PBFVMC: A New Pseudo-Boolean Formulation to Virtual-Machine Consolidation

Bruno César Ribas*‡, Rubens Massayuki Suguimoto†, Razer A. N. R. Montaño*, Fabiano Silva* and Marcos Castilho*
* LIAMF/UFPR - Laboratório de Inteligência Artificial e Métodos Formais
†LARSIS/UFPR - Laboratório de Redes e Sistemas Distribuídos
UFPR - Federal University of Paraná
‡ UTFPR - Federal University of Technology
Email: {ribas,rubensm,razer,fabiano,marcos}@inf.ufpr.br

***Abstract*—Over the course of the last decade, there have been several improvements in the performance of Boolean Satisfiability (SAT), Integer Linear Programming (ILP) and Pseudo-Boolean Optimization (PBO) solvers. These improvements have encouraged the applications of SAT, ILP and PBO techniques in modeling complex engineering problems. One such problem is the *Virtual Machine Consolidation*. The Virtual Machine Consolidation problem consists in placing a set of virtual machines in a set of hardware in a way to increase workload on hardware where they can operate more energy-efficient. This paper proposes an improved PBO formulation of the Virtual Machine Consolidation problem, PBFVMC. The improved formulation and enhancements are built on top of a previous work and a new set of constraints is created and rationalized to work more friendly with current PBO solvers. It is observed that this new formulation goes a step ahead and more problems can now be solved.**

I. Introduction

Over the last decade, Boolean Satisfiability (SAT) and Integer Linear Programming (ILP) solvers have improved significantly through the introduction of new intelligent algorithms that allowed the solvers to handle a wider ranger of challenging and real-world problems.

Within these advances it has come an interest to apply real-world problems to generic SAT,PBO and ILP solvers. One such problem is the *Virtual Machine Consolidation* (VMC) problem in a Cloud Computing infraestructure.

Cloud Computing is a recent paradigm of distributed computing that offers virtualized resources and services over the Internet [1], [2]. Using Cloud Computing is possible to offer a pool of easily usable and accessible virtualized resources. These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs [3].

One of the service models offered by Clouds is Infrastructure-as-a-Service (IaaS) in which virtualized resources are provided as virtual machine (VM). With VMs, users obtain a personalized and isolated execution environment to execute applications. A VM also uses virtualized resources such virtual CPU, virtual RAM, virtual network and virtual storage devices.

Many Cloud providers use a large data center with a huge amount of physical resources (server, disks, wired networks) to offer IaaS. Unfortunately, most of large data center usage ranges from 5% to 10% of capacity on average. In order to maximize the resources utilization a IaaS Cloud provider can apply server consolidation technique [4], [5], [6]. Server consolidation is a technique to reallocate VMs, distributed on many physical servers, on less amount of physical servers. Usually physical servers have capacity to run many VMs at the same time.

A server consolidation can increase workloads on servers from 50% to 85% where they can operate more energy efficiently [7] and, in some cases, a consolidation can save 75% of energy [8]. Reallocating virtualized resources allow to shutdown physical servers, reducing cooling costs, headcount, hardware management and energy consumption costs.

To maximize Cloud data center usage, an optimal VM consolidation has been topic of research in Cloud Computing. There are several works [4], [5], [6], [9], [10] that pursues an optimal resource utilization. In addition to these approaches this paper revisits a previous work [10] that models Virtual Machine consolidation to Pseudo-Boolean constraints and rework all the constraints in order to create a better formula that will take into consideration the limitation of available Pseudo-Boolean solvers and generate a more friendly formula to achieve better results.

We perform our experiments using data from the Google Cluster project which is a trace of real-life scenario. The results show that is possible to decrease the amount of variables in 50% and it is possible to execute huge sets of VM instances.

This paper is organized as follows. In section II, we present background information on SAT and PBO and related work. Section III revisits our previous work and is followed by section IV where we discuss the modifications and additions made to achieve a better formulation. In section V we evaluate the proposed approach using data from real scenario. Finally, in section VI we present conclusion and future work.

II. Background

A. Pseudo-Boolean Optimization

Pseudo-Boolean Optimization involves minimizing or maximizing a function subject to certain constraints where the optimal function and constraints are in Pseudo-Boolean constraints.

A Pseudo-Boolean function in a straightforward definition is a function that maps Boolean values to a real number. The term pseudo-Boolean is given to these functions that are not Boolean but remain very close to Boolean functions [11], [12], [13]. In a *Pseudo-Boolean* (PB) formula, variables have Boolean domains and constraints, known as PB constraints [13], are linear inequalities with integral coefficients. In *PB Optimization*, a cost function is added to a PB formula.

PB functions are a very rich subject of study since numerous problems can be expressed as the problem of optimizing the value of a PB function. PB constraints offer a more expressive and natural way to express constraints than clauses and yet, this formalism remains close enough to the *Satisfiability* (SAT) [11], [12] problem to benefit from the recent advances in SAT solving.

Simultaneously, PB solvers benefit from the huge experience in *Integer Linear Programming* (ILP) and, more specifically, *0-1 programming*. This is particularly true when optimization problems are considered. Inference rules allow to solve problems polynomially when encoded with PB constraints while resolution of the problem encoded with clauses requires an exponential number of steps. PB constraints appear as a compromise between the expressive power of the formalism used to represent a problem and the difficulty to solve the problem in that formalism [13].

A detailed description of modern SAT solver, maximum satisfiability and Pseudo-Boolean optimization can be found, respectively in [11], [12], [13].

B. Related work

Advances in virtualization technology allowed migration of VMs or entire virtual execution environment across physical resources. It also allowed a VM consolidation which has been investigated with different aspects [14], [8], [15] such performance of VM, energy consumption, costs of resource and costs of migration.

Optimal VM consolidation has been explored and solved using *Linear Programming* formulation [6], [9] and Distributed Algorithms [4] approaches.

Ferreto *et. al.* [6] presents a *Linear Programming* formulation and add constraints to control VM migration on VM consolidation process. The migration control constraints use CPU and memory to avoid worst performance when migration occurs.

Binary Integer Program (BIP) Bossche *et. al.* [9] propose and analyze a *Binary Integer Programming* (BIP) formulation of cost-optimal computation to schedule VMs in Hydrid Clouds. The formulation uses CPU and memory constraints and the optimization is solved by *Linear Programming*.

III. Existing PB Formulation of the Consolidation Problem

The goal of our problem is to deploy K VMs $\{vm_1 \ldots vm_K\}$ inside N hardware $\{hw_1 \ldots hw_N\}$ while minimizing the total number of active hardware. Each VM vm_i has an associated need such as number of VCPU and amount of VRAM needed while each physical hardware hw_j has an amount of available resources, number of CPU and available RAM.

In our previous work[10] it was proposed a formulation of the consolidation problem using PB constraints to take advantage of the advances of the PB solvers and many techniques that were aggregated to PB solvers from SAT solvers and, to the best of our knowledge, it is the only contribution to this problem using a formulation that is dispatched to a PB solver.

This formulation has 6 types of constraints which led to $(2\times N+2\times N\times K)$ variables and $(2+2\times N+K)$ constraints, where N and K represents the number of available hardware and virtual machines, respectively. This formulation is quite compact but on the other hand is very hard to a PB solver since the biggest formula that we could prove satisfiable in a 14400 seconds of time limit was a formula with 128 hardware and 1277 virtual machines and the best optimal proved was a formula with only 32 hardware and 98 virtual machines.

The formulation of our previous work is provided below:

In order to create the PB Constraints each hardware consists of two variables, one that relates hw_i to the amount of RAM hw_i^r and one that relates to the amount of CPU hw_i^p. Per hardware, a VM has 2 variables, one to relate the VM vm_j required amount of VRAM vm_j^r to the hardware hw_i amount of RAM hw_i^r, denoted as $vm_j^{r\cdot hw_i}$. The another variable relate the required VCPU vm_j^p to the amount of CPU available hw_i^p, denoted as $vm_j^{p\cdot hw_i}$. The total amount of VM variables is $2\times N$ variables.

A hardware is considered ON when its hw_i^r and hw_i^p are *True*, otherwise it is OFF.

$$minimize : \sum_{i=1}^{N} hw_i \tag{1}$$

$$\sum_{i=1}^{N} R_{hw_i} \cdot hw_i^r \geq \sum_{j=1}^{K} R_{vm_j} \tag{2}$$

$$\sum_{i=1}^{N} P_{hw_i} \cdot hw_i^p \geq \sum_{j=1}^{K} P_{vm_j} \tag{3}$$

$$\forall\ i \in 1..N \left(\sum_{j=1}^{K} R_{vm_j} \cdot vm_j^{r\cdot hw_i} \leq R_{hw_i} \right) \tag{4}$$

$$\forall\ i \in 1..N \left(\sum_{j=1}^{K} P_{vm_j} \cdot vm_j^{p\cdot hw_i} \leq P_{hw_i} \right) \tag{5}$$

$$\forall\ j \in 1..K \left(\sum_{i=1}^{N} vm_j^{p\cdot hw_i} \cdot vm_j^{r\cdot hw_i} \cdot hw_i^p \cdot hw_i^r = 1 \right) \tag{6}$$

The objective function is the summation of the ON servers. Constraints (2) and (3) guarantee the the necessary amount of ON resources are enough to power all the VMs. To limit the upper bound of hardwares, constraints (4) and (5) are the upper limit of the resources each hardware can provide. Finally constraint (6) guarantees that the VM is running in exactly one hardware. Due to the non-linear nature of this constraint, it is implicitly defined that if a VM is running on a hardware, this hardware must be ON.

As noted at the beginning of this section, this formulation is very compact and it is possible to achieve this succinctness because it is a non-linear formula where constraint 6 has a sum of four multiplication.

IV. Proposed PB Formulation

This paper proposes a PB formulation that modifies the previous formulation in the past section to new set where it can be more comparable to a Pigeon Hole formulation than to a Bin Packing formulation. This model improves on weaknesses present in previous PB Formulation [10]. From now we will refer this new formulation as PBFVMC.

Previous formulation were more comparable to a Bin Packing problem, specially on the constraint (6). The PBFVMC acts more like a Pigeon Hole formulation, with a special hole (hardware) that can handle more than one pigeon (virtual machines) limited to the amount of resource available (RAM and CPU) in each hole (hardware).

A. *Proposed Base Model*

Most of the structure defined in [10] and recalled in section III, are kept. All the pseudo-boolean variables that were associated to the RAM (and VRAM) and CPU (and VCPU), before named, hw_i^r (and vm_j^r) and hw_i^p (and vm_j^p) are now merged as hw_i for hardware and vm_j for virtual machines.

That said, our new formulation contains the following variables:

- N : Total number of available hardware (hw);
- K : Total number of virtual machines (VM);
- hw_i : Hardware $i \in N$;
- $vm_j^{hw_i}$: Virtual Machine $j \in K$ that runs in hw_i;

A hardware is considered ON when hw_i is *True*, otherwise it is OFF.

Although there is no separation between the pseudo-boolean variable that is related to the amount of RAM and processing power each VM has one variable that relates it to a running hardware. That said this new formulation contains $(N + N \times K)$ variables.

Our constraints are defined as follows:

$$minimize : \sum_{i=1}^{N} hw_i \tag{7}$$

$$\sum_{i=1}^{N} R_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} R_{vm_j} \tag{8}$$

$$\sum_{i=1}^{N} P_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} P_{vm_j} \tag{9}$$

$$\forall_{i \in 1..N} \left(\sum_{j=1}^{K} (R_{vm_j} \cdot \neg vm_j^{hw_i}) + R_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} R_{vm_j} \right) \tag{10}$$

$$\forall_{i \in 1..N} \left(\sum_{j=1}^{K} (P_{vm_j} \cdot \neg vm_j^{hw_i}) + P_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} P_{vm_j} \right) \tag{11}$$

$$\forall_{j \in 1..K} \left(\sum_{i=1}^{N} vm_j^{hw_i} \geq 1 \right) \tag{12}$$

$$\forall_{j \in 1..K} \left(\sum_{i=1}^{N} \neg vm_j^{hw_i} \geq N - 1 \right) \tag{13}$$

The objective function is the summation of the ON servers. Inequalities constraints (8) and (9) guarantees that the summation of memory and processing power of the powered ON servers fit the needs to power all virtual machines. Constraints (10) and (11) are the upper limit on the total resources each hardware may provide in relation to the virtual machines that may run on this hardware. Constraint (12) states that a virtual machine must be running in some hardware. Constraint (13) ensures that the virtual machine is running in exactly one hardware.

This new formulation generates $(2 + 2 \times N + 2 \times K)$ constraints and $(N + N \times K)$ variables which has only K more constraints than previous formulation and half of the variables.

B. *Discussion on new formulation*

The proposed formulation is an improvement from previous as the amount of variables were cut in half and this is good for the solver since the search space is smaller now, and the most important difference is that we no longer maintain non-linear constraints.

Non-linear constraints, as seen in constraint (6) are very hard to solve and most, if no all, solvers translate non-linear constraints into an equivalent linear instance. This is easily done, as seen on [13], but many methods introduce a significant number of auxiliary variables, increasing the search space causing a direct impact on the running time, specially when running against formulas with higher N and K values.

Constraints (10) and (11) are just an algebraic manipulation of constraints (4) and (11), reworked in a way to include the variables hw_i on the left side of the constraint to dictate that the hw_i must be on if some of the $vm_j^{hw_i}$ are running on this server, after that the constraints are normalized in a way that all coefficients are non-negative and the relational operator becomes $\geq$.

Constraints (12) and (13) represents the constraint (6) resembling a CNF-Pigeon Hole formulation instead of a Bin Packing formulation. While Bin Packing formulation are strictly written in a summation that equals one as constraint (6), a Pigeon Hole formulation, on the other side, is easily done with clauses leading to $(n+1)$ clauses, where n is the number of holes, saying that a pigeon has to be placed in some hole as shown in (14), and then for each hole we have a set of clauses ensuring that only one single pigeon is placed into that hole, and it is defined as constraint (15). The problem on using this formulation is that the number of clauses increases rapidly as the number of pigeons grow, and this is more critical within our problem as we may have thousands of virtual machines in hundreds of hardware, and this notation becomes really hard to use. In order to reduce the size we strength the formula by the strengthening preprocess where these clauses will be rewritten in one PB constraint.

$$\forall j \in 1..K \left(\sum_{i=1}^{N} vm_j^{hw_i} \geq 1 \right) \tag{14}$$

$$\forall j, i, k \in 1..K, 1..N, i+1..K \ (\neg vm_j^{hw_i} + \neg vm_k^{hw_i}) \geq 1 \tag{15}$$

1) Strengthening Formula: Dixon [16], [17], [18], rescues the discussion on taking advantage of advances in operations research techniques to preprocess formulas, in special it discussed how strengthening can be applied to pseudo-boolean constraints directly.

Strengthening is a method where a literal, or a set of literals are fixed a value and then a propagation is applied to the formula. Some assumptions will cause some constraints to become oversatisfied, i.e. suppose that after setting a literal, l_0 to true, we discover that a constraint c is given by $\sum w_i l_i \geq r$ becomes oversatisfied by an amount s in that the sum of the left hand side is greater (by s) than the amount required bu the right hand side of the inequality. The oversatisfied constraint c can now be replaced by the following:

$$s \cdot \neg l_0 + \sum w_i l_i \geq r + s \tag{16}$$

As proved in [16], if l_0 is true, we know that $\sum w_i l_i \geq r + s$, so (16) holds. If l_0 is false, then $s \cdot \neg l_0 = s$ and we still must satisfy the original constraint $\sum w_i l_i \geq r$, so (16) still holds. The new constraint implies the original one, so no information is lost in the replacement. The power of this method is that it allows us to build more complex axioms from a set of simple ones. The strengthened constraint will often subsume some or all of the constraints involved in generating it.

In the case of a pigeon hole formulation, constraint (15) will be strengthened and will subsume all constraints (15), which will be replaced by constraint (13), generating a smaller and richer set of constraints, taking advantage of all the power pseudo-boolean provides and yet keeping with linear and normalized constraints.

V. Experiments

For the implementation and evaluation of the PB Constraints, we wrote a simple program that reads the amount of physical hardware followed by its amount of RAM and CPU, the amount of VM and its requirements of virtual memory (VRAM) and virtual processing power (VCPU), and solved the formula using open source PB solver/optimizer *Sat4j-PB* [19], *SCIP* [20] and *BSOLO* [21].

Our experiments were executed on a Intel Xeon 2.1GHz with 256GB of memory and our PB consolidation formulation is applied against the Google Cluster Data Project workloads.

We also used a *subset of workloads* to see the progress on the use of different amount of VM or tasks. A *subset of workload* is the larger subset of VMs or tasks which sum of VCPU or VRAM requirements does not exceed σ percent of sum of physical servers CPU or RAM capacities. In this experiment we assume σ equals to 25%, 50%, 75%, 85%, 90%, 95%, 98%, and 99%. Although we ran all tests with various solvers, only the results of the SAT4J-PB are shown since this solver was the one that had the best performance overall the formulas.

A. Google Cluster Data Project

Google Cluster Data [1] is a Google project to intend for the distribution of data about workloads running on Google Cluster. The workloads contain data traces about *12k* hardware describing events and resource capacity of each server. The traces also describes around *132k* tasks workloads with respective resource requirements.

Due to the time constraints we selected five subsets of hardware. The sizes of each subset are **32**, **64**, **128**, **256**, **512** hardware. For each size of subset hardware, we used the before *subset of workload* to perform experiments. Table I shows the amount of hardware, virtual hardware and the size of the formulas in the previous formulation and PBFVMC.

As a result, table II shows time results for the set of formulas explained above. For each instance a time limit of 14400 seconds was given. When the solver runs out of time limit and does not find any solution it is show a Time Limit Exceeded (TLE), formulas that timed out in both formulations are omitted. When the solver proved optimum result it is shown in bold the time of the optimum time with the value of the objective function. When the solver could find an satisfiable assignment but could not prove optimum result it is show in normal font the time that the best solution was found and the best value of the objective function. Table III shows the time spent by the solver to find a satisfiable assignment.

Running these experiments over the new proposed formulation can be noticed a great break through, while with previous formulation couldn't solve most of the formulas with 128 hardware and just half of the formulas with 64 hardware, on top the new formulation most of the formulas with 32 hardware could be proven optimum in less than 1000s and most of the formulas up to 512 hardware could be proven SATISFIABLE, which are 4 times bigger than the bigger formula solved with the previous formulation. Also table III shows that most of the formulas could be proven satisfiable in under 200s. This represents that the solver used most of the time is to optimize the formula, while in the previous formulation most formulas could not be proven satisfiable.

VI. Conclusion

This paper presented an enhanced VM consolidation model using an artificial intelligence based on Pseudo-Boolean (PB)

[1] http://code.google.com/p/googleclusterdata/

TABLE I: Comparison of the size of the formulas from the previous and PBFVMC formulation to the problem. Table shows workloads of 25%, 50%, 75%, 85%, 90%, 95%, 98% and 99% for the subsets of 32, 64, 128, 256 and 512 hardware.

		Previous		PBFVMC	
HW	VMS	Vars	Constr	Vars	Constr
hw32-vm25p	98	6336	164	3168	262
hw32-vm50p	173	11136	239	5568	412
hw32-vm75p	278	17856	344	8928	622
hw32-vm85p	320	20544	386	10272	706
hw32-vm90p	325	20864	391	10432	716
hw32-vm95p	348	22336	414	11168	762
hw32-vm98p	364	23360	430	11680	794
hw32-vm99p	366	23488	432	11744	798
hw64-vm25p	174	22400	304	11200	478
hw64-vm50p	371	47616	501	23808	872
hw64-vm75p	559	71680	689	35840	1248
hw64-vm85p	629	80640	759	40320	1388
hw64-vm90p	665	85248	795	42624	1460
hw64-vm95p	707	90624	837	45312	1544
hw64-vm98p	712	91264	842	45632	1554
hw64-vm99p	713	91392	843	45696	1556
hw128-vm25p	368	94464	626	47232	994
hw128-vm50p	713	182784	971	91392	1684
hw128-vm75p	1048	268544	1306	134272	2354
hw128-vm85p	1155	295936	1413	147968	2568
hw128-vm90p	1277	327168	1535	163584	2812
hw128-vm95p	1321	338432	1579	169216	2900
hw128-vm98p	1368	350464	1626	175232	2994
hw128-vm99p	1410	361216	1668	180608	3078
hw256-vm25p	712	365056	1226	182528	1938
hw256-vm50p	1407	720896	1921	360448	3328
hw256-vm75p	2119	1085440	2633	542720	4752
hw256-vm85p	2372	1214976	2886	607488	5258
hw256-vm90p	2480	1270272	2994	635136	5474
hw256-vm95p	2583	1323008	3097	661504	5680
hw256-vm98p	2619	1341440	3133	670720	5752
hw256-vm99p	2678	1371648	3192	685824	5870
hw512-vm25p	1432	1467392	2458	733696	3890
hw512-vm50p	2771	2838528	3797	1419264	6568
hw512-vm75p	4035	4132864	5061	2066432	9096
hw512-vm85p	4431	4538368	5457	2269184	9888
hw512-vm90p	4745	4859904	5771	2429952	10516
hw512-vm95p	5068	5190656	6094	2595328	11162
hw512-vm98p	5319	5447680	6345	2723840	11664
hw512-vm99p	5402	5532672	6428	2766336	11830

Constraints. Formulas are solved by a generic PB solver, avoiding the need to write specific algorithms. The use of a generic PB solver benefits our approach as improvements in PB solving are incorporated in the solvers, these formulas automatically becomes easier to solve.

Results described in this paper shows a break through in the generated formulas, follow experimental results, by removing equal constraints, non-linear constraints and cutting in half the number of variables we can identify an increase of 4 times in the size of the hardware and virtual machines coded in the formulas being solved and 7 time bigger in terms of PB constraints. Also experiments shows that with PBFVMC solvers spend most the running time dedicated to optimize the formula while in the previous formulation most of the time are spent trying to decide whether the formula is satisfiable.

Although optimum could not be proved for bigger formulas a work on the formulation will continue to achieve formulas that are easier to solve, also we will start to work on the PB solvers to improve performance when running these formulas to achieve optimum results in bigger formulas.

References

[1] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?" *Journal of Computer*, vol. 42, no. 1, pp. 15–20, Jan. 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep., 2009.

[3] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: towards a Cloud Definition," *Journal of ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[4] M. Marzolla, O. Babaoglu, and F. Panzieri, "Server Consolidation in Clouds through Gossiping," in *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE Computer Society, Jun. 2011, pp. 1–6.

[5] W. Vogels, "Beyond Server Consolidation," *Journal of ACM Queue*, vol. 6, no. 1, p. 20, Jan. 2008.

[6] T. C. Ferreto, M. a.S. Netto, R. N. Calheiros, and C. a.F. De Rose, "Server Consolidation with Migration Control for Virtualized Data Centers," *Journal of Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, Oct. 2011.

[7] R. Harmon and N. Auseklis, "Sustainable IT Services: Assessing the Impact of Green Computing Practices," in *Proceedings of International Conference on Management of Engineering & Technology.* IEEE Computer Society, Aug. 2009, pp. 1707–1717.

[8] A. Corradi, M. Fanelli, and L. Foschini, "Increasing Cloud Power Efficiency through Consolidation techniques," in *Proceeding of IEEE Symposium on Computers and Communications.* IEEE Computer Society, Jun. 2011, pp. 129–134.

[9] R. Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads," in *Proccedings of IEEE 3rd International Conference on Cloud Computing*. IEEE Computer Society, Jul. 2010, pp. 228 –235.

[10] B. C. Ribas, R. M. Suguimoto, R. A. M. no, F. Silva, L. de Bona, and M. A. Castilho, "On modelling virtual machine consolidation to pseudo-boolean constraints," in *Advances in Artificial Intelligence – IBERAMIA 2012*, ser. Lecture Notes in Computer Science, J. Pavón, N. D. Duque-Méndez, and R. Fuentes-Fernández, Eds., vol. 7637. Springer, 2012, pp. 361–370.

TABLE II: Execution time per instance for Sat4j-PB solver running against the previous formulation and PBFVMC. Time Limit was set to 14400s and TLE represents when Time Limit is Exceeded, when the result is bold means that the optimum was found, when not means the solver could not prove optimum and the value is the time of the best solution found

Formula	Previous	PBFVMC
hw32-vm25p	**249.897/7**	**191.912/7**
hw32-vm50p	35.696/16	4.134/16
hw32-vm75p	23.628/25	**772.657/24**
hw32-vm85p	1175.103/29	159.86/28
hw32-vm90p	108.361/31	**948.924/29**
hw32-vm95p	3442.92/32	**319.041/31**
hw32-vm98p	TLE	**45.651/32**
hw32-vm99p	TLE	**5566.491/32**
hw64-vm25p	4248.893/17	8.541/16
hw64-vm50p	6477.271/33	200.261/33
hw64-vm75p	8784.933/50	8608.38/47
hw64-vm85p	603.393/59	490.656/55
hw64-vm90p	1272.89/62	869.421/58
hw64-vm95p	TLE	679.719/62
hw64-vm98p	TLE	4135.757/64
hw64-vm99p	TLE	**240.642/64**
hw128-vm25p	TLE	10319.859/29
hw128-vm50p	14661.134/75	4856.869/64
hw128-vm75p	16209.656/105	12538.628/98
hw128-vm85p	11203.456/122	1117.772/115
hw128-vm90p	13491.676/128	11295.761/117
hw128-vm95p	TLE	65.916/128
hw256-vm25p	TLE	12381.653/68
hw256-vm50p	TLE	3576.626/136
hw256-vm75p	TLE	11468.942/204
hw256-vm85p	TLE	10537.747/230
hw256-vm90p	TLE	2704.592/243
hw256-vm95p	TLE	2003.068/255
hw256-vm98p	TLE	7737.502/256
hw512-vm25p	TLE	4471.005/140
hw512-vm50p	TLE	5406.047/281
hw512-vm75p	TLE	4378.66/408
hw512-vm85p	TLE	4919.328/461
hw512-vm90p	TLE	14426.6/487
hw512-vm95p	TLE	6864.151/510

TABLE III: Execution time per instance for Sat4j-PB solver to find a satisfiable assignment. Time Limit was set to 14400s

Formula	Previous	PBFVMC
hw32-vm25p	92.756	0.433
hw32-vm50p	35.643	0.542
hw32-vm75p	3.43	0.588
hw32-vm85p	4.516	0.911
hw32-vm90p	6.795	9.716
hw32-vm95p	3442.92	8.129
hw32-vm98p	TLE	45.589
hw32-vm99p	TLE	5566.28
hw64-vm25p	3118.029	0.706
hw64-vm50p	18.306	0.892
hw64-vm75p	50.687	1.15
hw64-vm85p	60.38	1.365
hw64-vm90p	121.006	1.423
hw64-vm95p	TLE	7.512
hw64-vm98p	TLE	4135.757
hw64-vm99p	TLE	240.538
hw128-vm25p	TLE	1.731
hw128-vm50p	4015.592	2.753
hw128-vm75p	5975.386	4.026
hw128-vm85p	7676.653	7.984
hw128-vm90p	13491.676	7.904
hw128-vm95p	TLE	65.916
hw256-vm25p	TLE	4.379
hw256-vm50p	TLE	14.244
hw256-vm75p	TLE	33.259
hw256-vm85p	TLE	48.298
hw256-vm90p	TLE	126.506
hw256-vm95p	TLE	389.329
hw256-vm98p	TLE	7737.502
hw512-vm25p	TLE	28.436
hw512-vm50p	TLE	162.289
hw512-vm75p	TLE	508.322
hw512-vm85p	TLE	287.437
hw512-vm90p	TLE	5604.022
hw512-vm95p	TLE	4222.892

[11] J. P. Marques-Silva, I. Lynce, and S. Malik, *Conflict-Driven Clause Learning SAT Solvers*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 4, pp. 131–153.

[12] C. M. Li and F. Manyà, *MaxSAT, Hard and Soft Constraints*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 19, pp. 613–631.

[13] O. Roussel and V. Manquinho, *Pseudo-Boolean and Cardinality Constraints*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 22, pp. 695–733.

[14] H. Umeno, M. C. Parayno, K. Teramoto, M. Kawano, H. Inamasu, S. Enoki, M. Kiyama, T. Aoyama, and T. Fukunaga, "Performance Evaluation on Server Consolidation using Virtual Machines," in *Proceedings of SICE-ICASE International Joint Conference*. IEEE Computer Society, 2006, pp. 2730–2734.

[15] S. Mehta and A. Neogi, "ReCon: A Tool to Recommend Dynamic Server Consolidation in Multi-cluster Data Centers," in *Proceeding of IEEE Network Operations and Management Symposium 2008*. IEEE Computer Society, 2008, pp. 363–370.

[16] H. E. Dixon, M. L. Ginsberg, and A. J. Parkes, "Generalizing boolean satisfiability i: Background and survey of existing work," *J. Artif. Intell. Res. (JAIR)*, vol. 21, pp. 193–243, 2004.

[17] H. E. Dixon, M. L. Ginsberg, E. M. Luks, and A. J. Parkes, "Generalizing boolean satisfiability ii: Theory," *J. Artif. Intell. Res. (JAIR)*, vol. 22, pp. 481–534, 2004.

[18] H. E. Dixon, M. L. Ginsberg, D. K. Hofer, E. M. Luks, and A. J. Parkes, "Generalizing boolean satisfiability iii: Implementation," *J. Artif. Intell. Res. (JAIR)*, vol. 23, pp. 441–531, 2005.

[19] D. Le Berre, "SAT4j: a Reasoning Engine in Java based on the SATisfiability Problem," http://www.sat4j.org.

[20] T. Achterberg, "Scip: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009, http://mpc.zib.de/index.php/MPC/article/view/4.

[21] V. Manquinho, "BSOLO: A Solver for Pseudo-Boolean Constraints," http://sat.inesc-id.pt/~vmm/research/.

[22] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185.