Automated Verification of Stand-alone Solar Photovoltaic Systems

Alessandro Trindade[a,*], Lucas Cordeiro[b]

[a] *Federal University of Amazonas, Av. Rodrigo Octvio, 6200, Coroado I, 69077-000 Manaus-AM-Brazil*
[b] *University of Manchester, School of Computer Science, Kilburn Building, Manchester M13 9PL*

Abstract

With declining costs and increasing performance, the deployment of renewable energy systems is growing faster. Particular attention is given to stand-alone solar photovoltaic systems in rural areas or where grid extension is unfeasible. Tools to evaluate electrification projects are available, but they are based on simulations that do not cover all aspects of the design space. Automated verification using model checking has proven to be an effective technique to program verification. This paper marks the first application of software model checking to formally verify the design of a stand-alone solar photovoltaic system including solar panel, charge controller, battery, inverter, and electric load. Case studies, from real photovoltaic systems deployed in five different sites, ranging from 700W to 1,200W, were used to evaluate this proposed approach and to compare that with specialized simulation tools. Data from practical applications show the effectiveness of our approach, where specific conditions that lead to failures in a photovoltaic solar system are only detected by our automated verification method.

Keywords: Formal verification, model checking, photovoltaic power systems, power system modeling, solar power generation

*Corresponding author
Email address: alessandrotrindade@ufam.edu.br (Alessandro Trindade)

1. Introduction

There are presently 1.3 billion people with no access to electricity worldwide [1]. Only a niche market a few years ago, solar photovoltaic (PV) systems are now becoming a mainstream electricity provider, with an increase of approximately 50% from 2016 to 2017 in terms of new installations of PV [2].

In order to simulate or evaluate a PV system there are a myriad of specialized tools available in the market, such as RETScreen, HOMER, PVWatts, SAM, and Hybrid2 [3, 4, 5, 6, 7]; and even general purpose simulation tools such as PSpice, Saber, or MATLAB/Simulink package [8, 9]. However, those tools are based on running experiments in simulation models. Simulation has the advantage of being cheap (if compared to test in real systems) and can be employed before the system design is concluded but it has the drawback of an incomplete coverage since the verification of all possible combinations and potential failures of a system is unfeasible [10].

Formal methods based on model checking offer a great potential to obtain a more effective and faster verification in the design process [10]. Any kind of system can be specified as computer programs using mathematical logic, which constitutes the intended (correct) behavior; then, one can try to give a formal proof or otherwise establish that the program meets its specification. In this study, a mathematical model of each component of a stand-alone PV system, as panel solar, charge controller, batteries, inverter, and electrical load is created. The behavior of each system component can be analyzed and observed with the support of those formal models, as a joint operation of the components, which in this case represents the operation of the solar PV system itself. The project requirements, as battery autonomy and power demand, besides weather conditions, as solar irradiance and ambient temperature, are translated as part of the computer program and automatically verified during the formal process. The model checking tool reports in which conditions a system does not meet the user requirements. A key benefit to this approach is that it helps in the detection of flaws in the design phase of system development, thereby consid-

erably improving system reliability [11]. The implementation of the proposed tool is carried out by means of the efficient SMT-based bounded model checker (ESBMC) [12].

In prior studies, the evaluation of PV systems w.r.t. user requirements were performed by simulation tools using MATLAB/Simulink [9, 13], or HOMER Pro [14]. Related to formal methods, studies were carried out toward the modeling of power smart grids [11] and solar panels [15]; however, those studies do not perform automated verification. In addition, recent research performed a formal study of PV farms, where the focus has been the interaction with the grid [16]; or to model a PV system and to verify the maximum power point of solar panels with the use of Jmodelica tool [17]. Both studies restrict to PV panels, and do not include batteries, inverters, and charge controllers.

This paper makes two original contributions. Firstly, we propose an automated verification method that formally checks the design of a given PV system using incremental model checking based on Satisfiability Module Theories (SMT), and that was never did before. Secondly, experimental results show that this proposed approach can find subtle design errors in PV systems, which are not easily detected by other approaches based on simulation.

Outline. Section 2 gives the background about solar PV systems, design and validation of PV systems, and the mathematical modeling. Section 3 presents the automated verification technique. The methodology is presented in section 4. Section 5 is devoted to the results. Section 6 presents the conclusion and describes future work.

2. Solar Photovoltaic System

PV systems are classified into distinct types [18]. Specifically for remote rural areas of developing countries or places where the grid extension is not feasible, the most suitable configuration is the regulated stand-alone system with battery and AC load; this configuration is the focus of this study.

Table 1: Comparative coverage of reference software

Characteristic	PVWatts	SAM	HOMER	RETScreen	Hybrid2
Support	X	X	X	X	
Off-grid systems			X	X	X
Hybrid systems			X	X	X
Photovoltaics	X	X	X	X	X
Batteries			X		X
Main technical (T) or economical(E)	T	T	E	E	T
Optimization			X	X	
Sensitive analysis			X	X	

2.1. Design and Simulation of Solar PV systems

Here we evaluated the most popular tools available in the literature and summarized at Table 1: PVWatts, SAM, HOMER, RETScreen, and Hybrid2 [3, 4, 5, 6, 7].

As highlights, only Hybrid2 does not have technical support; HOMER and Hybrid2 perform off-grid system with battery backup analysis. Additionally, HOMER and RETScreen include economical analysis or even optimization-sensitive analysis. However, commercial version of those tools, called RETScreen Expert and HOMER Pro, are available only for Microsoft Windows and the annual subscription typically range from US$504.00 to US$657.00.

In this study, only HOMER has the minimum features for a comparative with our proposed verification approach. That evaluation is presented in Section 5.3.

2.2. Component models for a stand-alone PV system

A stand-alone PV system is illustrated in Fig.1. The PV generator (panel or array) is a semiconductor device that can convert solar energy into DC electricity. For night hours or rainy days, we need to hold batteries, where power can be stored and used. The use of battery as a storage form implies the presence of a charge controller [19]. The PV arrays produce DC and therefore when the PV system contains an AC load, a DC/AC conversion is required (inverter). The AC load dictates the behavior of AC electrical load from the house that will be fed by the system.

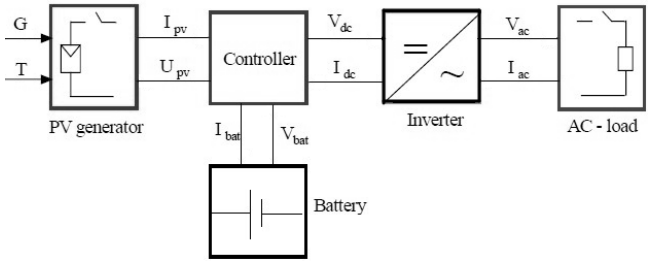

Figure 1: Block diagram for a typical stand-alone PV system [19].

2.3. PV generator model

A wide variety of models exists for estimation of power output of PV modules (and $I-V$ or $P-V$ curves). However, the present work will rely on the simplified model of 1-diode. It was demonstrated that it has a small error rate, between 0.03% and 4.68% from selected PV panels tested [20].

The 1-diode model, illustrated in Fig. 2, whose equation relates the output current, I, to the output voltage, V, is described by (1).

$$I = I_{ph} - I_{D1} = I_{ph} - I_0 \left[exp\left(\frac{V}{NaV_T} \right) \right], \tag{1}$$

where I_{ph} is the photocurrent delivered by the constant current source; I_0 is the reverse saturation current corresponding to the diode; N is the number of series-connected cells; a is the ideality or quality factor ($a = 1$ for ideal diodes and between 1 and 2 for real diodes); V_T is the thermal voltage ($V_T = k_B T/q$); k_B

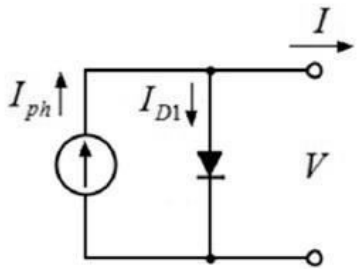

Figure 2: 1-diode equivalent PV cell/panel circuit model [21].

is the Boltzmann constant ($1.3806503 \times 10^{-23} J/K$); T the temperature of cell in Kelvin; q is absolute value of the electron's charge ($-1.60217646 \times 10^{-19} C$).

The voltage and the current at the maximum power point tracking (MPPT), can be described by (2), (3), and (4) as follows [20]:

$$V_m = \frac{aNk_BT}{q} ln\left(\frac{aNk_BT}{qI_0}\frac{I_{sc}}{V_{oc}}\right). \tag{2}$$

$$I_m = I_{ph} + I_0 - \frac{aNk_BT}{q}\left(\frac{I_{sc}}{V_{oc}}\right). \tag{3}$$

$$P_m = V_m I_m. \tag{4}$$

However, the photocurrent delivered by the constant current source (I_{ph}) or even the reverse saturation current (I_0) are not given by PV manufacturers. Therefore, (5) is used to calculate the photocurrent as function of irradiance and temperature [22]:

$$I_{ph} = \frac{G}{G_{ref}}\left[I_{ph,ref} + \mu_I\left(T - T_{ref}\right)\right], \tag{5}$$

where the reference state (STC) of the cell is given by the solar irradiance $G_{ref} = 1000 W/m^2$ and the temperature $T_{ref} = 298.15K (= 25^oC)$; μ_I is the short-circuit current temperature coefficient (A/K) (provided by PV manufacturers). $I_{ph,ref}$ can be approximated to the reference short-circuit current [22] that is provided by PV manufacturers ($I_{sc,ref}$). The cell temperature (T) in degree

Celsius is described by 6 [23]:

$$T = T_{air} + \frac{NOCT - 20}{800} G, \tag{6}$$

where T_{air} is the ambient temperature, $NOCT$ is the nominal operating cell temperature (in ^{o}C) that is found at the PV manufacturer's data-sheet [23], and G is the solar irradiance (W/m^2) of the place where the PV system is deployed.

Furthermore, (7) permits the saturation current (I_0) to be expressed as a function of the cell temperature as [22]

$$I_0 = \frac{I_{sc,ref} + \mu_I (T - T_{ref})}{exp\left[\frac{q(V_{oc,ref} + \mu_V (T - T_{ref}))}{aNk_BT}\right] - 1}, \tag{7}$$

where $V_{oc,ref}$ is the reference open-circuit voltage and μ_V is an open-circuit voltage temperature coefficient (V/K).

Using the maximum power point current (cf. (3)) and the saturation current in the reference temperature given by (7), the diode ideality factor is determined by (8):

$$a = \frac{q(V_{m,ref} - V_{oc,ref})}{Nk_BT} \frac{1}{ln\left(1 - \frac{I_{m,ref}}{I_{sc,ref}}\right)}, \tag{8}$$

where V_{mref}, $V_{oc,ref}$, $I_{m,ref}$, and $I_{sc,ref}$ are key cell values obtained under both actual cell temperature and solar irradiance conditions.

The PV generator model is now completely determined. It requires the actual cell temperature (or the air temperature), the actual solar irradiance and common data provided by manufacturers.

2.4. The Battery Storage Model

Various models have been described in the literature and the most common ones are based on lead-acid batteries [24, 25]; that kind of battery has relative low cost and wide availability [24]. Here, the model adopted uses only manufacturer's data without empirical tests [24]. The discharge voltage equation is described by (9).

$$V_d = [2.085 - 0.12(1 - SOC)] - \frac{I}{C_{10}} \left(\frac{4}{1 + I^{1.3}} + \frac{0.27}{SOC^{1.5}} + 0.02 \right) (1 - 0.007\Delta T), \tag{9}$$

where C_{10} means 10h of rated capacity, which is standard on the manufacturer's data-sheet, ΔT is temperature variation ($\Delta T = T - T_{ref}$), SOC or state of charge indicates how much electric charge is stored in the cell at a given time. Mathematically, it is the ratio between the present capacity and the nominal capacity. If $SOC = 1$, then the battery is totally charged; and if $SOC = 0$, then the battery is fully discharged. The depth of discharge (DOD) or the fraction of discharge, is $DOC = 1 - SOC$.

For the charging process, however, the parameters are described by (10) as

$$V_c = [2 + 0.16SOC] + \frac{I}{C_{10}} \left(\frac{6}{1 + I^{0.86}} + \frac{0.48}{(1 - SOC)^{1.2}} + 0.036 \right) (1 - 0.025\Delta T). \tag{10}$$

Note that SOC can be calculated easily at any point during the discharge period, thereby considering the current drained from batteries during a certain time period.

2.5. Charge Controller Model

To protect the battery against an excessive charge, the PV arrays are disconnected from the system, when the terminal voltage increases above a certain threshold V_{max_off} and when the current required by the load is less than the current delivered by the PV arrays [19]. PV arrays are connected again when the terminal voltage decreases below a certain value V_{max_on}. In order to protect the battery against excessive discharge, the load is disconnected when the terminal voltage falls below a certain threshold V_{min_off} and when the current required by the load is larger than the current delivered by the PV arrays [19]. The load is reconnected to the system, when the terminal voltage is above a certain value V_{min_on}. The steps in the modeling of the controller process are summarized in Table 2.

Table 2: Summary of the controller process (Source: [19])

Step	Constraint	Command
(1)	If $V > V_{max\_off}$ and $I_{load} < I_{pv}$	Disconnect PV array from the system
(2)	If command (1) is done and $V < V_{max\_on}$	Reconnect PV array to the system
(3)	If $V < V_{min\_off}$ and $I_{load} > I_{pv}$	Disconnect the load from the system
(4)	If command (3) is done and $V > V_{min\_on}$	Reconnect the load to the system

The output power (P_{out}) of controller is given by (11).

$$P_{in}\eta_c = P_{out}. \tag{11}$$

Assuming that the efficiency of the controller (η_c) is a manufacturer's data, from (11) we compute (12).

$$V_{in}I_{in}\eta_c = V_{out}I_{out}, \tag{12}$$

where V_{in} is the voltage across the PV array, I_{in} is the output current of PV array, $V_{out} = V_b = V_{system}$ is the DC bus voltage, and I_{out} is the output current from the converter.

2.6. The inverter model

The role of the inverter is to keep the voltage constant on the AC side, i.e., at the rated voltage, and to convert the input power P_{in} into the output power P_{out} with the best possible efficiency η_i as described by (13) [19]:

$$\eta_i = \frac{P_{out}}{P_{in}} = \frac{V_{AC}I_{AC}cos\varphi}{V_{DC}I_{DC}}, \tag{13}$$

where I_{DC} is the current required by the inverter from the DC source to be able to keep the rated voltage on the AC side, V_{DC} is the input voltage to the

inverter delivered by the DC source (PV panel or battery), V_{AC} and I_{AC} are the output voltage and current, respectively, and $cos\varphi$ can be obtained from the inverter's data-sheet.

3. Automated Verification Using Model Checking

Validation is the process of determining whether a design meets the user requirements, whereas verification is the process of determining whether a design meets a set of requirements, specifications, and regulations [10]. If the requirements, specifications, and regulations are given in a formal language, then it may be possible to automate the verification process, thus resulting in a process known as *formal verification*. Verification may form part of a validation process. While simulation and testing explore some of the possible behaviors and scenarios of the system, leaving open the question of whether the unexplored trajectories may contain a flaw, formal verification conducts an exhaustive exploration of all possible behaviors. Thus, when a design is pronounced correct by a formal verification method, it implies that all behaviors have been explored, and the questions of adequate coverage or a missed behavior becomes irrelevant [26].

Formal verification is a systematic approach that applies mathematical reasoning to obtain guarantees about the correctness of a system; one successful method in this domain is model checking [26].

The process of model checking can be split into three main components: modeling, specification, and verification method. In modeling, a model (normally mathematical) of the system is created; in specification, normally a list of properties to be satisfied by the system is established, i.e., the requirements, normally expressed in a temporal logic form as Computational Tree Logic (CTL) or Linear Temporal Logic (LTL). The model checking algorithm can be described as [10]:

- Given the model M and a CTL (or LTL) formula ϕ as input;

- Model checking algorithm provides all the states of model M which satisfies ϕ;
- It returns *YES* if ϕ is *TRUE*, or returns *NO* if ϕ is *FALSE*.

Specifically for the *FALSE* verification result, the algorithm returns a *counterexample* (i.e., a sequence of states that leads to a property violation), which is useful as diagnostic of the system to discover in which situation the model is violated; this is the most important advantage of the use of model checking [10]. Fig. 3 shows a real PV system, and Fig. 4 the conversion the system to be verified by a model checking procedure.

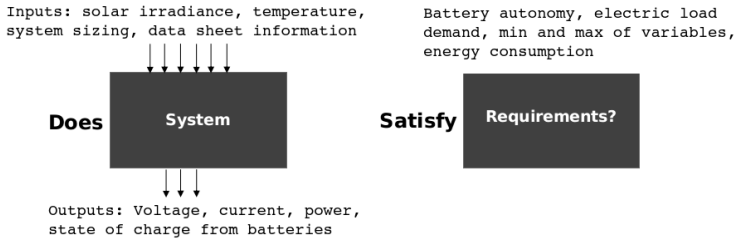

Figure 3: Real system verification (adapted from [10]).

Figure 4: Model checking verification [10].

However, there is a main disadvantage of model checking: the state explosion problem. In order to tackle this problem, many different techniques were developed in the last decades. One of the first major advances was symbolic model checking with binary decision diagrams (BDDs). In this approach, a set of states is represented by a BDD instead of by listing each state individually, which is often exponentially smaller in practice. Another promising approach to overcome state explosion problem is Bounded Model Checking (BMC) [27].

BMC is a method that checks a model up to a given path in the path length. BMC algorithms traverse a finite state machine for a fixed number of steps, k, and checks whether a property violation occurs with this bound. It uses Boolean Satisfiability (SAT) or Satisfiability Module Theories (SMT) solvers to check the generated formula from BMC.

SAT problem is a problem of determining whether there are certain conditions or interpretations that satisfy a given Boolean expression [10]. SMT decides the satisfiability of a fragment of first-order formulae using a combination of different background theories and thus generalizes SAT by supporting uninterpreted functions, linear and non-linear arithmetic, bit-vectors, tuples, arrays, and other decidable first-order theories [10]. The SAT or SMT solvers search the model for conditions (value of variables) that make the formula satisfiable. If a SAT or SMT solver finds a substitution for the formula/function then the substitute induces a counterexample and is said to be *satisfiable*, i.e., it is satisfiable *iff* the verified system contains errors. ESBMC is one of the most representatives bounded model checkers for embedded C/C++ software based on SMT solvers [12]. ESBMC comes as an alternative to overcome limitations of the system modeling, especially considering that the system complexity is increasing and SMT has richer theories than SAT to represent models.

3.1. ESBMC

ESBMC (or Efficient SMT-based Bounded Model Checker) is an open source, permissively licensed (Apache 2), cross platform bounded model checking for C and C++ programs [12], which supports the verification of LTL properties with bounded traces [28]. ESBMC's verification flow can be summarized in three stages: (i) a front-end that can read and compile C/C++ code, where the formal specification of the system to be verified is first handled; (ii) preprocessing steps to deal with the representation of the code, control flow and unwinding of loops, and the model simplification, thereby aiming to reduce the verification effort; and finally (iii) the SMT solving stage, where all the constraints and properties of the system to be verified are encoded into SMT and checked for

satisfiability. If the SMT formula is shown to be satisfiable (SAT), a counterexample is presented; otherwise, the formula is unsatisfiable (UNSAT), i.e., there are no errors up to the given unwinding bound.

ESBMC exploits the standardized input language of SMT solvers (SMT-LIB[1] logic format) to make use of a resource called *assertion stack*. An assertion, in SMT solvers, is a constraint over the variables in a formula that must hold if the formula is satisfiable [29]. New assertions can be added to or old assertions removed from this stack, depending on the evaluated value of variables. This enables ESBMC, and the respective solver, to learn from previous checks, optimizing the search procedure and potentially eliminating a large amount of formula state space to be searched, because it solves and disregards data during the process, incrementally. This technique is called "incremental SMT" [30] and allows us to reduce the memory overhead, mainly when the verified system is complex and the computing platform does not have large amount of memory to deal with all the design space state.

4. Model Checking Stand-alone Solar Photovoltaic Systems

The flowchart of the proposed automated verification method is illustrated in Fig. 5.

In Step 1, the PV input data (e.g., load power demand and load energy consumption) and the formulae to check the sizing project, the mathematical model, the limits of the weather non-deterministic variables, are all written as an ANSI-C code [31]. In Step 2, the sizing check of the PV system takes place to make sure that the components were selected according to critical month method [25]. The equations to check the PV sizing were not presented at this paper, however the standard from Engineering Manual was the reference [25]. In Step 3, weather variables (e.g., solar irradiance and ambient temperature) will be systematically explored by our verification engine based on maximum

[1] http://smtlib.cs.uiowa.edu/

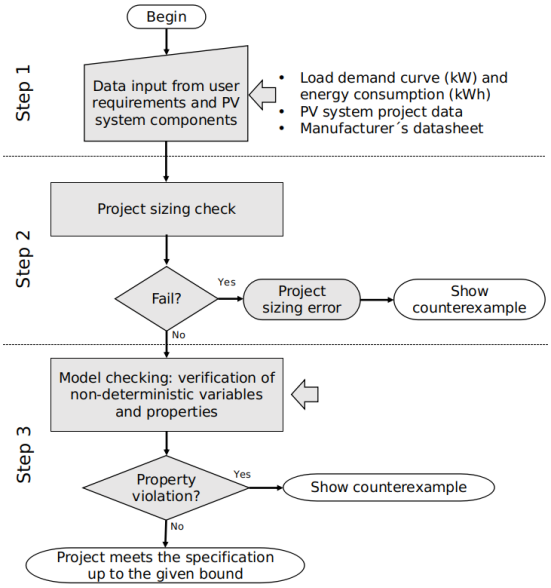

Figure 5: Flowchart of the proposed automated verification of PV systems.

and minimum values from the site, where the PV system will be deployed. In addition, depending on one of the desired properties of the system such as battery autonomy, energy availability, or even system power supply, our verification engine is able to indicate a failure if those properties are not met; in this particular case, it provides a diagnostic counterexample that shows in which conditions the property violation occurred.

In a nutshell, ESBMC will process the ANSI-C code with constraints and properties from the PV system that are provided by the user, and the tool will automatically verify if the PV system requirements are met. If it returns a failure (i.e., SAT), then the tool provides a counterexample, i.e., a sequence of states that leads to the property violation; this information can be used as a feedback to improve the PV system design. However, if the verification succeeds (i.e., UNSAT), there is no failure up to the bound k; therefore, the PV system will present its intended behavior up to the bound k.

Algorithm 1 describes the pseudo-code used to perform the automated verification. The batteries are assumed to have SOC of 100% (Line 1), i.e., the

batteries are charged. That is important because the system is said ready to use when the batteries are charged.

Information related to average temperature (T) and solar irradiance (G), maximum and minimum annual, are given to the algorithm in Lines 7 to 10 using non-deterministic variables from ESBMC to explore all possible states and the *assume* macro to constrain the non-deterministic values using a given range. In order to reduce the computational effort of the algorithm, every 24h-day was considered as a time-step of 1 hour, and it was split into two parts: (a) one where it is possible to occur PV generation, during daylight, with a duration in hours depending on each site (but dependent on the sun and weather conditions); and (b) one that includes all the remaining day (without any PV generation). Therefore, our approach depends on specific data about the solar irradiation levels to define the average amount of hours of PV generation.

After that, the model from PV generator is used in the function call of Line 7, to produce the voltage and current considering the states of G and T. With respect to every hour considered, the conditional *if-else-endif* statements from Lines 8, 11, 13, 16 and 19, will perform the charge or discharge of batteries according to the value of different variables: if there is PV generation (which depends on G and T), the updated state of charge from batteries, the house's load and the set-up information of the PV system.

Next, representing the time of the day where PV generation is not possible anymore, starting in Line 22, the algorithm will only discharge the batteries (using the 1 hour step) until a new charging process (at the next day) starts. Specific *assert* in Line 26 will check the state of charging from batteries, and they will violate the property if their levels reach the minimum that represents a discharged battery; therefore, the PV system is unable to supply energy to the house. Nevertheless, if the verification engine does not fail, then we can conclude that the PV system does not need further corrections up to the given bounds.

Algorithm 1 Model checking algorithm for stand-alone PV

1: $SOC \leftarrow 100\%$

LOOP Process

2: **for** $h = 1$ to $Hours\,of\,PV\,generation$ **do**

3: $G \leftarrow nondet\_uint(\,)$ {G is non-deterministic variable}

4: $T \leftarrow nondet\_uint(\,)$ {T is non-deterministic variable}

5: assume $(Gmin \leq G \leq Gmax)$ {restricting G values}

6: assume $(Tmin \leq T \leq Tmax)$ {restricting T values}

7: $Imax, Vmax \leftarrow PV\,generation MODEL(G, T)$

{If-then-else sequence to imitate charge controller work}

8: **if** ($battery\,is\,empty$) AND ($PV\,is\,generating$) **then**

9: charge battery in 1h

10: PV feed the house

11: **else if** ($battery\,is\,empty$) AND NOT($PV\,is\,generating$) **then**

12: FAIL with assert macro

13: **else if** NOT($battery\,is\,empty$) AND ($PV\,is\,generating$) **then**

14: stop battery charge

15: PV feed the house

16: **else if** NOT($battery\,is\,empty$) AND NOT($PV\,is\,generating$) **then**

17: discharge battery in 1h

18: Battery feed the house

19: **end if**

20: $h \leftarrow (h + 1)$

21: **end for**

Start of battery autonomy verification:

22: $AutonomyCount \leftarrow 1$

23: **while** $AutonomyCount \leq autonomy$ **do**

24: $SOC \leftarrow SOC - (24 - Hours\,of\,PV\,generation)\,h\,discharge$

25: $AutonomyCount \leftarrow AutonomyCount + 1)$

{autonomy verification during discharge period}

26: assert NOT($Battery\,empty$)

{Perform similar **for**-LOOP as defined in line 2-21}

27: **end while**

28: **return** ( )

5. Experimental Evaluation

5.1. Description of the Case Studies

We have performed five case studies to evaluate our proposed verification method: (a) four PV systems (three in series 325W PV panels, four 220 Ah batteries in a configuration with two series and two parallel with 48h autonomy, 700 W inverter with surge of 1,600W, charge controller with MPPT with 35A/150V of capacity) deployed in four different houses in an indigenous community (GPS coordinates 2^o44'50.0"S 60^o25'47.8"W) situated nearby Manaus (Brazil), with each house having a different power demand (house 1 = 253 W, house 2 = 263 W, house 3 = 283 W, and house 4 = 501 W); and (b) one case concerning a system deployed as an individual system in Manaus (GPS coordinates 3^o4'20.208"S 60^o0'30.168"W), supporting 915 W of the house's load (house 5 with four 325W PV panels in a configuration two series and two parallel, four 120Ah batteries in series and autonomy of just 6 h, 1,200 W inverter with surge of 1,600 W, charge controller with MPPT of 150V/35A).

Note that the annual average temperature (T) in Manaus is from 23^oC to 32^oC; and irradiance (G) varies from 274 W/m^2 to 852 W/m^2 when there is sunlight (that information is provided in Lines 5 and 6 of Algorithm 1). Another characteristic of Manaus, based on historical weather data [32], is related to the fact that only during 8 hours of the day is possible to have PV generation, from 8:00h to 16:00h (that information is provided in Algorithm 1 as well).

5.2. Objectives and Setup

Our experimental evaluation aims to answer two research questions:

RQ1 **(soundness)** Does our approach provide correct results?

RQ2 **(performance)** How does our approach compare against other existing tools?

All experiments were conducted on an otherwise idle Intel octa core Xeon CPU E5-2640 with 2.4 GHz and 264 GB of RAM, running Ubuntu 18.04.1 LTS

64-bits. Concerning our verification engine, ESBMC v6.0.0 was used [2]. We have also used the SMT solver Boolector version 2.4.1 [33].

In order to evaluate the proposed verification method and its performance, we have considered five case studies and also compared its performance to the HOMER Pro tool. Experimental setup of HOMER Pro: Intel Core i5-4210 (4-cores), with 1.7 GHz and 4 GB of RAM, running Microsoft Windows 10; we have used HOMER Pro v3.12.0.

5.3. Results

The description of our experimental results can be broken down into two parts: (a) the 1,200 W PV system (house 5) failed during the sizing check since the number of panels was *incorrectly* sized; in particular, the counterexample provided by our verification engine indicated 3 PV panels in parallel and the actual project has 2 in series and 2 in parallel. This verification took approximately 0.57 hours to be performed. In order to validate this possible flaw, it was surveyed the owner of the 1,200 W system; we identified that, in fact, the system does not meet the battery autonomy most of the time (mainly when all loads are turned on), and this was double checked with the monitoring system from the charge controller from the manufacturer who showed that the maximum power or surge power were not exceeded, thus affirming RQ1; this behavior is expected since the system was purchased as an off-the-shelf solution and not as a customized design for the electrical charges of the house; (b) For the 700 W PV systems of houses 2, 3, and 4, the sizing check was successful during verification, but our verification engine has found flaws related to the battery autonomy, particularly when SOC reached a empty-battery level. Our verification engine identified those flaws (for those three houses) right after the first night-discharge cycle, i.e., before the solar system started to recharge the batteries. Our verification engine took approximately 5.1 hours to find this flaw

[2]The command-line used to perform the verification is: $ esbmc filename.c --no-bounds-check --no-pointer-check --no-div-by-zero-check --unwind 300 --boolector

in house 2; 5.8 hours for house 3, and 4.4 hours for house 4. These flaws were confirmed with the dwellers who own the systems by an interview process and from a local monitoring system with the Raspberry Pi controller: at least once a month is usual the system to turn off, normally in raining or clouds days, thereby reaching the situation described in Step 3 of Table 2, further affirming RQ1; after the sun rises, the systems returns to normal condition operation. One more time the flaws were not caused by misuse of the system by the dwellers. Fig. 6 shows the monitoring result for the load curve of house 2 from June to July of 2018 (5 weeks), without undesired peaks from power demand of even unexpected energy consumption. The house 1 had an inconclusive result with the automated verification, because the solver did not presented any result from the SMT verification.

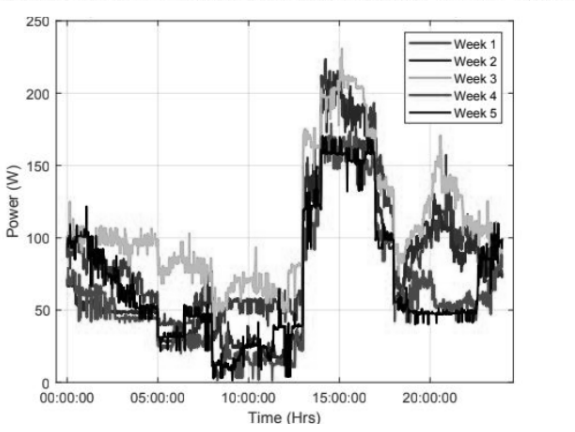

Figure 6: Five weeks monitored load curve from House 1.

The same five case studies were evaluated by HOMER Pro. The simulation results showed that the project restrictions were met by four 700 W PV systems (house 1, 2, 3 and 4), without any indication of sizing error or even performance related issues. The case study that was unsuccessful during simulation was the 1,200 W (house 5); however, without any indication about the failures of this PV system. All the simulations took less than 5 seconds (each) to be performed by HOMER Pro. Fig. 7 shows one of the screens presented by HOMER Pro

software, specifically to house 1.

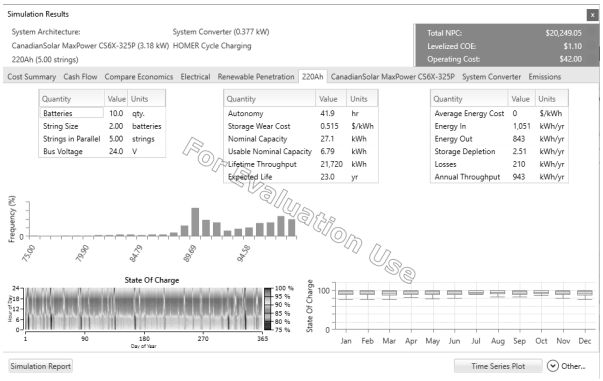

Figure 7: Homer simulation screen.

Despite the divergence of results for the houses 2, 3 and 4 w.r.t. our proposed approach, it is evident that the information collected from the dwellers and from the monitoring systems indicate that our approach provides the correct evaluation of the PV system, thus answering RQ2. House 5 presented flaws from both tools; however, only our approach indicated which design error was responsible for the flaw (number of PV panels), further answering RQ2.

5.4. Threats to Validity

We have reported a favorable assessment of the proposed method. Nevertheless, we have also identified three threats to the validity of our results that can further be assessed.

Model precision: each component of the PV system is mathematically modeled. A careful evaluation in a PV laboratory to validate the model could add more reliability to the results produced by our method.

Time step: The run-time complexity of our proposed method is an issue; the time step of one hour can be further reduced to approximate the algorithm to the real-world scenario, where a solar irradiance and ambient temperature can change in fractions of minutes.

Case studies: Our case studies are performed only in one municipality. A

more complete evaluation can be performed if other places around the world could become case studies.

6. Conclusion and Future Work

We have described and evaluated an automated verification method to check whether a given PV system meets its specification using software model checking techniques. We have considered five case studies from real photovoltaic systems deployed in five different sites, ranging from 700 W to 1,200 W. Although this verification method takes longer than simulation methods, it is able to find specific conditions that lead to failures in a PV system previously validated by a commercial simulation tool. In particular, the proposed method was successful in finding sizing errors and indicating in which conditions a PV system can fail. As future work, the proposed method will be extended to start from a list of commercial equipment, where each equipment is verified and the final solution, which satisfies the project specification, is found, thus leading to an optimum sizing of PV systems. We will also consider other types of renewable energy and even hybrid ones to allow our method to design and verify typical rural electrification.

Acknowledgments

The authors would like to thank Newton Fund (ref. 261881580) and Coventry University for the availability of the case studies (real PV systems) to validate the tool and to support the mobility of the researcher. FAPEAM (Amazonas State Foundation for Research Support, call 009/2017 and PROTI Pesquisa 2018), for the financial support related to the Virtual Machine rent, to the one-month PhD scholarship, and to support the mobility of the researcher. And Sustainable Amazonas Foundation (FAS) for the PhD scholarship.

References

[1] S. Coelho, A. Sanches-Pereira, L. Tudeschini, J. Escobar, M. Poveda, N. Coluna, A. Collin, E. L. Rovere, A. Trindade, O. Pereira, Biomass residues as electricity generation source in low HD source in regions of Brazil, in: UNESP (Ed.), The XI Latin. Cong. of Elec. Gener. and Transm. CLAGTEE, 2015, pp. 1–8. doi:10.13140/RG.2.1.2747.7208.

[2] EPIA, Global market outlook for photovoltaics 2017-2021, European Phototoltaic Industry Association, Belgium, 2017.

[3] S. Pradhan, S. Singh, M. Choudhury, D. Dwivedy, Study of cost analysis and emission analysis for grid connected PV systems using RETSCREEN 4 simulation software, Int. J. of Eng. Res. & Tech. 4 (4) (2015) 203–207.

[4] N. Swarnkar, L. Gidwani, R. Sharma, An application of HOMER Pro in optimization of hybrid energy system for electrification of technical institute, in: Int. Conf. on Energ. Eff. Tech. for Sust. (ICEETS), 2016, pp. 56–61. doi:10.1109/ICEETS.2016.7582899.

[5] A. Dobos, PVWatts Version 5 Manual, Tech. rep., National Renewable Energy Laboratory, Colorado (2014).

[6] N. Blair, A. Dobos, J. Freeman, T. Neises, M. Wagner, System Advisor Model, SAM 2014.1.14: General Description, Tech. rep., National Renewable Energy Laboratory, Colorado (2014).

[7] A. Mills, S. Al-Hallaj, Simulation of hydrogen-based hybrid systems using Hybrid2, Int. J. of Hydrog. Energy 29 (10) (2004) 991–999. doi:10.1016/j.ijhydene.2004.01.004.

[8] J. Gow, C. Manning, Development of a photovoltaic array model for use in power-electronics simulation studies, in: Proceedings of the 14th IEE Electric Power Applications Conference, Vol. 146, 1999, pp. 193–200. doi:10.1049/ip-epa:19990116.

[9] A. Benatiallah, D. Benatiallah, T. Ghaitaoui, A. Harrouz, S. Mansouri, Modelling and simulation of renewable energy systems in Algeria, Int. J. of Sc. and App. Inf. Tech 7 (1) (2017) 17–22.

[10] E. M. Clarke, T. A. Henzinger, H. Veith, Introduction to model checking, in: Handbook of Model Checking., 2018, pp. 1–26. doi:10.1007/978-3-319-10575-8_1.

[11] W. Akram, M. A. Niazi, A formal specification framework for smart grid components, Complex Adaptive Systems Modeling 6 (1) (2018) 5. doi:10.1186/s40294-018-0057-3.

[12] M. Gadelha, F. Monteiro, J. Morse, L. Cordeiro, B. Fischer, D. Nicole, ESBMC 5.0: An industrial-strength C model checker, in: 33^{rd} ACM/IEEE Int. Conf. on Aut. Soft. Engin. (ASE'18), ACM, New York, NY, USA, 2018, pp. 888–891. doi:10.1145/3238147.3240481.

[13] E. Natsheh, A. Albarbar, Solar power plant performance evaluation: simulation and experimental validation, in: J. of Physics: Conf. Ser., Vol. 364, 2012. doi:10.1088/1742-6596/364/1/012122.

[14] M. Lamnadi, M. Trihi, A. Boulezhar, Study of a hybrid renewable energy system for a rural school in Tagzirt, Morocco, in: Int. Ren. and Sust. Energ. Conf. (IRSEC), 2017. doi:10.1109/IRSEC.2016.7984079.

[15] Y. Driouich, M. Parente, E. Tronci, Modeling cyber-physical systems for automatic verification, in: 14th Int. Conf. on Synth., Mod., Appl. to Circ. Des. (SMACD 2017), 2017, pp. 1–4. doi:10.1109/SMACD.2017.7981621.

[16] A. Abate, Verification of networks of smart energy systems over the cloud, in: S. Bogomolov, M. Martel, P. Prabhakar (Eds.), Num. Soft. Verif., Vol. LNCS 10152, 2017, pp. 1–14. doi:10.1007/978-3-319-54292-8.

[17] Y. Driouich, M. Parente, E. Tronci, A methodology for a complete simulation of cyber-physical energy systems, in: IEEE Work. on Envir., En-

erg., and Struc. Monit. Syst. (EESMS), 2018. doi:10.1109/EESMS.2018.8405826.

[18] P. Mohanty, K. Sharma, M. Gujar, M. Kolhe, A. Azmi, Solar Photovoltaic System Applications, Springer International Publishing, 2016, Ch. PV System Design for Off-Grid Applications, pp. 49–83. doi:10.1007/978-3-319-14663-8.

[19] A. Hansen, P. Sørensen, L. Hansen, H. Bindner, Models for a stand-alone PV system, no. 1219, Forskningscenter Risoe, 2001.

[20] E. Saloux, A. Teyssedou, M. Sorin, Explicit model of photovoltaic panels to determine voltages and currents at the maximum power point, Solar Energy 85 (5) (2011) 713–722. doi:10.1016/j.solener.2010.12.022.

[21] J. Cubas, S. Pindado, F. Sorribes-Palmer, Analytical calculation of photovoltaic systems maximum power point (MPP) based on the operation point, Applied Sciences 7 (9) (2017) 870–884. doi:10.3390/app7090870.

[22] M. Villalva, J. Gazoli, E. Filho, Comprehensive approach to modeling and simulation of photovoltaic arrays, IEEE Trans. on Power Elec. 24 (2009) 1198–1208. doi:10.1109/TPEL.2009.2013862.

[23] R. Ross, Flat-plate photovoltaic array design optimization, in: C. San Diego (Ed.), 14th IEEE Photovoltaic Specialists Conference, 1980, pp. 1126–1132.

[24] J. Copetti, E. Lorenzo, F. Chenlo, A general battery model for PV system simulation, Prog. in Photovoltaics: Res. and App. 1 (4) (1993) 283–292. doi:10.1002/pip.4670010405.

[25] J. Pinho, M. Galdino, Manual de Engenharia para Sistemas Fotovoltaicos, CEPEL CRESESB, Rio de Janeiro/RJ, 2014.

[26] E. Clarke, W. Klieber, Miloš Nováček, P. Zuliani, Model Checking and the State Explosion Problem, Springer, Berlin, 2012, pp. 1–30. doi:10.1007/978-3-642-35746-6_1.

[27] A. Biere, A. Cimatti, E. M. Clarke, Y. Zhu, Symbolic Model Checking without BDDs, in: TACAS, Vol. 1579 of LNCS, 1999, pp. 193–207.

[28] J. Morse, L. C. Cordeiro, D. A. Nicole, B. Fischer, Model checking LTL properties over ANSI-C programs with bounded traces, Software and System Modeling 14 (1) (2015) 65–81. doi:10.1007/s10270-013-0366-0.

[29] J. Morse, Expressive and efficient bounded model checking of concurrent software, Ph.D. thesis, University of Southampton (2015).

[30] P. Schrammel, D. Kroening, M. Brain, R. Martins, T. Teige, T. Bienmüller, Incremental bounded model checking for embedded software, Formal Asp. Comput. 29 (5) (2017) 911–931. doi:10.1007/s00165-017-0419-1.

[31] International Organization for Standardization, ISO/IEC 9899:2018 Information Technology – Programming Languages – C, acessed: 14.11.2018. URL https://www.iso.org/standard/74528.html.

[32] EnergyPlus, Weather data by location, acessed: 09.07.2018. URL https://goo.gl/A82Jqh

[33] R. Brummayer, A. Biere, Boolector: An efficient SMT solver for bit-vectors and arrays, in: Tools and Alg. for the Const. and An. of Sys. (TACAS), Vol. LNCS 5505, 2009, pp. 174–177. doi:10.1007/978-3-642-00768-2_16.