We want to thank the opportunity to answer the questions asked by the reviewers and the suggestions for the article submitted to CAV 2020. It is always an opportunity for self-reflection and improvement of the research. We split the answer into two parts, one related to some errors of understanding that we consider having been practiced by reviews 3 and 4, and a second part containing answers to questions, which can help clarify some questions.

**PART I -** We would like to highlight that reviews 3 and 4 contain significant misreadings on the results, scope, and contributions of our paper; we provide evidence of factual errors and ask the Program Chairs to discard them.

***"CBMC is able to generate an optimal configuration for all but one of the settings"*** -- Review 3 overlooks Section 4.4, where we describe: *"CPAchecker was able to synthesize the optimal sizing in six out of seven case studies (cases 1 to 6)"*. In particular, CBMC is unable to produce any conclusive results, as stated in Section 4.4, as follows: *"CBMC and ESBMC are unable to produce any conclusive results since time-outs or memory-outs occurred."*

***"The specific domain of solar photovoltaics is outside my area… One potential reason to accept this at CAV would be if the technique involved some novelty..."*** -- Review 3 overlooks the scope of our paper submitted to *"Industrial Experience Reports and Case Studies"*. The CfP explicitly states that: *"These papers are expected to describe the use of formal methods techniques in industrial settings or in new application domains. Papers in this category do not necessarily need to present original research results but are expected to contain novel applications of formal methods techniques as well as an evaluation of these techniques in the chosen application domain"*. Review 1 points out *“I appreciate the application of formal methods in industrial applications…, and believe that the paper... could be a nice addition to CAV”* and review 2 *"The paper is one of the first to apply formal methods to PV system design... I think it's fantastic to see researchers applying formal methods in areas related to renewable energy"*.

***"First, the target problem here is one of optimization, but the synthesis algorithm does not model the optimality objective explicitly, instead relying on standard CEGIS."*** -- Review 4 overlooks Section 2.1, where we describe that we do not rely on standard CEGIS: *"In our CEGIS variant, there exist four differences related to the traditional one..."*. In this section, we provide details of the main differences of our approach to standard CEGIS. Besides, we model the optimality objective explicitly as stated in Section 3.: *"The optimal sizing of PV systems is made by the best compromise between two objectives: power reliability and system cost..."*. As described in Section 2.1, the VERIFY phase is responsible for obtaining the lowest cost (minimum cost is the optimization objective function).

***"The paper claims… more accurate results than existing commercial tools…*** ***this claim is not validated..."*** Review 3 overlooks EG3 (cf. Section 4). We report that HOMER Pro does not use costs related to charge controllers, which were introduced into our synthesis model. This charge controller cost makes the synthesis approach more precise and real. We observed that there exists a divergence in case study 2, where the costs presented by HOMER Pro were 54% higher than our synthesis tool.

**PART II -** We present below some answers to the questions asked by all reviewers to explain better the method, the results, or even the research that resulted in this application with case studies.

Review 1

***"Isn't CBMC a model checker for C/C++? Can't you use directly an SMT solver like Z3?"*** --

When we started this research, we tried to use SMT solvers directly using the SMT-lib format, but they have all struggled with the discharged verification conditions (VCs) resulting from Algorithm 1 (cf. Section 3); in particular, we tried the SMT solvers Z3 and Boolector. State-of-the-art software model checkers are doing an excellent job of simplifying the VCs before invoking the underlying SMT solvers. Since we also wrote Algorithm 1 in the C programming language, we were able to find one software model checker (CPAchecker), which could deal with the VCs produced from Algorithm 1.

***"Is this optimization problem beyond the scope (or ability) of probabilistic programming?"*** -- Unfortunately, we have not investigated the application of probabilistic programming yet within our research. We are aware that using this method we can work even with incomplete models, however the answer or solution presented by the technique could be a local optimal and not to a global optimal; we will address this suggestion for future work in a compatison with other heuristics that can produce fast response (as genetic algorithm, for example), but with limitation about precision. In this future work we can analyse the trade-off between performance and precision.

Review 2

***"Currently, the CEGIS loop results in a \*single input\* being added at each iteration. This may require a large number of iterations for the synthesis component to produce a candidate that succeeds in the verifier. Is there a more \*general\* constraint that the verifier can produce, to help the synthesizer more quickly rule out bad candidates?"*** -- We realized this observation just after the paper submission. In an updated version of the algorithm (not the one presented in this paper), we have included a minimum cost for the beginning of the iterative loop (see Chapter 4, page 75, Algorithm 2 in shorturl.at/nJT78). This cost is the lowest among the list of equipment used, considering a configuration of just one solar PV panel, one charge controller, one battery, and one inverter. Indeed, if we use an extensive list of equipment, then the time to obtain the result tends to be more significant. We will address this comment in a revised version of our manuscript.

***"Where does the verification step occur in Algorithm 1? Unless I'm missing something, the VERIFY and SYNTHESIZE phases don't show up explicitly in Algorithm 1."*** -- The SYNTHESIZE phase is obtained by a possible combination of equipment that meets the properties collected from the electrical requirements of the equipment itself, e.g., power, current, or voltage (cf. Section 3). This phase produces one feasible solution in line 20 of Algorithm 1. The VERIFY phase is responsible for obtaining the lowest cost (minimum cost is the optimization objective function), thereby considering the iterative steps taken from the for-loop, which ranges from line 6 to 22 in Algorithm 1. The code will lead to "FAIL" if the optimal sizing is found or lead to "SUCCESS" if there exists no feasible solution.

***"My understanding is that the verification step uses a model checker. What exactly does the model look like?"*** -- We do use symbolic model checking for our VERIFY phase. As described in Section 3, the model (incl. constraint and properties) is obtained from the mathematical model used for the sizing, from the electrical characteristic of the equipment (to combine electrical compatibility among equipment, as current, voltage, or power), from the “assumes” statements included explicitly in Algorithm 1, and from the objective function (minimum cost of the feasible PV sizing solution). We created a specific page for specifying the resulting VCs (constraints and properties): see <https://drive.google.com/file/d/1ByNJJTd9Xhb1cYijGKpCxLq-fm7jprSG/view?usp=sharing>.

***"In regards to the mismatched Windows/Ubuntu machine specs in section 4.3, why couldn't you just use separate Windows/Ubuntu VMs on the same host machine?"*** The ideal scenario would be to use the same hardware configuration for the experiments. However, we faced restrictions concerning the license for the HOMER Pro; besides, we did not have the autonomy to change the Linux VM machine installed in the servers of our university due to the internal policy.

Review 4

**“the application demands reasoning about constraints in nonlinear real arithmetic, which is a hard problem. It is not clear how this problem will be solved.”**– At the end of Section 2.2, we describe that all the provided equations model the continuous-time behavior of the PV system; they produce real numbers except for the batteries and panels, where real numbers must be converted into integer ones, considering the minimum or maximum according to each equation. The underlying symbolic verifiers perform bit-precise verification based on the Floating-Point theory using the command-line provided in the footnotes of page 8, which allows us to handle nonlinear real arithmetic.