E17361 comments1:

Real root isolation is a fundamental  problem in computational real algebraic geometry.

The authors proposed new methods for computing positive bounds of real roots and

implemented a new tool for real root isolation in C.

The experimentation shows that their tool is much more efficient than state of art implementations in many test cases.

The result certainly deserves publication. Overall, the proof of the theorems and the presentation of the algorithms are sound.

But some clarifications are needed.

\*\*To make the experimental results more convincing, I have the following questions and suggestions

1) The paper says that logcf is much more efficient than other tools. But it is unclear why it is more efficient.

    Is it mainly because of the improved root bound or because you have better implementation techniques?

    Is the improvement related to particular types of polynomials?

    Have you found the cases when your method is less efficient than state of art implementations and why?

    It will be useful to demonstrate the bounds computed by your methods and other methods by benchmark examples.

    If in your implementation, you use the same bound by other tools,  how efficient will it be w.r.t. other tools?

2) for random generated examples, you did not mention the size of the polynomials.

\*\*The presentation of the paper mixes the theorems with algorithms. I would suggest you separate them.

   First state your improved bound in theorems and present your algorithms.

\*\*The presentation of the algorithm looks like a direct map from the code, which makes the algorithm hard

   to understand than necessary. Some simplifications and informal explanations of the main idea are needed.

   Some suggestions:

   1) Algorithm 1, line 9 and 10 seems incorrect. You mean p := p(-x)? and I.add(-cf(p))?

   2) It's better to replace Algorithm 2 by an operation.

  3) In Algorithm 3, is p a pointer?

  4) You need a simple example to illustrate your algorithm by tracing the main steps.

   5) In Algorithm 5 and 6, why not simply assume that a\_n > 0?

   6) Algorithm 6, step 2, is lessOne(p) is false, why the root bound is 2?

   7) Algorithm 6, line 9, the use of loop condition i==n is weird, you mean while true, since at the beginning of

      each iteration, the value of i is for use n. Line 25, should you return 2^(base-1)?

   8) Corollary, you can not say "it cost at most ...." if you did not specify how you compute 2^({n-j}\*base).

\*\*\*The second trick does not seem obvious. Some explanation or reference is needed.

\*\*\*The proof of Theorem 5 is unclear.

\*\*\*Some English problems

 1) Page 3, the line before Algorithm 2, ==> ....this replacement will further....

 2) Page 3, second line after Definition 2, has only a little ==> is a slight..

 3) Page 5, right shifts ==>shifts...right