Rebuttal Letter

1 Review #1

CGP is a specific field of investigation which should deserve a deeper illustration for the sake of reader's comprehension. On the contrary, Sec. 2 is hastily sketched and several points remain unclear. Some examples: - How are the nodes formalized and in which way are the functions (which functions?) and connections represented into the genes? - Figure 1 needs some clarifications: what is the meaning of the "f=" pieces of information reported into the nodes? - What about population initialization, parent selection, stop criteria? - Some part of the text are somewhat misleading: "inactive nodes" are mentioned before their actual definition; "gate" (logic gates?) are abruptly introduced even if the employment of CGP in the field of combinational circuits design has not been still announced.

The CGP section was more detailed and the highlighted points were clarified and explained in the text and in the images. In addition, a CGP pseucode was added.

The following Sec. 2.1 presents some problems. First of all, a major part of the text is copied--pasted from ref.[18]. Actually, such a bibliographical reference is mentioned by the authors, but that should not allow to reproduce the very same text. More importantly, the authors produce a kind of extract (realized as a collage from [18]), thus compromising the overall comprehension of the text. Some examples: - "variables" (logical variables?) are not properly introduced; - "ascending order" is undefined; - "Sat-Count" function (misspelled at start) is undefined; - the authors claim they used the "Buddy Package"; however, no information is provided about its authorship and the URL referring to the package (reported in the footnote) is broken. That is quite strange, since it appears that the URL has been modified some years go, as simply revealed by a quick web search; - "fitness" and "initialization of the program" are abruptly introduced without descending into details and the mentioned "desired circuit" should refer (maybe?) to the truth table, but the overall argumentation is hurried and unclear.

The BDD section has been completely rewritten. The link to Buddy has been fixed and the reference added. The questions about "fitness" and "initialization of the program" was added and explained in CGP section.

Sec. 3 introduces the proposed approach, namely the GAM operator. The authors specify the general algorithm of such an operator, but they do not provide any low-level detail. How is the mutation implemented in practice? That is an unanswered question, and this kind of perplexity is related to the previously mentioned absence of the node formalization (mutation is supposed to be applied on nodes, but the reader is still unaware of their format). A sentence appears to be quite relevant, but it has been formulated in an obscure way (I am referring to "Furthermore, in cases that outputs share..."), so that I have the feeling to be missing something interesting about Figs. 2(c)-2(d). Also, at the end of the section the possibility to combine SAM and GAM operators is hinted as a minor afterthought. Instead, it is quite a relevant position, as I am going to discuss later.

The way in which GAM works were detailed in the paper, including technical information as the use of arrays to store the desired information (such as the number of matches with respect to the truth table). Furthermore, it is clarified that the mutation is performed on nodes and these nodes compose the subgraphs that are analyzed by the proposed GAM. Figure 2 was re-explained. The section "The Proposed Approach" was rewritten to clarify the proposed approaches: the GAM operator and the combination of both operators. They are two distinct proposed approaches. In this way, the individual characteristics of the proposals are detailed in their respective sections.

The fourth section is devoted to the presentation of the experimental results, and here the major perplexities of mine are lying. Some examples: - As far as I can see, the number of Gates reported in Table 2 should come from the LGSynth'91 document whose URL is reported in a footnote. However, I cannot find any trace of the reported "Gates" values into the referred pdf

In fact the information of the number of gates is not provided by LGSynth91' but by RevLib. Links and references have been fixed.

- "Balancing" and "Simplification" appear to be some key-elements to conduct the experiments. However, they are briefly introduced, without any hint about their usefulness in this context. In other words, what does the Balancing measure actually evaluate?

The balancing and simplification rates were detailed in the text and their importance reflects on the number of objective function evaluations allowed for the problems.

- Is the number of objective function evaluations related to some stop criteria of the algorithm? In which way? What is the sense of the

imposed numbers (800,000; 1,600,000) related to some (apparently) arbitrary ranges?

The stopping criteria consists of obtaining a feasible solution or reaching the maximum number of objective function evaluations. This was highlighted in the text. In addition, values of 800,000 and 1,600,000 were determined empirically.

- As far as I can see, SAMGAM simply consists in applying both the SAM and the GAM operators. In other words, it is not a matter of support, i.e. the GAM operator is not applied to aid SAM in its operation: they are simply applied together (i.e., in the overall genetic engine) separately (i.e., they do not influence each other). If this is the case, what is the rationale behind this kind of approach? Hadn't the GAM operator been introduced as an alternative for SAM? How can we manage the combination of both operators during the experimentation? Without further information, I would argue that the obtained SAMGAM results cannot be properly interpreted: the optimization may come from SAM or GAM, without the possibility to understand their specific contributions. This concept will come again in what I am going to comment shortly.

The SAM and GAM operators are actually applied separately. Each mutation operator acts in a different region of the search space. In view of the questioning about which mutation operator is actually leading to obtain the good results, new experiments were performed to prove that the GAM actually assists the evolutionary process.

- Table 3-4 report the numerical results. The application of the proposed operator (GAM) is not convincing since the results obtained over the explored datasets are far from encouraging. All in all, SAM is still "the only method capable of achieving 100% success rate on all problems": that appears to be quite a relevant remark. Most notably, in several cases GAM is far from those percentage values (by the way, I cannot see the point to include in the tables some datasets with missing values related just to the proposed approach!). To a certain extent, the reader is led to believe that resorting to SAMGAM is an attempt to improve the overall results. Indeed, SAMGAM works generally better than GAM, but I should recall what I was commenting before: I suspect that it is the SAM component which is working in SAMGAM (instead of the GAM one), since they are applied together separately, as I was arguing in the previous point. If there are some replies to these kind of perplexities, I was not able to find them in the paper. - As a final remark, it should be highlighted the notable difference between SAM and GAM: SAM leads to mutation of inactive nodes, while GAM doesn't. This implies a reduction of the

search space for GAM (it is somewhat admitted by the authors themselves when they talk about the Neutral Genetic Drift in SAMGAM, which of course it's all thanks to SAM). In this sense, the unsuitability of the proposed operator was to be expected from the very beginning. Of course, in principle GAM could be able to introduce some different kind of advantages (maybe related to the efficiency of computation), but I believe that efficacy is of paramount importance in this context. In any case, these specific issues were not adequately addressed by the authors.

As highlighted throughout the text, SAM is responsible for achieving high success rates and GAM by reducing the number of objective function evaluations required to obtain a feasible solution. Presenting results in which GAM does not find solutions serves as a basis for comparison for the next proposal (SAM + GAM combination) and that this combination actually achieves the expected objectives (increase success rate and reduce number of evaluations). Again, the belief that SAM is the component responsible for the good results obtained by SAMGAM is refuted in the new experiments added to the paper where the number of offspring generated by SAM and GAM are varied. In this way it is possible to notice that the use of GAM actually helps in reducing the number of evaluations of the objective function mainly in the major problems presented.

2 Review #3

The paper is a little confuse. The technique is well described, but in the abstract it says "The main advantages observed are the reduction of the number of objective function evaluations required to find a feasible solution and the improvement in the success rate." Nevertheless, in the conclusions section, it says "SAM was the only method capable of achieving 100% success rate on all problems tested here."

The abstract has been rewritten in order to differ which method decreases the number of evaluations and which method is best at success rate. The same is done in the conclusions.

It would be good a deepest study and characterization of kinds of instances in which one approach or the other work better, with some kind of theoretical arguments supporting the experimental results.

The good results are highlighted in the conclusions and discussion of results, according to the characteristics of the problems used (balancing and simplification).

The state of the art and bibliography seem biased to this kind of heuristic approaches. Are there other techniques for circuit design?

The focus is the evolutionary design of digital circuits so the literature is more focused on heuristic approaches. However, the first section introduces traditional design methods: Karnaugh maps and ESPRESSO for simplification.
Also, the writing errors were corrected.

3 Review #4

An effet is made to illustrate the problem / method. It is a good point. However, for those who are not familiar with Cartesian Genetic Programming, Figure 1 should be still more explained. What does represent 'f=800' in nodes? How can be defined the best individual. This is not well explained.

The text relating to the image was re-made explaining all the details and the CGP was detailed in its respective section. In addition, the way in which CGP determines the best individual has been highlighted, as well as explaining how we determine the best individual in our specific application case.

I part 2, a paragraph is dedicated to Binary Decision Diagrams. The transition to this paragraph is not motivated. We do not know why it is presented here. Moreover, it this section 2, only one subsection is proposed. The organisation of the section should be revised.

The BDDs section was rewritten and the transition has been motivated.

Page 4, line 17: "Modifications in subgraphs which generate correct outputs will be worst or equal to the preliminary candidate circuit." This statement must be better justified.

Regarding the modifications in the subgraphs that already matches its entire table truth: means that if it already matches the entire table truth any modification will be inutile (the candidate solution still matches the entire truth table) or will worsen the number of matches with respect to the truth table. This makes GAM only where it is needed. This fact was highlighted in the text.

Page 5,line 6: "In generation 1, output O1 is the worst one, as it matches 3 of 8 possible bits of the truth table. The other outputs of this circuit hits 8 and 6 of the values of the truth table." I do not understand the way it is evaluated.

The way in which an individual is evaluated is the amount of matches with respect to the truth table. This was highlighted in the new version.

It is quite strange to label the first Input (output) node I0 (O0). It should be I1? Then the second node would be I2 rather than I1 !

Although in the engineering field we denote the indexes starting with 1, in programming languages it is common to use the first index as 0.

Also, the writing errors were corrected.

4 Review #6

The paper is so full of specialist application field acronyms that it is hard to read and follow.

Most acronyms used are proper to the field and all acronyms are explained when they appear for the first time.

Genetic programming using branching tree structure rather than a matrix structure is much more efficient and accurate. John Koza of Stanford and others did excellent development and applications using these type of algorithms twenty years ago. Why is this paper opting not to even mention these pioneering efforts?

Koza's pioneering efforts were highlighted in the text.