. It consists of $(i,k)$ index pairs that specified the task and the *worker* it is sent to.

$S_2$ - set of tasks transmitting from *servers* to the *master*. It consists of $(i,k)$ index pairs that specified the task and the *worker* it is sent from.

**Events** The events related to the transmission are defined as following.

$E_{r1}(i,k)$ - the event to send the task $k$ to *worker i* from the master. It updates $S_1 \leftarrow S_1 \bigcup \{(i,k)\}$, and schedules $E_{r2}(i,k)$ after $t_\tau$ sampled from $t_\tau \sim T_\tau$.

$E_{r2}(i,k)$ - the event of receiving the task $k$ at the *worker i*, it removes $(i,k)$ from $S_1$ and serve as the output event of the transmission.

$E_{r3}(i,k)$ - the event to send the task $k$ from *worker i* to the master. It updates $S_2 \leftarrow S_2 \bigcup \{(i,k)\}$, and schedules $E_{r4}(i,k)$ after $t_\tau$ sampled from $t_\tau \sim T_\tau$.

$E_{r4}(i,k)$ - the event of receiving the task $k$ at the *master* from *worker i*, it removes $(i,k)$ from $S_2$ and serve as the output event of the transmission.

Note that, the connection among the three type of atomic components are specified in Figure 3. At the level of coupled system, the static parameter $n$ is used to defined the number *workers* in the system. And the coupled system is initialized by executing the event $E_{m0}(\{1,\ldots,w\})$ of the *master* where $w = n$.

## 3.5 Searching improved architecture parameters

Algorithm 1 illustrates a simple heuristic procedure that attempts to explore Pareto configurations aiming for balanced throughput rates between *master* and *workers*, as well as to exploit the utilization of the *workers*. In the specified algorithm, let $n_{\max}$ to be the maximum number of *workers* to be explored.

Alternatively, we could also apply existing multi-objective sim-opt algorithms in the literature, e.g., MO-COMPASS (Li et al. 2015), to search for the Pareto configurations. Indeed, the numerical experiments

**Algorithm 1:** Exploring Pareto Configuration for Parallel Sim-Opt Deployment

```
1  Initialize the set of candidate configurations $\Theta \leftarrow \emptyset$;
2  forall the $n \in \{1, \ldots, n_{\max}\}$ do
3      Let $m$ the minimum sample size that enable the master to reach the target throughput rate of
       the workers, i.e., $1/\mathrm{E}[T_s(m)] \geq \sum_{i=1}^{n} 1/\mathrm{E}[T_e(i)]$;
4      Set $\beta_m \leftarrow 0$, and $\beta_w(i) \leftarrow 0, \forall i \in \{1, \ldots, n\}$;
5      while true do
6          Record the candidate configuration $\mathbf{x} = [n, m, \beta_m, \beta_w(i)]$ as $\Theta \leftarrow \Theta \bigcup \{\mathbf{x}\}$, and evlauate
           corresponding $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$;
7          Observe the average utilizaiton $\mu$ of all workers, i.e., the proportion of time that $q(i) \neq 0$,
           break the loop if $\mu$ exceeds a threshold $\mu_0$, e.g., $\mu_0 = 0.99$;
8          Increase either $\beta_m$ or $\beta_w(i)$ by a unit, whichever results in a greater $\Delta f_1$ observed from the
           simulation evaluation;
9      end
10     Identify the Pareto configuration $\Pi \subseteq \Theta$;
11 end
```

in Section 4 shows that MO-COMPASS algorithm is able to achieve a better set of configurations compared with the results from the simple heuristic rule.

## 4 NUMERICAL EXPERIMENTS

To test the proposed framework, we apply the developed simulation model to evaluate a scenario specified by assuming that the sampling time, evaluation time, and transmission time follow truncated normal distributions (take only the positive part) with respective means and variance, specifically, $T_s \leftarrow \mathbf{Norm}(2 + 0.1 \times m, 0.1)^+$, $T_\tau \leftarrow \mathbf{Norm}(2, 0.3)^+$, and $T_e \leftarrow \mathbf{Norm}(1, 0.2)^+$. Note that, the model is flexible enough to tackle various distributions. In practice these can be observed from computational experiments with specific optimizers, simulators and computing infrastructure.

We set the number of workers to be $n = 4$. This is due to the fact that we are given the workers based on the architecture we can use. The decision variables are the sample size $m$, master and workers buffer sizes $\beta_m$, and $\beta_w$. We allow each of these parameters to vary in the range $\{1, \ldots, 20\}$, thus obtaining $20^3$ possible configurations.

The gray color "scatter points" in Figure 5 shows the performance for possible configuration, which provide an overview of how the solutions are distributed in the objective space, as well as to form a "true" global Pareto front. This is possible since there is no true simulation model running in this experiment. Nevertheless, in reality, this figure cannot be obtained, due to the fact that the evaluation task is really expensive. As a result, we need to be able to identify promising designs with far less evaluations.

Also, from the figures we can observe that in general a larger throughput rate translates into larger age of information, therefore a trade-off needs to be managed. To quickly identify a set of Pareto-efficient configurations, the heuristic rule as proposed in Algorithm 1 is applied, from which the results in shown on Figure 5 with the "circles". The result shows that the algorithm only works well towards the higher-end, where the *workers* are highly utilized. For comparison, we adopt the MO-COMPASS (Li et al. 2015) algorithm that is aiming to identify the optimal Pareto configurations. As shown by the "solid line" in the figure, with 360 sample points, the algorithm is able to identify a solution set that is close to the true Pareto front in terms of both dominance and spread.

In reality, the number of *workers* is usually a primary constraint due to physical and financial limitation, and thus a critical decision variable needs to be determined. Therefore, we extend our study to the case where the number of *workers n* varies.

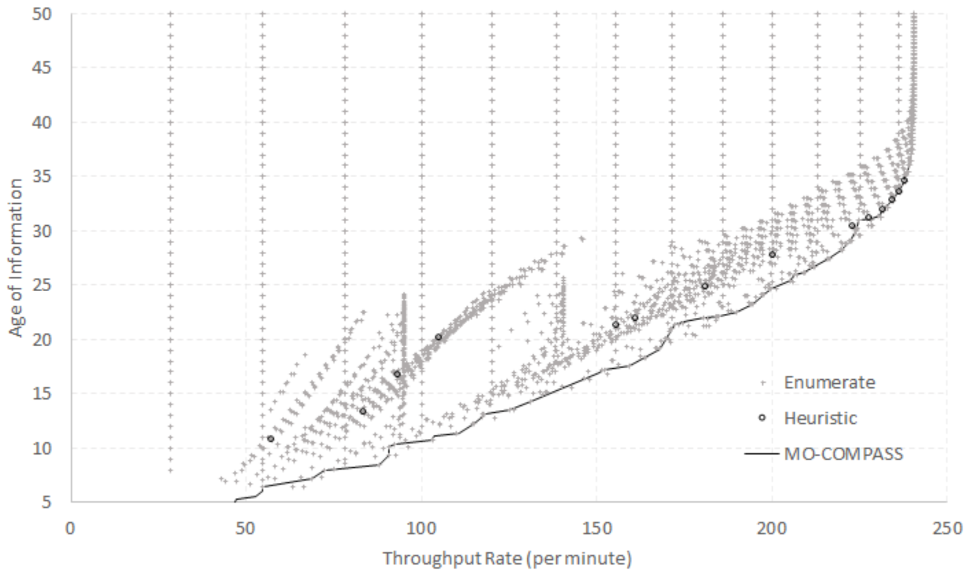


Figure 5: Performance of various configurations for parallel sim-opt with 4 *workers*, and comparison of optimization results.

We firstly apply the heuristic rule proposed in Algorithm 1 for *n* varying from 1 to 9 (as the maximum throughput is constrained at the *master* as below 600 per minute which corresponding to the capacity of maximum 9 *workers*). The performance of all candidate configurations explored by the heuristic rule is indicated as "circles" in Figure 6.

In addition to the heuristic algorithm, MO-COMPASS is applied with the number of *workers n* constrained at different levels, i.e., 4, 8, and 16 respectively. As different from the heuristic rule, we do not pursue the exploited utilization of the *workers*, but leave it to the search algorithm to explore the optimal configuration. The result is promising as the Pareto front identified from MO-COMPASS dominates the heuristic results.

Moreover, we identify that relaxing of the constraint on *n* helps to achieve a more efficient Pareto front. However, in practice, it may imply a higher cost. Therefore, this study provides evidence to show how much improvement in both criteria can be achieved by the additional investment in the number of *workers*.

On the other hand, from the comparison of experiment results, we also observe that ensuring the high utilization of *workers* is not always the best choice, as it comes with a cost of wasting the time value of the information.

## 5 CONCLUSION

In this study, we formulate and solves the design problem for a master-worker architecture dedicated to the implementation of a parallelized simulation optimization algorithm. In particular, we refer to the master-workers paradigm and identify as key performance indicators the throughput regarding the number of evaluations and the age of information. Also, we list the decision variables which characterize the design of the master-worker architecture.

To analyze the proposed paradigm, a discrete-event simulation model is constructed to mimic the execution of the simulation optimization algorithm over in a parallel master-worker environment. Alternative optimization procedures are proposed to identify the Pareto configurations under a specified scenario and constrained computational resources. As a result, the problem of optimal design of a parallelized sim-opt method is formulated as a simulation based optimization algorithm. Numerical experiments show that the problem is solved effectively with the suggested approach.

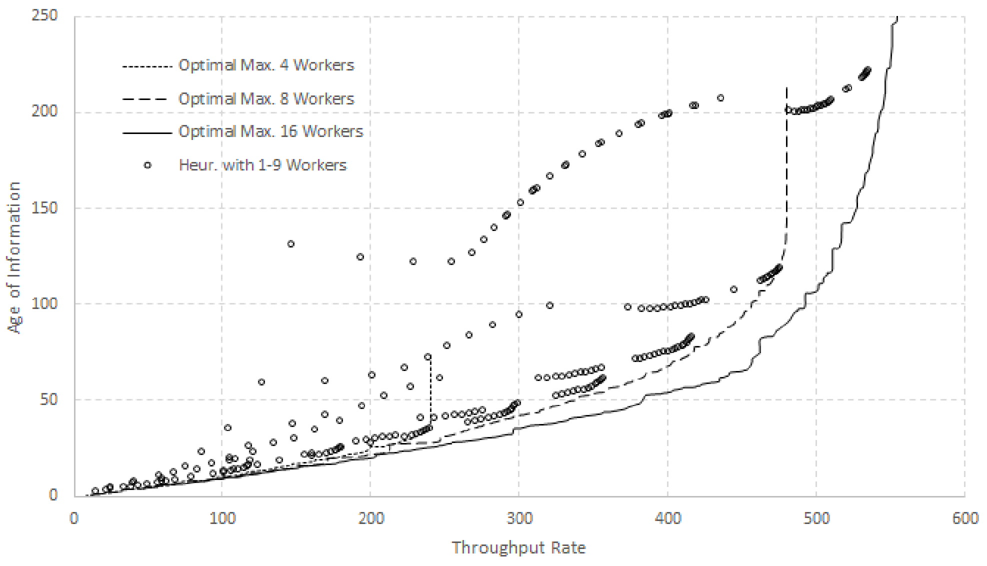


Figure 6: Comparison of optimized configurations for parallel sim-opt in the example scenario.

Future study will be directed to both the modeling of such execution according to stochastic processes as well as the analysis of the effect of dynamic parameters settings.

## REFERENCES

Laganá, D., P. Legato, O. Pisacane, and F. Vocaturo. 2006. “Solving simulation optimization problems on grid computing systems”. *Parallel Computing* 32 (9): 688–700.

Li, H., L. H. Lee, E. P. Chew, and P. Lendermann. 2015. “Mo-compass: a fast convergent search algorithm for multi-objective discrete optimization via simulation”. *IIE Transactions* 47 (11): 1153–1169.

Li, H., Y. Zhu, Y. Chen, G. Pedrielli, and N. A. Pujowidianto. 2015. “The Object-oriented Discrete Event Simulation Modeling: A Case Study on Aircraft Spare Part Management”. In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, 3514–3525. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Luo, J., L. J. Hong, B. L. Nelson, and Y. Wu. 2015. “Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments”. *Operations Research* 63 (5): 1177–1194.

Luo, Y.-C., C.-H. Chen, E. Yücesan, and I. Lee. 2000. “Distributed Web-based Simulation Optimization”. In *Proceedings of the 32nd Conference on Winter Simulation*, edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1785–1793. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mühlenbein, H. 1989. “Parallel genetic algorithms, population genetics and combinatorial optimization”. In *Workshop on Parallel Processing: Logic, Organization, and Technology*, 398–406. Springer.

Nelson, B. L., J. Swann, D. Goldsman, and W. Song. 2001. “Simple procedures for selecting the best simulated system when the number of alternatives is large”. *Operations Research* 49 (6): 950–963.

Ni, E. C., D. F. Ciocan, S. G. Henderson, and S. R. Hunter. 2017. “Efficient ranking and selection in parallel computing environments”. *Operations Research*.

Pierreval, H., and J.-L. Paris. 2000. “Distributed evolutionary algorithms for simulation optimization”. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 30 (1): 15–24.

Zeigler, B. P. 1987. “Hierarchical, modular discrete-event modelling in an object-oriented environment”. *Simulation* 49 (5): 219–230.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press.

## AUTHOR BIOGRAPHIES

**HAOBIN LI** is Scientist for the Institute of High Performance Computing, A*STAR Singapore. He received his B.Eng. degree (1st Class Honors) in 2009 from the Department of Industrial and Systems Engineering at National University of Singapore, with minor in computer science; and Ph.D. degree from the same department in 2014. He has research interests in operations research, simulation optimization and designing high performance optimization tools with application on logistics and maritime studies. His email address is lihb@ihpc.a-star.edu.sg.

**GIULIA PEDRIELLI** is Assistant Professor for the School of Computing, Informatics, & Decision Systems Engineering, Arizona State University. Her research focuses on stochastic simulation-optimization in both single and multiple–objectives framework. She is developing her research in meta-model based simulation optimization and learning for simulation and simulation optimization. Her email address is giulia.pedrielli@asu.edu.

**LOO HAY LEE** is Associate Professor for the Department of Industrial Systems Engineering & Management, National University of Singapore. He received his B. S. (Electrical Engineering) degree from the National Taiwan University in 1992 and his S. M. and Ph. D. degrees in 1994 and 1997 from Harvard University. He is currently a senior member of IEEE, a committee member of ORSS, and a member of INFORMS. His research interests include production planning and control, logistics and vehicle routing, supply chain modeling, simulation-based optimization, and evolutionary computation. His email address is iseleelh@nus.edu.sg.

**XIUJU FU** is Senior Scientist in Department of Computing Science at Institute of High Performance Computing, A-STAR. She received her bachelor and master degrees in Beijing Institute of Technology in 1995 and 1999, respectively. And she received her Ph.D. in Information System, School of EEE, Nanyang Technological University, 2003. Her research areas are in intelligent modeling and simulation based on data analysis, modeling/simulation and optimization techniques, with the applications in maritime port and transportation, logistics and supply chain, and public health. Her email address is fuxj@ihpc.a-star.edu.sg.

**XIAO FENG YIN** is Scientist for the Institute of High Performance Computing, A*STAR Singapore. He received both his PhD and his Masters degrees from Nanyang Technological University (NTU), Singapore. His research interests include maritime, logistics and supply chain optimization, planning and scheduling, knowledge discovery and artificial intelligence. His email address is yinxf@ihpc.a-star.edu.sg.