Modelling user satisfaction for power-usage optimisation of computer fleets

Ramón Medrano Llamas[a,*], Joaquín Entrialgo[b], Daniel F. García[b]

[a] *Google, Zürich, Switzerland*
[b] *University of Oviedo, Gijón, Spain*

Abstract

Power consumption costs of computer fleets can be one of the main operational costs of medium or large-sized office sites. In order to optimise the power consumption of the fleet, a set of optimal power management policies must be generated and enforced.

Generating these policies is an optimisation problem of finding the power off timeout value for each computer that maximises energy savings while guaranteeing user satisfaction. To solve this problem, understanding the computer utilisation patterns of the users and defining a metric of user satisfaction is fundamental.

This paper presents a method to analyse user activity and inactivity, extract models from previously recorded utilisation logs and use them to manage a whole computer fleet. A tool that implements this method is also introduced. This tool generates power management policies from utilisation logs. It analyses the effects of variations in fleet characteristics on policies by means of discrete event simulation. It also seeks to understand the behavioural patterns of users over weekly periods. Finally, it generates utilisation logs from high level descriptions of fleets. This tool offers a simulation to study diverse fleet configurations and generation of synthetic fleets.

Keywords: power management, PC fleet, simulation, user activity, user satisfaction

1. Introduction

By 2050, governments will need to ensure a reduction of up to 90% of the CO_2 emissions in order to prevent a raise of 2 °C in global temperature [1]. Power usage and size of the computing infrastructures needed by companies is increasing; not only the fleets of large data centres, but also the computing equipment used daily by employees: laptops, workstations, tablets and smartphones. The fraction of the total computing cost of these types of devices is significant [2]. There are more and more devices with ever-increasing power

*Corresponding author
Email addresses: `niobium@google.com` (Ramón Medrano Llamas), `joaquin@uniovi.es` (Joaquín Entrialgo), `dfgarcia@uniovi.es` (Daniel F. García)

needs. Korn et al. [3] predicted that \$1.5B could be saved on energy costs, while Belkhir et al. [4] indicated that the footprint of ICT systems will grow up to 6.9% year-over-year until 2040, including manufacturing.

A particular case of devices used by employees are workstations or desktop computers. They are typically powered on during the whole day and they use less than half of their computing capabilities, idling most of the time [5]. Furthermore, though they comply with the latest energy regulations [6], the power consumption of desktop computers is typically one or two orders of magnitude greater than that of new mobile devices such as laptops or tablets [7].Therefore, many organisations that run large desktop computer fleets can benefit from the energy savings derived from implementing dynamic power management [8], from both power savings and discounts [9].

There are many widely-used tools for managing software updates, configurations or security, such as Puppet [10]. However this type of configuration management tool does not support dynamic power management of the fleet. Other tools, specifically designed for power management of computer fleets, such as NightWatchman [11], monitor and collect information about fleet behaviour, but they do not consider user satisfaction when optimising power management policies.

This work presents a method for modelling user activity and inactivity, and a tool that generates power management policies following this method. In addition, it allows the fleet administrator to experiment with variations in the fleet and different merging mechanisms to generate customised policies for each user.

The activity analysis method is based on the analysis of utilisation data from the users of the fleet, collected by monitoring agents installed in every computer of the fleet. The method generates models that determine the optimal policy to ensure the satisfaction level of users of the fleet. The models are then used by the tool for four functions: firstly, to validate the policy determined analytically by using discrete event simulation; secondly, to compare different merging mechanisms and their impact on the efficiency achieved by the policy; thirdly, to simulate variations of existing fleets (for example, with different numbers of users); and finally, to create logs from which models can be generated from scratch by providing high level parameters that describe the fleet. This tool allows rapid prototyping of power management policies for the continuous improvement of fleets through analysis and planning.

We define here some common terms used across the rest of the paper:

User / Computer Through the paper we assume that a computer is used by a single user, i.e. we do not try to infer if the behaviour recorded from a computer is from one or more users.

Power manager The power manager is the component that enforces the power policies generated. Generally, the operating system will be enforcing the policy by setting an idle timer to set the computer on a lower power state.

Activity Period of time in which the user has been interacting with the computer.

Inactivity Period of time when the computer is idle: it is turned on, but the user is not present or interacting with it.

The rest of the paper is organised as follows: the next section analyses in depth the current and previous work on the area. This is followed by a section describing how user satisfaction is measured and modelled. Next, the tool design and implementation is described. The paper ends with a detailed description of the validation performed and conclusions drawn.

2. Survey of previous work

In this section, we analyse previous work, grouped in three sections: techniques based in stochastic methods, techniques based in machine learning and commercial solutions. Finally we review in detail those techniques that consider the user satisfaction for power optimisation.

2.1. Stochastic and control based techniques

The work of Srivastava et al. [12] determines the length of future inactivity intervals by using a regression method. It operates on a system modelled as power state automata, which determines the power usage of the system and its possible transitions.

Lu et al. [13] define a basic power manager by tracking the resource usage of each of the tasks being run by a computer. When the task in execution was not using a device, and the break-even point to amortise the state change was passed, the controller would immediately turn it off with no delay. Ramanathan et al. [14] define a basic adaptive algorithm and do a competitive analysis with a non-adaptive method.

The work of Hwang et al. [15] uses an estimation method based on moving averages of the inactivity of the user, and then a very simple-state automata to model the system. They predict the duration of inactivity and decide to turn the computer off and schedule a turn on for the estimated duration of the inactivity period. They establish that short inactivity periods are generally worth ignoring due to the noise that these periods introduce for prediction.

Benini et al. [16, 17] establish stochastic methods. In this seminal work, the system and users are modelled as a queue system: the system has different service rates depending on the power state it is set to. They propose solving a Markov model in order to find the best policy.

The work of [18] follows the same approach, but adds a comparison of the power usage to the inter-arrival time and queue length. Later, this work was extended to incorporate a partially observable Markov decision process (POMDP) [19]. With this, the system can be controlled optimally without the need to estimate and discover all of its parameters. Tan et al. [20] use a similar approach, but also measure the probability of finding a fit with the number of states on the POMDP. The work of Jung et al. [21] incorporates a similar method, and shows the usage logs in a 3D representation.

Simunic et al. [22] develop a method implemented both by renewal theory and a time indexed Markov model. Both implementations converge to global optimal solutions, but the latter is more complex. Another novel analysis was the comparison of how well the exponential and Pareto distributions match the user inactivity, determining that the exponential is not a suitable fit. They use usage logs and also consider the time to change power state on the optimisation. Later, Phillips et al. [23, 24] propose a model based on Markov systems and validate it.

The work of [25] introduces two novel concepts: first, it considers a sliding window of user actions over time, which allows to calculate richer probabilities of transfer between computer states; second, it allows to dynamically adapt the system and the policies dynamically. A policy is defined as a set of actions for each of the computer states. Luiz et al. [26] extended this work by incorporating multisize sliding window for the load estimation, which allows to balancing the resolution of the system in contrast to the speed of adaptation to changes in the load profile. They keep several estimators running in parallel, with different window sizes, and choose the best one by means of a maximum likelihood estimation.

Control theory is another approach used by some methods for power management. Jung et al. [27] developed one of the first works on controls for power management, it was applied on to wireless network optimisation. Juang et al. [28] introduced a PID controller to govern the frequency of the processor cluster on multi-core machines, making the scheduler aware of the demand and the state of the rest of the processor nodes. The work of Minerick et al. [29] incorporates d a feedback loop for the control to be refined over time. Bulay et al. [30] used a control to manage the A/C systems of a computer laboratory, providing an interesting insight of into state management, usable for general power control.

Lin et al. [31] developed a method to model the power usage of computers that incorporates the proportionality of power usage with load.

Table 1 summarises the research work on stochastic methods as well as the control based-methods.

Work	Year	Scope	Method	User satisfaction	Preemptive Turn on	Performance metric	System modelling	User modelling
[12]	1996	computer	regression	○	○	power	state automata	inactivity regression
[32]	1999	computer	n/a	●	○	battery lifetime	n/a	n/a
[16–18]	1999/00	computer	stochastic	○	○	service queue size	state automata	Markov system
[13]	2000	computer	tracking	○	○	power	lists of tasks	n/a
[14]	2000	computer		○	○	power, latency	n/a	interval count
[15]	2000	computer	stochastic	○	●	energy gain	simple state automata	moving average
[22–24]	2001, 2008	computer	stochastic	○	○	service queue size	state automata	Markov system
[27]	2002	Wi-Fi	control	○	○	throughput	transfer window	n/a
[29]	2002	processor	control	○	○	power	energy conservation	n/a
[25]	2002	computer	stochastic	○	○	service queue size	state automata	Markov system
[28]	2005	computer	control	○	○	energy-delay product	PID controller	n/a
[19–21]	2007/08	computer	stochastic	○	○	latency, energy	state automata	POMDP
[33]	2009	processor	control	◐	○	power, temperature	n/a	user events
[26]	2010	computer	stochastic	○	○	energy	load sliding windows	Markov system
[34]	2016	computer	stochastic	●	○	power	n/a	activity probability
[30]	2018	laboratory	control	○	○	power	state automata	static
[35]	2019	computer	stochastic	◐	○	idle time	Hidden Markov system	Hidden Markov system
This work	2020	fleet	analytical	●	○	removed inactivity	queue system	empirical distribution

Table 1: Comparison of related research work based stochastic and control methods.

Only two of these methods fully support user satisfaction. Generally, the implementation of stochastic methods is based on modelling computers as a state automaton and then using a Markov system to model user behaviour. Some of the problems of using these kinds of models are the complexity of the Markov model when many states are considered (for example, on fleets of many computers), and the fact that these are often partially observable models, which are expensive to solve.

2.2. Techniques based on machine learning

One of the first machine learning techniques applied to computer power management was decision tree based prediction [36]; these trees are updated during periods of inactivity. The new model predicts inactivity times and schedules the computer to turn on automatically so it can be ready when the user comes back. The authors also introduce the filtering of short inactivity periods so as not to affect user satisfaction.

The work of [37] introduces a system to choose between different experts (policies) based on machine learning. In particular, online learning [38].

Kong et al. [39] use a genetic algorithm to predict the future inactivity intervals based on past inactivity. They set up genes as sequences of past activity and inactivity intervals and then use a fitness function to take into account the break-even time of state change cost while performing well with bursting workloads.

The work of Prabha et al. [40] incorporates some basic aspects of user satisfaction to a reinforcement learning based method by adding an agent to determine rewards and penalties to the learning engine. This engine chooses among existing Dynamic Power Management (DPM) policies and is implemented in hardware.

The work of Jung et al. [41] applies supervised learning to the frequency scaling of a multi-core processor. The work defines several features related to the occupancy and arrival of requests to use as input for the method, which is based on a Bayesian classifier. With this, a power policy is generated to govern the processor frequency on run time.

Liu et al. [42] use reinforcement learning with the Q-learning method to select the best of several power management policies. They measure the performance of the system with synthetic load and compare the pairs of power usage and latency observed for each load level and policy. Candrawati et al. [43–45] also use Q-Learning to manage the power of a computer. They model the computer as a set of time series that provide knowledge for a controller to actuate over the different components.

Skakun et al. [46] and later Csurgai et al. [47] use a method based on a neural network to predict user activity.

Table 2 summarises the research work on machine learning methods.

Work	Year	Scope	Method	User satisfaction	Preemptive Turn on	Performance metric	System modelling	User modelling
[36]	1999	computer	learning tree	○	●	prediction hit rate	state automata	n/a
[46]	2005	computer	neural network	○	○	power	n/a	n/a
[39]	2006	computer	genetic algorithm	○	○	custom	inactivity genes	n/a
[37]	2006	computer	expert system	○	○	custom	state automata	Markov system
[48]	2006	computer	C4.5, k-nearest	●	○	power	state automata	user annoyance
[40]	2007	computer	reinforcement learning	◐	○	energy savings	state automata	agent issuing rewards
[49]	2008	processor	neural network	◐	○	energy savings	performance counters	n/a
[41]	2010	processor	supervised learning	○	○	energy	ocupancy features	n/a
[42]	2010	HDD	Q-Learning	○	○	energy, latency	queue system	synthetic load
[47]	2011	computer	neural network	○	○	power	n/a	n/a
[43–45]	2016	computer	Q-Learning	○	○	power	components	time series
This work	2020	fleet	analytical	●	○	removed inactivity	queue system	empirical distribution

Table 2: Comparison of related research work based on machine learning methods.

One of the strongest points of using systems based on reinforcement learning is that the system can dynamically adapt the models with the system evolution, instead of having to generate a new policy from time to time. On the other hand, these models tend to need much more data to produce precise models and are more difficult to explain and debug than the stochastic methods. Finally, user satisfaction is only present in one of these works.

2.3. Comparison of commercial solutions

Commercially available solutions typically implement basic methods, in general, static: the fleet administrator selects the timeout value as factoring in some information provided by the tool, such as time series and recommendations. Some other tools directly indicate which timeout to use, but this does not depend on metrics such as user satisfaction or detailed activity models. There are other tools that are closed sourced, for which the publishers do not provide details of their implemented method.

Few tools [11, 50–53] support user clustering or merging to generate models and policies for entire groups of the fleet. Only one [54] is able to extrapolate the models in situations when the fleet changes substantially (such as with the opening of new buildings), or when new users are added to it. Also, just one tool [11] supports preemptive turn ons of computers to have them ready when the users need them.

These tools are very good at delivering the policies to the computers and enforcing them. They offer good reporting graphs and integrate with other management systems within the company, such compliance or software management systems.

Tool	Scope	User Satisfaction	Preemptive Turn on	Policy	User Clustering	Dynamic Models	Model Extrapolation	Reporting
1E NightWatchman [11]	fleet	○	●	n/a	●	◐	○	●
Autonomic Software ANSA [55]	fleet	○	○	n/a	○	○	○	○
Avob Energy Saver [50]	fleet, DC	○	○	n/a	●	○	○	●
Certero PowerStudio [56]	fleet	○	○	static	○	○	○	●
Data Synergy PowerMAN [51]	fleet	○	○	static	●	○	○	●
Dell KACE [57]	fleet	○	○	n/a	○	○	○	●
EventZero Greentrac [58]	fleet	○	○	n/a	○	○	○	●
Faronics Power Save [59]	fleet	○	○	static	○	○	○	●
IBM Tivoli [54]	fleet	○	○	static	○	○	◐	●
Ivanti Power Manager [60]	fleet	○	○	static	○	○	○	○
New Boundary PwrSmart [52]	fleet	○	○	static	●	◐	○	●
Aptean Verdiem Surveyor [53]	fleet, DC	○	○	static	●	◐	○	●
Verismic Power Manager [61]	fleet	○	○	static	○	○	○	●
This work	fleet	●	○	dynamic	●	●	●	●

Table 3: Comparison of commercial power management systems

Table 3 contains a summary of the most prominent commercial tools. All of the tools reviewed here produce static policies (i.e. fleet administrator has to set a timeout). The work presented in this paper introduces dynamic policies per user, allows for merging by user and gives time to generate policies.

2.4. Techniques that consider user satisfaction

Finally, and considering the importance of the user satisfaction in power optimisation, we compare the research work that mentions user satisfaction more throughoutly.

One of the first works to incorporate the concept of user satisfaction was [32], in which power management is directed by the applications. In this way, each of the applications can determine what they understand by user satisfaction and adapt their behaviour to achieve the goals. In the research, the authors focus on battery time as the user satisfaction metric. They manage to reduce the power usage by up to 30%, however they need to instrument each of the four classes of application (web browser, speech recognition, video player and map viewer) independently and specifically for each of it.

There have been additional attempts at measuring user satisfaction; for example, a commonly used metric to determine if a service is providing satisfaction to its customers is the Application Performance Index(*Apdex*) [62]. This index defines novel concepts, such as the possibility of defining partial satisfaction of customers. However, this not suitable for power management at it focuses satisfaction solely on the latency of the service time of transactional systems.

Theocharous et al. [48] define user annoyance and use machine learning methods (logistic regression, C4.5, k-nearest) to learn the policies that find a better balance between user annoyance and power saved. The definition of user annoyance in this work has some similarity with the satisfaction definition of our work; in addition Theocharous et al. found that the relation of user annoyance and power savings follow a law similar to a logarithmic rule, which we have also confirmed in our work. They found that with annoyance levels of 5%, power savings are between 20% and 50%. Their work applies to a single computer and some major problems they present are very high variability in power savings and Markov decision models that are not easy to explain.

Lin et al. [33] implemente a governor for both the voltage and frequency directed by the process and the user, respectively. They estimate the dissatisfaction of the users of the system at different frequency points to tune the parameters of their algorithm. While they save up to 49.9% of power usage, their work is effective on very specific tasks (like word processing) and requires direct instrumentation of voltage of the processor.

Setz et al. [34] use a metric of user satisfaction to incorporate negative feedback into the power optimisation process. The users need to manually report events that are unsatisfactory for them. An increase of this negative feedback in turn produces higher timeout values. This work presents a representation of the user activity that is similar to the one we use in this work, defining the probability of user activity over time in slotted intervals (in this case of one minute).

Shye et al. [49] studied the relation of the user satisfaction with several performance indicators of the system (using the Windows Performance API). They demonstrated that the satisfaction and performance indicators follow a non-linear rule and introduce a neural network to map the performance metrics to a power management policy. While saving 25% of power, the satisfaction is generally affected and power savings are not consistent.

The last proposal that incorporates user satisfaction is the work by Turkin et al. [35], which uses a subjective measurement to understand the impact of power management. This measurement is performed after the fact on specific studies.

In comparison with all the works reviewed, the proposal presented in this article incorporates user satisfaction as the central metric to optimise power management, via timeout policies delivered to the computer fleets by a central agent. All the policies are personalised for each computer. In general, we find that our method is simpler and more adaptive, providing a single parameter that can be used to balance the user satisfaction and the power savings.

3. Modelling user behaviour and satisfaction for power management

This section introduces in detail the method we propose for managing the power usage of computer fleets.

3.1. User behaviour modelling

Users of the computers of the fleet can behave in different ways. In order to understand and model their behaviour, we consider a series of different actions they can do at a given moment in time. The users can be actively using their computer; they might have it turned on, but not developing any activity on their computers (idling); or they might just decide to turn off their computer explicitly. Computers can also be turned off by the power manager of the operating system running on them, after they idle for more time than the configured idle timer. In practical terms, these are the four states a computer can be at any point in time:

- Turned on, with the user actively utilising the computer (*activity*).
- Turned on, without the user utilising the computer (*inactivity*). Computers on this state consume typically an amount of energy comparable to that spent while being actively used.
- Turned off, triggered directly by the user (*user turn off*). Computers that are turned off are assumed to consume a negligible amount of energy.
- Turned off, triggered by the power manager (*auto turn off*). This only happens when the power management is enabled.

Figure 1 depicts a small sample of a fleet that has no control enabled (i.e. there is no power management applied to it and the computer events are only directed by the user) and the state of each computer over one day. Each of the computers cycle through the states above, independently from the other computers of the fleet.

Note that the following examples are illustrative examples to explain the cycle of states and intervals that the computers follow through the day. The intervals on a real fleet, particularly the activity intervals, will be much smaller and numerous, depending on the collection frequency. Longer inactivity intervals, like those on breaks or during the night, are commonly seen on the collected usage logs from the fleet studied, where each user shows their own behaviour.

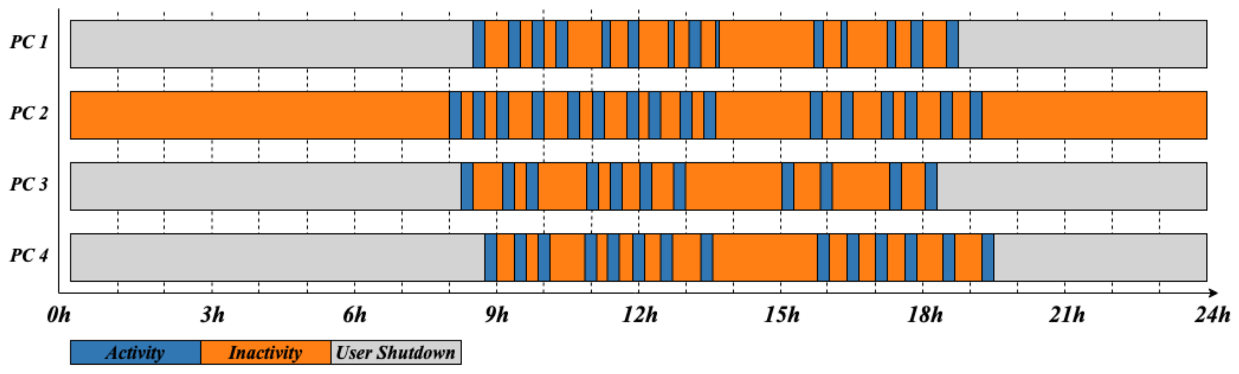


Figure 1: Example of computer states on a PC fleet.

The typical cycle for a workstation aligns with the working hours. In this example the workday starts between 8:00 and 9:00. There is a lunch break and the working day finishes at approximately 18:00. Activity intervals happen throughout the day, interleaved with inactivity intervals of relatively short duration. During the breaks, a longer inactivity interval occurs, typically aligned for several users, for example those of similar departments or units.

Most of the time between the evenings and morning hours in this example the computers are off. There may be users that leave their computer on during the nights (such as the *PC 2*). Users usually start using their computers at similar times of the days when arriving at their work spaces, and finish at similar times.

The activity time is interleaved with inactivity intervals: people are typing, moving their mouse or performing other activities. They also stop for other activities such as meetings, reading and resting.

Establishing dynamic power management policies reduces the inactivity time, by preemptively turning off computers. This reduces inactivity time and conversely increases the time computers are off, saving energy.

Figure 2 shows the same example fleet, but with the dynamic power management enabled. The power manager turns off computers automatically by defining a timeout period for the operating system power manager. In this way the majority of the time that computers are inactive can be removed by turning them off. This is especially impactful for users that leave their computers turned on all night.

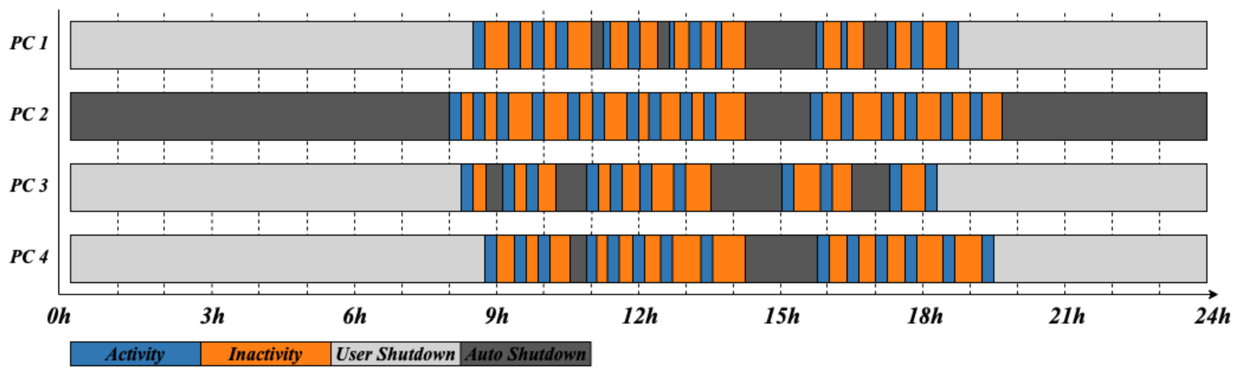


Figure 2: Example of computer states on a PC fleet, with power management.

Recording all events that happen in fleet will take storage that scales with the size of the fleet and the frequency of collection. A fleet of 10,000 computers with a collection interval of 60 s will generate 10,000 queries per minute, or about 167 QPS (queries per second). This accumulates up to 5.25×10^9 events/year (438×10^6 events/month). For these reasons, the events are summarised by creating a model using a distribution fitted on the raw data.

3.1.1. Modelling of activity and inactivity times

In order to know how real users behave, the user behaviour of a company was studied by collecting utilisation logs from their fleet of computers with no power control.

To increase the precision of the models, one distribution per hour is fitted. This represents the different user behaviour during work hours or over weekends more accurately.

The data shows that the behaviour of users repeats over a weekly cycle. For this reason, our analysis uses cycles of one week,with each day divided into 24 periods (slots) of one hour, reducing the modelling of each user to 168 time slots.

Within each of these slots, each user has fixed models for their activity and inactivity and a determined user satisfaction target. Therefore, during the whole slot, the idle timeout is set to a constant value depending on the model that the user has for each time slot [63].

The probability distribution function (PDF) is the function that gives the probability of outcomes of an experiment [64]. In the case of the inactivity intervals, it represents the probability of each interval duration to occur. In [63], we have concluded that the distribution function of activity and inactivity intervals follows an exponential law, where the shorter intervals have much more probability than the longer ones.

Figure 3 shows the PDF of the inactivity intervals of several periods of one hour on a Tuesday. Since the interval duration (horizontal axis) has a logarithmic scale, if the data is a good fit for a power law, the PDF will be a straight line. A log-normal distribution fitted from the data with the methods from the `powerlaw` package [65] is also shown for comparison.

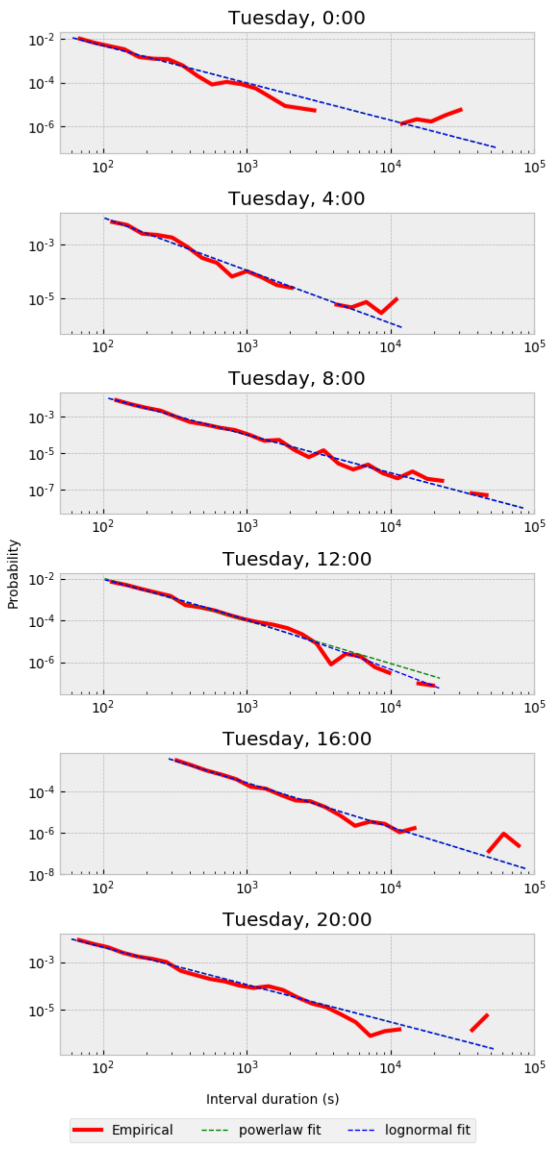


Figure 3: PDF of the inactivity intervals on logarithmic scales.

Several probability distributions were tested to fit each of the variables. Initially, it seemed that an exponential law was underlying the data. A pure exponential distribution, a Pareto distribution and a log-normal distribution were tested with a variety of data sets in [63], but although the fit was clearly good on the mass of the distribution, the results were underwhelming due to the peculiarity of the tail of the observations and the minimal amount of data that some of the models have. This exacerbated the generation of longer inactivity intervals, which were not matching at all the reality observed, some times even generating weeks long inactivity intervals.

For some users, there may be density concentrated around some inactivity duration that corresponds to users that leave the computer permanently on. There are intervals of inactivity for 12 to 14 hours, corresponding to the inactivity of the whole night. Other users have this period of inactivity over the weekend: a very long inactivity interval starting on Friday evening, and lasting approximately two days.

This is why there are hours where the actual data separates from the expected line, indicating that an exponential law is not a good fit for the data. This pattern is frequently observed in the activity logs collected from users: the inactivity is very right skewed, indicating that the majority of the inactivity intervals are very short, but it also shows a long tail. In this tail, there is a notable probability mass at its end (like on the PDFs shown in Figure 3 –0:00, 16:00 and 20:00). This is especially intense in the evening hours, indicating that some very long inactivity intervals are starting at that time. This mass corresponds to inactivity periods of users that, for example, leave their computer on during the weekend [63].

Due to the particularity of the data, especially on the tail, instead of an exponential law, an empirical distribution fitting is used. The empirical distribution provides much more precision for fitting the specific probability mass on the tail of the data (which represents very long inactivity intervals, like those of the weekends).

We have tested the goodness of fit of several implementations of the empirical distribution by means of the Kolmogorov-Smirnov test [66] over different data sets (available on the source code [67]). Other exponential laws have been tested and compared in [63], in particular the Pareto and Log-normal distributions.

The activity time is modelled with the same method.

3.2. Power management based on user satisfaction

Establishing a timeout value based only on the inactivity as modelled above requires determining how much inactivity time should be removed. If, for example, the fleet administrators want to remove 25% of the inactivity, it will suffice to establish the timeout value to 75% of the cumulative inactivity. Conversely, if the administrator is more aggressive and aims to remove 90% of the inactivity time, the timeout value will be set to 10% of the cumulative inactivity.

However, this way of calculating the idle timer does not take into account the impact on the usability of the computer. Depending on the inactivity profile and the objectives of the administrator, it might yield a completely unusable system that needs to be turned on after every inactivity interval.

User satisfaction is a crucial property to make a power management system friendlier for users. If the power manager causes inconvenience, the user will disable it or find ways to work around it, reducing its ability to save power. Thus, measuring and determining power management policies depending on the user satisfaction are an important feature of the method presented in this work.

Our method performs power optimisation by generating policies for the power manager of the operating system. These policies control the idle timeout value that the operating system uses as the time after which it puts the computer in a lower energy state. In order to balance the minimisation of the power usage while satisfying users, the power policies are a trade-off between the minimisation of energy consumption and the reduction of user satisfaction.

3.2.1. User Satisfaction

To calculate the user satisfaction, the inactivity intervals are collected and analysed before the control is enabled on the fleet. This helps to define an appropriate metric for satisfaction of the users and understand how a policy affects it.

Satisfaction is defined on a per turn off event basis. We consider a turn off interval unsatisfactory if the user has to turn the computer back on to continue working after the power manager triggers the power off. Otherwise, the turn off interval is considered satisfactory. The user satisfaction ratio is measured as the ratio of satisfactory turn off intervals to all of the turn off intervals triggered by the power manager [68].

To produce a formal definition of the user satisfaction for a value, T, of the idle timer of the operating system, I_i can be defined as the value of the duration of the i-th inactivity interval on a series of n total inactivity intervals. An inactivity interval is labelled *unsatisfactory* if its duration is greater than T, because the user has come back to a computer that was turned off by the power manager; and *satisfactory* otherwise.

Equation 1 and Figure 4 represent the user satisfaction for the i-th inactivity interval, $s(I_i)$, which is either 1 (the user is satisfied) if the interval was shorter than the idle timer set for the system or 0 otherwise (the user is completely dissatisfied). The values of the satisfaction per interval can be aggregated to calculate the global user satisfaction, S, as in Equation 2.

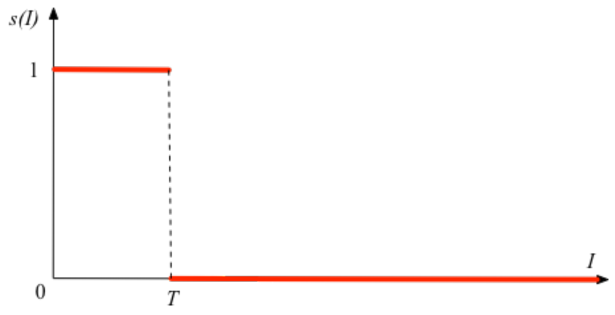


Figure 4: A step function for user satisfaction.

$$s(I_i) = \begin{cases} 1 & \text{if } I_i < T \\ 0 & \text{if } I_i \geq T \end{cases} \tag{1}$$

$$S = \frac{\sum_{i=1}^{n} s(I_i)}{n} \tag{2}$$

With this rule, if the user has an inactivity interval which lasts longer than the idle timer, they have to turn the computer back on when coming back to use the device and therefore, the idle interval should have been longer. Of course, there is a trade-off between satisfaction and saved power that the fleet administrator must take into account. This metric for user satisfaction, S, is subject to a problem as depicted on Figure 4: it makes no difference whether the user comes back to their computer one second after it is turned off or two days after. Any kind of inactivity interval is considered unsatisfactory if the idle time is equal to or longer than the value of the idle timer, T. However, the impact on the satisfaction is not same for the user after two days of inactivity as after a few minutes. Having to turn on the computer is less annoying in the first case.

Using the plain user satisfaction definition as presented in Equation 1 makes the timeout values generated prone to noisiness due to this long tail of longer inactivity intervals, as seen in Figure 3. It also reduces the effectiveness of the power manager to define shorter idle timer values on high satisfaction configurations.

Another way to measure user satisfaction is to use common industry metrics such as *Apdex* [62]. The original definition of *Apdex*, is represented in Equation 3. t is the target response time for a system; requests fulfilled in less than t are satisfying and those completed in less than $4t$ are tolerated by the user.

$$Apdex_t = \frac{SatisfiedCount + \frac{ToleratingCount}{2}}{TotalSamples} \tag{3}$$

Apdex needs to be adapted in order to be used for power management: we define the "tolerated" events as those with very long inactivity: the user is always less satisfied if their computer is turned off after a short period of inactivity than after a long time of inactivity. This yields an adapted *Apdex*, which is called S_a (see Equation 4 and 5) and shown in Figure 5:

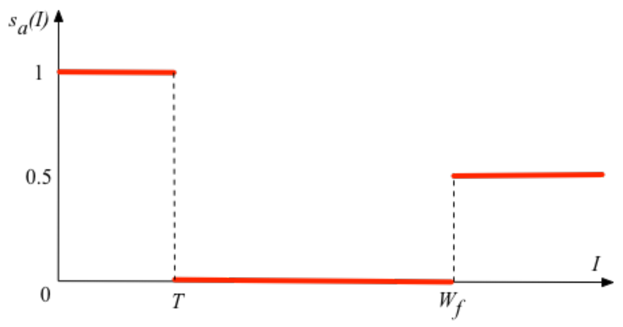


Figure 5: Three-stepped satisfaction weighting function.

$$s_a(I_i) = \begin{cases} 1 & \text{if } I_i < T \\ 0 & \text{if } T \le I_i < W_f \\ 0.5 & \text{if } I_i \ge W_f \end{cases} \tag{4}$$

$$S_a = \frac{\sum_{i=1}^{n} s_a(I_i)}{n} \tag{5}$$

The *Apdex* method has low progressivity and a strong memory for long inactivity intervals. For example, very long inactivity periods do not count as satisfactory, even if the user comes back after few days.

To further understand the user preferences around satisfaction, we have conducted a survey ($n = 49$), asking for the satisfaction in relation to the time away from the computer.

The survey showed that the insatisfaction decreases the longer the user have been away from the computer (from 93% when considering 5 minute inactivity to 33% when considering inactivity of more than one hour). In the case of using an intermediate power state, ACPI S3 ("Suspension"), the insatisfaction over having the computer turned off by a control also decreased further, since the session can be easily recovered.

To overcome these problems, the user satisfaction can be weighted as a function of the length of the inactivity interval in order to model the fact that the longer the inactivity, the less impact on user satisfaction. We can say that any inactivity interval that does not trigger a turn off is satisfactory, and those that trigger a turn off will be decreasingly unsatisfactory as the duration of the inactivity interval increases.

With a weighting function, w, like the one depicted on Figure 6, user satisfaction will grow linearly between the *starting point*, W_s, and the *final point*, W_f. Any idle time longer than W_f will be considered of full satisfaction, given that the user has spent more time away from the computer than usual. Equations 6 and 7 show the estimation of this function.

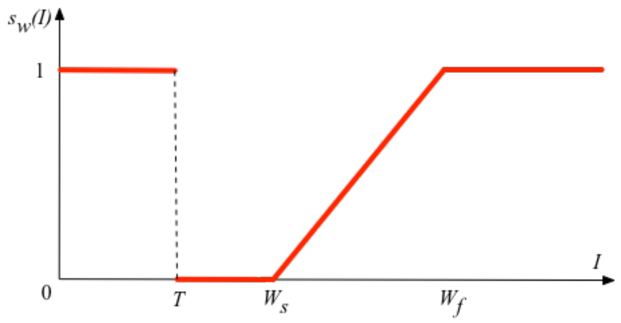


Figure 6: A simple linear satisfaction weighting function.

$$s_w(I_i) = \begin{cases} 1 & \text{if } I_i < T \\ 0 & \text{if } T \leq I_i < T + W_s \\ \dfrac{I_i - T - W_s}{W_f - T - W_s} & \text{if } I_i \in [T + W_s, T + W_f] \\ 1 & \text{if } I_i > W_f \end{cases} \tag{6}$$

$$S_w = \frac{\sum_{i=1}^{n} s_w(I_i)}{n} \tag{7}$$

By using a weighting function, a more realistic metric of user satisfaction is obtained. Furthermore, the metric converges to high values quicker than before, and allows for smaller timeout values, increasing the energy saved, as shown in the experiment section.

A survey can be carried out when analysing the properties of the fleet that is going to be managed, to determine the W_s and parameters W_f. The fleet administrator can adjust them depending on the target they have: higher power savings or better user experience.

In our experiments, we have chosen $W_s = 60$ and $W_f = 1800$ as an approximation that represents a common user pattern on the fleets we have collected data from.

3.2.2. Removed Inactivity

In addition to user satisfaction, the other fundamental measurement is how much energy is saved by a given policy. In this work this is defined as *removed inactivity*, R. This is the proportion of time that the computer is powered off after applying the power management optimisation rather than wasting energy by idling inactive. Using this metric instead of raw power allows for an easier characterisation of savings without physically measuring the power draw of each computer.

Any removed inactivity value can be converted to energy with the function of energy consumption that characterises the computer being studied. The R value is used directly as it is generic and does not depend on the specific computer model to estimate the potential and actual savings in energy. Nor does it depend on the fleet size or the characteristics of the computers installed in the fleet.

The value of R can be estimated by the ratio of time that the computer is off to the total inactivity time. For any inactivity interval that is longer than the idle time, the computer idles for a period equal to the idle timer value, T. After the timer expires, the computer is turned off. Therefore, the inactivity time is reduced from I_i to $I_i - T$. Equation 8 depicts the formal definition of the R estimator. This calculation is done before the power control is enabled, since otherwise, the automatic turn off of the computers would alter the user behaviour.

$$R = \frac{\sum_{i=1}^{n} (I_i - T) \text{ if } I_i > T}{\sum_{i=1}^{n} I_i} \tag{8}$$

There are some interesting properties of R. First, the expected value of R is bounded by the total inactivity time (i.e. more than the current idle total time cannot be saved since the computers are already off or being used).

3.2.3. Timeout calculation with user satisfaction

The definition presented on Equation 2 is very interesting for the definition of user satisfaction *a posteriori*, but the interest for the power manager lies on the ability to define a proper value of the idle timer in a way that guarantees a certain level of satisfaction for users defined by the administrator. For this purpose, we can make use of the inverse function of S, which should be evaluated at the target user satisfaction value.

For a simple user satisfaction estimator, like the one defined on Equation 2, the timeout value can be estimated as the percentile, p, of the inactivity model of the user, p being the target satisfaction.

The power manager uses a set of utilisation logs to estimate the initial value to set the idle timer. It can then be evolved as the users develop new behaviours.

The satisfaction function in Equation 7, does not have an inverse. Therefore, another approach should be used to find the optimal timeout, T. We have used a root-finding algorithm (the Brent Method [69]) to estimate the timeout that brings user satisfaction to the desired target.

Other approaches, such as genetic algorithms or more classical root finding might also be used, but the cost of the calculation and the precision has to be considered, as this calculation must be done for each user in each time slot.

4. Fleet power optimisation tool

In order to allow PC fleet administrators to generate power management policies analytically and experiment with fleet parameters, a tool has been developed, implementing the method described in this work. A discrete event simulation approach is used to experiment with fleet parameters.

If the fleet itself is used in the design and experimention process of the power management of variables, a significant amount of time and resources is required. Furthermore the process can affect the users of the fleet. Therefore, a simulation approach was used to calculate the value of idle timers and the models for different variables of the fleet.

The analysis carried out by the tool is based on the collection of usage logs from the users of the fleet before the control is enabled transparently for them. These logs contain crucial information that represents the user behaviour in order to model it. Logs are collected with collection agents installed in each computer of the fleet. These agents report events (such as activity or a turn off determined by the user) to a central database. This aggregated and anonymised data is used by the tool.

Collecting these usage logs is not always possible, for example, when a new fleet is being planned. Generating usage logs for hypothetical fleets is also possible by using the simulation code with some high-level configuration parameters, such as the number of users and basic schedule information.

The simulation has been implemented as a discrete event simulation using Python and the SimPy framework. The simulation is implemented with a simulation agent style, with each agent modelled as an independent SimPy process that interacts with the rest.

4.1. Power management tool use cases

Power management optimisation follows a cycle that starts with the collection of the usage logs from the unmanaged fleet. With those logs, certain analysis can be performed, such extracting power management policies, studying best options for user merging or understanding how different changes and variations to the fleet will affect its power usage. Finally, cross validation can be used to select the policy that best fits the needs of the fleet.

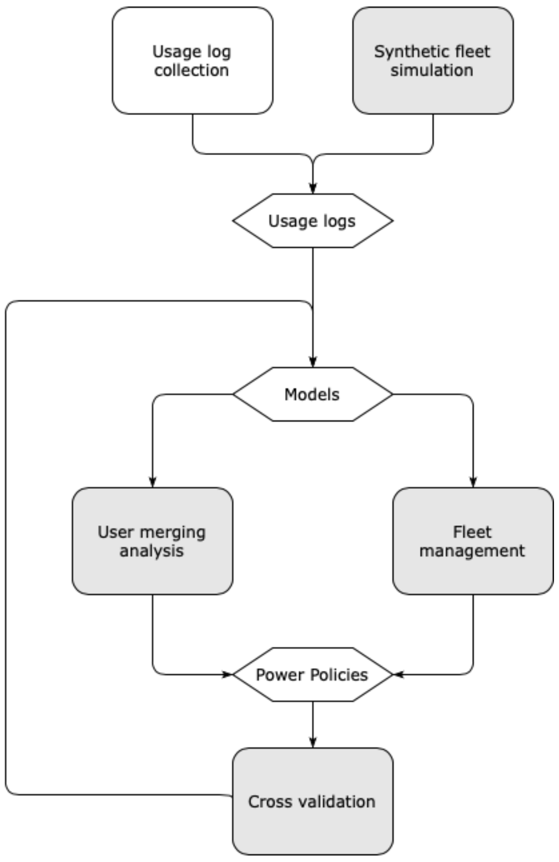


Figure 7: Flow diagram of the life cycle of fleet power management.

The tool described in this work implements four use cases, represented in grey in Figure 7 (usage log collection is performed by agents developed separately from this tool):

Synthetic fleet simulation is possible with the tool. With few parameters, such as the probability distributions for user activity, inactivity periods and the turn offs initiated by the user, a fleet can be simulated and the logs generated can be used for testing and developing policies with no need to monitor and collect real logs from a fleet.

User merging analysis can be used as a tool to find the most appropriate merging of user behaviour. By harnessing the data merging techniques that are designed to handle very large fleets, policies can be generated using groups of users and/or with time aggregations to find groups of users or times of the week for which behaviour is similar.

Fleet power management determines the idle timer for power management policies by analysing the usage logs from the users of the fleet. This tool can be used directly to optimise the power consumption of a fleet for which logs are available. The calculation of the idle timer is done analytically, by applying the weighted satisfaction technique described above.

Cross validation offers a simulation tool for further understanding of user behaviour. Policies can be cross validated with other usage data sets or variables of the fleet, such as comparing usage logs produced during a regular week with those of a holiday week. New methods for managing the fleet can be tested. These may differ from the operating system's idle timeout functionality.

Thus, this is a flexible tool which gives the fleet administrator great insight into the fleet and to quickly iterate in order to compare several policies. The tool shows the effects of using a small number of users as sample group for setting policies for the whole fleet. This is especially interesting for very large fleets. It is also useful for researchers who can iterate very quickly on new power manager modules and policies.

4.2. Enforcement of power management policies

The actuation of the power management policies will generally be done by means of the embedded power manager of the computer operating system, controlled by the idle timer that our method and tool will be calculating for each hour of the week and user.

The ACPI standard [70] specifies several intermediate power states (from S1 to S5). The initial implementation on this work is using S5 ("Soft Off") as the power state that the user takes their computer to when turning it off. Other common intermediate states like S3 ("Standby") or S4 ("Hibernation") can also be used, with the impact of higher power consumption in comparison to S5, but lower recovery time. The former needs to be considered in the calculation of removed inactivity while the latter will have impact on the satisfaction function, which should allow for higher satisfaction even when the agent is turning off computers more frequently.

Although modern operating systems will generally allow progressively putting the computer into lower energy states (i.e. first into stand by and later soft off or hibernation), and also manage different devices independently (e.g. put the PCI Express buses in low energy mode or spin down the disks); we are managing a single idle timer per computer.

4.3. Usage log collection agents

In order to generate power management policies for existing fleets, usage logs from unmanaged fleets are collected during a period of time determined by the administrator. These logs, which monitor the state of the computer at any given time, are collected by a collection agent installed in each computer of the fleet.

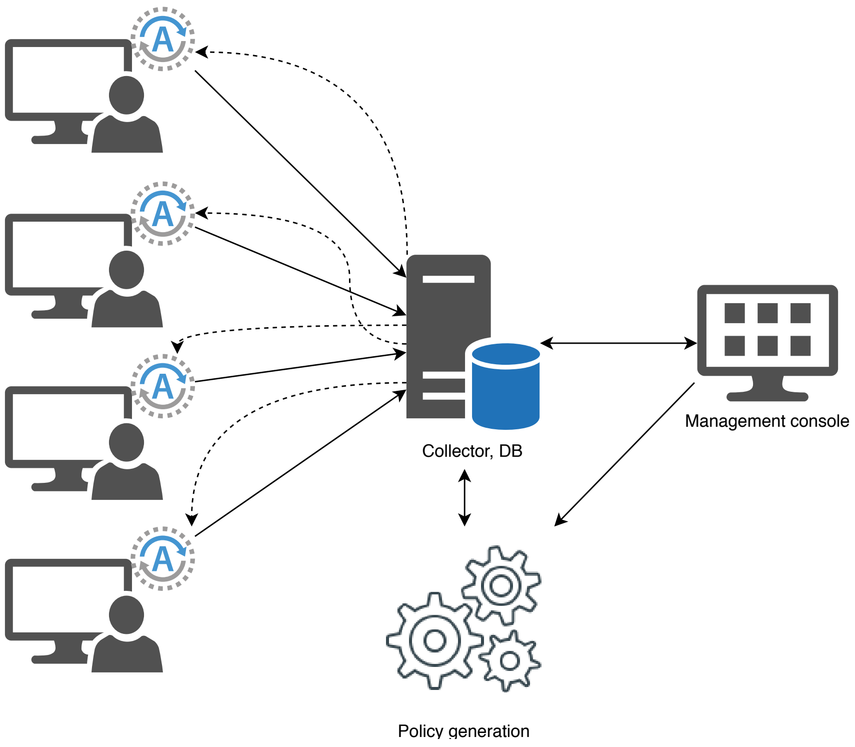


Figure 8: Log collection and management architecture.

Figure 8 represents the architecture of the system. Each of the workstations have a collection agent installed which reports activity and inactivity intervals to the central collector, which in turn stores the logs on a database. These are the logs used as input to the policy generation engine described in this paper, that is able to generate power management policies for each of the workstations. The central collector shows reports and summaries to the administrator.

The collection agent works by periodically polling the system API `GetLastUserInput()` [71] in Windows systems. As yet no measurement is supported on GNU/Linux or macOS machines, although equivalent system calls do exist for these operating systems.

The collection agents installed determine the computer state every 60 s and report state updates to a central database node. There is some basic data cleanup, such as removing duplicate entries, in this central node, and the usage log files are produced with anonymised data.

The agent is a very lightweight process that consumes between 5 and 10 MiB of RAM and a negligible amount of CPU: it wakes every minute as default, performs a system call and then a single API call over the network. Even on higher frequency setups, the cost of running the agents is trivial for current computers. The implementation is based on a push model from the computers of the fleet into the central database to avoid wasteful polls when computers are off and to leave computers pace themselves when under load. The timer to wake up the computer gets randomised per computer to avoid pushing data synchronously to the central database and overload it.

Collection for intervals longer than 60 s necessarily trigger duplicate entries: every 60 s, a report from the node to the database indicates that an interval is active and started at time t, until the poll to `GetLastUserInput()` indicates that the interval is finished. The central node retains only the last report, which indicates the complete interval.

Furthermore, this approach incurs in a *quantisation* effect: since polling and reporting happen every 60 s (configurable), intervals shorter than this time are not detectable. Practical experience during data collection with agents shows that this is not a problem, However, for some fleets this collection interval might need tuning to more frequent polling.

These logs are structured as a list of time intervals that are reported to a central database node. Each of the intervals is described as a tuple $(computer, state, starttime, duration)$. This list of intervals is the basis for the modelling of the user activity and inactivity, and the implementation of the tool for generating power management policies.

4.4. Model implementation

This section explains the implementation of the models for power optimisation, and the options the simulation implemented provide.

4.4.1. Modelling of user directed turn off events

When a user decides to turn off their computer, an event is recorded in the log by the usage collector agent, the same way activity or inactivity intervals are recorded. Modelling user directed turn off events is different from the other variables, since it requires modelling not only the duration of the intervals, but also their probability of occurrence.

While the activity and inactivity intervals always happen in succession, turn off events might or might not happen in a given cycle of the simulation loop. There are therefore two characteristics to be modelled for the turn off events that are triggered by the user: duration and probability. The duration is modelled in the same way as the rest of events, however the probability is estimated by the number of turn off events that happen in the given time slot in the logs.

There are several options to model the probability of a user turning off their computer in a given time slot. For example, a binomial distribution could be fitted so it can generate turn off events as needed, with the expectation of triggering the same number of positive events as turn offs in a given time slot as those recorded in the logs. However, for this option, an estimation of the total number of simulation runs is necessary, and this figure is not usually available.

The option selected for this work is to trigger the same number of turn off events in a time slot as those recorded on the usage logs, and randomising their appearance over the time slot uniformly.

4.4.2. Model merging

Generating a different model for each user can lead to over fitting or excessively noisy models when there is little data provided by the utilisation logs. The effect of over-fitting can be mitigated by using data merging adequately if not enough data is available or changing the distribution to fit (both cases are supported by the tool).

The default operation mode is with merging disabled, to ensure that the most personalised model can be fitted for each user, and merging will only be activated if the administrator requests it.

In the unmerged operation mode, different models are fitted per user and hour of the week, meaning that 168 different models per user are generated. Each model will contain the four variables used for the analysis and simulation. Two merging modes, which can be combined, have been implemented: by user and by time.

When merging by user, the data for all the users is merged and shared hourly so only 168 models in total are needed (one per hour). Conversely, when merging by time, data will be merged on the time dimension and each of the users will have a single model for each of the 168 hours. When merging by both dimensions, a single, global, model per variable is shared by all users in all hours.

In the example fleet of 265 computers described in sections 4.5.1 and 5.1, the number of models is reduced from 44,520 to 265 when merging by time, to 168 when merging by user and only one when merging by both dimensions.

For merging, all the data of the models to be merged is joined and a single model is then fitted on the totality of intervals. When using different log partitions for fitting and validating the models, the merging is done for the fitting log only.

The advantages of data merging include a reduction in the number of models and the fact that each of the models works with substantially more data, which reduces noise. However, the drawbacks are that the preprocessing cost for the simulation is larger and drawing random numbers for each of the estimated variables can be much more expensive. This is the case when the empirical distribution is based on interpolating *splines*: drawing random numbers from them is of quadratic complexity [72].

It is necessary to indicate that merging is not clustering. The latter is a technique that will identify groups of users that have similar behaviour and will make it possible to model them together. Merging, as implemented in this work, will either group all of the users together or none at all. Clustering has not been implemented due to the complexity of the methods and the small difference in removed inactivity found with the fleets studied between fully merged and fully unmerged models, which indicates that there is low probability of improving the results with clustering.

4.4.3. Model extrapolation

A further model transformation must be done when simulating fleets of different sizes or of different durations than that of the logs used for the modelling.

When running a simulation for a fleet larger than the one that generated the logs, additional users will be extrapolated from the existing data. One of the simplest methods is to do a merge per computer and then simply increase the number of users in the simulation. Since all users and computers share a single set of models, the expected average behaviour of all users is exactly the same or very similar. This is the approach implemented in the tool.

Other options for generating additional users are to create random new users by generating random combinations of models from existing users.This can be done by using randomly selected hourly models from existing users.

In the case of reducing the fleet size, either random or systematic sampling can be applied to reduce the number of users. In the implementation of the tool, we use random sampling of the users in the case of a reduction on the number of users to simulate.

Finally, when expanding or reducing the duration of the simulation, there is no modification of the models to perform, since we can keep using them as much as we would like: since the models are cycling over the length of a week, we can run less or more weeks of simulation as desired.

4.5. Simulation processes and software design

Each of the elements of the simulated fleet (users and computers) are independent simulation processes, supported by other elements such as the models, or *activity distributions*, and statistic-gathering utilities.

Each simulation process is represented as a different Python class, which implements the SimPy generator API. These processes operate as an infinite loop yielding SimPy timeouts to schedule the discrete events.

The UML diagram in Figure 9 shows the object oriented design of the main simulation processes.

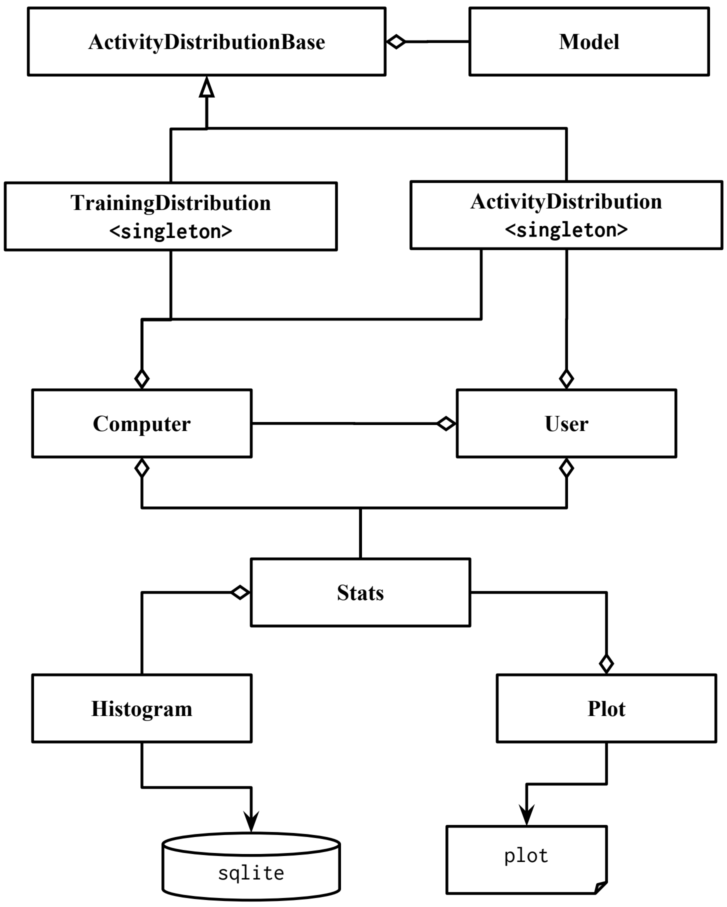


Figure 9: UML diagram for the main simulation processes.

The main processes are the `User` and `Computer` classes. These are instantiated in pairs of one `User` instance and one `Computer` instance and are provided a unique Computer ID, called CID, typically the computer host name, or the MAC address.

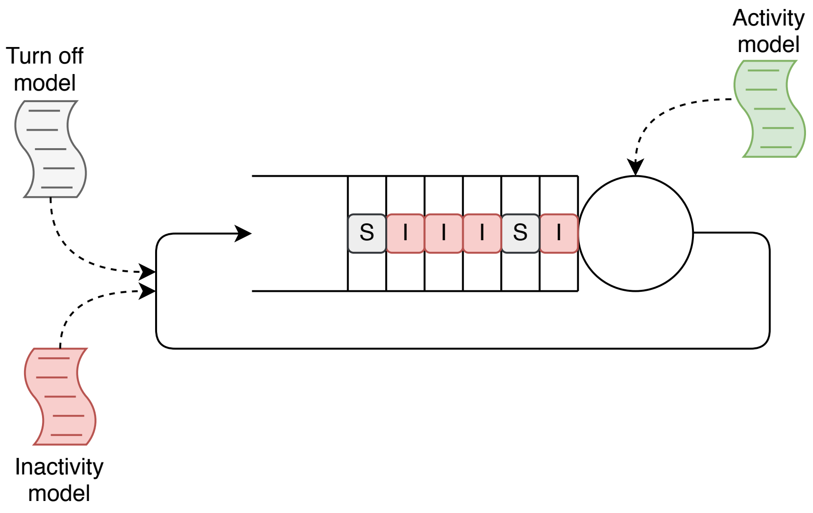


Figure 10: G/G/1 queue model for a simulated user.

This pair of components behave as an generalised queue ($G/G/1$), as depicted on Figure 10. The inter-arrival time for the client requests is determined by the inactivity and user turn off models fitted from the usage logs. Each of the models is used to generate different types of events to be processed by the computer (the queue server), responding with an activity time generated by the activity model, or by changing its own state to turn off for the established time.

Each of the computers has a simulated software agent that determines the timeout to be set on the power manager of the operating system of the computer every hour. The timeout is defined hourly with the data on the logs.

The `ActivityDistribution` object (a singleton) encodes all the behaviour of the users and computers contained in the activity logs provided as simulation input. This data set is used as the basis for the behaviour of the users at run time. The second object, `TrainingDistribution`, behaves in exactly the same manner, but is used as the data set for fitting the models. This data set is used to establish the initial timeout during the model fitting process.

In summary, `TrainingDistribution` contains the models used to determine the timeout of each time slot, and `ActivityDistribution` is used to simulate the behaviour of the users and to validate the quality of the settings of the policies generated by the logs.

All of the statistics generated by the simulation are stored on the `Stats` singleton object. This object is backed by a set of `Histogram` objects (one per hour and metric) that are backed by a sqlite3 database. This minimises the impact of storing statistics directly on memory: the simulator stores all metrics of interval duration in raw format. It also provides persistence for further analysis and comparison between simulation runs.

The `Stats` object is used for basic health validation at the end of the simulation (correctness of total simulated time, event count, etc.), and to generate plots of the results of the simulation. These plots are generated by the `Plot` object, which iterates over and aggregates statistics to display the time series describing the simulation results.

4.5.1. Optimisation of the model fitting

There are two major code fragments that require optimisations and on which the simulator spends most of the run time: fitting the distributions used by the models and generating optimal models when merging them.

Given the number of models managed by the simulator, when fitting the distributions from the input data one of the most effective ways to make the simulator effective is to optimise the empirical distribution by fitting and random number generation (sampling).

In [73, p. 462] there is a proposal for a fast implementation that runs in $O(1)$ time for sampling. It also proposes pre-calculating the difference of the cumulative distribution function (CDF) values in order to reduce the number of operations. This latter optimisation has also been implemented, and although the memory usage is higher, the performance is notably improved.

However, when using Law's implementation with very large data sets, the fitted objects have an excessively high memory footprint. A further optimisation is performed in order to implement the fitted distribution as a *spline* function [74] that keeps the memory footprint low while retaining fast sampling properties.

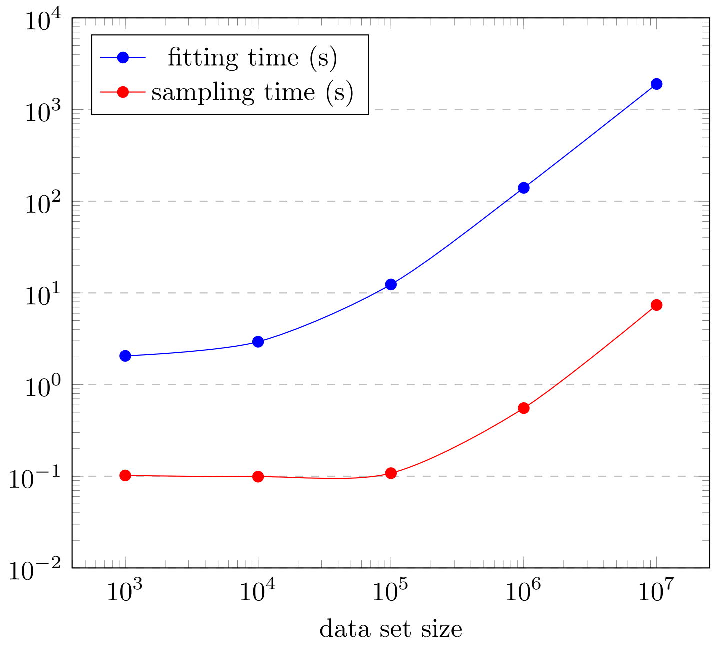


Figure 11: Benchmark for empirical distribution fitting and sampling.

Figure 11 shows the results of a benchmark performed with different sample sizes (up to 10^7 elements) indicating how both the fitting and sampling of the empirical distribution grows linearly with the sample size.

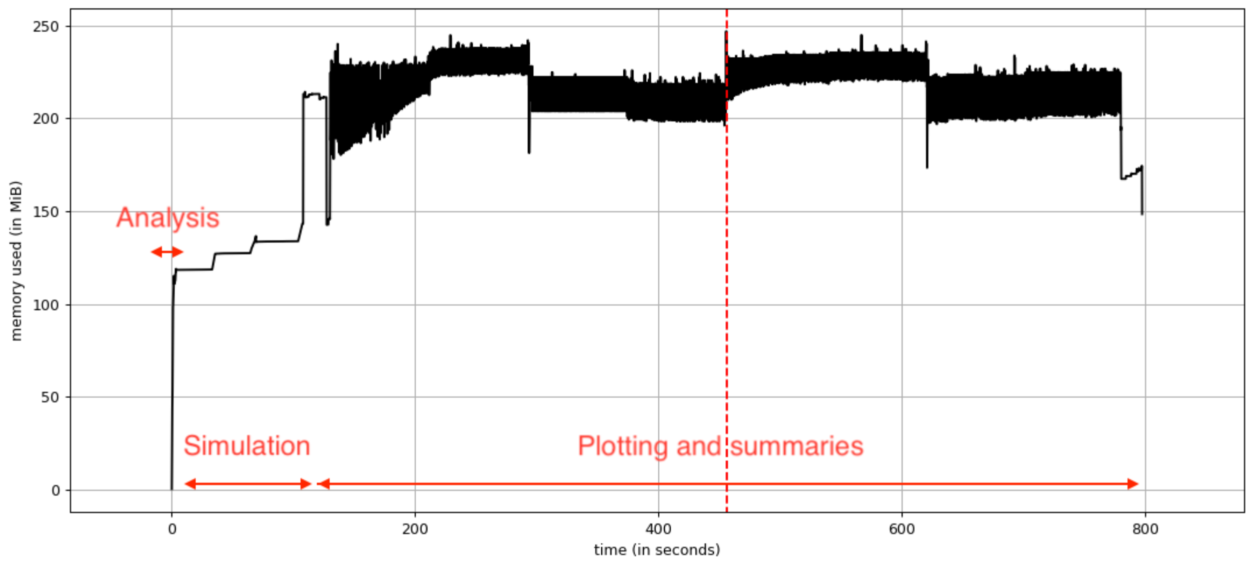


Figure 12: Memory usage of the simulator over time.

Figure 12 indicates the memory usage profile for the simulator over time, indicating how a simulation of 265 PCs behaves and scales. This simulation run needed 3 runs in order for the results to converge with confidence intervals of 0.5 % points width at 95% confidence. The process has three phases: analysis, simulation and plotting and summaries.

The analysis is almost instantaneous, spending about 2 seconds parsing and constructing the models and emitting the policy. The second phase, the simulation, scales with the number of runs. Figure 12 shows that the whole process took three runs due to quick convergence, and memory usage is proportional to the number of runs needed to converge. Summary generation and plotting of results is the activity that uses most memory and takes the most time.

Table 4: Performance of different empirical distribution implementations

Variant	Time	Memory usage
Law, unmerged	2m44	281 MiB
Law, merged	>20m	—
spline, unmerged	2m10	287 MiB
spline, merged	3m37	237 MiB

Table 4 shows a comparison of the memory footprint and the execution time of the simulation phase with 265 PCs over two months, with two implementations of the empirical distribution and with either no merge at all or merge by the two dimensions, in order to compare the performance with the highest number of elements to fit.

The *Law* version is the naive implementation. As expected, with large data sets like the one produced when merging, the simulation takes over 20 minutes. Using the *spline* implementation keeps the runtime low and, since the raw data is not kept in memory, also reduces the memory footprint when merging. Both implementations have *merged* and *unmerged* variables.

The models stored in the `ActivityDistribution` and distribution objects are laid out in a tree-like structure with three levels: computer, day and hour. There are eight instances of this tree: for each of the interval types modelled as the four variables (activity, inactivity, off duration and off probability). There are two trees: one with the models used for the analysis and policy generation and the other for the simulation runs to validate the policies. The biggest cost of the fitting phase is the huge fan out factor that this structure has, because look ups on this data structure are expensive and not optimal for any cache. Furthermore, merging is a complex operation that needs careful iteration over the structure, which is slow and bug-prone.

A new object is introduced in the simulation, the `Model`. This object is meant to encode all four distributions of each user in a time slot. Encoding the models together reduces the size of the hourly tree and improves the memory access speed. Only two instances of this new tree are needed (one for the trace used to fit the models and another for the trace used to cross-validate). Also, it makes data merging more efficient and easier to push fitting to the last moment with a lazy evaluation strategy.

4.5.2. Scalability of the simulator

The simulator must scale to model and simulate the fleets of large corporations, which can be up to tens of thousands computers. The time needed to simulate grows linearly with the number of computers being simulated [63], while the model fit is dependent on the size of the log files being used, in particular on the number of events it contains.

5. Validation and Results

This section summarises the experiments carried out to validate our method and the implementation provided, both of the analytical method and the simulation tool.

5.1. Usage log file partition

The usage log files collected empirically can be partitioned to generate input log files for the tool: one part is used to fit the models of the users, and the others to validate the models prior to applying them to the real fleet.

The implementation of the method presented in this work uses a simple method of temporal partition: all log files collected from the fleets are split in two halves by cutting them off at half their duration.

We have chosen this method since the utilisation logs collected from the company span up to 8 weeks during two different periods of time. In this way, each of the partitions have at least data for 4 weeks and allow to not incur in overfitting when, for example, a holiday happens.

Data set	Collection dates	Users	Duration	Notes
1.1	Oct 5 - Nov 1, 2014	265	3.9 weeks	No laptops
1.2	Nov 2 - Nov 30, 2014	265	4 weeks	
2.1	Apr 1 - Apr 30, 2015	64	4.1 weeks	No laptops, no suspensions, min. 50 inactivity intervals
2.2	May 1 - May 29, 2015	75	4 weeks	

Table 5: Data set partition schema.

The partition of the two original data sets collected, shown in Table 5, produces two pairs of data sets, with different properties. Data sets 1.1 and 1.2 do not include laptops because the power manager of laptops is optimised differently from workstations and their behaviour depends on the power source (battery or electrical grid). Also, users are more accustomed to suspending these devices when not in use.

Data sets 2.1 and 2.2 contain only desktop computers with 50 or more events on the collected utilisation logs. This avoids including users that were wrongly logged or computers that were on or off for the whole period. It also contains only computers without automatic suspensions, since having power control enabled on the machine prior to the optimisation would make the inactivity intervals shorter.

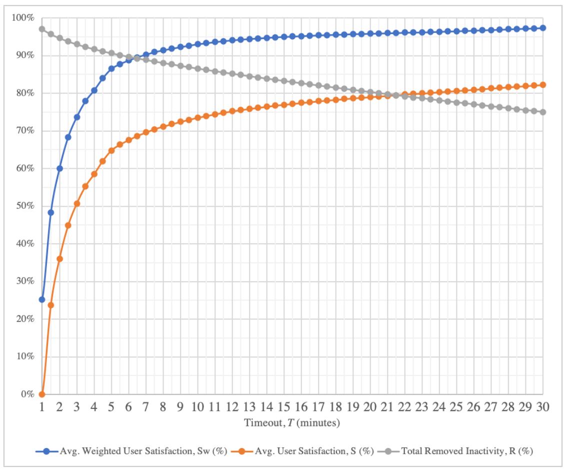
5.2. Empirical distribution validation

In order to validate the fitting of the empirical distribution implementation based on *splines*, Kolmogorov-Smirnov tests [66] are added to the implementation. The testing code is available with the project source [67].

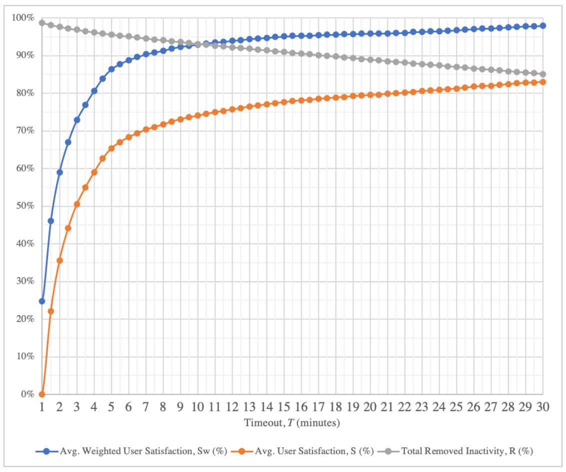
The tests run automatically, and generate a random sample of size 50,000 for a set of distributions (normal, exponential and Pareto) to validate the goodness of the fit. This sample is then used to fit an empirical distribution, generate a second sample from the fitted distribution and finally execute the Kolmogorov-Smirnov test on the two samples, which pass at 99.9% confidence level.

5.3. Comparison of S and S_w

We have compared the performance of the S and S_w estimators for the user satisfaction, in relation to the timeout value, T and the removed inactivity achieved, R.



(a) Data set 1.1.



(b) Data set 2.1.

Figure 13: S, S_w and R as function of the idle timer, T.

Figure 13 shows how the values of S, S_w and R evolve as functions of the value of the idle timer calculated from utilisation logs of two fleets (data sets 1.1 and 2.1 in Table 5). Shorter idle timers yield excellent energy savings, with values of R as high as 95%, meaning that almost all the idle time of computers is removed. However, the user satisfaction, S, is very poor. Increasing the user satisfaction, either S or S_w reduces the power savings, as expected. There can be an important gap in power savings when moving the satisfaction target: for example, in Figure 13a going from 90% to 95% target for S_w will reduce R from 90% to approximately 85%; while going from 95% to 99% target S_w will reduce R from 85% to 69%.

There is a big gap between the weighted S_w and non-weighted S satisfaction functions. S penalises heavily any inactivity interval that is longer than the timeout value (T), whereas S_w will consider longer intervals satisfactory after the W_f interval. For very long inactivity intervals or big values of W_f, S_w and S will converge.

5.4. Validation and performance of the analysis

In order to carry out the validation, a target weighted satisfaction (S_w) of 90% is selected and the corresponding policy for this satisfaction is computed with the adjusting data set indicated in the second column of Table 6. Then, this policy is used in the validation data set indicated in the third column. The satisfaction (both S_w and S) and removed inactivity are shown in the last columns. The models are adjusted with no merging at all, to provide the policies that are most personalised to users.

Test case	Adjusting data set	Validation data set	Average timeout ($\bar{T}$)	Avg. Weighted Satisf. ($\bar{S_w}$)	Average Satisf. ($\bar{S}$)	Total Rem. Inactivity (R)
1	1.1	1.1	385.30 s $T_m = 223.54s$	90.56% ± 3.26	74.87% ± 28.74	86.76%
2		1.2		90.79% ± 11.15	75.92% ± 28.92	87.48%
3	1.2	1.2	388.56 s $T_m = 220.23s$	90.75% ± 4.23	77.30% ± 28.10	87.11%
4		1.1		86.37% ± 12.96	68.52% ± 33.21	86.99%
5	2.1	2.1	350.34 s $T_m = 195.36s$	90.49% ± 3.54	75.65% ± 28.70	94.16%
6		2.2		89.75% ± 10.40	66.12% ± 37.97	95.07%
7	2.2	2.2	307.73 s $T_m = 169.38s$	90.84% ± 3.57	75.94% ± 29.05	94.42%
8		2.1		78.83% ± 12.69	59.46% ± 37.60	95.22%

Table 6: Validation results for the power management policy generation.

Combinations of both partitions of the utilisation logs are used to validate that the method is sufficiently robust, yielding results of Weighted User Satisfaction (S_w) that in most cases slightly exceed the target satisfaction. The variability (expressed as the standard deviation) of the results obtained when the same data set is used for adjusting and validation is due to rounding long inactivity intervals. Also, there is an increase in the variability of S_w when validating with a data set different from the adjusting data set, as expected due to the different behaviour of the users.

Test cases 2 and 4 are interesting since the data set 1.1 contains one holiday day (November 1st). The effect can be observed when using the data set 1.2 for adjusting the models (test case 4): since it does not have the longer inactivity of the holiday for adjusting the models, it drives slightly down the resulting satisfaction when using the data set with the holiday as validation. In test case 8, a similar situation happens, as data set 2.1 contains the Easter week, with three holidays. The resulting timeout is noticeably higher in test case 4 while the satisfaction on test case 6 gets reduced. The table also shows the median timeouts, T_m, which also reflect this behaviour.

The Removed Inactivity (R) remains between 86% and 95% in all the cases, indicating that most of the inactivity time can be converted to power savings by turning off computers that spend very long periods of time idling (in the night and over weekends and holidays).

5.4.1. Impact of model merging

The optimiser can merge models by time, by user, or by both time and user. As explained in Section 4.4.2, model merging allows for different policies to fit the models. Table 7 and 8 summarise the impact of different merging combinations for the pair of data sets used in test cases 2 and 6 respectively. It shows the average metric, the median idle timer (T_m), plus the standard deviation of each of the metrics that vary per user

(T, S_w and S).

	Merged by user	Unmerged by user
Merged by time	$\bar{T} = 407.99s$ $T_m = 407.99s$ $\bar{S_w} = 91.83\% \pm 14.75$ $\bar{S} = 76.87\% \pm 32.82$ $R = 84.87\%$	$\bar{T} = 463.36s \pm 142.46$ $T_m = 430.83s$ $\bar{S_w} = 92.75\% \pm 14.78$ $\bar{S} = 77.44\% \pm 32.59$ $R = 82.89\%$
Unmerged by time	$\bar{T} = 429.43s \pm 195.04$ $T_m = 374.56s$ $\bar{S_w} = 91.04\% \pm 14.11$ $\bar{S} = 76.55\% \pm 32.14$ $R = 86.35\%$	$\bar{T} = 385.30s \pm 454.93$ $T_m = 223.54s$ $\bar{S_w} = 90.79\% \pm 11.15$ $\bar{S} = 75.92\% \pm 28.92$ $R = 87.48\%$

Table 7: Comparison of results and data merging for test case 2.

	Merged by user	Unmerged by user
Merged by time	$\bar{T} = 428.00s$ $T_m = 428.00s$ $\bar{S_w} = 91.43\% \pm 12.88$ $\bar{S} = 69.24\% \pm 32.16$ $R = 93.46\%$	$\bar{T} = 494.93s \pm 452.30$ $T_m = 366.17s$ $\bar{S_w} = 90.37\% \pm 15.86$ $\bar{S} = 67.28\% \pm 33.12$ $R = 93.87\%$
Unmerged by time	$\bar{T} = 517.61s \pm 297.46$ $T_m = 483.63s$ $\bar{S_w} = 91.93\% \pm 13.67$ $\bar{S} = 71.12\% \pm 32.24$ $R = 92.80\%$	$\bar{T} = 350.34s \pm 417.90$ $T_m = 195.36s$ $\bar{S_w} = 89.75\% \pm 10.40$ $\bar{S} = 61.12\% \pm 37.97$ $R = 95.07\%$

Table 8: Comparison of results and data merging for test case 6.

According to the results, while there is some change on both data sets, offering a slightly positive effect in overall satisfaction, the impact is certainly marginal.

It is worth noting how the standard deviation of $\bar{T}$ decreases when merging happens: when merging, all the intervals are put together into a single distribution to fit and extract a timeout (where in the case of using S, it will directly be the percentile corresponding to the target satisfaction). Under these conditions, the longer inactivity intervals on the long tail of the distribution influence less the timeout calculation and the computers will get policies that tend to be more homogeneous, and in the case of merging by both dimensions just one policy for all computers. At the same time, the deviation of satisfaction increases, as the policies start to match less the user behaviour than when no merging is used.

This effect is also explained by looking at the median idle time, T_m, which is an indicator of centrality that is not as affected by outliers as the mean. In both test cases, merging increases the median timeout value, meaning that the timeouts are covering more cases to fulfil the satisfaction target.

Both the increase of the median timeout and the reduction of the standard deviation are explaining that the main effect of the merging is reducing the influence of long inactivity intervals on the power policy, at the expense of less personalised policies.

5.5. Generation of new fleets

The tool is able to generate simulations of computer fleets with no input usage log, by providing simple parameters to describe the fleet under study (number of users, start and end time of the day, distributions of activity and inactivity). There are two types of parameters: global, which describe general attributes of the fleet to be generated; and temporal, which model attributes subject to change over the course of the week or the work day.

Table 9 lists the default global parameters of the simulator: the number of users and the length of the work days (with start and end times).

Simulation time	4 weeks
Number of users (N)	1000
Work start time of users	9:00 AM
Work end time of users	5:00 PM

Table 9: Default global parameters for fleet generation and their default values.

Table 10 summarises the temporal parameters; all of them follow a random distribution with a specific mean and deviation. The Off duration time, which determines the duration of the turn off intervals triggered by the user, is worthy of special note. This parameter is dependent not only on the day and hour of the week, but also, on how much time is remaining until the next day start is available. Outside of working hours, should a user decide to turn the computer off, it will be in that state until the next start of a working day.

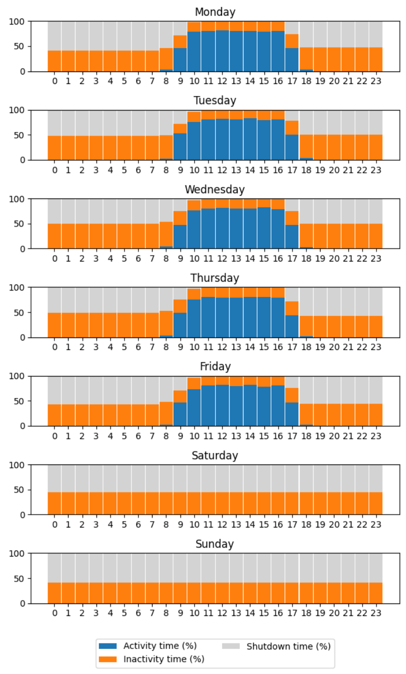
These parameters have been chosen qualitatively, with similar values to the ones encountered on the utilisation logs of the fleet studied, and serve as an example of a potential fleet to be generated by the simulator. Any other parameter value describing different fleets can be chosen by the user.

	During working hours	Outside of working hours
Activity	$A \sim \mathcal{N}(\mu = 3600, \sigma = 60)$	$A \sim \mathcal{N}(\mu = 360, \sigma = 30)$
Inactivity	$I \sim \mathcal{N}(\mu = 1800, \sigma = 60)$	$I \sim \mathcal{N}(\mu = 1800, \sigma = 60)$
Off fraction	$O_f = 1\%$	$O_f = 70\%$
Off duration	$O_d \sim \mathcal{N}(\mu = 1800, \sigma = 900)$	"until next working hour"

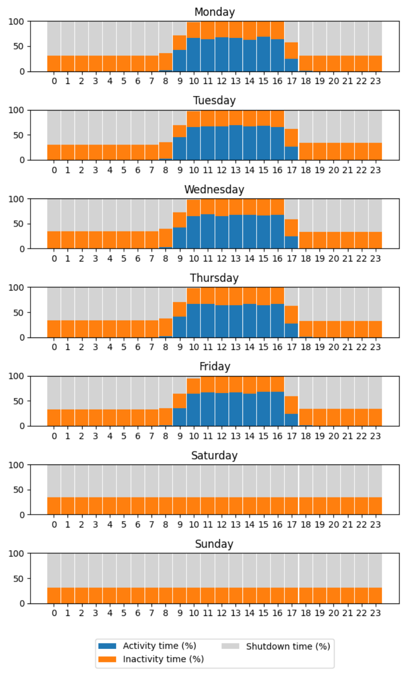
Table 10: Default temporal parameters for fleet generation and their default values.

We have validated the fleet generated by the simulator qualitatively and observationally. Several experiments are performed, altering the values of global and local parameters: two different work start and end times of the users, and two distributions for the activity and inactivity.

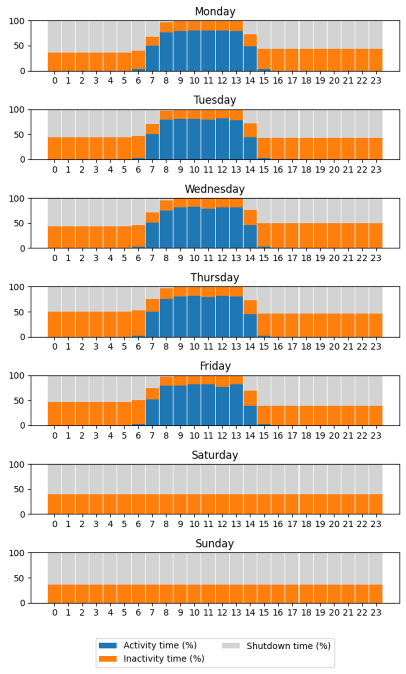
Figure 14 compares two modified parameters of a fleet generated by the simulator: the distribution of the activity time and the working hours start and finish times. Fleets with longer activity intervals (Figures 14a and 14b) show a higher proportion of activity time per hour. Figures 14c and 14d show the shift of the work hours to represent the flexibility of the tool to generate multiple configurations.



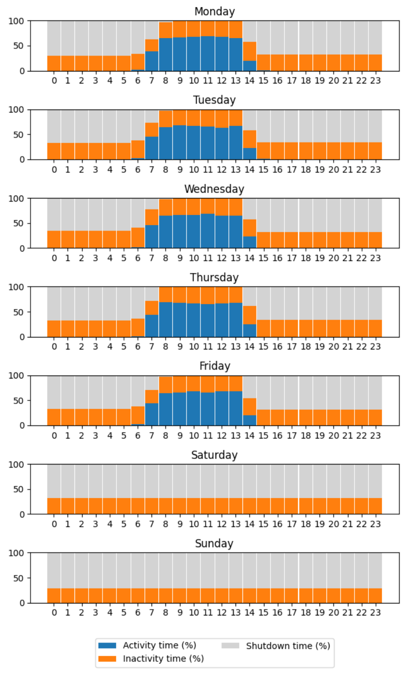
(a) Day from 9:00 AM to 5:00 PM, $A \sim Lognormal(\mu = 3600,\ \sigma = 600)$



(b) Day from 9:00 AM to 5:00 PM, $A \sim Lognormal(\mu = 1800,\ \sigma = 600)$



(c) Day from 7:00 AM to 2:00 PM, $A \sim Lognormal(\mu = 3600,\ \sigma = 600)$



(d) Day from 7:00 AM to 2:00 PM, $A \sim Lognormal(\mu = 1800,\ \sigma = 600)$

Figure 14: Fleet generation with different distribution of activity (A) and working hours

6. Conclusions and Future Work

This paper presents a novel technique to model computer fleets in large organisations and optimise power consumption by turning off computers preemptively during large periods of inactivity. The model the technique uses allows for a simple data collection protocol. By deploying an agent across the fleet it offers a variety of data aggregation and customisation properties and, most importantly, can simulate different power saving policies and their impact on energy savings and user satisfaction. The application of the policies is simple: they are the timeout for the computer to turn off itself, which can be adjusted fleet-wide by any existing fleet management software.

User satisfaction is the central concept of the optimisation: the analytical method uses this metric to optimise the timeout policy and the power savings. It is also a novel approach to determine the timeout for each of the users of the fleet.

This method can save most of the energy wasted by a fleet when computers are left idling, while guaranteeing a high level of user satisfaction. Furthermore, the tool provided makes it possible to study other variables of the fleet before they are deployed, to understand the cost and optimal setup.

Our method is highly explainable, since all the usage logs are summarised into a set of four distributions (instead of more complex Markov or machine learning methods) while also retaining a high level of customisability by allowing to balance the energy savings and user satisfaction with a single knob: the user satisfaction target. Fleet administrators that are more interested in optimising for savings can reduce the satisfaction target to have shorter timeouts, while administrators that prefer to conserve user satisfaction can do so by raising the target at the expense of more power usage.

Future work includes the addition of multiple ACPI low power states of computers, utilisation of clustering for additional personalisation of policies per user and dynamic adaption of policies via continuous policy generation for the fleets. Finally, using a clustering algorithm could be a good evolution in the future to find more suitable sets of users where the merging can yield better results.

References

[1] S. Solomon, D. Qin, M. Manning, Z. Chen, M. Marquis, K. Averyt, M. Tignor, H. Miller, et al., The physical science basis, Contribution of working group I to the fourth assessment report of the intergovernmental panel on climate change 2007 (2007) 235–337.

[2] S. Murugesan, Harnessing green IT: Principles and practices, IT professional 10 (1) (2008) 24–33.

[3] D. Korn, R. Huang, D. Beavers, T. Bolioli, M. Walker, Power management of computers, in: IEEE International Symposium on Electronics and the Environment, 2004. Conference Record. 2004, 2004, pp. 128–131. `doi:10.1109/ISEE.2004.1299701`.

[4] L. Belkhir, A. Elmeligi, Assessing ict global emissions footprint: Trends to 2040 & recommendations, Journal of Cleaner Production 177 (2018) 448 – 463. `doi:10.1016/j.jclepro.2017.12.239`.

[5] C. A. Webber, J. A. Roberson, M. C. McWhinney, R. E. Brown, M. J. Pinckard, J. F. Busch, After-hours power status of office equipment in the usa, Energy 31 (14) (2006) 2823 – 2838. doi:10.1016/j.energy.2005.11.007.

[6] E. Reinhard, B. Champion, N. N. Schulz, R. Gould, E. Perez, N. Brown, Computer power consumption and management: "earth, wind, and fire: Sustainable energy for the 21stcentury" research experience for undergraduates, in: 2011 IEEE/PES Power Systems Conference and Exposition, 2011, pp. 1–8. doi:10.1109/PSCE.2011.5772621.

[7] G. Procaccianti, L. Ardito, A. Vetro, M. Morisio, Energy efficiency in the ICT-profiling power consumption in desktop computer systems, Prof. Moustafa Eissa, 2012.

[8] M. Chetty, A. Brush, B. R. Meyers, P. Johns, It's not easy being green: understanding home computer power management, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2009, pp. 1033–1042.

[9] J. M. Walker, Power management for networked computers: A review of incentive programs, in: 2009 IEEE International Symposium on Sustainable Systems and Technology, 2009, pp. 1–6. doi:10.1109/ISSST.2009.5156753.

[10] Puppet [cited August 2020].
URL https://puppet.com

[11] 1e nightwatchman [cited May 2020].
URL https://www.1e.com/products/nightwatchman-pc-power-management/

[12] M. B. Srivastava, A. P. Chandrakasan, R. W. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 4 (1) (1996) 42–55. doi:10.1109/92.486080.

[13] Y.-H. Lu, L. Benini, G. D. Micheli, Operating-system directed power reduction, in: ISLPED'00: Proceedings of the 2000 International Symposium on Low Power Electronics and Design (Cat. No.00TH8514), 2000, pp. 37–42. doi:10.1109/LPE.2000.155250.

[14] D. Ramanathan, R. Gupta, System level online power management algorithms, in: Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537), 2000, pp. 606–611. doi:10.1109/DATE.2000.840847.

[15] C.-H. Hwang, A. C.-H. Wu, A predictive system shutdown method for energy saving of event-driven computation, ACM Trans. Des. Autom. Electron. Syst. 5 (2) (2000) 226–241. doi:10.1145/335043.335046.

[16] L. Benini, A. Bogliolo, G. A. Paleologo, G. D. Micheli, Policy optimization for dynamic power management, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 (6) (1999) 813–833. doi:10.1109/43.766730.

[17] L. Benini, A. Bogliolo, G. D. Micheli, A survey of design techniques for system-level dynamic power management, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8 (3) (2000) 299–316. doi:10.1109/92.845896.

[18] Q. Qiu, M. Pedram, M. Pedram, M. Pedram, M. Pedram, Dynamic power management based on continuous-time markov decision processes, in: Proceedings of the 36th Annual ACM/IEEE Design Automation Conference, DAC '99, ACM, New York, NY, USA, 1999, pp. 555–561. doi:10.1145/309847.309997.

[19] Q. Qiu, Y. Tan, Q. Wu, Stochastic modeling and optimization for robust power management in a partially observable system, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '07, EDA Consortium, San Jose, CA, USA, 2007, pp. 779–784.

[20] Y. Tan, Q. Qiu, A framework of stochastic power management using hidden markov model, in: 2008 Design, Automation and Test in Europe, 2008, pp. 92–97. doi:10.1109/DATE.2008.4484668.

[21] H. Jung, M. Pedram, Dynamic power management under uncertain information, in: 2007 Design, Automation Test in Europe Conference Exhibition, 2007, pp. 1–6. doi:10.1109/DATE.2007.364434.

[22] T. Simunic, L. Benini, P. Glynn, G. D. Micheli, Event-driven power management, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20 (7) (2001) 840–857. doi:10.1109/43.931003.

[23] C. Phillips, S. Singh, D. Sicker, D. Grunwald, Applying models of user activity for dynamic power management in wireless

devices, in: Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '08, ACM, New York, NY, USA, 2008, pp. 315–318. doi:10.1145/1409240.1409277.

[24] C. Phillips, S. Singh, An empirical activity model for wlan users, in: IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, 2008, pp. 2065–2073. doi:10.1109/INFOCOM.2008.272.

[25] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, G. D. Micheli, Dynamic power management for nonstationary service requests, IEEE Transactions on Computers 51 (11) (2002) 1345–1361. doi:10.1109/TC.2002.1047758.

[26] S. O. Luiz, A. Perkusich, A. Lima, Multisize sliding window in workload estimation for dynamic power management, IEEE Transactions on Computers 59 (2010) 1625–1639. doi:10.1109/TC.2010.90.

[27] E.-S. Jung, N. H. Vaidya, An energy efficient mac protocol for wireless lans, in: IEEE infocom, Vol. 3, Institute of Electrical Engineers Inc. (IEEE), 2002, pp. 1756–1764.

[28] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, D. W. Clark, Coordinated, distributed, formal energy management of chip multiprocessors, in: ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005., 2005, pp. 127–130. doi:10.1145/1077603.1077637.

[29] R. J. Minerick, V. W. Freeh, P. M. Kogge, Dynamic power management using feedback (2002).

[30] L. L. D. Bulay-og, R. C. Gustilo, Power management control system for computer laboratories, Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 10 (1-8) (2018) 61–66.

[31] W. Lin, H. Wang, Y. Zhang, D. Qi, J. Z. Wang, V. Chang, A cloud server energy consumption measurement system for heterogeneous cloud environments, Information Sciences 468 (2018) 47 – 62. doi:10.1016/j.ins.2018.08.032.

[32] J. Flinn, M. Satyanarayanan, Energy-aware adaptation for mobile applications, SIGOPS Oper. Syst. Rev. 33 (5) (1999) 48–63. doi:10.1145/319344.319155.

[33] B. Lin, A. Mallik, P. Dinda, G. Memik, R. Dick, User- and process-driven dynamic voltage and frequency scaling, in: 2009 IEEE International Symposium on Performance Analysis of Systems and Software, 2009, pp. 11–22. doi:10.1109/ISPASS.2009.4919634.

[34] B. Setz, F. Nizamic, A. Lazovik, M. Aiello, Power management of personal computers based on user behaviour, in: Proceedings of the 5th International Conference on Smart Cities and Green ICT Systems - Volume 1: SMARTGREENS,, INSTICC, SciTePress, 2016, pp. 409–416. doi:10.5220/0005762604090416.

[35] I. Turkin, O. Vdovitchenko, Model and Methods of Human-Centered Personal Computers Adaptive Power Control, Springer International Publishing, Cham, 2019, pp. 323–348. doi:10.1007/978-3-030-00253-4_14.

[36] E.-Y. Chung, L. Benini, G. De Micheli, Dynamic power management using adaptive learning tree, in: Proceedings of the 1999 IEEE/ACM International Conference on Computer-aided Design, ICCAD '99, IEEE Press, Piscataway, NJ, USA, 1999, pp. 274–279.

[37] G. Dhiman, T. S. Rosing, Dynamic power management using machine learning, in: Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design, ICCAD '06, ACM, New York, NY, USA, 2006, pp. 747–754. doi:10.1145/1233501.1233656.

[38] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1) (1997) 119 – 139. doi:10.1006/jcss.1997.1504.

[39] F. Kong, P. Tao, S. Yang, X. Zhao, Genetic algorithm based idle length prediction scheme for dynamic power management, in: The Proceedings of the Multiconference on "Computational Engineering in Systems Applications", Vol. 2, 2006, pp. 1437–1443. doi:10.1109/CESA.2006.4281864.

[40] V. L. Prabha, E. C. Monie, Hardware architecture of reinforcement learning scheme for dynamic power management in embedded systems, EURASIP Journal on Embedded Systems 2007 (1) (2007) 065478. doi:10.1155/2007/65478.

[41] H. Jung, M. Pedram, Supervised learning based power management for multicore processors, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29 (9) (2010) 1395–1408. doi:10.1109/TCAD.2010.2059270.

[42] W. Liu, Y. Tan, Q. Qiu, Enhanced q-learning algorithm for dynamic power management with performance constraint, in: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10, European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 2010, pp. 602–605.

[43] R. Candrawati, N. L. Hashim, Analysis of control, learn, and knowledge model for computer power management using unified modelling language, Advanced Science Letters 22 (5-6) (2016) 1327–1330. `doi:10.1166/asl.2016.6759`.

[44] R. Candrawati, N. L. Hashim, Adaptive approach in handling human inactivity in computer power management, Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 8 (8) (2016) 65–69.

[45] R. Candrawati, N. L. B. Hashim, A learning approach for computer power management, Advanced Science Letters 22 (10) (2016) 2738–2741.

[46] S. V. Skakun, N. N. Kussul, A. G. Lobunets, Implementation of the neural network model of users of computer systems on the basis of agent technology, Journal of Automation and Information Sciences 37 (4) (2005).

[47] L. Csurgai-Horváth, J. Bitó, Primary and secondary user activity models for cognitive wireless network, in: Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on, IEEE, 2011, pp. 301–306.

[48] G. Theocharous, S. Mannor, N. Shah, P. Gandhi, B. Kveton, S. Siddiqi, C.-H. Yu, Machine learning for adaptive power management., Intel Technology Journal 10 (4) (2006).

[49] A. Shye, B. Ozisikyilmaz, A. Mallik, G. Memik, P. A. Dinda, R. P. Dick, A. N. Choudhary, Learning and leveraging the relationship between architecture-level measurements and individual user satisfaction, in: 2008 International Symposium on Computer Architecture, 2008, pp. 427–438.

[50] Avob energy saver [cited May 2020].
URL `https://www.avob.com/gestion-energetique/`

[51] Data synergy PowerMAN [cited May 2020].
URL `http://www.datasynergy.co.uk/products/powerman/`

[52] New boundary PwrSmart [cited May 2020].
URL `https://www.newboundary.com/products/pwrsmart-service`

[53] Aptean verdiem surveyor [cited May 2020].
URL `https://www.aptean.com/solutions/vertical-business-applications/aptean-verdiem/`

[54] IBM tivoli [cited May 2020].
URL `https://www-01.ibm.com/software/tivoli`

[55] Autonomic software ansa [cited May 2020].
URL `https://autonomic-software.com/products/power-manager.html`

[56] Certero PowerStudio [cited May 2020].
URL `https://www.certero.com/products/powerstudio/`

[57] Dell KACE [cited May 2020].
URL `https://www.quest.com/products/kace-systems-management-appliance/`

[58] Eventzero greentrac [cited May 2020].
URL `https://enoten.com/Greentrac/`

[59] Faronics power save [cited May 2020].
URL `https://www.faronics.com/en-uk/products/power-save`

[60] Ivanti power manager [cited May 2020].
URL `https://help.landesk.com/ld/help/en_US/LDMS/10.0/ISV/tv-Powman.htm`

[61] Verismic power manager [cited May 2020].
URL `https://www.cloudmanagementsuite.com/`

[62] P. Sevcik, Defining the application performance index, Business Communications Review 20 (2005).

[63] R. M. Llamas, D. F. García, J. Entrialgo, J. Garía, User inactivity modeling and simulation for power management of PC fleets, in: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Spects '15, Society for Computer Simulation International, San Diego, CA, USA, 2015, pp. 1–6. `doi:10.1109/SPECTS.2015.7285276`.

[64] M. J. Evans, J. S. Rosenthal, Probability and statistics: The science of uncertainty, 2nd Edition, W. H. Freeman, 2009.

[65] J. Alstott, E. Bullmore, D. Plenz; powerlaw: a python package for analysis of heavy-tailed distributions, PLoS ONE 9 (1) (2014). `doi:10.1371/journal.pone.0085777`.

[66] F. J. M. Jr., The Kolmogorov-Smirnov test for goodness of fit, Journal of the American Statistical Association 46 (253) (1951) 68–78. `doi:10.1080/01621459.1951.10500769`.

[67] Project source code in GitHub.
URL `https://github.com/asi-uniovi/power-simulation`

[68] D. García, J. Entrialgo, R. Medrano, J. García, A technique to minimize the power consumption of computer fleets, in: Proc. of the Intl. Conf. on Advances In Computer and Electronics Technology- ACET 2014, 2014, pp. 100–105.

[69] R. P. Brent, Algorithms for Minimization without Derivatives, Prentice-Hall, 1973.

[70] Advanced configuration and power interface specification, version 6.3 [cited August 2020].
URL `https://uefi.org/sites/default/files/resources/ACPI_6_3_final_Jan30.pdf`

[71] GetLastUserInput function [cited May 2020].
URL `https://msdn.microsoft.com/en-us/library/windows/desktop/ms646302.aspx`

[72] K. Toraichi, K. Katagishi, I. Sekita, R. Mori, Computational complexity of spline interpolation, International Journal of Systems Science 18 (5) (1987) 945–954. `doi:10.1080/00207728708964021`.

[73] A. M. Law, Simulation Modeling and Analysis, 5th Edition, McGraw-Hill, 2015.

[74] Ž. Ivezić, A. J. Connolly, J. T. VanderPlas, A. Gray, Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data, Princeton University Press, 2014.