Providing Software Asset Management Compliance in Green Deployment Algorithm

Noëlle Baillon-Bachoc, Eddy Caron, Arthur Chevalier, Anne-Lucie Vion

Providing Software Asset Management Compliance in Green Deployment Algorithm

Noëlle Baillon-Bachoc[1], Eddy Caron[2], Arthur Chevalier[1,2], and Anne-Lucie Vion[1]

[1] Orange S.A.
noelle.baillon@orange.com
annelucie.cosse@orange.com
[2] Univ Lyon, EnsL, UCBL, CNRS, Inria, LIP
F-69342, LYON Cedex 07, France
{firstname.surname}@ens-lyon.fr

Abstract. Today, the use of software is generally regulated by licenses, whether they are free or paid and with or without access to their sources. The world of licenses is very vast and unknown. Often only the public version is known (a software purchase corresponds to a license). For enterprises, the reality is much more complex, especially for main software publishers. Very few, if any, deployment algorithm takes Software Asset Management (SAM) considerations into account when placing software on Cloud architecture. This could have huge financial impact on the company using theses software. In this article, we present the SAM problem more deeply, then, after expressing our problem mathematically, we present *GreenSAM*, our multi-parametric heuristic handling performance and energy parameters as well as SAM considerations. We will then show the use of this heuristic on two realistic situations, first with an Oracle Database deployment and second with a larger scenario of managing a small OpenStack platform deployment. In both cases, we will compare *GreenSAM* with other heuristics to show how it handles the performance/energy criteria and the SAM compliance.

Keywords: Licensing, Cloud, Software Asset Management, Ressources Management, Deployment

1 Introduction

In contrast with public licensing, we have to license software by fulfilling what we call a metric. A metric defines a way to calculate the number of licenses required for software, so we can license it with x licenses under metric A or y licenses under metric B. The price of licenses depends on the used metric. The metrics of software licenses are defined contractually either by the general conditions of sale found on the publisher's website or by a contract between the customer and the publisher. The general terms and conditions of sale available online can be updated, but a license purchased before a change in the legal text must follow the old version of the metric definition, so it is necessary to have a continuous

monitoring of these metrics and the ability to identify and retrieve the legal documents of each license when needed.

One problem stems from the fact that we can interpret the metrics definitions in contracts in different ways and thus the customer is not in conformity because of an unintended misunderstanding. A concrete and recent example of this is the trial between SAP and Diageo [1], a company that uses SAP products. Because of a lack of usage rights understanding due to legal uncertainty about licenses, the company was fined £55 million ($75 million) for non-compliancy and more recently a disagreement between SAP and AB InBev [2] where the latter was facing a $600 million penalty. Most companies can't afford or face such penalties. Another telling number is the fact that 85% [3] of companies sanctioned for counterfeiting were unknowingly non-compliant which shows the need for tools to verify and ensure compliance automatically.

The Cloud also brings issues, there is no more obvious relation between software and hardware. Without a border, every instance of an application may run on every server and this dynamicity impact metrics. To solve this case, it is necessary to recreate links between the resources used by software and the hardware layer by any means and at the deepest levels. The problem is that monitoring software usage in the Cloud is a difficult task, made easier by software inventory tools provided by publishers. The problem arises when it is necessary to link physical and software inventories while keeping track of when they were made because a virtual machine or container can move much more easily in the Cloud than in traditional architectures. These problems are critical as compliance is expressed as 'yes' or 'no'. A simple mistake can cause serious damages as seen before. Then comes the problem of identifying the software itself. Indeed, the discovery tools go back to the names of the executables and try to recognize the associated products. Several techniques exist to improve this identification as mentioned in the state of the art [4, 5] but it remains generally difficult. We will focus on the management of these products here under the assumption that the identification is total and accurate.

We can see that the larger a Cloud is, the greater is the need for tools that can track software usage, identify licenses used, verify compliance and manage effective placement. This tool is part of a process called "Software Asset Management" (SAM) which must be able to carry out the actions described above but also have some industrial objectives allowing to have a return on use and therefore to conduct a supplier strategy or even carry out portfolio consolidations but we will not focus on these aspects.

In addition, the functioning of metrics forces the user to measure the usage of the Cloud in a very extensive and very precise way. The metric function takes in resources ranging from simple physical resources (number of cores, number of users) to much more complicated resources (number of indirect accesses). Each metric will target one to several of these and to be able to manage all the metrics of each application in the Cloud we need to know exactly how much material or immaterial resources are being consumed as mentioned above.

One extra problem is that the compliance verification is done retrospectively (after the use so it is too late to take action) and take a lot of time: for a large Cloud, it can take several months to check the compliance for all installed software without appropriate tools. Besides, to reduce and optimize the license consumption according to the needs, software asset managers should have information from all sides (machine management, human resources, accounting) because of the lack of standardization of the metrics. Indeed, editors can use in their metrics everything they want from physical attributes of servers to number of employees in different teams.

Finally, there is a total lack of deployment algorithm considering compliance and license cost. Only few papers showed an interest in SAM and often with the wrong assumption of one license per software. Also, only considering SAM during the deployment is not possible, as reducing the number of licenses with a processor number based metric could lead to placing all the software on a single processor server. Therefore, we need to use a multi-parametric heuristic. In 2014, data centers in the U.S. consumed an estimated 70 billion kWh, representing about 1.8% of total U.S. electricity consumption. Current study results show data center electricity consumption increased by about 4% from 2010-2014, and is expected to continue increasing in the near future by with the same rate. Based on current trend estimates, U.S. data centers are projected to consume approximately 73 billion kWh late 2020. To answer this trend and Software Asset Management problem we introduce *GreenSAM* which is a multi-parametric deployment heuristic taking into account Software Asset Management considerations as well as performance and energy. Performance and energy are contradictory - forming the basis for multi-objective optimization.

This paper is organized as follows: We start by presenting the state of the art of Software Asset Management and the very few papers talking about deployment algorithm with license considerations in Section 2. Then, in Section 3, we describe the major advancement of this paper, the *GreenSAM* heuristic before introducing two use cases for it in Sections 4 and 5. Finally, in Section 6, we conclude and discuss future directions to enhance the *GreenSAM* heuristic.

2 Related work

Genesis of Software Asset Management was in 1999 when Holsing and Yen [6] proposed a study leading to the first considerations on the model and identification of software. In 2004, Ben-Menachem and Marliss [7] underscored the need for investment and the creation of tools for such processes to ensure long-term management of software assets. In 2011, McCarthy and Herger [8] offered a solution to combine Information Technology (IT), processes and Software Asset Management: this requires the ability to scan all the infrastructure, make a license inventory, implement contract management and to produce reports on the state of readiness for compliance and verification. In 2014, Gocek *et al.* [9] described the SAM as tools for discovering and collecting information on instances of software used in monitored environments. Recently in 2017, Vion *et al.* [10]

made a brief survey of the existing SAM tools, their benefits and proposed a SAM model for a Cloud architecture. Moreover, in 2018, Chevalier *et al.* [11] proposed an efficient and economical way of handling metrics in Cloud environments for Oracle Database and showed that a deployment algorithm focused on software licenses could save money. During the same year [12] proposed optimization of the placement of virtual machines with many other parameters including license costs and showed that handling both problems of mapping virtual machines to physical machines and mapping applications to virtual machines leads to better results than considering the two problems in isolation. Even so, the problem is well formulated, it uses the fact that an application uses one license at most as we see in his UML model and that the number of licenses does not rely on underlying architecture which is an unrealistic view of the licenses. We need to tackle this problem.

On the other hand, to identify software several works were first carried out in 2014, Han *et al.* [5] proposed a way to identify open-source software. Cho *et al.* [4] proposed a technique on proprietary software to reduce counterfeiting through a birthmark technique located in the executable. A data discovery agent could use this birthmark to correctly identify the software. Then in 2018 Vion *et al.* [13] proposed another way to identify the software following a purchase by relying on ISO 19770 -3 and -4 [14] standards to correctly recognize the software and the associated rights.

As many IT services and analyses use the Cloud and are dependent on large infrastructures that can be local or remote [15], more and more questions are being asked about the huge energy consumption of these infrastructures to supply and cool computing machines [16]. Research has been conducted to reduce these consumptions [17] and heuristics have been proposed such as *GreenPerf* [18] for example which introduces a performance and power consumption ratio to improve energy efficiency or energy-efficient framework dedicated to Cloud [19]. In the context of 5G, very recent research shows that it is possible to reduce consumption [20, 21]. The field of multi-parameter optimizations is very wide and many researches are based on Pareto fronts in fields ranging from sensor networks [22, 23] to virtual machine deployment [24].

3 GreenSAM: Energy and Software Asset Management

The *GreenSAM* heuristic aims at ensuring compliance during deployment of applications while saving energy and keeping a sufficient level of performance. This heuristic takes as input the set of servers we can deploy on and for each servers, its attributes which we describe later. First, *GreenSAM* will remove servers that will causes loss of compliance because of contractual rules. Then, it will compute the Pareto front based on the servers' attributes (here energy, performance, and license number) to eliminate non Pareto efficient ones giving us a subset. Then we compute scores for each server in this subset based on a formula optimizing the parameters described in Subsection 3.2. We finally deploy the software on the server with the best score. If two server get the same score,

then we choose at random. We can see in Fig. 1 the flow of the deployment algorithm using the *GreenSAM* heuristic.

These different steps are described in the following subsections:

3.1 Servers attributes

As performance and energy as huge research domains and evolve quickly, *GreenSAM* is based on agnostic parameters. It means that every parameter we input into *GreenSAM* are scores given by the user. The goal is to be able to compare servers based on these scores. If the user wants to use cutting-edge computing method for energy consumption, he will not have to use another heuristic each time and *GreenSAM* will not have to understand this cutting edge method. This way, we can enhance the accuracy of *GreenSAM* by enhancing the method to compute scores we give to it. Therefore, *GreenSAM* manipulate parameters

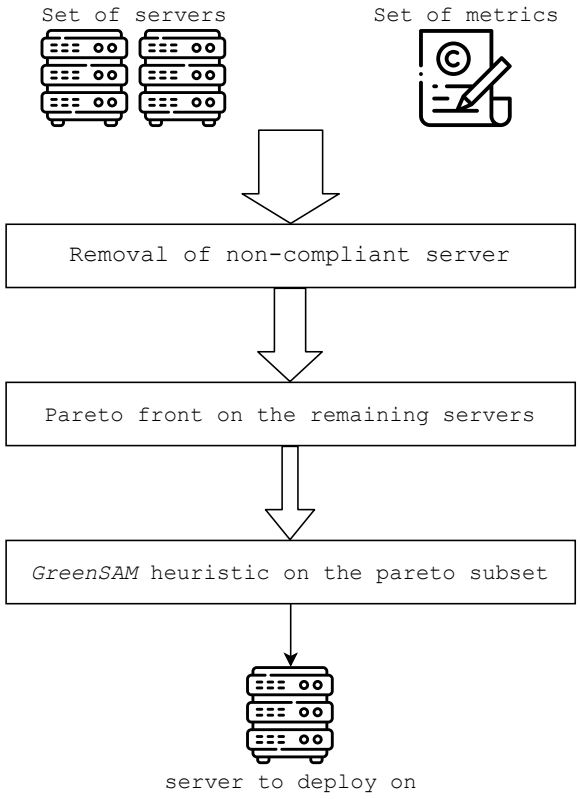

Fig. 1. Flow of deployment algorithm using *GreenSAM* heuristic. The first step removes the servers that cause a non-compliance situation if the software is deployed on them. Then, in second step, we apply a Pareto front to the remaining servers to keep only the Pareto optimal servers. Finally, to choose one in the Pareto subset, we apply the *GreenSAM* heuristic and take the best server.

without units but is able to compare servers by normalizing the attributes. Having two servers with 50 and 30 energy consumption will have the same order relation than two servers having 100 and 60 energy consumption.

In our case, the energy consumption is computed with the number of active cores on the server. The SAM score is given by the price of the licenses we would have to possess if we place the software on one particular server. Finally, we used a modular performance indicator. For example, the performance of a distributed high-performance computing application will be affected by the network speed but also by the parallel power of the machine (heterogeneous architecture or not, for example). Storage service in the Cloud will be efficient if many people can access it at the same time and if it has enough storage (more or less fast). We define different performance classes in *GreenSAM* to describe the performance calculation of each product if we want to have a precise index of whether or not a product is performing in a given environment. A product is therefore defined by its metrics, and its performance calculation function (performance class).

3.2 Multi-objective optimization

Any optimization problem will have design parameters whose best possible values from the viewpoint of the objectives are sought to be attained in the optimization process. The optimization task here is to map a set of software onto available resources, here servers. The three objective functions are defined with the following variables:

n The number of servers noted s
m The number of applications noted a
A_s The attributes of a server s
f_a The formula for the metric of the application a that takes into account attributes of servers
E_s The energy consumed by the server s for the first installation. When we deploy an application a on a server already containing one then $E_s = 0$.
PI_a The performance class of the application a. It is a function waiting for attributes of a server to give performance score. If the application a is not installed on the server s then $PI_a(A_s) = 0$.

Minimize energy consumption The total energy consumption 'E_T' of our deployment is then expressed as:

$$E_T = \sum_{s=1}^{n} E_s \times (\neg(\exists\ a \in [1..m]/a \in s)) \tag{1}$$

Maximizing performance To avoid putting all applications on the same server we added operational constraints. Each application will require at least 2 cores not used so $PI_a(A_s) = 0$ if a doesn't fit on s. The overall performance P_T is expressed as:

$$P_T = \sum_{s=1}^{n} \sum_{a=1}^{m} PI_a(A_s) \tag{2}$$

Minimizing software cost This objective function will stop the process if the metric computation brings a non-compliant state. The total license consumption L_T is expressed as:

$$L_T = \sum_{s=1}^{n} \sum_{a=1}^{m} f_a(A_s) \tag{3}$$

In most cases, machines that bring performance will have higher energy consumption, implying that objectives P_T and E_T are contradictory - forming the basis for multi-objective optimization.

We first compute the three criteria each time we deploy a product because the results per server may vary between each deployment, indeed the performance indicator can vary because of the past deployment as well as the license cost (for example the metric depends on the number of instance of the product on the server) which provide the Cloud dynamic state. Then we filter servers that are not compliant with the current product metrics (for example there can be constraints with country or language). Afterwards, we apply a Pareto front to this dataset to have a subset of potentially good servers. We still have to choose one of the "best servers" given by the Pareto front and so we implemented a function to give a score to each server of the subset. This score function will simply divide normalized performance score by the sum of normalized license and energy points. We add one to the denominator to avoid being in a case of a server having no energy and license consumption breaking the division. Finally, *GreenSAM* will return the first server of the sorted subset to deploy the software on it before starting again for the next product to deploy.

Note that in the case of the deployment of multiple products, we may not have the optimal deployment because of local optimization. *GreenSAM* goal is to optimize a unique deployment but could be enhanced later to handle multiple deployment at once.

4 Oracle Database Enterprise Edition Use Case

For both use cases, we use two datasets. The first one is a set of about 5000 servers coming from Orange™ Cloud (Orange™ is the first historical French multinational telecommunications corporation). This dataset allows us to express the efficiency of the Pareto front reducing these 5000 servers to subsets of tens of potential servers. The next datasets if used to compare *GreenSAM* with other heuristics described later in this Section. For each comparison between the four heuristics, we generated hundred datasets of 200 servers with random but realistic attributes. We then compute the average results of the four heuristics on these hundred datasets to make a fair comparison on many Cloud architectures.

In this use case, we deploy 10 Oracle databases in a new Cloud to allow development projects to use them. From the benefit of [11], we can prove that in our use case, it's better to focus on the processor metric which is defined as follows: To deploy a database d on a server s with c_s cores, a corefactor co_s and inside a cluster Cl_s then the number of licenses you need to have to install the database on t is the following:

$$L = \sum_{s=1}^{n} c_s \times co_s \times (s \in Cl_t) \times (\neg(\exists\ a \in Cl_t)) \quad \text{where } a \text{ is a previously installed database} \tag{4}$$

meaning you have to sum all the cores of the cluster in which the server t is, except if there is already a database somewhere in this cluster.

This upstream calculation avoids the deployment algorithm having to compute each metric of each product before making the Pareto front.

As we have only one product to deploy ten times the only performance indicator is defined as follows: We must have a minimum of 2 cores to be eligible and the performance score is the number of cores divided by the number of already installed databases on it.

We can see in Fig. 2 the Pareto front on the first deployment. In subsequent deployments, it is always interesting to deploy databases on the same server until this server no longer has enough cores available. Finally, in Fig. 3 we can see that the front of the Pareto only has a few servers left until the end of the deployment. In Fig. 4, we compared the result of this algorithm with three others on different sizes of Cloud:

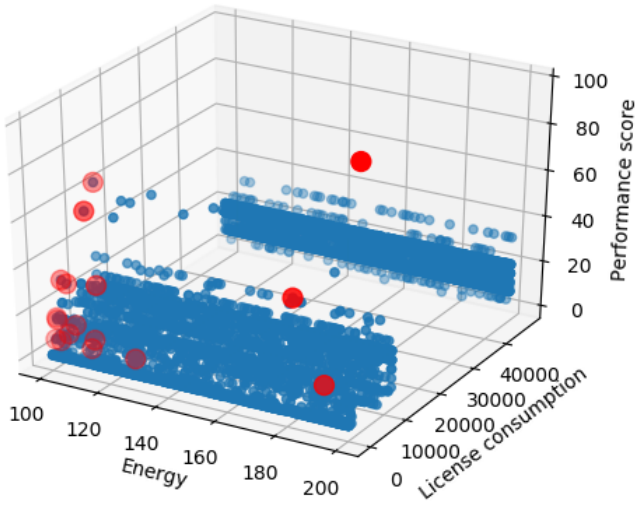

Fig. 2. 17 servers (in red) as a part of the Pareto front on the 5000 servers used for this simulation.

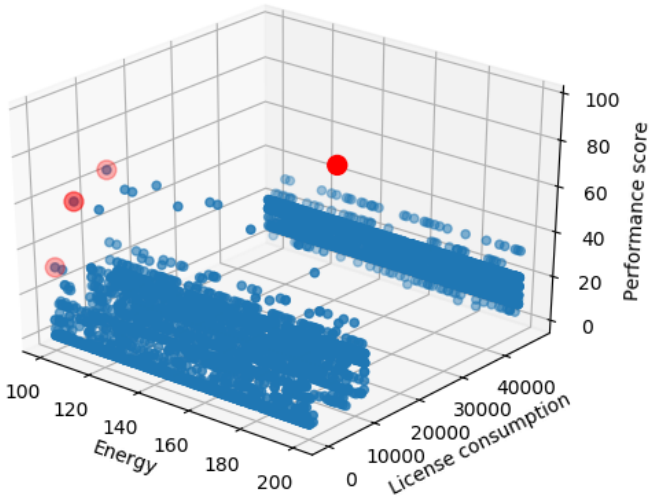

Fig. 3. Only four servers are part of the Pareto front at the end of the 5th deployment. Then until the end, there will be as many. In the end, 8 different servers will be used.

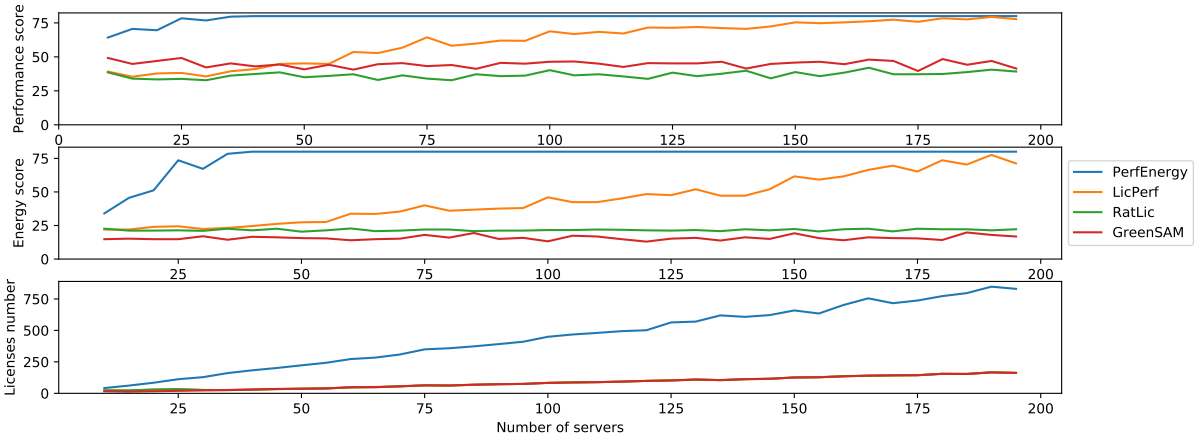

Fig. 4. *GreenSAM* is a lot better in performance and license criteria even though it loses around 30% of performance. RatLic is close but does not achieve the same performance and LicPerf is terrible in terms of energy as PerfEnergy.

- **PerfEnergy**: The first is an algorithm that promotes performance first, followed by energy.
- **LicPerf**: The second one focuses on the number of licenses and then on performance.
- **RatLic**: The last one focuses on the ratio performance by energy and then license consumption.

We can see in the results that *GreenSAM* is pretty good and succeeds in minimizing energy but at the price of performance. In the next section, we use *GreenSAM* in a more complex deployment that is an OpenStack environment.

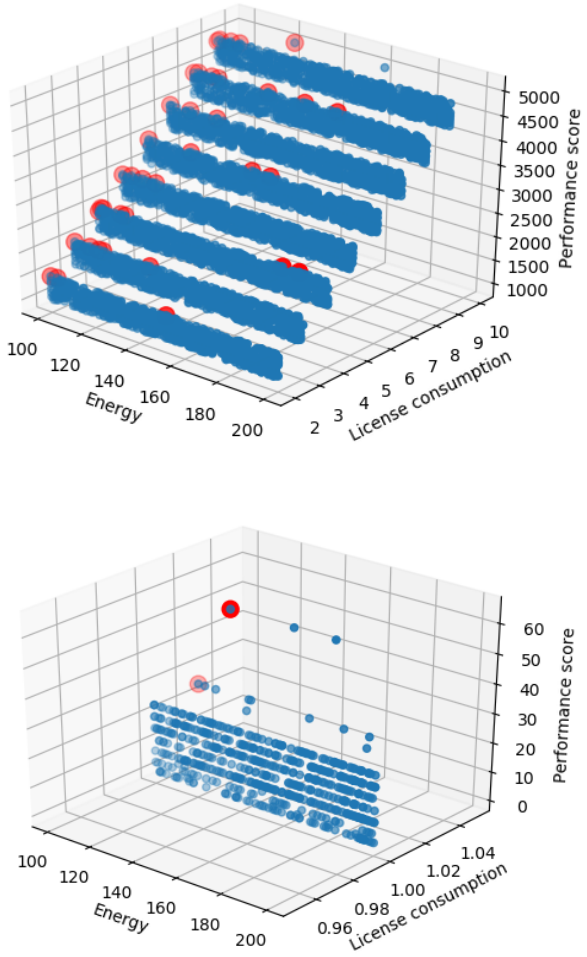

Fig. 5. Top: Second deployment of the CEPH product.
Bottom: Second deployment of compute product. The license dimension has no impact here as the metric demands one license per installation.

5 OpenStack Use Case

In this case we will deploy a minimal OpenStack platform with the metrics of the RedHat editor. The purpose of this deployment is to have a lead node (called director), ten compute nodes and twenty CEPH nodes. Each node has its metrics and performance class defined as follows:

Director It must be located in a cluster with as many servers as possible to be able to deploy as many compute and CEPH nodes as desired. The number of machines in the parent cluster will, therefore, be the performance index.

Compute It needs a lot of cores but must be deployed on single servers. A machine that has too many VMs cannot be reused. The performance index is the number of cores.

CEPH It requires a lot of memory. The performance index is the amount of memory available. We cannot reuse a server already used for another application.

For the volume of licenses, you need one license for the lead node, one license per compute node and one license per 500 memory slots for CEPH nodes. We can see in Fig. 5 the Pareto front of the CEPH and compute products.

Finally, in Fig. 6, we can see the comparison between *GreenSAM* and three other algorithms. *GreenSAM* still saves energy while keeping the number of licenses required low at the price of performance, which remains acceptable unlike the LicPerf or RatLic algorithm. It demonstrates that going only for the performance in the Cloud can lead to spending more money on software or disastrous

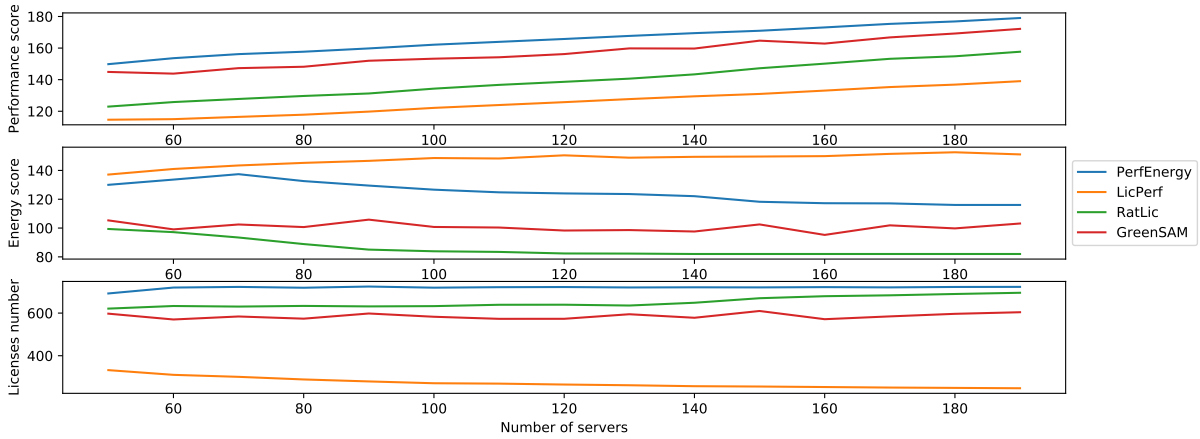

Fig. 6. This shows that *GreenSAM* still manages to get good results on a multi-products deployment. RatLic is saving more energy but fails at reducing license consumption and saving performance. LicPerf is very good at license criteria but still terrible at saving energy and performance.

scenarios like Diageo [1] and shows the need to have algorithms like *GreenSAM* that take into account software licensing.

6 Conclusion

This paper introduces *GreenSAM*, a multi-objective heuristic for deploying services in the Cloud that guarantees license compliance while reducing energy consumption but maintaining reasonable performance. This heuristic gets very good results on two use cases with two significant datasets (one simulated and one real) and compared to three others heuristics. In both experiments, *GreenSAM* achieved reducing energy and license consumption while maintaining acceptable performance and keeping full compliance by filtering bad servers. Compared to the prior State-Of-The-Art, we are now able to deploy automatically and while respecting compliance different products on a Cloud architecture all the while optimizing energy and performance.

This heuristic allows in the case of a 5G tool deployment to ensure good power consumption and software compliance and therefore could be used in network orchestrators such as ONAP [25] for 5G to manage the SAM part of the deployment while allowing energy savings in a technology that will be massively used. This use would remain very dependent on the execution time of the deployment calculation because they are done at high speed in the context of the virtualized network. To conclude, this heuristic offers a solution to manage an IT environment and the metrics associated with the products, even with complex licensing rules.

For the enhancement of *GreenSAM*, new approaches can be considered. Indeed, the implementation of a better heuristic function for sorting could allow choosing a better server from the subset of servers on the Pareto front. This cost function would be composed of the attributes that we want to optimize and would give a better score for each server to obtain a unique one. Another way would be to specify the objectives to have a finer deployment in our Cloud and add parameters to allow us to have a performance index closer to reality. It would be equally interesting to compare performance drop on each performance category to see the impact of software classes and improving the algorithm.

In addition, we could add more parameters to this heuristic to have a more realistic deployment. With the concept of agnostic parameters, we could modify the score function of *GreenSAM* to add networking or security considerations. Pushing further, it would be interesting to see if it is possible and profitable to make parameters fully generic. Allowing the user to add any parameter only giving attributes scores. Finally, we could study the problem of multiple deployment at once with the *GreenSAM* heuristic to avoid making only local optimisations.

Bibliography

[1] England, (Technology WHC, Court) C (2017) SAP UK Ltd v. Diageo Great Britain Ltd [2017] ewhc 189 (tcc). URL http://www.bailii.org/ew/cases/EWHC/TCC/2017/189.html

[2] Sayer P (2018) SAP settles licensing dispute with AB InBev. URL https://www.itworld.com/article/3264435/sap-settles-licensing-dispute-with-ab-inbev.html

[3] Flexera (2014) Flexera software and IDC research survey report: Software license audits and costs & risks to enterprises. URL http://learn.flexerasoftware.com/slo-wp-key-trends-audits-cost-risk

[4] Kim D, et al (2014) A birthmark-based method for intellectual software asset management. In: Lee S, et al (eds) The 8th International Conference on Ubiquitous Information Management and Communication, ICUIMC '14, Siem Reap, Cambodia - January 09 - 11, 2014, ACM, pp 39:1–39:6, DOI 10.1145/2557977.2558062, URL http://dl.acm.org/citation.cfm?id=2557977

[5] Han Y, et al (2014) A new detection scheme of software copyright infringement using software birthmark on windows systems. Comput Sci Inf Syst 11(3):1055–1069, DOI 10.2298/CSIS130918064H

[6] Holsing NF, et al (1999) Software Asset Management: Analysis, development and implementation. Information Resources Management Journal 12(3):14, DOI 10.4018/irmj.1999070102

[7] Ben-Menachem M, et al (2004) Inventorying information technology systems: supporting the "paradigm of change". IEEE Software 21(5):34–43, DOI 10.1109/MS.2004.1331300

[8] McCarthy MA, et al (2011) Managing software assets in a global enterprise. In: 2011 IEEE International Conference on Services Computing, pp 560–567, DOI 10.1109/SCC.2011.119

[9] Gocek P, et al (2017) Obtaining software asset insight by analyzing collected metrics using analytic services. URL https://www.google.com/patents/US9652812, uS Patent 9,652,812

[10] Vion A, et al (2017) Software License Optimization and Cloud Computing. CLOUD COMPUTING 2017 p 125

[11] Baillon N, et al (2018) Towards economic and compliant deployment of licenses in a Cloud architecture. In: Workshop: Cloud Management and Operations, In conjunction with IEEE International Conference on Cloud Computing (IEEE CLOUD 2018), San Francisco, USA, URL https://hal.inria.fr/hal-01808751, hal-01808751

[12] Mann ZA (2018) Resource optimization across the cloud stack. IEEE Transactions on Parallel and Distributed Systems 29(1):169–182, DOI 10.1109/TPDS.2017.2744627

[13] Vion A (2018) Software Asset Management and Cloud Computing. Phd, Université Grenoble Alpes, URL https://tel.archives-ouvertes.fr/tel-01901991

[14] Wikipedia (2019) ISO 19770 wikipedia. URL {https://www.iso.org/standard/68531.html}

[15] Foster I, et al (2001) Computational grids. In: Palma JMLM, Dongarra J, Hernández V (eds) Vector and Parallel Processing — VECPAR 2000, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 3–37

[16] Dongarra J, et al (2011) The international exascale software project roadmap. The International Journal of High Performance Computing Applications 25(1):3–60, DOI 10.1177/1094342010391989

[17] Berl A, et al (2010) Energy-efficient cloud computing. The Computer Journal 53(7):1045–1051, DOI 10.1093/comjnl/bxp080

[18] Balouek-Thomert D, et al (2015) Energy-aware server provisioning by introducing middleware-level dynamic green scheduling. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, pp 855–862, DOI 10.1109/IPDPSW.2015.121, URL https://hal.inria.fr/hal-01196908

[19] Orgerie AC, et al (2009) When Clouds become Green: the Green Open Cloud Architecture. In: PARCO, URL https://hal.inria.fr/ensl-00484321v1

[20] Zhang K, et al (2016) Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access 4:5896–5907, DOI 10.1109/ACCESS.2016.2597169

[21] Zappone A, et al (2015) Energy-efficient power control: A look at 5G wireless technologies. CoRR abs/1503.04609, URL http://arxiv.org/abs/1503.04609

[22] Sengupta S, et al (2013) Multi-objective node deployment in wsns: In search of an optimal trade-off among coverage, lifetime, energy consumption, and connectivity. Engineering Applications of Artificial Intelligence 26(1):405 – 416, DOI 10.1016/j.engappai.2012.05.018, URL http://www.sciencedirect.com/science/article/pii/S0952197612001248

[23] Khalesian M, et al (2016) Wireless sensors deployment optimization using a constrained pareto-based multi-objective evolutionary approach. Engineering Applications of Artificial Intelligence 53:126 – 139, DOI 10.1016/j.engappai.2016.03.004, URL http://www.sciencedirect.com/science/article/pii/S095219761630063X

[24] Xu B, et al (2015) Dynamic deployment of virtual machines in cloud computing using multi-objective optimization. Soft Computing 19(8):2265–2273, DOI 10.1007/s00500-014-1406-6

[25] Linux Foundation (2019) ONAP: Open Network Automation Platform. URL {https://onap.org}