**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 11, No. 1, February 2021, pp. 686~698
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp686-698 

# Optimization of energy consumption in cloud computing datacenters

**Ahmed Osman, Assim Sagahyroon, Raafat Aburukba, Fadi Aloul**
Department of Computer Science and Engineering, American University of Sharjah, Uni Emirat Arab

**Article Info**

***Article history:***

Received Jan 13, 2020
Revised Jul 29, 2020
Accepted Aug 8, 2020

***Keywords:***

Boolean satisfiability (SAT)
Cloud computing
Energy optimization
Integer linear programming

**ABSTRACT**

Cloud computing has emerged as a practical paradigm for providing IT resources, infrastructure and services. This has led to the establishment of datacenters that have substantial energy demands for their operation. This work investigates the optimization of energy consumption in cloud datacenter using energy efficient allocation of tasks to resources. The work seeks to develop formal optimization models that minimize the energy consumption of computational resources and evaluates the use of existing optimization solvers in testing these models. Integer linear programming (ILP) techniques are used to model the scheduling problem. The objective is to minimize the total power consumed by the active and idle cores of the servers' CPUs while meeting a set of constraints. Next, we use these models to carry out a detailed performance comparison between a selected set of Generic ILP and 0-1 Boolean satisfiability based solvers in solving the ILP formulations. Simulation results indicate that in some cases the developed models have saved up to 38% in energy consumption when compared to common techniques such as round robin. Furthermore, results also showed that generic ILP solvers had superior performance when compared to SAT-based ILP solvers especially as the number of tasks and resources grow in size.



***Corresponding Author:***

Assim Sagahyroon,
Department of Computer Science and Engineering,
American University of Sharjah, University City,
Sharjah, Uni Emirat Arab.
Email: asagahyroon@aus.edu

## 1. INTRODUCTION

In recent years, cloud computing has emerged as a practical paradigm for hosting and delivering services over the internet. This in turn has led to the creation of huge data centers which are power hungry entities with complex requirements and operational needs [1]. A critical design parameter for data centers is their power consumption characteristics and energy demands.

Cloud computing services are provided through datacenters. Datacenters contain large scale, networked compute infrastructure. The major physical resources in cloud computing can be classified into three categories: compute-related resources, storage-related, and network resources. For the cloud providers to be able to meet the consumers' requests with high quality of service (QoS), they need to have large numbers of these resources with high capabilities. Hence, datacenters hosting these resources end up consuming significant amounts of energy. In 2010 for example, the electricity usage of datacenters around the world represented (1.1%-1.5%) of the total worldwide electricity usage while in the United States, the datacenters usage represented up to 2.2% of the total energy consumed in the US [2]. This in turn stresses the need for energy efficient datacenters to minimize the energy costs and sustain the environment.

Some of the techniques that can contribute to energy efficiency in cloud computing include:

- Use of smart tasks allocation strategies including static as well as dynamic approaches to reduce energy consumption [3, 4].
- Use of virtualization techniques to enable the creation of multiple virtual resources on a single physical resource. This allows the reduction of the number of physical resources needed to perform a task and hence can contribute in reducing the energy consumption [5].
- Workload Consolidation to minimize the number of resources used and maximizes the utilization of the resources [6].
- Other techniques include dynamic VMs placement and VM migration, heat management and temperature-aware allocation, and load balancing and task scheduling [6].

Scheduling is the process of making decisions to allocate resources to tasks over time with the goal of optimizing one or more objectives [7]. In cloud computing, scheduling addresses the problem of allocating network resources, compute resources, or storage needs to tasks over a period of time, and it encompasses the following [8]:

- Resources: physical/virtual devices with the ability to execute or process tasks;
- Tasks: instructions to be executed by the resources.
- Constraints: conditions must be considered when scheduling the tasks to the resources. They may be task-based, resource-based, or a combination of these. They could also be hard constraints, meaning constraints that must be full-filled or soft constraints that can be relaxed.
- Objectives: the evaluation criteria that needs to be measured in order to assess the system performance.

Some of the related scheduling objectives can be minimizing the completion time, maximizing the profit, minimizing the energy consumption, maximize the resource utilization, or it can have multiple objectives. Our research aims to formulate the scheduling problem (with the objective being the minimization of power consumption) as a 0-1 ILP model. The generated ILP model will be transformed into a Boolean satisfiability problem (SAT) that can be solved using SAT solvers with the results being mapped back to the original scheduling problem. We will validate the ILP models and examine the capabilities of current state of the art generic ILP and SAT solvers when used to solve formulations of the scheduling problem. Preliminary results related to this work were reported in [9].

Optimizing the power consumption in a cloud computing environment can be achieved by considering it as a scheduling problem where the tasks submitted to the datacenter are scheduled to the different resources in the datacenter in such a manner to optimize the power consumed by computing resources. However, the scheduling problem can have multiple optimizations objectives such as increasing the utilization of resources, reducing the time taken to perform the tasks, maximizing the profit, or minimizing the energy consumption.

In recent years, the scheduling aspects of a datacenter have been tackled with different optimization objectives in mind. For example, in [10], the tasks or processes are assigned in a round robin fashion between the available processors. The tasks are distributed equally between all the processors; problem is, some processors will end up being heavily loaded compared to others. The authors proposed overcoming this limitation by using a weighted round robin approach. The Min-Min static load balancing algorithm is used in [11]. Here, parameters related to the job or task are assumed to be known in advance. The algorithm starts by identifying the minimum completion time needed by a task and assigning tasks by the cloud manager on ascending completion time needs. A drawback is the fact that tasks requiring long execution times have to wait and this may lead to starvation. Liu et. al [12] proposed an optimization model for minimizing the energy consumption of cloud computing datacenters using task scheduling. The authors formulated an integer programming model of the task scheduling problem with the goal of minimizing the number of active servers and thus minimizing the overall energy consumption of the datacenter without breaking the response time constraints. The formulation process considered two cases: an initial case where no task is received before and the new tasks are assigned immediately and a second case where the datacenter has backlogged tasks that result in queuing delays. In addition, the authors proposed a scheduling scheme named most-efficient-server first that assigns the most efficient server first. The model and the environment are then simulated with heterogeneous tasks. Their results showed that the proposed scheme minimizes the server energy consumption 70 times better than a random-based scheduling scheme on average. In [13] a scheduling algorithm for online scheduling of servers of cloud computing providers was proposed. The algorithm is called predict optimize dispatch (POD) based on the three main steps that the algorithm follows. It was assumed that the servers used have multiple processors with different speeds and different power consumption, and each server can be in busy, idle, or switched off mode. The objective of the scheduling algorithm is to minimize the energy consumption of a cloud computing environment. The algorithm is implemented in three steps: first prediction of the workload assigned to the datacenter. The workload is acquired using estimates based on the cloud provider's information or using a new kind of multi-arm bandit

workload prediction. Then, the algorithm modifies and optimizes the set of available servers to best fit the predicted workload. The final step is to schedule the jobs to the available processors using a modified version of Round Robin scheduling method. The authors then simulated their proposed solution and tested it against other online and offline scheduling algorithms and presented their results with a recommendation on which type of scheduling to use for different types of workload. Another technique that is used with scheduling to minimize the energy consumption in cloud computing environment is dynamic voltage and frequency scaling (DVFS). In [14] a DVFS based green energy-efficient scheduling algorithm was proposed. It works on increasing the utilization factor of the different servers in the datacenter in an effort to reduce its energy consumption. The environment consisted of heterogeneous servers with each server having different processor, memory, and storage specifications. Server's has frequency is varied or reduced to minimize the energy meeting the requirements of the workload. Next, a priority-based scheduling algorithm is used to assign jobs to created VMs. Experimental results indicated that the proposed algorithm reduce the energy consumption by a range that varies from 5 to 25% while maintaining the required performance and execution time.

Various combination of DVFS and node vary-on vary-off (VOVO) methods were evaluated in [15]. The objective of the research was to minimize the power consumption in clusters of servers during times where the workload is reduced. The authors explored five different policies with different degrees of complexity at the implementation level. The five policies are:

- Independent voltage scaling (IVS): uses processors that have voltage scaling property and can independently change their voltage and frequency according to their workload.
- Coordinated voltage scaling (CVS): exploits voltage scaled processors but coordinate their voltage scaling algorithms.
- Vary-on vary-off (VOVO): turning the whole server on or off depending on the size of the workload.
- Combination of IVS and VOVO.
- Combination of CVS and VOVO.

These different policies were studied and compared with a combination of VOVO and CVS producing the most savings in energy consumption.

In [16] authors defined an energy consumption ratio (ECR) to evaluate the efficiency of different frequencies under which to execute a task. They converted the energy-efficient task scheduling problem into minimizing the total ECR. Next they transformed the problem to the variable size bin packing, and showed that the minimization of ECR is NP-hard and proposed propose a task allocation and scheduling methods based on the feature of this problem. They also proposed a processor-level migration algorithm to reschedule remaining tasks among processors on an individual server and dynamically balance the workloads and lower the total ECR on a given server. In [17], researchers propose a consolidation algorithm which favors the most effective migration among Virtual Machines, containers and applications; and investigate how migration decisions should be made to save energy without any negative impact on the service performance.

Linear programming is a mathematical method for maximizing or minimizing a linear objective function under certain conditions or constraints consisting of different variables. Integer linear programming (ILP) is a special type of linear programming where some or all the variables in the objective function and constraints are limited to integer values. ILP problems can be categorized into two categories: 0-1 ILP problems where the variables in the problem are binary variables and have values of 0 or 1, and Generic ILP where the variables are not limited to binary variables [18].

In this work, the generated ILP model will be transformed into a boolean satisfiability problem (SAT) equations. These equations will be solved using SAT solvers and the result will be mapped back to the original scheduling problem. Boolean satisfiability problem refers to finding a satisfying assignment for a problem or proving that none exist. A satisfying assignment for a certain Boolean formula is an assignment of the binary variables in the formula to 0 or 1 to make the overall formula evaluate to 1. The assignment must satisfy all the constraints of the problem simultaneously [19]. Both the SAT problem and the 0-1 ILP problem are represented using binary variables but in SAT, the Boolean formulas used to represent the problem and its constraints must all be in a product-of-sums form or a conjunctive normal form (CNF) and the operators used in these forms are logical operators (AND, OR, and NOT) while in 0-1 ILP problem the constraints are simple inequalities of the form $Ax \leq b$ where $b \in \mathbb{Z}^n$, $A \in \mathbb{Z}^m \times \mathbb{Z}^n$, and $x \in \{0,1\}^n$. These constraints are referred to as Pseudo Boolean (PB) constraints and they are considered as a general form of the CNF constraints [20].

The SAT problem is considered to be NP complete problem. The solving time increases exponentially with the increase in the number of variables. The SAT solvers are algorithms that determine if a SAT problem is satisfiable or not and find a satisfying assignment in case the problem was satisfiable [19-21]. In the past few years the use of SAT models to represent different problems in various domains increased

rapidly due to the huge improvement in the SAT solving algorithms and the availability of better and more powerful computing capabilities. One of the great improvements to SAT-based solvers is the extension to handle PB constraints because they are easier to write and understand and can replace an exponential number of CNF constraints. Before this extension SAT models were limited to decision problems but with the help of PB constraints SAT can now model optimization problems and solve them using the different SAT-based PB solvers [20].

Pseudo Boolean SAT Solvers: which are basically SAT solvers that can handle pseudo Boolean constraints allowing the SAT solvers to be used in solving optimization problems. In [22], for example, two power estimation problems were formulated as SAT problems and evaluated using generic ILP and SAT-based solvers. It was shown that the SAT based solver performed better than random based approaches in terms of finding the optimal values of the proposed problem.

In this work, PB-SAT solvers are used to solve the proposed optimization model. The testing process involves using different solvers and comparing their results. The solvers are selected according to the following criteria:

- A proven performance of the solver from the reported literature.
- The ability of the solver to handle large problem size.

Based on the above criteria, we identified three PB-SAT solvers, namely:

- NaPS [23]: Nagoya Pseudo Boolean Solver, is a solver for PB constraints that are linear and contain binary variables only. It was the best performing solver in the Pseudo Boolean competition 2016 in both categories of optimization with small integers and large integers and was among the best solvers in the other categories. The information and results of the competition is available in [24].
- Minisat+ [25]: is a modified version of the Minisat solver to be able to deal with pseudo Boolean constraints, and despite being a simple solver it performed well in the pseudo Boolean competition through the years.
- Sat4j [26]: is a Boolean reasoning java library that is used to solve Boolean satisfaction and optimization problems. The sat4j solver is different from the other two solvers because it does not prioritize being the fastest in solving a problem. However, the solver is full featured, robust, and user friendly. The solver design follows the Java design guidelines and code conventions.

Furthermore, Commercialized and open source optimization tools with various embedded algorithms exist today. Those tools deal with generating a solution to a defined linear, non-linear, or integer models. Some of the popular tools include:

- CPLEX [27]: is a high-performance solver for linear programing, mixed integer programing, and quadratic programing. It is a well-known and wildly used commercial solver developed over 20 years ago and is currently owned by IBM. The solver uses the Simplex exact algorithm, and the name CPLEX is actually taken from C-Simplex which is Simplex algorithm implemented in C language. The solver is improved and updated periodically to increase its performance and include different algorithms for different types of problems. some of the various methods used in CPLEX are: primal simplex algorithm, dual simplex algorithm, network simplex algorithm, and the barrier method. This solver will be used in testing the proposed model.
- LINGO [28]: is an efficient comprehensive tool used in building and solving different types of optimization models. It was developed by LINDO Systems as an optimization modeling software for linear, nonlinear, and integer programming. LINGO provides its users with a language to express the optimization models, an environment to build and edit their problem, and a set of solvers. These solvers include but not limited to: Primal and dual simplex solvers, Barrier solver, Integer solver, General nonlinear solver, and Quadratic solver.

The rest of the paper is organized as follows: in Section 2 we discuss the research methodology followed in this work; Section 3 includes the results obtained and a thorough discussion of it, the paper is concluded in Section 4.

## 2. RESEARCH METHOD

### 2.1. Environment setting and model formulation

This section presents the characteristics of the cloud computing environment, analyses the scheduling problem in the cloud datacenter, and formulates the scheduling problem as an integer linear program (ILP) model.

A cloud computing environment may consist of a single or multiple datacentres that may belong to a single or multiple providers. This work deals with a single cloud datacentre that belongs to a single provider. The cloud datacentre infrastructure contains three resource types: compute, storage, and network resources. Our aim in the work presented here is to focus on the computing component of the resources used

since it is shown to consume about 26% of the total energy of a datacentre [17]. The main computing units are the servers. The proposed model is formulated based on the following assumptions:

- Each CPU can run two or four tasks at the same time depending on number of cores it contains (dual or quad core); each core will run only one task at 100% utilization.
- A server is turned ON if at least a single core in one of its CPUs is executing a task and is switched OFF otherwise.
- A core is either ON (switched on and executing a task), idle (switched on but not running any task), or completely OFF (switched off).
- Each core is assumed to have two running modes: High and Low.
- In High mode, the core runs with the maximum available frequency and voltage, while in Low mode, the core runs at the minimum available frequency and voltage.
- Cores inside the same server can run in different modes.
- Tasks are allocated in batches that arrive at distinct times ($t_0$, $t_1$, …, $t_n$).
- All the tasks in a specific batch will be scheduled immediately and will start running simultaneously (no task will be assigned to start later).
- A task can only be executed by a core that meets its frequency and memory requirements.
- For any task there is at least one core that meets the requirements of the task (no task will be rejected).
- The execution of a task on a certain core is assumed to consume a specific amount of energy that will be called the energy cost in this model.
- Ne pre-emption of tasks is allowed.

### 2.2. Optimization model formulation

This section discusses the integer linear programming (ILP) model that minimizes the energy consumption of the servers in a cloud datacentre. The intended objective is to find the specific assignment of tasks to cores that gives the minimum energy value. The total power consumption is the power consumed by the cores when a task is allocated to it plus the power consumed by the idle cores. Table 1 presents the variables used in developing the model and their meaning. Figure 1 illustrates a simple datacentre configuration with two servers $S_1$ and $S_2$. $S_1$ has two dual-core CPUs and $S_2$ has one quad-core CPU as shown. A server has two states either ON or OFF. Depending on the server state, each core is assumed to be either active, idle, or switched off, as represented by the green, blue, and red coloured-squares respectively. In the scenario depicted by Figure 1, all cores were initially off, then two tasks are assigned to $C_{1,1,1}$ and $C_{1,2,1}$ with a cost of $X_{1,1,1,1}$ and $X_{1,2,1,2}$ respectively.

Table 1. Model notations

| Variables | Description |
|---|---|
| $S_i$ | Binary variable equals to 0 if server *i* is off, and 1 if it is on. |
| $P_{i,j}$ | Binary variable equals to 0 if CPU *j* in server *i* is not active and 1 otherwise. |
| $G_{i,j}$ | Binary variable, equal to 1 if CPU *j* in server *i* is idle and 0 otherwise. |
| $C_{i,j,k}$ | Binary variable, equal to 1 if core *k* in CPU *j* in server *i* is active, and 0 otherwise. |
| $Q_{i,j,k}$ | Binary variable, equal to 1 if core *k* in CPU *j* in server *i* is idle, and 0 otherwise. |
| $T_{i,j,k,n}$ | Binary variable, equal to 1 if task *n* is assigned to core *k* in CPU *j* in server *i*, and 0 otherwise. |
| $L_{i,j,k,n}$ | Binary variable, equal to 1 if task *n* is executed in core *k* in CPU *j* in server *i* in Low mode and 0 otherwise. |
| $H_{i,j,k,n}$ | Binary variable, equal to 1 if task *n* is executed in core *k* in CPU *j* in server *i* in High mode and 0 otherwise. |
| $X_{i,j,k,n}$ | The energy cost value of assigning task *n* to core *k* in CPU *j* in server *i* in Low mode. |
| $Y_{i,j,k,n}$ | The energy cost value of assigning task *n* to core *k* in CPU *j* in server *i* in High mode. |
| $Z_{i,j,k}$ | The energy cost value of core *k* in CPU *j* in server *i* when it is idle. |
| $RR_n$ | The minimum amount of RAM requested by task *n*. |
| $RF_n$ | The minimum amount of Frequency requested by task *n*. |
| $AR_{i,j,k}$ | The available amount of RAM in core *k* in CPU *j* in server *i*. |
| $AL_{i,j,k}$ | The available Low frequency in core *k* in CPU *j* in server *i*. |
| $AH_{i,j,k}$ | The available High frequency in core *k* in CPU *j* in server *i*. |

We describe the formulation steps taken to develop the comprehensive model. Initially, the formulation commences with an objective function whose primary goal is to minimize the energy consumption when assigning tasks to cores in a datacenter. From Figure 1 we observe that the energy consumption comes from the energy cost of active cores. This energy cost consists of the cost of cores running in Low mode ($X_{i,j,k,n}$) and the cost of cores running in high mode ($Y_{i,j,k,n}$). In addition to active cores there is

also the energy cost of idle cores ($Z_{i,j,k,n}$). Furthermore, the off cores are assumed to have zero energy needs. This is represented as follows in (1):

$$Min\left(\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N}X_{i,j,k,n}L_{i,j,k,n}+\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{n=1}^{N}Y_{i,j,k,n}H_{i,j,k,n}+\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K}Z_{i,j,k}Q_{i,j,k}\right) \quad (1)$$

The objective function described in (1) has three parts, the first two parts represent the energy consumed by the active cores. The active cores energy is calculated by summing the energy cost of all the cores that are running a task across the whole datacenter. The third component of the objective function sums the energy consumed by the remaining idle cores in the datacenter. This objective function is subject to the following set of constraints:

– If a core *k* is active, then the CPU *j* that contains the core *k* is considered active as well.

$$P_{i,j}=1 \qquad if \sum_{k=1}^{K} C_{i,j,k} \geq 1 \qquad \forall i \forall j$$

– If a CPU j is active, then at least one of its cores is active.

$$\sum_{k=1}^{K} C_{i,j,k} \geq 1 \qquad if\ P_{i,j}=1 \qquad \forall i,\ j \forall$$

– If a CPU is active, all the cores inside the CPU must be either active or idle but not switched off.

$$\prod_{k=1}^{K}(C_{i,j,k}+Q_{i,j,k})=1 \qquad if\ P_{i,j}=1 \qquad \forall i, \forall j$$

– A core cannot be active and idle at the same time.

$$C_{i,j,k}+Q_{i,j,k} \leq 1 \qquad \forall i, \forall j, \forall k$$

– If a CPU is idle, then all its cores must be idle.

$$\sum_{k=1}^{K} Q_{i,j,k}=K \qquad if\ G_{i,j}=1 \qquad \forall i, \forall j$$

– If a server is off all the cores inside the server are off.

$$\sum_{j=1}^{J}\sum_{k=1}^{K} C_{i,j,k}+Q_{i,j,k}=0 \qquad if\ S_i=0 \qquad \forall i$$

– If one or more of the CPUs in server i are active, then server i must be active.

$$S_i=1 \qquad if\ \sum_{j=1}^{J} P_{i,j} \geq 1 \qquad \forall i$$

– If a server is active, then at least one of its CPUs is active.

$$\sum_{j=1}^{J} P_{i,j} \geq 1 \qquad if\ S_i=1 \qquad \forall i$$

– If a server is active, then all the CPUs in an active server are either active or idle but not switched off.

$$\prod_{j=1}^{J}(P_{i,j}+G_{i,j})=1 \qquad if\ S_i=1 \qquad \forall i$$

– If a server i is off, all the CPUs inside the server are off.

$$\sum_{j=1}^{J} P_{i,j}+G_{i,j}=0 \qquad if\ S_i=0 \qquad \forall i$$

– A CPU cannot be active and idle at the same time.

$$P_{i,j}+G_{i,j} \leq 1 \qquad \forall i,\ \forall j$$

– A task n must be assigned to one core only at any given time.

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} T_{i,j,k,n} = 1 \qquad \forall n$$

– A core can execute a single task only at any given time or not execute any task.

$$\sum_{n=1}^{N} T_{i,j,k,n} \triangleq \begin{cases} 1 & if\ C_{i,j,k} = 1 \\ 0 & if\ C_{i,j,k} = 0 \end{cases} \qquad \forall i, \forall j, \forall k$$

– The number of active cores is equal to the total number of tasks to be scheduled.

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} C_{i,j,k} = N$$

– A core cannot be running in High and Low mode at the same time.

$$L_{i,j,k,n} + H_{i,j,k,n} \leq 1 \qquad \forall i, \forall j, \forall k, \forall n$$

– If a task is executed on a specific core, then it must be executed at either High or Low mode.

$$L_{i,j,k,n} + H_{i,j,k,n} = 1 \quad if\ T_{i,j,k,n} = 1 \qquad \forall i, \forall j, \forall k, \forall n$$

– if a task is not executed on any core then it cannot be executed at neither High nor Low mode.

$$L_{i,j,k,n} + H_{i,j,k,n} = 0 \quad if\ T_{i,j,k,n} = 0 \qquad \forall i, \forall j, \forall k, \forall n$$

– The required frequency by task n does not exceed the available high or low frequency of the core.

$$RF_n H_{i,j,k,n} \leq AH_{i,j,k} \qquad \forall i, \forall j, \forall k, \forall n$$
$$RF_n L_{i,j,k,n} \leq AL_{i,j,k} \qquad \forall i, \forall j, \forall k, \forall n$$

– The amount of RAM required by task n is less than or equal to the amount of RAM available in core *k*.

$$RR_n T_{i,j,k,n} \leq AR_{i,j,k} \qquad \forall i, \forall j, \forall k, \forall n$$

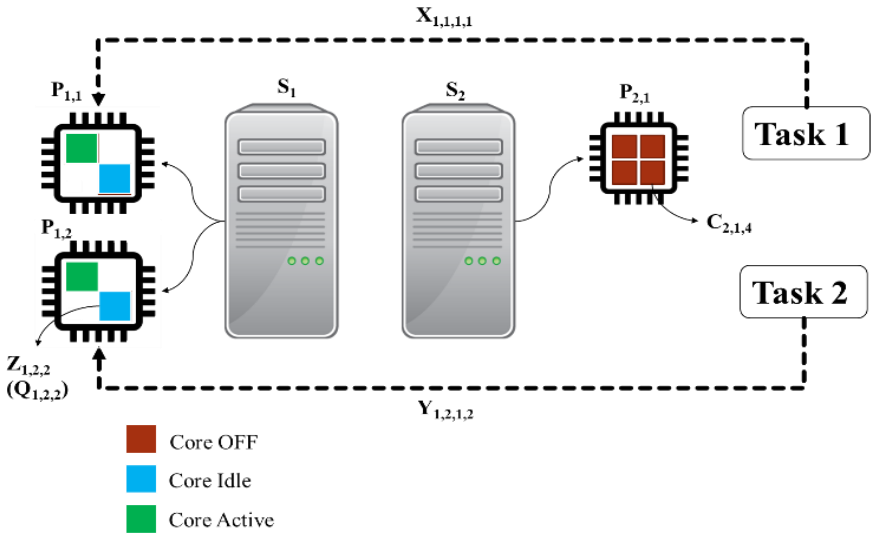


Figure 1. A simple datacenter configuration

### 2.3. Dynamic scheduling

Following the scheduling of the first batch at time $t_0$, in the second phase of this work, we assume the arrival of a new batch to be scheduled at time $t_1$. Here, the remaining free cores are used. Following the scheduling of the first batch and while preparing to schedule the second batch, the constraints below are added:

- All the cores that are still running a task from batch t0 are marked as busy,

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} C_{i,j,k} = B$$

where $B$ is the number of tasks from the first batch that are still running at t1.

- If a core $C_{i,j,k}$ is busy then no task from the second batch is assigned to it,

$$\sum_{n=1}^{M} T_{i,j,k,n} = 0 \qquad if\ C_{i,j,k} = 1$$

where $M$ is the number of the new tasks in the second batch at $t_1$.

- The total number of active cores is now equal to the total number of tasks to be scheduled plus the number of tasks that are still running from the first batch.

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} C_{i,j,k} = M + B$$

## 3. RESULTS AND DISCUSSION

The generated optimization model of (1) is validated and tested using SAT-based 0-1 PB solvers named NaPS, Minisat+, and Sat4j, and the generic ILP solver CPLEX. The solvers compute the minimum energy consumption of the datacenter and the corresponding assignment of tasks to the cores that would achieve this power. To test the model, several datacenter instances were created with different number of servers, different number and types of CPUs (dual-core or quad-core) in each server, and different number of tasks to be scheduled. The tests are carried out as described in the following sections.

The comprehensive model in (1) is tested using the three PB-SAT solvers and CPLEX. The testing instances used consist of a number of servers with each server containing dual or quad core CPUs. The instances also include the number of tasks to be assigned to the cores of the datacenter. The tasks to be scheduled have several characteristics including: task's size (in number of instructions), task's required frequency, and task's required RAM. Each core in the datacenter runs in one of two modes: High, or Low. As mentioned earlier, in High mode the core runs at the maximum frequency and voltage allowed, while in low mode, the core runs at the minimum frequency and voltage possible. The testing instances are generated randomly from a pool of servers. The tasks requirements are created depending on the tasks' size. Table 2 shows the task size ranges and the corresponding tasks' requirements. Table 3 includes partial servers' specifications. Table 4 lists the instances created for this test. Each setup is tested twice using different number of tasks. The instances are first solved using all the solvers to find the minimum energy consumption and compare the solvers' performance. Then, the same instances are solved using round robin and maximum possible value methods to compare their performance against the proposed model.

Table 2. Tasks' size ranges and requirements used in Test *A*

| Task Size Range (#instructions) | Required Frequency (MHz) | Required RAM (GB) |
|---|---|---|
| 500 – 2000 | 800 | 1 |
| 2500 – 4000 | 1000 | 2 |
| 4500 – 6000 | 2400 | 2 |
| 6500 – 8000 | 2600 | 3 |
| 8500 – 10000 | 2800 | 4 |

Table 3. Test *A* servers' information

| Server No. | Server Type | High Freq. (MHz) | High Volt. (Volt) | Low Freq. (MHz) | Low Volt. (Volt) | RAM (GB) |
|---|---|---|---|---|---|---|
| Server1 | Dual | 2400 | 1.45 | 800 | 0.9 | 2 |
| Server2 | Dual | 3000 | 1.35 | 1000 | 1.0 | 4 |
| Server3 | Quad | 2400 | 1.35 | 1000 | 0.9 | 2 |
| Server4 | Quad | 2600 | 1.45 | 800 | 1.0 | 4 |

Table 4. The testing instances for Test *A*

| Instance No. | No. of servers | Total no. of CPUs | Total no. of cores | No. of Tasks | Total Tasks Size (#instructions) |
|---|---|---|---|---|---|
| 1 | 2 | 4 | 12 | 2 | 7,500 |
| 2 | 2 | 4 | 12 | 6 | 25,500 |
| 3 | 3 | 6 | 20 | 5 | 23,000 |
| 4 | 3 | 6 | 20 | 10 | 41,000 |
| 5 | 4 | 8 | 24 | 8 | 29,500 |
| 6 | 4 | 8 | 24 | 12 | 47,000 |
| 7 | 8 | 16 | 52 | 18 | 68,500 |
| 8 | 8 | 16 | 52 | 26 | 98,500 |

In the first part of the test, the PB-SAT solvers and CPLEX solved the testing instances to find the minimum energy consumption and the results are presented in Table 5. All solvers managed to find the minimum energy consumption in the first six instances. The CPU times for the PB-SAT solvers are relatively close to each other, with NaPS taking the lead in most of the instances. However, if we include CPLEX in the comparison we can see that except in instance 1 it was always faster than the other solvers with an appreciable difference as can be seen in Figure 2.

Table 5. The minimum energy consumption and CPU time after adding High and Low execution modes

| Instance No. | Min Energy Consumption | CPU Time (seconds) | | | |
|---|---|---|---|---|---|
| | | NaPS | Minisat+ | Sat4j | CPLEX |
| 1 | 151 | 1.05 | 0.05 | 0.18 | 0.33 |
| 2 | 536 | 2.63 | 1.94 | 7.79 | 0.35 |
| 3 | 488 | 9.86 | 15.31 | 13.71 | 0.33 |
| 4 | 690 | 44.0 | 184.74 | 151.68 | 0.53 |
| 5 | 516 | 65.09 | 140.63 | 150.36 | 0.51 |
| 6 | 730 | 215.9 | 705.13 | 68.21 | 0.68 |
| 7 | 1101 | > 1000 (1108) | > 1000 (1261) | > 1000 (1143) | 3.02 |
| 8 | 1567 | > 1000 (2080) | > 1000 (1833) | > 1000 (1756) | 4.96 |

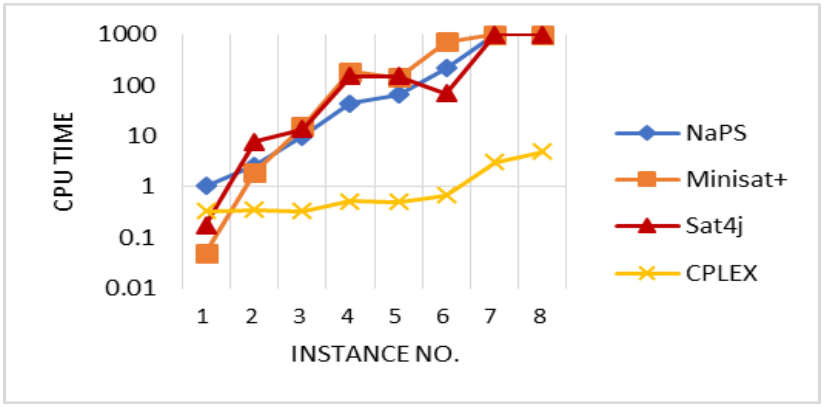


Figure 2. The performance of the PB-SAT solvers and CPLEX in Test *A*

In the last two instances (7 and 8) the PB-SAT solvers are stopped after exceeding the 1000 second mark. The values between the brackets in Table 5 are the best solution found by the solvers up to that point in time. Only CPLEX have successfully found the minimum energy consumption in those instances and within less than 5 seconds. From Figure 3 we notice that the minimum energy consumption is directly related to the total number of instructions of the tasks to be scheduled. The time taken by NaPS is proportional to the total number of instructions as well.

Table 6 depicts the results of the second part of the test where the proposed model is compared with maximum possible value (MPV) and round robin (RR). We observe that overall the proposed model has consistently returned the minimum energy consumption when compared with the other two methods. The average amount of energy saved is 27.3% of the maximum energy consumption with a maximum of

35.6% energy saved in instance 7. We also notice that round robin method failed to schedule two of the instances (4 and 6). This can be attributed to the fact that round robin method schedules the tasks to the first available core that can execute these tasks even if the task is small and can be executed on some other lower capability core. This leads to larger tasks not finding any core that can be assigned to them because all the capable cores are executing smaller tasks and that exactly what happened in both instances 4 and 6. The only way round robin can schedule the tasks of instances 4 and 6 is by waiting until one of the busy cores finishes its task. Hence, in all cases the proposed model proved its superiority. Figure 4 is a graphical representation of the results of Table 6.

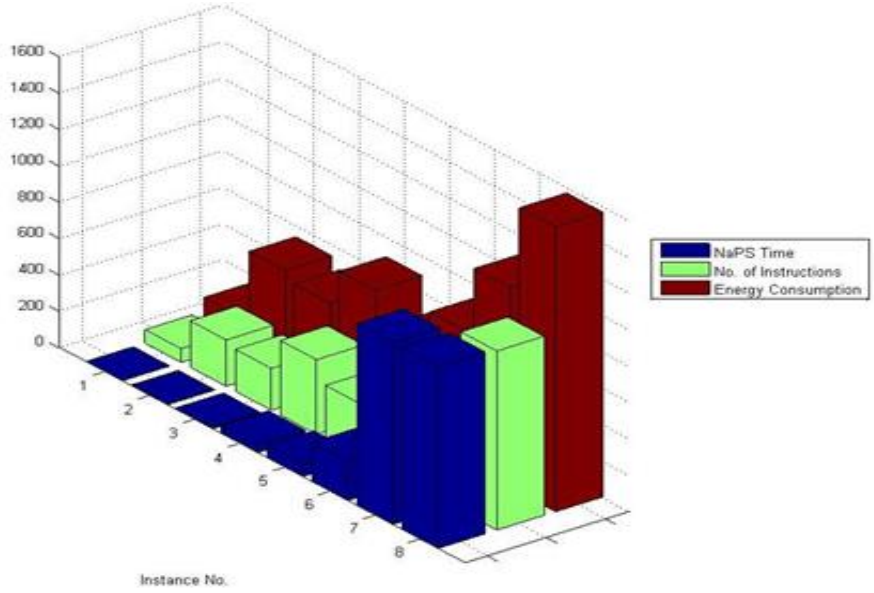


Figure 3. Illustration of high and low model results

Table 6. Comparison of the proposed model with round robin and maximum possible value methods

| Instance No. | Total Tasks Size | Energy Consumption | | |
|---|---|---|---|---|
| | | MPV | Round Robin | Proposed Model |
| 1 | 7,500 | 233 | 189 | 151 |
| 2 | 25,500 | 584 | 536 | 536 |
| 3 | 23,000 | 599 | 491 | 488 |
| 4 | 41,000 | 939 | UNSAT | 690 |
| 5 | 29,500 | 742 | 560 | 516 |
| 6 | 47,000 | 1081 | UNSAT | 730 |
| 7 | 68,500 | 1710 | 1383 | 1101 |
| 8 | 98,500 | 2277 | 1830 | 1567 |

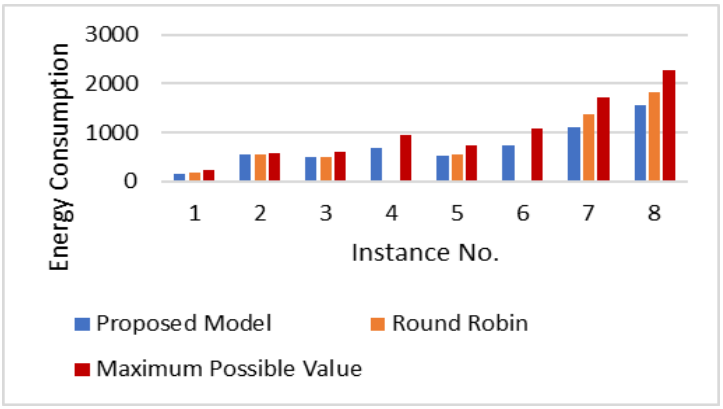


Figure 4. Energy consumption of the proposed model, round robin, and maximum possible value methods

In all of the analysis above we assumed that all the tasks arrive as one batch and scheduled to execute simultaneously at time $t_0$. In this segment of our tests, we consider the case of another batch of tasks arriving at time $t_1$. The new batch of tasks is to be assigned to the remaining free cores of the datacenter while minimizing the energy consumption. The tasks of the first batch ($t_0$) that are still running at $t_1$ will not be rescheduled. However, their energy consumption will be taken into account when calculating the overall energy consumption of the second batch. Table 7 shows the testing instances used with a second batch of tasks at time $t_1$. Number of remaining tasks at $t_1$ refers to how many tasks from batch $t_0$ are still running when the second batch of tasks arrived at $t_1$. Number of new tasks at $t_1$ is how many new tasks arrived at $t_1$. Both the proposed model and round robin are used to schedule the tasks at $t_0$ and $t_1$ and the minimum energy consumption in each case is compared.

Table 7. Testing Instances used in Test *B*

| Instance No. | No. of servers | Total no. of CPUs | Total no. of cores | No. of Tasks at $t_0$ | No. of remaining Tasks at $t_1$ | No. of new Tasks at $t_1$ |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 8 | 2 | 1 | 2 |
| 2 | 2 | 4 | 12 | 4 | 2 | 3 |
| 3 | 3 | 6 | 20 | 6 | 2 | 6 |
| 4 | 6 | 18 | 68 | 10 | 4 | 8 |
| 5 | 8 | 24 | 92 | 20 | 8 | 16 |

Table 8 shows the results obtained. The second column (Min Energy at $t_0$) is the energy consumption value of executing all the tasks in batch $t_0$. The minimum energy consumption at $t_1$ is the total of the energy consumed by the tasks that are still running from $t_0$ plus the energy consumed by the new tasks at $t_1$ and is listed under the fourth column.

Table 8. The results of scheduling a second batch of tasks at $t_1$

| Instance No. | Min Energy at $t_0$ | CPU time in seconds at $t_0$ | Min Energy at $t_1$ | CPU time in seconds at $t_1$ |
|---|---|---|---|---|
| 1 | 24 | 0.015 | 32 | 0.015 |
| 2 | 40 | 0.015 | 97 | 0.031 |
| 3 | 92 | 0.078 | 108 | 0.078 |
| 4 | 80 | 0.562 | 96 | 0.515 |
| 5 | 184 | 10.17 | 224 | 3.921 |

From the results we observe that the modification made to the original model has successfully allowed scheduling a second batch of tasks at $t_1$ while minimizing the overall energy consumption of the datacenter. We also observe that scheduling the second batch of tasks took less time in instance 4 and 5 because some cores were already busy running the remaining tasks from batch $t_0$, and hence were removed from the pool of available cores. Further testing is carried out to compare the proposed modified model to round robin algorithm in scheduling a second batch of tasks. The results are tabulated in Table 9. In the small instances (instances 1 and 2), round robin and the proposed model results are close and even equal in some cases, but as the number of servers, CPUs, and cores increase, the proposed model results became better compared to round robin. Considering the largest instance tested (instance 5) we observe that round robin results are 235.7% worse than the proposed model. The difference between the two algorithms' results at $t_0$ and $t_1$ is illustrated in Figures 5 and 6.

Table 9. Comparing round robin and the proposed model in scheduling a second batch of tasks

| Instance No. | Min Energy at $t_0$ Proposed model | Min Energy at $t_0$ RR | Min Energy at $t_1$ Proposed model | Min Energy at $t_1$ RR |
|---|---|---|---|---|
| 1 | 24 | 24 | 32 | 32 |
| 2 | 40 | 40 | 97 | 119 |
| 3 | 92 | 134 | 108 | 164 |
| 4 | 80 | 226 | 96 | 228 |
| 5 | 184 | 288 | 224 | 528 |

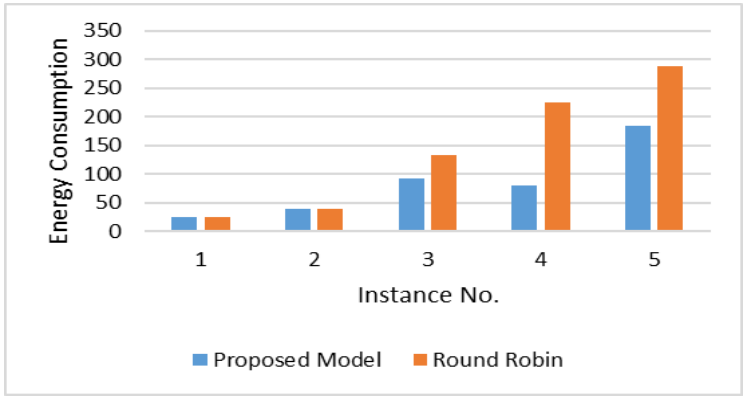


Figure 5. Energy consumption after scheduling the task of $t_0$

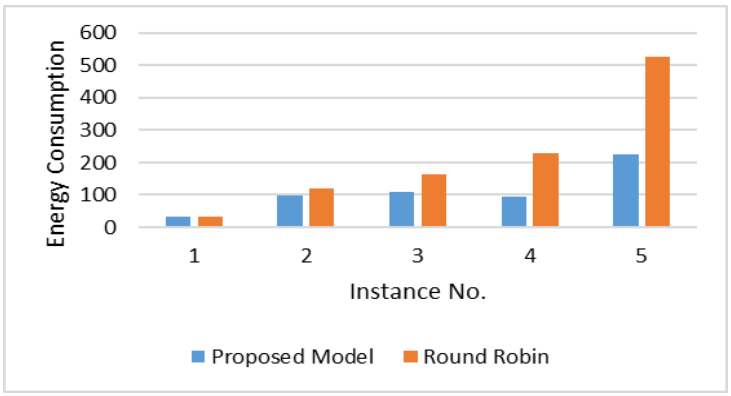


Figure 6. Energy consumption after scheduling the task of $t_1$

## 4. CONCLUSION

In this work, the energy consumption of a datacentre is minimized using energy efficient tasks scheduling and allocation of resources. We proposed a framework to tackle the energy efficiency in the context of a datacentre operation through energy conscious scheduling and reduction in the number of active servers. The resources considered in this research are the computing resources only. We developed ILP optimization model for the task allocation and scheduling problem in a datacentre with the objective of assigning the tasks to the resources that would execute them with the least amount of energy consumption.

The model included tasks' characteristics such as: Task' frequency requirement, its size and RAM requirement. It also included two different running modes (high and low) for the cores inside the same server. The model was tested with different datacentre setups and compared with round robin and maximum possible value methods. It was found that the proposed comprehensive model was able to find the minimum energy consumption in all of the tested cases. Finally, the proposed model was modified to handle the arrival and scheduling of a second batch of tasks after the initial batch is already assigned to various servers in the datacentre. The performance of this model was compared to round robin algorithm, and it was evident that the modified model was able to allocate the two batches of tasks with less energy consumption than round robin.

Various datacentres configurations are studied using different scenarios. The results of these tests show that the energy consumption of a datacentre was found as expected to be proportional to the utilization level of the datacentre, that is, as the number of active cores increases the energy consumption as well increases. Upon testing the complete model and comparing it with round robin and maximum possible value methods, results demonstrated that the proposed model can save up to 35.6% of the energy consumption in some cases when compared to maximum possible value method. Furthermore, while round robin algorithm was not always successful in finding a solution for allocating all the tasks, still the proposed model outperformed it in the few cases where it did find a solution.

Another contribution in this work involved using the proposed ILP formulations to test the performance of three PB-SAT solvers namely: NaPS, Minisat+ and Sat4j, and a generic ILP solver named CPLEX in solving the energy optimization problem for datacentres. The performance of the three PB-SAT

solvers was relatively close to each other, and they were able to find the optimum energy for most of the tested instances. However, as the size of the search space increased the PB-SAT solvers started to time-out. Therefore, for PB-SAT solvers the time needed to explore the search space makes them impractical for finding solutions for large scale datacentres. On the other hand, the generic ILP-solver CPLEX outperformed all the PB-SAT solvers and was able to handle relatively large datacentres configurations and find the optimal solution (the minimum energy consumption) in a shorter amount of time. Potential future directions of this work include extending the ILP models developed in this work to include scheduling of network and storage resources and also extending the proposed model to have a multi-objective function such as reliability and availability of services while optimizing energy consumption.

## REFERENCES


[1] P. Sharma, et al., "Design and operational analysis of a green data center," *IEEE Internet Computing*, vol. 21, no. 4, pp. 16-24, 2017.

[2] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732-794, 2016.

[3] L. Wang, and E. Gelenbe, "Experiments with smart workload allocation to cloud servers," *Proc. of IEEE 4th Symposium on Network Cloud Computing and Applications (NCCA)*, pp. 31-35, 2015.

[4] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 381-384, 1992.

[5] B. Priya, E. Pilli, and R. Joshi, "A survey on energy and power consumption models for Greener Cloud," *Proc. of IEEE 3rd International Advance Computing Conference (IACC)*, pp. 76-82, 2013.

[6] S. Garg and R. Buyya, "Green cloud computing and environmental sustainability," *Harnessing Green IT: Principles and Practices*, vol. 2012, pp. 315-340, 2012.

[7] M. Pinedo, "Scheduling: theory, algorithms, and systems," *Springer*, pp. 35-42, 2016.

[8] R. Aburukba, H. Ghenniwa, and W. Shen, "Agent-based approach for dynamic scheduling in content-based networks," *Proc. of the IEEE International Conference on e-Business Engineering (ICEBE)*, pp. 425-432, 2006.

[9] A. Osman, et al, "Towards Energy Efficient Servers' Utilization in Datacenters," *Intelligent Computing: Proceedings of the 2019 Computing Conference*, vol. 1, pp. 254-262, 2019.

[10] S. Aslam and M. Shah, "Load balancing algorithms in cloud computing: A survey of modern techniques," *Proc. of National Software Engineering Conference (NSEC)*, pp. 30-35, 2015.

[11] R. Kaur and P. Luthra, "Load balancing in cloud computing," *Proc. of International Conference on Recent Trends in Information, Telecommunication and Computing (ITC)*, pp. 374-381, 2012.

[12] N. Liu, Z. Dong, and R. Rojas-Cessa, "Task scheduling and server provisioning for energy-efficient cloud-computing data centers," *Proc. of the IEEE 33rd International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 226-231, 2013.

[13] A. Dambreville, et al., "Load Prediction for Energy-Aware Scheduling for Cloud Computing Platforms," *Proc. of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2604-2607, 2017.

[14] C. Wu, R. Chang, and H. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp.141-147, 2014.

[15] E. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," *International Workshop on Power-Aware Computer Systems*, pp. 179-197, 2002.

[16] S. Wangi, et al., "A DVFS Based Energy-Efficient Tasks Scheduling in a Data Center," *IEEE Access*, vol. 5, 2017.

[17] A. Khan, et al., "An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters," *Journal of Network and Computer Applications*, vol. 150, 2020.

[18] A. Schrijver, "Theory of linear and integer programming," *John Wiley & Sons*, 1998.

[19] S. Malik and L. Zhang, "Boolean satisfiability from theoretical hardness to practical success," *Communications of the ACM*, vol. 52, no. 8, pp. 76-82, 2009.

[20] F. Aloul, et al., "Generic ILP versus specialized 0-1 ILP: An update," *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 450-457, 2002.

[21] C. Gomes, et al., "Satisfiability solvers," *Foundations of Artificial Intelligence*, vol. 3, pp. 89-134, 2008.

[22] A. Sagahyroon, and F. Aloul, "Using SAT-based techniques in power estimation," *Microelectronics Journal*, vol. 38, no. 6-7, pp. 706-715, 2007.

[23] M. Sakai and H. Nabeshima, "Construction of an ROBDD for a PB-Constraint in Band Form and Related Techniques for PB-Solvers," *IEICE Transactions on Information and Systems*, vol. 98, no. 6, pp. 1121-1127, 2015.

[24] Pseudo-Boolean Competition, [Online] Available at: http://www.cril.univ-artois.fr/PB16/, 2016.

[25] N. Eén and N. Sörensson, "An extensible SAT-solver," *International Conference on Theory and Applications of Satisfiability Testing*, pp. 502-518, 2003.

[26] D. Le Berre and A. Parrain, "The sat4j library, release 2.2, system description," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, pp. 59-64, 2010.

[27] IBM ILOG CPLEX Optimizer, [Online] Available at: http://www01.ibm.com/software/integration/optimization/cplex-optimizer/.

[28] LINDO Systems Inc, "Powerful LINGO Solvers," 2017. [Online] Available at: http://www.lindo.com/index.php/products/lingo-and-optimization-modeling/89-products/lingo/88-powerful-lingo-solvers.