1. Subrings of quotient rings

We may have overlooked the case where the ambient ring is a quotient ring. This use case is important to some applications and it is discussed the paper "Using SAGBI bases to compute invariants" by Stillman and Tsai. If you aren't already aware, this paper's authors are the package's original programmers. In the paper, they specifically mention a specific implementation which is an earlier version of the package.

The code is actually close to being capable of handling subrings of quotient rings. After looking into it, I think it should work correctly in all cases where a finite Sagbi basis happens to exist. However, a problem arises when a finite Sagbi basis doesn't exist. In this case, it is possible for the algorithm to falsely succeed. The calculation they give in Example 1 exhibits this behavior:

```
i1 : gndR = kk[x,y, MonomialOrder => Lex];

i2 : I = ideal(x^2 - x*y);

o2 : Ideal of gndR

i3 : Q = gndR/I;

i4 : subR = sagbi subring {x};

i5 : gens subR

o5 = | x |

              1       1
o5 : Matrix Q  <--- Q
```

In this example, the algorithm terminated successfully because it has no way of knowing that $x^2 = xy$ inside of the ambient ring. Technically, one could argue that this isn't a bug because it's actually doing what it is supposed to do according to the specification. Notice that the Sagbi algorithm's proof of correctness (Proposition 3) doesn't apply when a finite Sagbi basis doesn't exist. Fixing this "issue" is really a question of whether or not it is possible to improve upon the underlying algorithm.

In cases like the above example, it would be an improvement if there was a way to at least distinguish between failure and success. In section 2.2, Stillman and Tsai give some information about how this can be done. However, I find this part difficult to understand and I haven't been able to apply successfully apply their solution. Another problem is that it complicates the design because there would be really be two types of failure, for a total of three possible outcomes:

(1) Successfully found a finite Sagbi basis.
(2) Failed to find a finite Sagbi basis before the limit.
(3) (The new condition) Ran out of S-polynomial candidates.

I wonder if it's possible to generate a useful partial Sagbi basis in case (3).