A Cloud-based Framework for Sensitivity Analysis of Natural Hazard Models

Ujjwal KC[a,1], Saurabh Garg[a], James Hilton[b], Jagannath Aryal[c]

[a]*Discipline of ICT, University of Tasmania, Hobart, Australia*
[b]*Data61, CSIRO, Melbourne, Australia*
[c]*Melbourne School of Engineering, University of Melbourne, Parkville, and Discipline of Geography and Spatial Sciences, University of Tasmania, Australia*

Abstract

Computational models for natural hazards usually require a large number of input parameters that affect the model outcome in a complex manner. The sensitivity of the input parameters to the output variables can be quantified using sensitivity analysis, which provides insight into the key factors driving the model and can guide modeling optimization. However, performing a sensitivity analysis typically requires a large number of simulations, which can be prohibitively time-consuming on workstations or local servers. To address this issue, this study proposes a Cloud-based framework that takes advantage of scalable Cloud resources. The efficacy of the framework is demonstrated by the scalability achieved while running large-scale wildfire simulations. Moreover, a comprehensive sensitivity analysis of the input parameters used in these models is presented. The ability to efficiently perform sensitivity analysis using the framework could allow such analysis to be performed as an

Email addresses: Ujjwal.KC@utas.edu.au (Ujjwal KC), Saurabh.Garg@utas.edu.au (Saurabh Garg), James.Hilton@data61.csiro.au (James Hilton), Jagannath.Aryal@unimelb.edu.au (Jagannath Aryal)

[1]Corresponding author

on-demand service for operational disaster management.

Keywords: Sensitivity Analysis, Parameter Uncertainty, Uncertainty Quantification, Wildfire Modeling, Cloud computing, Spark

---

Highlights

- Developed and investigated an efficient Cloud-based sensitivity analysis framework.
- Implemented various sensitivity analysis methods - Morris, Sobol, and FAST.
- Demonstrated the efficiency of the analysis of large-scale wildfire simulations.

Software Availability

The python library SALib can be accessed at `https://github.com/SALib/SALib`. The fire propagation tool *Spark* is available at `https://research.csiro.au/spark/`. The code for the framework is available upon request.

1. Introduction

Natural hazard models are required both for risk assessment to identify vulnerable regions and assets and faster-than-real-time operational applications during an unfolding disaster. These models are necessarily complex as many environmental factors must be taken into account. For example, wildfire models require input parameters such as the fuel condition, local

weather, the type of land coverage, and local topography [1, 2]. Each of these input parameters is subject to uncertainties that affect the outcome of the model, such as fire area or maximum heat intensity [3]. To take these uncertainties into account, ensemble predictions can be used, which perform multiple simulations with input values sampled from these uncertain inputs. Risk metrics can then be derived by performing statistical analyses on the ensemble predictions. However, the interrelation between the parameters and outputs from these models is complex and usually non-linear [4]. As such, running multiple simulations with different input combinations in an ensemble prediction, for risk assessment requires information on the sensitivity of the outcome of the model to various inputs.

Sensitivity analysis (SA) is one means to determine the influence of input parameters on a model outcome and its uncertainties [1, 5, 6]. In local SA, the impact of the parameters is studied around a specific point while in global SA, the entire range of the input parameters is considered [**?** **?**] This can provide crucial information about the range of use of the model [4][7], and help identify the parameters in the model on which the additional research must be focused in order to strengthen the knowledge base [8].

Conventionally, SA analyzes the variability of deterministic model outputs produced by possible combinations of the input parameters [9]. Computational natural hazard models are characterized by different, often complex, mathematical relationships that must be calculated multiple times for each combination of input parameters to produce a set of outputs. As natural hazard models often require a large number of input parameters, accurate sensitivity analyses require a large number of combinations, making such

analyses compute-intensive and time-consuming. These analyses can take several hours to days to complete for complex models. Such analyses also practically require a high degree of maintenance for data handling, orchestration, and management of results for the calculation of the final required metrics. The ability to automate SA and reduce the time taken for such analysis could benefit operational disaster management by rapidly determining the dominant factors affecting a particular local natural hazard to guide efficient response and planning.

Different methods such as variance-based sensitivity analysis [4][5], Bayesian analysis [10][11][12], Generalized likelihood uncertainty estimation (GLUE) framework and Metropolis algorithm [13][14], neural networks [15][16] and Taylor Series methods [17] have also been used for uncertainty quantification in an environmental context. Nossent et al. [4] performed Sobol' SA for flow simulations given by a SWAT model to calculate the sensitivity indices of 26 different input parameters. Similarly, sensitivity analysis of SWAT model was carried out in [18][19][20][21]. Yang et al. [22] assessed five different SA techniques applied to a hydrologic model. Brohus et al. [23] used the Morris method to analyze the sensitivity of fire dynamics simulation, while Hilton et al. [24] used polynomial chaos for similar models. Similar works have been done to perform SA of different fire models in [25][7][1]. These mentioned works have applied different sensitivity analysis methods to environmental models without directly considering the computational needs of such analyses.

Researchers have developed several methods and tools including Matlab-based [26][27][28] and Python-based libraries [29] to calculate the sensitivity

indices of input parameters of any environmental models. Wagener et al. [30] developed the Monte Carlo Analysis Toolbox (MCAT) enabled by a Matlab library of different visual and numerical analysis tools for sensitivity analyses of hydrological and environmental models. Another Matlab-based toolbox called *Eikos* [26] was developed by Ekstrom, which is capable of calculating the sensitivity indices of different models developed in Matlab/Simulink environments. D'Augustine has developed MATLODE [27] as a tool for SA of the models described by ordinary differential equations (ODEs) in direct and adjoint approaches. Pianosi et al. [28] constructed a Matlab/Octave-based toolbox called SAFE (Sensitivity Analysis For Everybody) (available now in R and Python as well) to improve the diffusion and quality of global SA in the environmental modeling community. Herman and Usher developed a Python framework called SALib [29], that facilitate the sensitivity analysis of environmental models using different existing SA methods. Roy et al. [31] developed a python-based Bayesian tool for uncertainty quantification. Andrianov et al. [32] developed an open-source software platform called *OpenTURNS (Open source Treatment of Uncertainty, Risk 'N Statistics)* that could treat uncertainty by dedicated to uncertainty treatment by probabilistic methods. Simlab [33] was developed as a free software package by the Joint Research Centre (JRC) of the European Commission. It generates a set of random samples of different parameters and the simulations can be run to compute the measure of sensitivity based on the method used. A package called *sensitivity* in R was developed by Iooss et al. [34] that can calculate the sensitivity indices using various popular methods. These tools and libraries can easily estimate the measure of sensitivity for mathematical

models and even for computational models but only after the sets of input and output values are available after model runs.

To deal with the high computational needs of the global SA of computational models, researchers have adapted a wide range of approaches. Stanfill et al. [35] proposed an easy to set up and inexpensive emulator based sensitivity indices estimators and applied the estimator to perform the sensitivity analysis to APSIM [36]. To deal with the curse of dimensionality in Global Sensitivity analysis, Sheikholeslami et al. [37] proposed a grouping strategy using boot-strapping-based clustering to enable GSA to high-dimensional environmental models. Saltelli et al. [5] highlighted the importance of using surrogate models with a subset of input factors that contribute to most of the variability of model output for model simplification. Efforts have been made to estimate different measures of sensitivity using generic sets of model input and output sets. Pianosi and Wagener [38] improvised their density-based sensitivity measure method (PAWN [39]) with an approximation measure such that the method was applicable to a generic sample of inputs and output for a model. Borgonovo et al. [40] proposed an ensemble of sensitivity measures, based on the different purposes (parameter prioritization, trend identification, and interaction quantification), to provide insights into environmental models without increasing the computational burden. The approach in the work used data-driven estimation of global sensitivity measure along with hybrid local-global method DELSA [41] such that the ensemble of sensitivity measures could be estimated simultaneously. Eldred et al. [42] proposed a multi-level parallel object-oriented framework called *DAKOTA* that provided an extensible interface between

simulation runs and iterative sensitivity methods. The framework enabled a problem-solving environment for performance analysis of computational models, but on high-performance computers. All of these efforts addressed the high computational needs of global sensitivity analysis with various approximation methods and approaches to better estimate the effects.

Cloud Computing has come forward as an attractive solution to support high computational demands with its almost unlimited scalable compute resources, storage, and network capacity. Several studies have verified the capability of Cloud Computing to accommodate the computational complexities of different environmental models [22][43][44]. Consequently, global SA of computational models, previously thought to be very difficult (or infeasible) [45, 46], can be conducted on the Cloud. However, to authors' knowledge, there are no systems or services that offer such analyses in a scalable, time-efficient, and convenient manner. As such, this study proposes a cloud-based framework that can efficiently handle the high computational need of a large number of environmental model simulations. The framework uses scalable Cloud resources to run the computational models with sampled input set to obtain the set of output values for further analyses in a time-efficient manner, which would take several hours to days in a conventional system. The set of input values to the model can be sampled as required and the set of output values obtained after numerous model runs, along with input sets, can be used for various mathematical analyses including sensitivity analyses using different global SA methods. In our work, to validate and demonstrate the capability of the framework, we utilize the sets of input and output values of the model to calculate the sensitivity indices of input parameters to model

output using a set of different popular SA methods. These are the Morris method [47], the Sobol' method [48] and the Fast Amplitude Sensitivity Test (FAST) [49]). These methods are chosen as a modular block in the framework based on the standard comparison presented in [50] that highlights the suitability of SA methods for different purposes (ranking, screening, and mapping) with the trade-offs between accuracy and cost taken into consideration. The sampling strategy and index calculation are customized based on the user input and method chosen before a job is launched in the framework. All data management and intermediate calculations are automatically handled to produce the metrics from the SA method. The framework is demonstrated specifically here for sensitivity analysis of wildfire models using the Spark wildfire modeling system, although the method can easily be extended to other natural hazard models. The model input and output set obtained after the model runs in the framework can be further analyzed using any suitable approaches.

In the following section, different SA methods used in the framework are detailed, and the framework software is described. The framework is then applied to a wildfire natural hazard model and, finally, the implications of the analysis in the context of wildfire modeling are discussed.

2. Sensitivity Analysis Methods

Sensitivity Analysis (SA) deals with the study of the variation or uncertainty in the model output due to the variation in one or more input parameters. The global SA methods overcome the limitations of local SA such as linearity, normality assumptions, and local variation and are widely used

for sensitivity analysis of parameters in different models [51]. We consider three widely adapted global SA methods (one-at-a-time and variance-based) [4, 28], detailed in the following sections.

2.1. Morris Method

Morris Method [47] is one of the screening-based SA methods. It is often called 'one at a time' (OAT) analysis as each input parameter is varied while keeping the other parameters constant during the model runs. This method classifies the input parameters into three distinct categories - input parameters with negligible effect, parameters with large linear effects without interactions, and parameters with large non-linear and/or interaction effects. The method calculates the sensitivity indices for the parameters j in terms of mean (μ_j^*) and standard deviation (σ_j) of the absolute value of the elementary effects. μ_j^* is the measure of the effect of j^{th} input parameter on the output, where greater values indicate a greater influence of j^{th} input parameter on the variability of the output. σ_j is the measure of the non-linear and interaction effects of the j^{th} input parameter. Smaller values of σ_j signify fewer interaction effects, while higher values of σ_j signify higher interaction effects with at least one other input parameter and/or non-linearities.

For a sample size argument of N (N samples within the range of input and k parameters in a model, calculation of sensitivity indices in Morris method requires $(N + 1) \times k$ model runs [47].

2.2. Sobol' Indices

Sobol' SA [52] is a variance-based SA method that quantifies the input and output variability as probability distributions. The analysis breaks the

output variability into the individual input variability and the variability caused by the interaction between the inputs. Consequently, the method quantifies the variability of the input parameters in terms of first-order indices, second-order indices, and total sensitivity indices. The first order index $S1_j$ defines the variability of the model output caused by the variability of input parameter j without considering any interaction with other input parameters. The second-order index $S2_{i,j}$ explains the variability in the model output caused by the non-linear interaction between parameter i and parameter j. The total sensitivity index ST_j defines the total variability caused by the variability in the input parameter j and its non-linear interaction with one or more other input parameters.

For a sample size argument of N and k parameters in a model, calculation of the sensitivity indices requires $2N(k+1)$ model runs if the calculation of second-order indices is enabled [53]. The number of model runs needed is $N(k+2)$ if the calculation of second-order indices is disabled [53]. The second-order index calculation is enabled throughout this study.

2.3. Fourier Amplitude Sensitivity Test (FAST)

Fourier amplitude sensitivity test (FAST) is a variance-based global sensitivity analysis method. It defines the sensitivity indices based on the conditional variance of the input parameters indicating the individual or joint effects of the parameters on the model output. FAST first uses coefficients of multiple Fourier series expansion of the model output function to represent the conditional variances of the inputs. It then applies the ergodic theorem to transform the multi-dimensional integral to a one-dimensional integral for the evaluation of the Fourier coefficients [49]. The continuous integral

function can be recovered from a set of finite sampling points if the Nyquist-Shannon sampling theorem [54] is satisfied. The integral can be evaluated from the summation of the function values at the generated sampling points. FAST gives the indices in terms of first-order indices $S1$ and total effect indices ST. $S1$ quantifies the standalone impact of an input parameter, while ST measures the overall impact of the parameter, including the effects of its non-linear interactions with other parameters.

For a sample size argument of N and k parameters, the calculation of the sensitivity indices in FAST requires $N \times k$ model runs [55].

3. Cloud-based Framework

Our Cloud-based framework enables sensitivity analyses of natural hazard models using various well-established methods, as explained in the previous section in a time-efficient and convenient manner to address the prohibitively time-consuming issue of such analyses. The components of our Cloud-based SA framework are shown in Figure 1. The framework handles the computational complexities of multiple model runs among the distributed Cloud resources and calculates the sensitivity indices for the input parameters to the model. The user uploads a configuration file for running the models and enters the required inputs into a web interface. These are - 1) the SA method to be used, 2) the required sample size and 3) the number of input parameters. In the framework, three different SA methods are implemented. The sample size input allows the user to specify the total number of samples of the inputs within a predefined range. The user can also specify the number of input parameters for the model through the interface, which, together

with the sample size, defines the total model runs required for the analysis. It should be noted that the number of total model runs can be different for different SA methods due to differences between the SA algorithms.

A *Master* retrieves the user input and generates the required samples from the possible input parameter combinations for the SA method selected. The Master then distributes the required model runs to several *Workers* (or Cloud instances) to complete all the required model runs in a time-efficient manner. The Master finally collects the model outputs from all the workers and calculates the sensitivity indices for the input parameters. The calculated indices are stored and can be downloaded from the web interface by the user. In addition to the calculated indices, the user can download the model input and output set of values to perform further relevant analyses. The components description and the features offered by the framework are described further as follows.

3.1. Web Interface

Users initiate a service request for the calculation of sensitivity indices through a Web Interface. The Web Interface is the only point of interaction between the users and the framework, encapsulating all operations within a graphical user interface. Users can initiate a request by uploading the required configuration and input files into the web interface and launching a job. The interface reflects the status of the service request at different instants of time during the operation. Finally, users can download a text file containing sensitivity indices after the execution of the model runs from the web-interface. Moreover, the user can also download the input and output set of values for the model from the master using the interface.

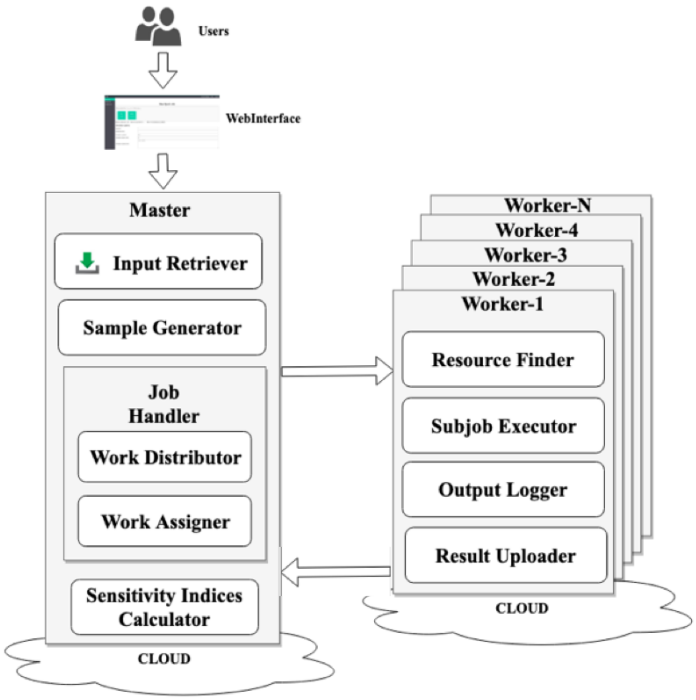


Figure 1: Proposed Framework. A master-slave based framework where master assumes all the control functions and slaves executes multiple model runs and sends the output variable to the master for the calculation of sensitivity indices.

3.2. Master

The Master is the central point of the proposed service framework, controlling how the system serves the service requests in an efficient, scalable, and timely manner. Based on the user input, the Master generates required input parameter combinations. It then divides the required model runs into several sub-jobs, assigns these sub-jobs to multiple Workers, collects the model outputs from the workers upon the completion of the execution, and calculates the sensitivity indices using these outputs. The Master makes use of different mechanisms to distribute the computational complexity of a large number of model runs over multiple Cloud Workers.

The *Input Retriever* retrieves key information from the files uploaded and input fields in the web-interface as per the service request (*job*) initiated by the user. Based on the information retrieved by the Input Retriever, the *Sample Generator* generates sets of input parameter combinations within predefined ranges for the SA method. Each combination results in one model run, producing one model output. It should be noted that for different SA methods chosen, the total number of samples (combinations) generated is different even for the same sample size. For example, for a sample size of 1000, the total number of input parameter combinations generated for the Morris method is 4000, while the number is only 3000 for the FAST method (for three input parameters in the model).

The *Job Handler* manages the computational complexity of each job by creating multiple independent tasks with a fixed number of model runs, referred to as a *subjob*. Each subjob contributes a fraction to the job. The subjobs are independently executed in multiple workers. The Job Handler

consists of two sub-components - the *Subjob Creator* and the *Subjob Assigner*. The Subjob Creator creates several independent subjobs ($S_1, S_2, ..S_N$) with each subjob possessing their respective sample combinations. The Subjob Assigner finds suitable workers for each subjob and assigns the subjob to the worker for the required number of model runs. In the framework, a suitable worker can be a new Cloud instance or an idle worker within the system.

Upon completion of all the required model runs, the *SA Indices Calculator* aggregates the model outputs from the files uploaded by the workers. This component uses SALib python library to calculate the sensitivity indices for the input parameters of the model. The calculated indices are stored and can be downloaded by users through the web-interface.

3.3. Workers

Workers are the Cloud instances created by the Master to execute the model runs to produce outputs. After the subjobs are assigned, the Workers find and download the required files. The Workers then execute the models multiple times (under a subjob), collect the model outputs, and upload the input combinations along with the respective outputs and time information to the Master. Each worker operates independently within the framework. It is noteworthy that the workers should have the computational model tool pre-installed on them. The workers have sub-components assuming different functions.

The *Resource Finder* finds all the necessary relevant files in the Master, based on the identifier attached to the subjob assigned for the worker and downloads them in the respective directories in the worker. The *Subjob Executor* runs the model in the worker for as many input parameter

combinations in the file downloaded by Resource Finder. The model runs can run as an ensemble to save the time required for multiple data fetch, as one data fetch is enough for all the model runs in such mode. The *Output Logger* employs a text processor to extract the reduced information on the input parameters' combinations, the model output produced by the respective combination, and the time taken for each model run. The reduced information makes the data exchange between Workers and Master more efficient. The *Result Uploader* sends out the requested information extracted by the Output Logger to Master, where the results are stored in a centralized fashion.

3.4. System Setup

Algorithm 1 outlines the steps used to perform the sensitivity analysis of an environmental model in the framework. The symbols used in the algorithm are listed in Table 1. Java is the main programming language used to enable different mechanisms within the framework. Python scripts are used to generate the samples of input parameters' combinations and calculate the sensitivity indices using SALib. Python is used as a programming tool for text processing and synthesis. Nectar Cloud [56], an OpenStack-based Cloud infrastructure, is used to provide the Cloud resources for the model runs to produce the model outputs. For simplicity, we use only one kind of instance flavor (m2.small) for the experiments. The setup can be easily extended to accommodate different types of instance flavors for further optimizing the resource utilization and operation time and cost within the framework. The creation of new Cloud instances is handled by JClouds, which provides Java-based wrapper APIs for OpenStack. The web-interface of the proposed

service framework is implemented using VueJS to offer concurrent access to multiple users. The Spark modeling framework is pre-installed on the Cloud image, based on which the new instances are created.

Algorithm 1 Calculation of Sensitivity Indices

Input: $[u, N, k, Method]$

Output: $[Si_1, Si_2, ..S_k]$ (Sensitivity Indices)

Master:

For every service request u_k, Retrieve the values of N, k and $Method$
if Method == 'Sobol' **then**
    Generate $2N(k+1)$ input parameters combinations
else if Method=='Morris' **then**
    Generate $N(k+1)$ input parameters combinations
else if Method=='Fast' **then**
    Generate $N \times k$ input parameters combinations
end if
Calculate $N_S = min\{10, \lceil \frac{\#Samples}{x} \rceil\}$
Divide samples into N_S batches and create N_S subjobs $S_i ... S_{N_S}$
Find N_S workers (W_i) and assign S_i to worker W_i,
For every file uploaded by worker W_i, check if $\#files == N_S$
if $\#files == N_S$ **then**
    Calculate sensitivity indices $Si_1, Si_2, ..S_k$
end if

Worker W_i:

Find Configuration file and sample file F_i in the Master
Download files in respective directories
Execute subjob S_i
For each model run r_c, Extract input combination, model output and time information
if S_i == completed **then**
    Upload reduced result file rf_i to Master
end if
Make worker W_i free and available for other subjobs

Table 1: Description of Symbols used

Symbols	Description
u	User Request
N	Sample Size Argument
k	Number of model parameters
$Method$	SA Method
Si_i	Sensitivity Index for parameter k_i
N_S	Number of subjobs for a user request u
$\#samples$	Size of combinations generated
$\#files$	Number of uploaded result files
x	Number of model runs in each subjob
S_i	i^{th} subjob
W_i	i^{th} worker for user request u
F_i	Sample File for subjob S_i
r_c	c^{th} model run in any subjob
r_{f_i}	Reduced result file for subjob S_i

4. Framework Application Use Case

In this section, we describe the application of our Cloud-based SA framework to wildfire models and analyze the performance of the framework for different SA methods and sample size.

4.1. Wildfire model

The Spark [57] wildfire modeling system is used to simulate the example of natural hazards for the SA Cloud framework. Spark is a flexible platform for simulating wildfires allowing different types of fire behaviour to be defined using scripts, including rates-of-spread in different fuel types, firebrand dynamics, and risk metrics for fire impact and severity. Simulations in Spark typically require several input data sets for the fire behaviour models, including maps of the land classification, fuel type, topography, fuel informa-

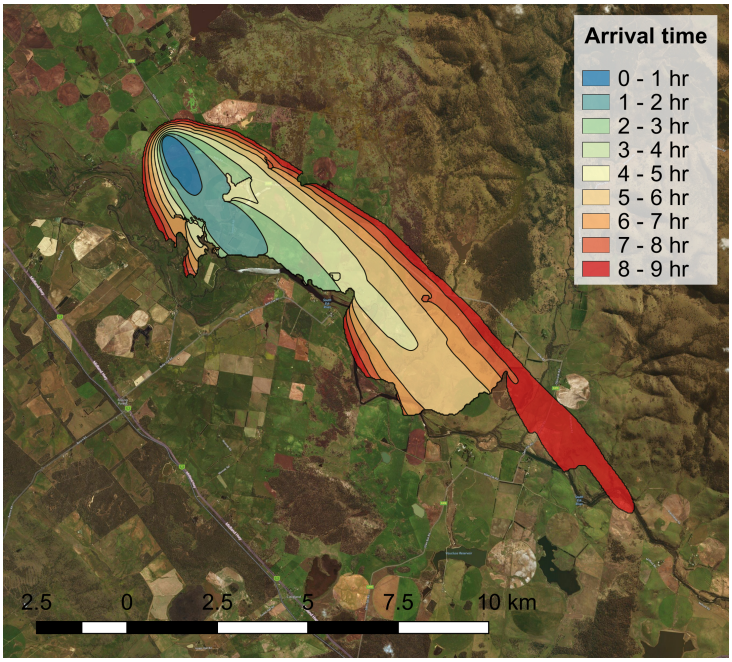


Figure 2: Visualization of the spread of fire in Spark for a location in Tasmania, Australia. The colour scale indicates the time of arrival of the fire, with blue being the area covered in the first hour and red the final hour of a nine-hour simulation. The fire is constrained to the south by river.

tion, and meteorological data. Calculations in Spark are parallelized using the OpenCL framework to enable the efficient execution of the simulations. Figure 2 shows an example simulation for the predicted areas burnt over different periods of time.

For an example of SA analysis, an area in Tasmania, Australia was chosen. Tasmania is one of the most wildfire-prone regions in Australia during the fire season. From 2018 to 2019, 841 wildfires were reported, and 310,311 hectares were burnt by wildfires [58]. As a part of their ongoing effective wildfire management strategy, the Tasmania Fire Service (TFS) and State

Emergency Service (SES) have been actively working to create and manage high-quality land data sets relevant to wildfires which were used for this study. The simulations used a number of different empirical fire models for fuels found in Tasmania. Vegetation types from the TasVeg data set [59] were mapped to a number of Australian empirical fire spread models. These were the McArthur [60] and Dry Eucalypt model [61] for forest, a model for buttongrass moorland [62], a model for heathland [63] and grasslands [64].

The parametric sensitivity study was conducted for the meteorological data inputs common to all the empirical models used: the air temperature, relative humidity and wind speed. The simulations were run for nine hours at a specified single start location within Tasmania. The total fire area (in hectares) burned by the wildfire was considered as the output variable for each simulation in Spark. The ranges of weather data used were based on observations by McArthur [65] and reported in [66]; these are listed in Table 2. For simplicity, we assigned a uniform distribution to the parameters while creating samples for the analysis. These distributions, as well as the ignition location of the wildfire, can straightforwardly be changed and the values used here are simply to demonstrate the utility of the framework.

Table 2: Probability Density Function (PDF) of Input Parameters

Parameters	**pdf**	**Range**
Temperature	Uniform Distribution	[10, 40]
Relative Humidity	Uniform Distribution	[10, 90]
Wind Speed	Uniform Distribution	[10,60]

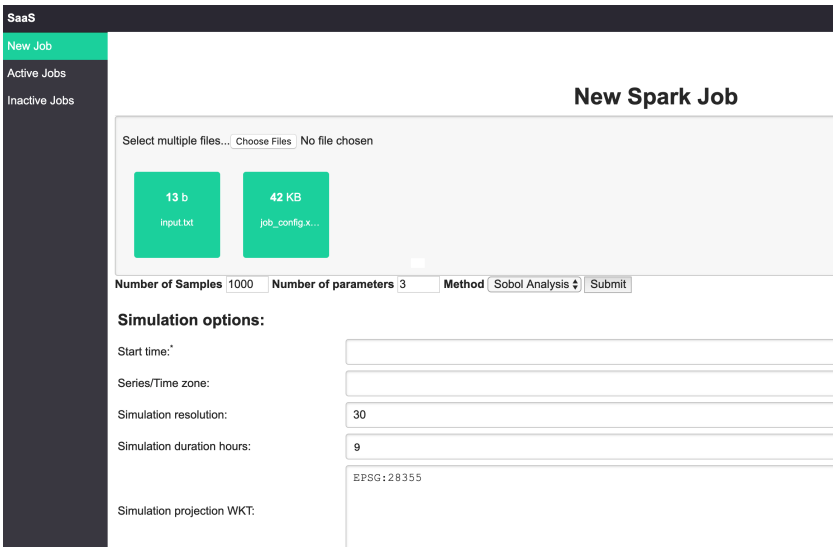


Figure 3: User Interface. A user uploads the required configuration file for Spark simulation and enters the sample size argument and desired SA method to run the analysis as a new job in the framework.

4.1.1. Calculation of Sensitivity Indices at Sample Size Argument = 1000

For a SA calculation of sample size $N = 1000$ the numbers of model runs required were 8000, 4000, and 3000 respectively for Sobol, Morris, and FAST method. Here, the sample size argument of 1000 has been chosen to reflect the high computational demand for sensitivity analyses. Further analysis on the choice of the sample argument for convergence is included in Section 5.2. The value can be changed to suit the nature of analysis to be carried out. To perform the SA, a service request was initiated in the framework by uploading a configuration XML file and input file (with information about the sample size argument, number of parameters, and SA method) into the web-interface as shown in Figure 3. For this study, the value of x (total number of total model runs in a worker) as defined in Algorithm 1, was taken as 100. The effect of x on the overall time performance of the framework is detailed in a subsequent section. Based on the value of x and the total numbers of samples created, the Master creates a corresponding number of subjobs and assigns them to the Workers. Table 3 lists the values of sample size and the total number of subjobs/workers created for different SA methods. Upon completion of the models runs in the workers, Master combines the result files and calculates the sensitivity indices, which can be downloaded from the web-interface, as shown in Figure 4. In the framework, we use the cloud instances of flavor type *m2.small* with 1 VCPU, 4 GB RAM, and 10 GB memory Ubuntu 16.04 LTS 'Xenial' amd64. The discussion on the analysis of the sensitivity indices is made in the next section.

Figure 5 represents the total time taken by the Cloud framework using a sample argument of 1000. The total time includes the time taken for the

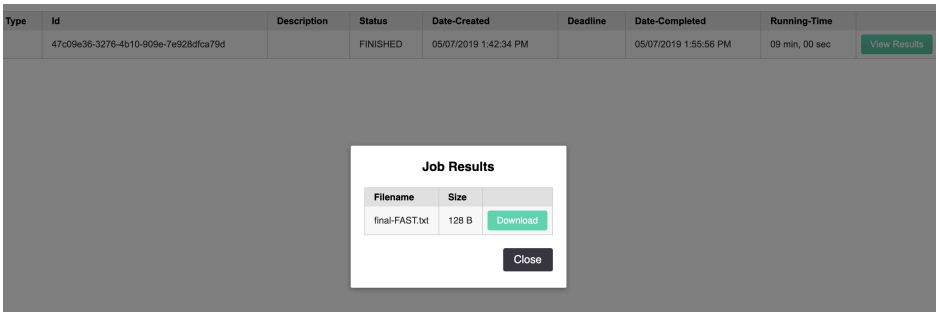


Figure 4: A Sample Downloadable File. After the completion of the job execution, the user gets to download a text file with the values of sensitivity indices calculated based on the chosen SA method.

Table 3: Total model runs (N) and workers for different SA methods

S.N	SA Method	Model Runs	Workers/Subjobs
1	Sobol	8000	80
2	Morris	4000	40
3	FAST	3000	30

creation of new Cloud instances, downloading the files, required model runs, and calculation of the indices. The infrastructure used for the study, Nectar Cloud [56], can experience delays when required to create a large number of instances simultaneously. Such delays appear due to various hardware and physical limitations, including memory size. As such, the time required for the creation of new instances varies from 1 minute to 5 minutes. Due to selective downloads, the time needed for downloading the required files and resources is minimal (a few seconds).

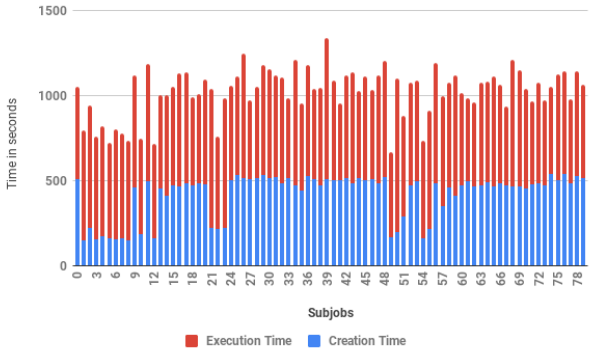
Since the indices are calculated only once after the completion of all the subjobs, the time required for indices calculation is also minimal (1 second). A typical user request for calculation of Sobol indices for a sample argument

of 1000 takes around 22 minutes, while the same for Morris method takes around 36 minutes. The calculation of the indices using the FAST method takes around 17 minutes. The values of input parameters govern the fire simulations in Spark, and the overall simulation time is strongly dependent on the various combinations of these input parameters. This dependency explains the difference in the time performance of the framework even when Workers have subjobs with the same number of simulations.

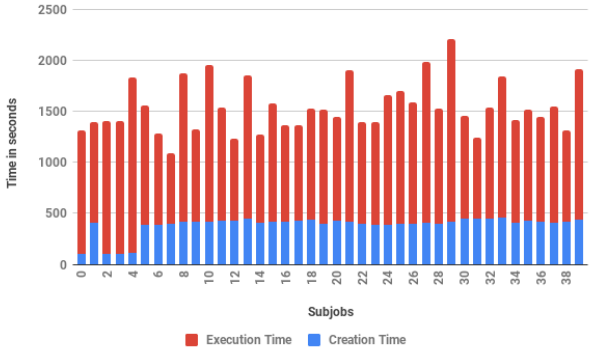
Figure 6 compares the total time taken for the SA using the proposed Cloud framework and performing the analysis on a single local machine for a sample argument of 1000 and three different SA methods. For rational comparison, we consider a single local machine with the same hardware specifications as the Cloud instance has (4 GB RAM, 1 VCPU, and 10 GB memory, Ubuntu 16.04 LTS 'Xenial' amd64). For the same set of input parameter combinations, the Cloud framework takes only 3.0% of the time taken by a comparable local system for calculating the indices using Sobol Method. This comparison includes the time taken to create the instances within the framework. Moreover, the Cloud framework further decreases the waiting time for SA using Morris and FAST method as the Cloud framework takes only 4.5% and 6.3% of the time taken by a single machine. In addition to the improvement in waiting time, the Cloud framework offers the benefits of flexibility, scalable resources, ease of use, and efficient handling of model outputs.

4.1.2. Performance Analysis

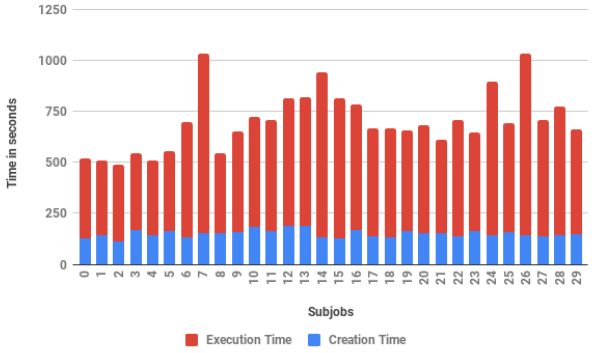
In this section, we analyze the performance of the framework by varying the sample size argument (N) and the number of simulations (x) in a subjob.



(a) Sobol Method



(b) Morris Method



(c) FAST Method

Figure 5: Time required for calculation for SA indices ($x = 100$). The time required for the calculation of the SA indices varies based on the SA method chosen, which is contributed by different sampling methods. The Cloud instances in Nectar Cloud take more to start up when subjected to a large number of simultaneous spun-off requests.

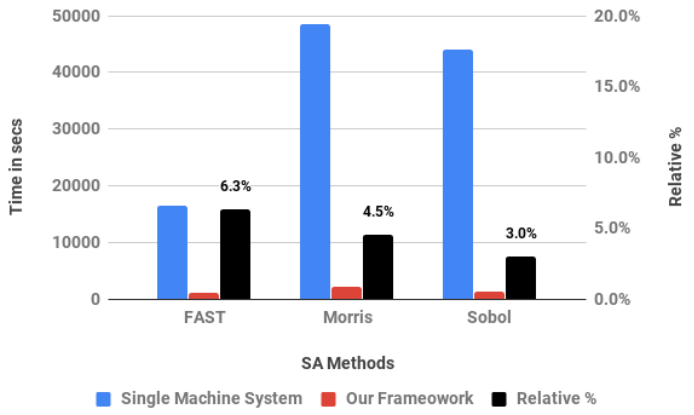


Figure 6: Time Performance Comparison of our framework against a single-machine system. Our framework completes the analysis in 3-7% of the total time taken by a local system with a single machine, which is at least 15 times faster. The framework offers additional benefits of flexibility and convenience.

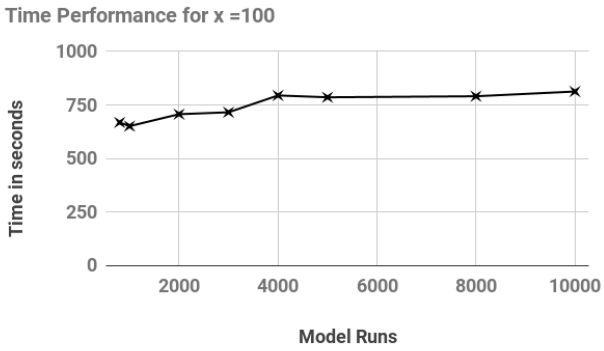
In our study, creation time is the time required to create a cloud instance after the request has been initiated while execution time is the time taken by a worker to execute all simulations in a subjob. The execution time includes data fetch time and computative cycle time for all the simulations as explained in [44]. Additionally, we present the impact of parallelizing the model runs in a distributed computing environment of the Cloud. The Cloud instance creation time does not affect the distribution of simulations among the workers and thus, the time taken for the creation of the instances is not considered for the analysis of the impact of parallelization of the model runs. The Cloud instances are assumed to be available and ready to run the models.

The change in the number of sample size argument ultimately changes

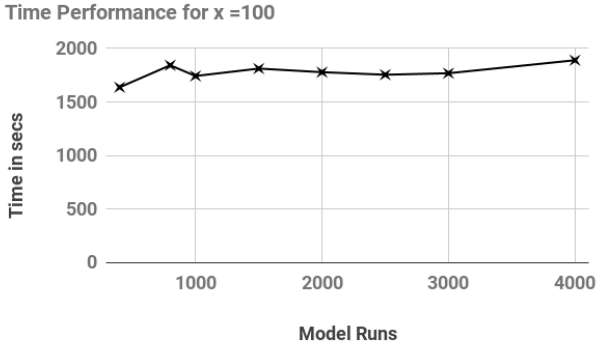
the total number of model runs for the analysis. The time taken for the calculation of the SA indices for the input parameters for the varied number of samples (simulation runs) is represented in Figure 7. Figure 8 represents the time taken for the framework to complete the analyses for different values of x.

In Figure 7, it is evident the change in the total time taken for the sensitivity analysis is not directly proportional to the change in the number of model runs in a job. The maximum absolute difference in the operation time for a varied number of sample sizes (model runs) for the Sobol method is 162 seconds (Figure 7a). For Morris, the performance analysis shows that the total operation time has changed by 252 seconds (see Figure 7b) when the total model runs changed from 400 to 4000. The same statistics for the FAST method stands at 222 seconds (Figure 7c). Even when the total model runs increased by a factor of 10, the total operation time in the framework did not increase in the same proportion. The Cloud framework distributes the increase in the computational complexity with increasing model runs over multiple Cloud instances. As such, the entire analysis is completed in a time-efficient manner for a large sample size argument. However, there are relative differences between the operation time for each method, which are the result of various combinations of parameter samples resulting in longer simulations in the same worker. Currently, there are no optimizing mechanisms in the framework that intelligently distribute the parameter combinations. This will be the subject of future work to improve the performance of the framework.

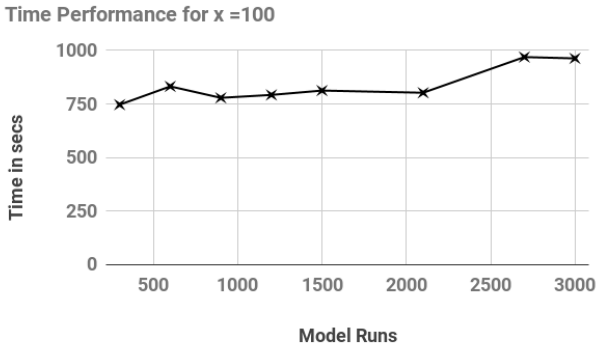
In Figure 8, it is clear that the number of model runs in a Worker, x, has a significant impact on the total time taken for a SA request. The total time



(a) Sobol Method



(b) Morris Method

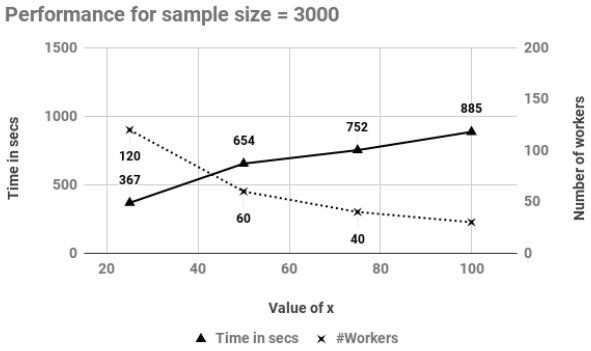


(c) FAST Method

Figure 7: Variation of total operation time with the sample size (for x=100). There is a variation of total operation time with the change in the value of x but, even when the total model runs (N) increased by a factor of 10, the framework distributes the computational complexity of the analysis over more number of Cloud instances and finishes the entire operation in a time-efficient manner.

(a) Sobol Method

(b) Morris Method



(c) FAST Method

Figure 8: Variation of total operation time with values of x (for $N = 1000$). The total operation time increases with the increase in the number of model runs in a subjob (running in a worker) but, the total workers allocated for the job decreases with the increase in the value of x.

taken for the completion of a subjob (with multiple model runs) increases with an increase in the number of the model runs in the subjob. The same applies to all the methods in the framework, where the total operation time consistently increases with the increase in the value of x. The number of workers required to serve the requests decreases with an increase in the value of x, as shown in Figure 8b. The increase in the operation time is non-linear and appears to be due to competing data fetching and computing requests on the Cloud instance from the multiple subjobs. Future work will aim to investigate this effect to optimize the size of the subjobs and allocation to the Cloud resources.

4.2. Impact of Parallelization of Model runs

Spark consists of a data fetch and computative cycle [44]. The system can be configured to run N simulations on a single machine, requiring only a single data fetch followed by N sequential simulations. On the Cloud, a job with N simulations can be divided into batches of size n where only one data fetch cycle is required for all simulations in the batch. Each simulation batch can be considered to be a parallelizable task and run in individual workers. The choice of the value of n depends on the availability of the workers, the desired time of job completion, and resource utilization within the system. As each batch, rather than the components of each simulation, can be parallelized on the Cloud the classic Amdahl's law relation [67] cannot be applied to calculate a relative speed up factor. Instead, we define a speed up factor involving the distribution of jobs to M nodes and the possible execution of n multiple simulations on each node. The speed up factor, s used here is the ratio of the time taken to complete the job in a single-machine

system, T_{single}, to the time taken to complete the job in our framework with multiple Cloud workers, T_{cloud}. The speed up factor s represents the factor by which the time required for the completion of the entire job improves when compared to execution in a single-machine system.

The time taken for a single simulation consists of the fetch time T_{fetch} plus an average time for a simulation, T_{sim} (for this analysis, we generalize the time taken for model runs and use an average unit execution time for T_{sim}). The fetch time T_{fetch} can be considered to be a constant term based on the type of the instance used. For N total simulations on a single-machine system the total time, T_{single}, is therefore $(T_{fetch} + NT_{sim})$. For the Cloud system, all the workers run in parallel (the time for the completion of the job would be the maximum of the time taken by each worker) and thus, the time taken, T_{cloud}, for N total simulations distributed over M nodes each carrying out $n = N/M$ simulations is:

$$T_{cloud} = T_{fetch} + nT_{sim} \tag{1}$$

The speed-up factor is therefore:

$$s = \frac{T_{single}}{T_{cloud}} = \frac{T_{fetch} + NT_{sim}}{T_{fetch} + nT_{sim}} \tag{2}$$

At the greatest possible cloud utilisation, $M = N$ giving $n = 1$ and an overall theoretical maximum speed up factor of:

$$s = \frac{T_{fetch} + NT_{sim}}{T_{fetch} + T_{sim}} \tag{3}$$

In the large simulation limit of $N \to \infty$ Eq. (2) gives:

$$\lim_{N\to\infty} \frac{T_{fetch} + NT_{sim}}{T_{fetch} + (N/M)T_{sim}} = M \tag{4}$$

Showing that the speed-up should be linear with the number of Cloud nodes, M, for large numbers of simulations.

It should be noted that the simulations can take different times for different input combinations and the fire start location. For example, fires that burn larger areas (due to a combination of high air temperatures and wind speeds with low relative humidity) take longer when compared to those with smaller burned areas (due to low relative values of air temperature and wind speed with high relative humidity). Fires starting closer to the water bodies cease quicker even in favorable weather conditions when compared to the fire starting at a location farther away from water sources. Due to this fact, the speed up factor calculated for a real system is usually less than the theoretical values of the speed up factor and should be considered as a reference point (upper limit) to further optimize the real system.

Figure 9 shows the speed up factor and the variation in unit simulation execution time with the increase in the number of workers for a sample size argument of 1000. For the Morris method, the number of total model runs required for the analysis, N, is 4000, taking 48,475 seconds to complete in a single machine system. Assuming 70 seconds on average for the data fetch cycle and 12.10 seconds as the average unit simulation execution time, the maximum possible speed up factor with an arbitrary number of workers (at least 4000) is 590 (calculated using Equation 4). As can be seen in Figure 9a, in our framework, the speed up factor linearly increases from 1 to 33 until 50 workers after which the value increases steadily to about 128 for 320 workers

(the analysis was limited to this maximum number of workers by our quota of computing nodes on the Cloud system used). The analysis continued for worker sizes beyond 320 would produce a similar increase in the speed up factor. Similar trends are evident with the Sobol and FAST methods, where the gradient in the speed up factor decreases earlier for the FAST method. The linear increase in the speed up factor demonstrates the effectiveness of the framework within the ranges considered.

We also studied the efficiency of using multiple workers in the framework by further analyzing the unit simulation execution time for different methods with an increase in worker size as summarized in Figure 9b. The unit simulation execution time represents the time required for the computative cycle of the simulations in the subjob. The data fetch time for any worker cannot be further reduced or parallelized and hence/, is not considered as a part of the unit simulation execution time. The unit simulation execution time is the least when all the simulations are run in a single machine. Consequently, running such a high number of model runs costs the least in a single-machine system, but takes several days to complete. Such delays are not acceptable in an operational environment. With the facilitation of multiple distributed workers in the framework, there has to be a data fetch cycle in each worker, which is then followed by model runs. Adding more workers in the framework does not necessarily mean an improvement in the unit simulation execution time. Adding more workers can decrease the total time for the completion of the job but, such addition cannot always ensure maximum resource utilization. Due to this fact, the value of unit simulation execution time saturates after a particular value of worker size. For example, the average time spent

Table 4: Sensitivity Indices for wildfire simulations (Sample Size Argument $N = 1000$)

Input	**Morris Method**			**Sobol Analysis**			**FAST**		
Parameters	μ	σ	%	**FO**	**Total**	%	**FO**	**Total**	%
Temperature	0.1	0.20	16.6%	0.01	0.09	8.8%	0.01	0.09	6.4%
Rel. Humidity	**0.31**	**0.41**	**51.8%**	**0.65**	**0.91**	**69.3%**	**0.59**	**0.91**	**67.4%**
Wind	0.19	0.31	31.6%	0.07	0.29	21.8%	0.07	0.35	26.2%

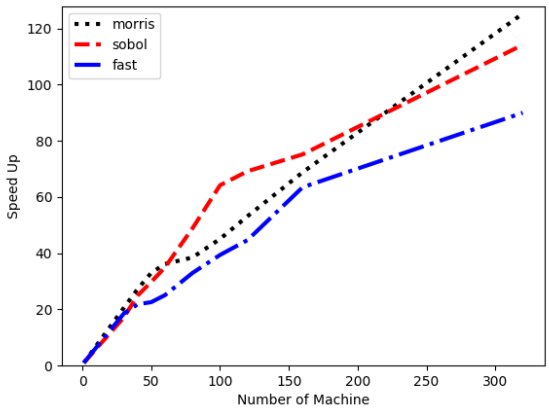
to run a simulation for Morris method with 100 workers is almost the same for a worker size of 200 for the same job, despite the entire job taking less to complete with worker size of 200. It is also clear from Figure 9b that workers can be best utilized (maximum resource utilization with a balanced trade-off between time and resources) at a size of 50, 100 and 30 for Morris, Sobol, and FAST methods respectively. Beyond these worker sizes, unit simulation execution time saturates indicating to the fact This can be further studied to define a suitable trade-off between the worker size and time for various situations ensuring better resource utilization.

5. Sensitivity Analysis Results

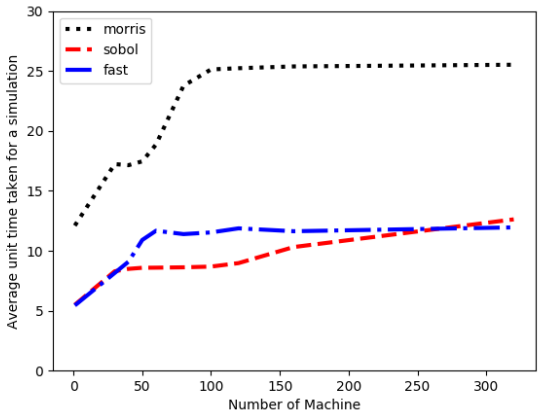
In this section, we explain in detail the results of the sensitivity analyses of wildfire models using our framework and discuss the implications of the findings.

5.1. Sensitivity Indices

The first order (FO) and the total effect of the input parameters on the area burned by the fire are summarized in Table 4. The analysis shows that relative humidity has the highest effect on the variability of fire size and the



(a) Speed Up factor vs Number of workers



(b) Unit Simulation execution time

Figure 9: Analysis of the impact of parallelization of simulations in the framework. Initially, the framework scales linearly with the addition of more workers, but the gradient flattens after a certain point. The linear scaling demonstrates the effectiveness of our framework. The framework can be best utilized at different sizes for different methods.

temperature has the least influence. The wind also has a significant effect, but the effect is less than that of relative humidity.

Similar to the first-order indices, the total sensitivity indices also confirm relative humidity as the parameter with the highest impact and temperature with the least impact on the model output variability. The interaction of wind with other parameters is shown by the Sobol analysis to have the greatest effect on the output variability when compared with other interactions. All three methods indicate the interaction of the temperature with other parameters has the least influence in the variance of the fire area. Even though the Morris and FAST methods show that interactions of relative humidity, with other parameters, have the greatest impact, the interactions of wind, with other parameters, also have a significant impact on the model output variability. Relative humidity contributes to 52-67% in the variability of fire area while temperature contributes to just 6-17% of the fire area variability.

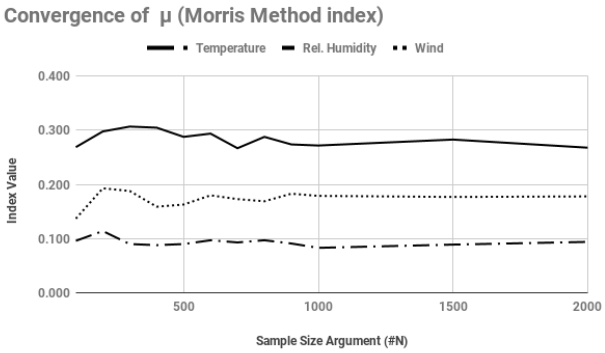
5.2. Convergence Test

For the convergence of sensitivity indices, we follow the three criteria defined by Sarazzin et al.[68] (consistent sensitivity indices values, parameter ranking, and partitioning between sensitive and least sensitive parameters). The ranks (order of the input parameters with the highest to the lowest impact) of the input parameters for the wildfire model are quite consistent for every sample size. The difference between the SA indices calculated using Sobol and FAST for the same input parameter is significant (more than 0.05) until the base sample size is 1000. Beyond the value of the base sample size (N) greater or equal to 1000, the indices converge as per the consistent value criterion. The consistent value criterion is fulfilled for Morris

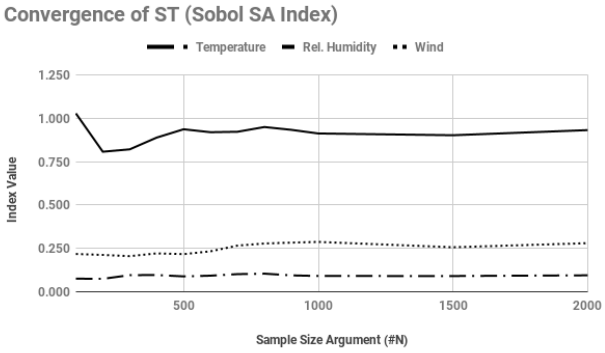
method at smaller sample size (at around 500) as Morris method is a semi-quantitative measure and can effectively be used as a proxy for variance-based SA methods with low computation cost and for ranking and screening of the input parameters [5, 69]. Similarly, the distance between the most significant and the least significant impact of the parameters is almost constant for all the methods after $N \geq 500$. Thus, for this study, the minimum base size of the sample for the convergence of SA indices is 1000 for Sobol and FAST and 500 for the Morris method, which requires 8000, 2000 and 3000 model runs respectively.

5.3. Repeatability Analysis

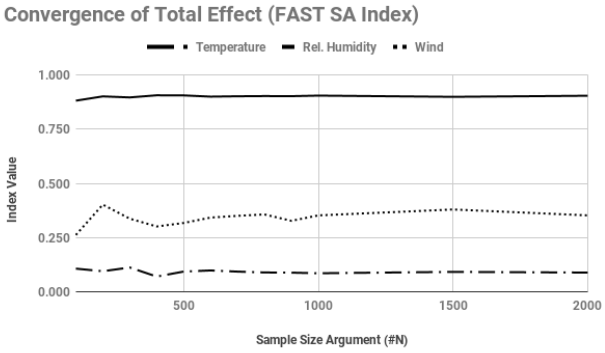
Figure 11 represents the scatter plot of the repeatability test for fire simulations where the fire area is calculated once by considering the variability of the temperature and then without considering the variability of the temperature. As represented in the figure, the values of correlation coefficients between the sets of fire areas are 0.92, 0.95, and 0.95 for the input parameter combinations obtained through the Morris, Sobol, and FAST methods respectively. These values (closer to 1) represent the degree of similarity between the two data sets, which again concludes that the temperature has the least impact on the variability of the simulated wildfire area. Such findings could, in practice, help to define a trade-off between the precision of results and the computational time for operational situations. Moreover, new operational tools could be built by cutting down the parameter space of less important input parameters.



(a) Morris Method

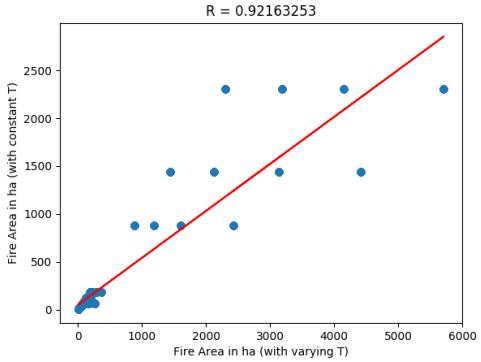


(b) Sobol Analysis

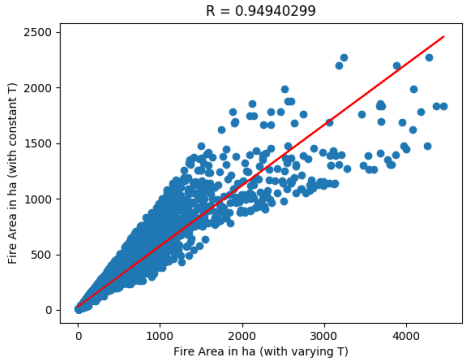


(c) FAST

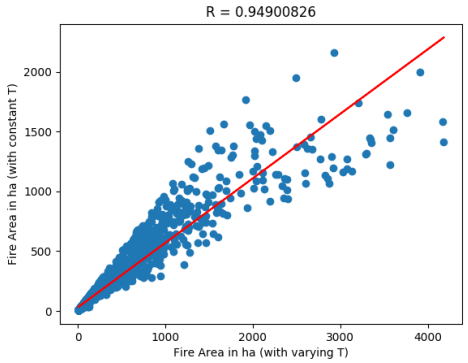
Figure 10: Convergence of SA indices for Spark input parameters. The minimum model runs required for the convergence of the indices vary according to the methods. It is fair to say the indices start converging for the value of sample argument ($N \geq 1000$) for all the methods.



(a) Morris Method



(b) Sobol Analysis



(c) FAST Method

Figure 11: Scatter Plot of Repeatability Test for Spark Simulations. The high values (closer to 1) of correlation coefficients calculated for all methods represent the similarities between two different data sets considered for repeatability analysis, thereby confirming the insignificant impact of temperature in fire area.

6. Conclusions and Future Directions

Natural hazard models are essential for modeling and mitigating risks from dangerous events. However, these models rely on a complex set of interconnected input variables. Here, we have introduced a Cloud framework for rapidly performing a large number of simulations and subsequently sensitivity analysis (SA) on such models, allowing the dominant components and degree of connection between the input parameters to be characterized. This characterization can be applied to either improve understanding of a natural hazard in progress by categorizing the current dominant factors driving the event and guide mitigation efforts, or allowing the parameter space for inessential input parameters to be reduced for risk modeling. Such practice can leverage the current state-of-the-art of natural hazard modeling systems. The data sets obtained after each analysis can be used for further analyses for better insights into the models.

We have demonstrated the efficiency of our framework with the scalability achieved while calculating sensitivity indices for simulated fires in Tasmania using the Spark wildfire modeling system. The framework was able to achieve a significant speed improvement (at least about 15 times faster) over a similar analysis on a local machine. The SA in our demonstration investigated the variation in the fire area caused by the input parameters temperature, relative humidity, and wind speed. Relative humidity was found to have the greatest impact on the area burned by the fire, while the temperature was the parameter with the least impact. Future work will involve optimization of the framework and extensions to create and assign tasks to the Cloud Workers in an optimal manner. Our framework is also model-agnostic and

is directly applicable to other environmental and disaster models.

Declaration of Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] L. Cai, H. S. He, Y. Liang, Z. Wu, C. Huang, Analysis of the uncertainty of fuel model parameters in wildland fire modelling of a boreal forest in north-east china, International Journal of Wildland Fire (2019).

[2] Y. Liu, E. Jimenez, M. Y. Hussaini, G. Ökten, S. Goodrick, Parametric uncertainty quantification in the rothermel model with randomised quasi-monte carlo methods, International Journal of Wildland Fire 24 (3) (2015) 307–316.

[3] J. Hilton, C. Miller, A. Sullivan, C. Rucinski, Effects of spatial and temporal variation in environmental conditions on simulation of wildfire spread, Environmental Modelling & Software 67 (2015) 118 – 127. doi:https://doi.org/10.1016/j.envsoft.2015.01.015.
URL `http://www.sciencedirect.com/science/article/pii/S1364815215000468`

[4] J. Nossent, P. Elsen, W. Bauwens, Sobol'sensitivity analysis of a complex environmental model, Environmental Modelling & Software 26 (12) (2011) 1515–1525.

[5] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global sensitivity analysis: the primer, John Wiley & Sons, 2008.

[6] X. Qin, H. Wang, Y. Li, Y. Li, B. Mcconkey, R. Lemke, C. Li, K. Brandt, Q. Gao, Y. Wan, et al., A long-term sensitivity analysis of the denitrification and decomposition model, Environmental modelling & software 43 (2013) 26–36.

[7] X. Li, G. Hadjisophocleous, X.-q. Sun, Sensitivity and uncertainty analysis of a fire spread model with correlated inputs, Procedia Engineering 211 (2018) 403–414.

[8] D. Hamby, A review of techniques for parameter sensitivity analysis of environmental models, Environmental monitoring and assessment 32 (2) (1994) 135–154.

[9] S. Tarantola, N. Giglioli, J. Jesinghaus, A. Saltelli, Can global sensitivity analysis steer the implementation of models for environmental assessments and decision-making?, Stochastic Environmental Research and Risk Assessment 16 (1) (2002) 63–76.

[10] E. Jacquier, N. G. Polson, P. E. Rossi, Bayesian analysis of stochastic volatility models, Journal of Business & Economic Statistics 20 (1) (2002) 69–87.

[11] S. An, F. Schorfheide, Bayesian analysis of dsge models, Econometric reviews 26 (2-4) (2007) 113–172.

[12] J. E. Oakley, A. O'Hagan, Probabilistic sensitivity analysis of complex models: a bayesian approach, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 66 (3) (2004) 751–769.

[13] A. Yegnan, D. Williamson, A. Graettinger, Uncertainty analysis in air dispersion modeling, Environmental Modelling & Software 17 (7) (2002) 639–649.

[14] G. Kuczera, E. Parent, Monte carlo assessment of parameter uncertainty in conceptual catchment models: the metropolis algorithm, Journal of Hydrology 211 (1-4) (1998) 69–85.

[15] C. Freissinet, M. Vauclin, M. Erlich, Comparison of first-order analysis and fuzzy set approach for the evaluation of imprecision in a pesticide groundwater pollution screening model, Journal of Contaminant Hydrology 37 (1-2) (1999) 21–43.

[16] A. D. Richardson, D. Y. Hollinger, Statistical modeling of ecosystem respiration using eddy covariance data: maximum likelihood parameter estimation, and monte carlo simulation of model and parameter uncertainty, applied to three simple models, Agricultural and Forest Meteorology 131 (3-4) (2005) 191–208.

[17] A. Bachmann, B. Allgöwer, Uncertainty propagation in wildland fire behaviour modelling, International Journal of Geographical Information Science 16 (2) (2002) 115–127.

[18] R. Cibin, K. Sudheer, I. Chaubey, Sensitivity and identifiability of

stream flow generation parameters of the swat model, Hydrological Processes: An International Journal 24 (9) (2010) 1133–1148.

[19] K. Holvoet, A. van Griensven, P. Seuntjens, P. Vanrolleghem, Sensitivity analysis for hydrology and pesticide supply towards the river in swat, Physics and Chemistry of the Earth, Parts A/B/C 30 (8-10) (2005) 518–526.

[20] A. v. van Griensven, T. Meixner, S. Grunwald, T. Bishop, M. Diluzio, R. Srinivasan, A global sensitivity analysis tool for the parameters of multi-variable catchment models, Journal of hydrology 324 (1-4) (2006) 10–23.

[21] K. L. White, I. Chaubey, Sensitivity analysis, calibration, and validations for a multisite and multivariable swat model 1, JAWRA Journal of the American Water Resources Association 41 (5) (2005) 1077–1089.

[22] C. Yang, Y. Xu, D. Nebert, Redefining the possibility of digital earth and geosciences with spatial cloud computing, International Journal of Digital Earth 6 (4) (2013) 297–312.

[23] H. Brohus, P. V. Nielsen, A. J. Petersen, K. Sommerlund-Larsen, Sensitivity analysis of fire dynamics simulation, in: Proceedings of Roomvent, Citeseer, 2007.

[24] J. E. Hilton, A. G. Stephenson, C. Huston, W. Swedosh, Polynomial chaos for sensitivity analysis in wildfire modelling.

[25] R. Salvador, J. Pinol, S. Tarantola, E. Pla, Global sensitivity analysis

and scale effects of a fire propagation model used over mediterranean shrublands, Ecological Modelling 136 (2-3) (2001) 175–189.

[26] P.-A. Ekstrom, Eikos: a simulation toolbox for sensitivity analysis in matlab, FACILIA AB (2005).

[27] A. F. D'Augustine, Matlode: A matlab ode solver and sensitivity analysis toolbox, Ph.D. thesis, Virginia Tech (2018).

[28] F. Pianosi, F. Sarrazin, T. Wagener, A matlab toolbox for global sensitivity analysis, Environmental Modelling & Software 70 (2015) 80–85.

[29] J. Herman, W. Usher, Salib: An open-source python library for sensitivity analysis, Journal of Open Source Software 2 (9) (2017) 97.

[30] T. Wagener, J. Kollat, Numerical and visual evaluation of hydrological and environmental models using the monte carlo analysis toolbox, Environmental Modelling & Software 22 (7) (2007) 1021–1033.

[31] P. T. Roy, S. Ricci, R. Dupuis, R. Campet, J.-C. Jouhaud, C. Fournier, Batman: Statistical analysis for expensive computer codes made easy., J. Open Source Software 3 (21) (2018) 493.

[32] A. Dutfoy, I. Dutka-Malen, A. Pasanisi, R. Lebrun, F. Mangeant, J. S. Gupta, M. Pendola, T. Yalamas, Openturns, an open source initiative to treat uncertainties, risks' n statistics in a structured industrial approach, 2009.

[33] S. Tarantola, W. Becker, Simlab software for uncertainty and sensitivity

analysis, Handbook of Uncertainty Quantification; Ghanem, R., Higdon, D., Owhadi, H., Eds (2016) 1–21.

[34] I. Bertrand, J. Alexandre, P. Gilles, Senstivity Analysis, r package version 1.18.0, `https://CRAN.R-project.org/package=sensitivity` (2020).

[35] B. Stanfill, H. Mielenz, D. Clifford, P. Thorburn, Simple approach to emulating complex computer models for global sensitivity analysis, Environmental Modelling & Software 74 (2015) 140–155.

[36] B. A. Keating, P. S. Carberry, G. L. Hammer, M. E. Probert, M. J. Robertson, D. Holzworth, N. I. Huth, J. N. Hargreaves, H. Meinke, Z. Hochman, et al., An overview of apsim, a model designed for farming systems simulation, European journal of agronomy 18 (3-4) (2003) 267–288.

[37] R. Sheikholeslami, S. Razavi, H. V. Gupta, W. Becker, A. Haghnegahdar, Global sensitivity analysis for high-dimensional problems: How to objectively group factors and measure robustness and convergence while reducing computational cost, Environmental Modelling & Software 111 (2019) 282–299.

[38] F. Pianosi, T. Wagener, Distribution-based sensitivity analysis from a generic input-output sample, Environmental Modelling & Software 108 (2018) 197–207.

[39] F. Pianosi, T. Wagener, A simple and efficient method for global sensi-

tivity analysis based on cumulative distribution functions, Environmental Modelling & Software 67 (2015) 1–11.

[40] E. Borgonovo, X. Lu, E. Plischke, O. Rakovec, M. C. Hill, Making the most out of a hydrological model data set: Sensitivity analyses to open the model black-box, Water Resources Research 53 (9) (2017) 7933–7950.

[41] O. Rakovec, M. C. Hill, M. Clark, A. Weerts, A. Teuling, R. Uijlenhoet, Distributed evaluation of local sensitivity analysis (delsa), with application to hydrologic models, Water Resources Research 50 (1) (2014) 409–426.

[42] M. S. Eldred, W. J. Bohnhoff, W. E. Hart, Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, sensitivity analysis, and uncertainty quantification, Sandia National Labs Report, No. SAND99-0000 (1999).

[43] K. Ujjwal, S. Garg, J. Hilton, J. Aryal, N. Forbes-Smith, Cloud computing in natural hazard modeling systems: Current research trends and future directions, International Journal of Disaster Risk Reduction (2019) 101188.

[44] K. Ujjwal, S. Garg, J. Hilton, An efficient framework for ensemble of natural disaster simulations as a service, Geoscience Frontiers (2020).

[45] F. Campolongo, A. Saltelli, Sensitivity analysis of an environmental model: an application of different analysis methods, Reliability Engineering & System Safety 57 (1) (1997) 49–69.

[46] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, T. Wagener, Sensitivity analysis of environmental models: A systematic review with practical workflow, Environmental Modelling & Software 79 (2016) 214–232.

[47] M. D. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (2) (1991) 161–174.

[48] T. Homma, A. Saltelli, Importance measures in global sensitivity analysis of nonlinear models, Reliability Engineering & System Safety 52 (1) (1996) 1–17.

[49] A. Saltelli, R. Bolado, An alternative way to compute fourier amplitude sensitivity test (fast), Computational Statistics & Data Analysis 26 (4) (1998) 445–460.

[50] G. Qian, A. Mahdi, Sensitivity analysis methods in the biomedical sciences, Mathematical Biosciences (2020) 108306.

[51] B. Iooss, P. Lemaître, A review on global sensitivity analysis methods, in: Uncertainty management in simulation-optimization of complex systems, Springer, 2015, pp. 101–122.

[52] I. M. Sobol', On sensitivity estimation for nonlinear mathematical models, Matematicheskoe modelirovanie 2 (1) (1990) 112–118.

[53] A. Saltelli, Making best use of model evaluations to compute sensitivity indices, Computer physics communications 145 (2) (2002) 280–297.

[54] H. Landau, Sampling, data transmission, and the nyquist rate, Proceedings of the IEEE 55 (10) (1967) 1701–1706.

[55] A. Saltelli, S. Tarantola, K.-S. Chan, A quantitative model-independent method for global sensitivity analysis of model output, Technometrics 41 (1) (1999) 39–56.

[56] Nectar cloud, `https://nectar.org.au/research-cloud/`, accessed: 2018-05-12.

[57] C. Miller, J. Hilton, A. Sullivan, M. Prakash, Spark–a bushfire spread prediction tool, in: International Symposium on Environmental Software Systems, Springer, 2015, pp. 262–271.

[58] T. F. Service, STATE FIRE COMMISSION ANNUAL REPORT, 2019.

[59] W. Tasmanian Department of Primary Industries, Parks, E. T. V. Monitoring, M. Program, Tasveg 3.0 (2013).

[60] A. McARTHUR, Fire behavior in eucalypt forest. canberra, department of development, Forestry and Timber Bureau (1967).

[61] N. P. Cheney, J. S. Gould, W. L. McCaw, W. R. Anderson, Predicting fire behaviour in dry eucalypt forest in southern australia, Forest Ecology and Management 280 (2012) 120–131.

[62] J. Marsden-Smedley, W. R. Catchpole, Fire behaviour modelling in tasmanian buttongrass moorlands. ii. fire behaviour, International Journal of Wildland Fire 5 (4) (1995) 215–228.

[63] W. R. Anderson, M. G. Cruz, P. M. Fernandes, L. McCaw, J. A. Vega, R. A. Bradstock, L. Fogarty, J. Gould, G. McCarthy, J. B. Marsden-Smedley, et al., A generic, empirical-based model for predicting rate of fire spread in shrublands, International Journal of Wildland Fire 24 (4) (2015) 443–460.

[64] N. Cheney, J. Gould, W. R. Catchpole, Prediction of fire spread in grasslands, International Journal of Wildland Fire 8 (1) (1998) 1–13.

[65] A. McArthur, Weather and grassland fire behaviour. commonwealth department of national development, Forestry and Timber Bureau Leaflet 100 (1966).

[66] M. G. Cruz, J. S. Gould, M. E. Alexander, A. L. Sullivan, W. L. McCaw, S. Matthews, A guide to rate of fire spread models for Australian vegetation, Australasian Fire and Emergency Service Authorities Council Limited and CSIRO, 2015.

[67] M. D. Hill, M. R. Marty, Amdahl's law in the multicore era, Computer 41 (7) (2008) 33–38.

[68] F. Sarrazin, F. Pianosi, T. Wagener, Global sensitivity analysis of environmental models: Convergence and validation, Environmental Modelling & Software 79 (2016) 135–152.

[69] F. Campolongo, J. Cariboni, A. Saltelli, An effective screening design for sensitivity analysis of large models, Environmental modelling & software 22 (10) (2007) 1509–1518.