Modular methods in commutative algebra

In the computation Gröbner basis over the rational $\mathbb{Q}$, an intermediate coefficient growth was observed. That is, we have intermediate results which are much larger than the output result. One way to remedy this is a modular approach. The computations are done over several prime fields and use the Chinese reminder as an accumulator and rational reconstrution to lift the result over the rational. This modular method has good performances in examples with large coefficients. Nevertheless, the algorithm is probabilistic. This is due to the choise of the primes to be used in the computation and the existence of primes which lead to a wrong result called *bad primes*. In practice, such primes cannot be identified but the use of error tolerant rational reconstruction ensures the correctness of the result even in presence of *bad primes*, provided that we have used a sufficiently large set of primes. Indeed, in the case involving Gröbner basis, the set of *bad prime* are finite. So that we can rely on this method. In another hand, this construction carries out a way to parallelize algorithms in commutative algebra because the computations over the prime fields are independent of each other. See below the general reconstruction scheme for modular algorithms.

Algorithm Reconstruct polynomial data

Input: Polynomial data I, an algorithm to compute $U(p)$ from I_p and a way of verifying that the computed data is equals to $U(0)$.

Output: The ideal $U(0)$.

1: Choose a random set of finite prime $\mathcal{P}$.
2: Compute $U(p)$ for all $p \in \mathcal{P}$.
3: Delete primes in $\mathcal{P}$ respecting to a majority vote on $\mathrm{LM}(G(p))$.
4: Use Chinese remainder theorem to lift the result to $U(N)$ where $N = \prod_{p \in \mathcal{P}} p$.
5: Reconstruct $U(N)$ via error tolerant reconstruction and get U.
6: **if** $U_p = U(p)$ for a random prime $p \notin \mathcal{P}$ **then**
7:     **if** $U = U(0)$ **then**
8:         **return** U.
9:     **else**
10:         Enlarge $\mathcal{P}$ and repeat from 2.
11:     **end if**
12: **end if**

- I_p is the reduction of I modulo the prime p.
- $U(p)$ is a result modulo p and U is the result expected .
- The majority vote is an attempt to remove *bad primes* by considering the leading monomial.
- The test in line 6 is called `pTest` which verrify the result modulo some prime p.
- In practical application, one can check in every loop of the algorithm if the result U has stabilized. Then, we continue with the `pTest` if and only if this is the case else we go directly to line 10.