Chapter 9

CONCLUDING REMARKS

9.1 Summary of Contributions

In this thesis, we looked at two different relaxation of the decoding problems for error correcting codes: list decoding and property testing.

In list decoding we focused on the achieving the best possible tradeoff between the rate of a code and the fraction of errors that could be handled by an efficient list decoding algorithm. Our first result was an explicit construction of a family of code along with efficient list decoding algorithm that achieves the list decoding capacity. That is, for any rate $0 < R < 1$, we presented folded Reed-Solomon codes of rate R along with polynomial time list decoding algorithms that can correct up to $1 - R - \varepsilon$ fraction of errors (for any $\varepsilon > 0$). This was the first result to achieve the list decoding capacity for any rate (and over any alphabet) and answered one of the central open questions in coding theory. We also constructed explicit codes that achieve the tradeoff above with alphabets of size $2^{O\left(\varepsilon^{-4}\log(1/\varepsilon)\right)}$, which are not that much bigger than the optimal size of $2^{\Omega(1/\varepsilon)}$.

For alphabets of fixed size, we presented explicit codes along with efficient list decoding algorithms that can correct a fraction of errors up to the so called Blokh-Zyablov bound. In particular, these give binary codes of rate $\Omega(\varepsilon^3)$ that can be list decoded up to a $1/2 - \varepsilon$ fraction of errors. These codes have rates that come close to the optimal rate of $\Theta(\varepsilon^2)$ that can be achieved by random codes with exponential time list decoding algorithms.

A key ingredient in designing codes over smaller alphabets was to come up with optimal list recovery algorithms. We also showed that the list recovery algorithm for Reed-Solomon codes due to Guruswami and Sudan is the best possible. We also presented some explicit bad list decoding configurations for list decoding Reed Solomon codes.

Our contributions in property testing of error correcting codes are two-fold. First, we presented local testers for Reed-Muller codes that use near optimal number of queries to test membership in Reed-Muller codes over fixed alphabets. Second, we defined a natural variation of local testers called tolerant testers and showed that they had comparable parameters with those of the best known LTCs.

9.2 Directions for Future Work

Even though we made some algorithmic progress in list decoding and property testing of error correcting codes, there are many questions that are still left unanswered. We have

highlighted the open questions throughout the thesis. In this section, we focus on some of the prominent ones (and related questions that we did not talk about earlier).

9.2.1 List Decoding

The focus of this thesis in list decoding was on the optimal tradeoff between the rate and list decodability of codes. We first highlight the algorithmic challenges in this vein (most of which have been highlighted in the earlier chapters).

- The biggest unresolved question from this thesis is to come up with explicit codes over fixed alphabets that achieve the list decoding capacity. In particular is there a polynomial time construction of a binary codes of rate $\Omega(\varepsilon^2)$ that be list decoded in polynomial time up to $1/2 - \varepsilon$ fraction of errors? (Open Question 4.1)

- A less ambitious goal than the one above would be to give a polynomial time construction of a binary code with rate $\Omega(\varepsilon)$ that can be list decoded up to $1 - \varepsilon$ fraction of *erasures*? Erasures are a weaker noise model that we have not considered in this thesis. In the erasure noise model, the only kind of errors that are allowed is the "dropping" of a symbol during transmission. Further, it is assumed that the receiver knows which symbols have been erased. For this weaker noise model, one can show that for rate R the optimal fraction of errors that can be list decoded is $1 - R$.

- Another less ambitious goal would be to resolve the following question. Is there a polynomial time construction of a code that can be list decoded up to $1/2 - \varepsilon$ fraction of errors with rate that is asymptotically better than ε^3?

- Even though we achieved list decoding capacity for large alphabets (that is, for rate R code, list decode $1 - R - \varepsilon$ fraction of errors), the worst case list size was $n^{\Omega(1/\varepsilon)}$, which is very far from the $O(1/\varepsilon)$ worst case list size achievable by random codes. A big open question is to come up with explicit codes that achieve the list decoding capacity with constant worst case list size. As a less ambitious goal would be to reduce the worst case list size to n^c for some constant c that is independent of ε. (See Section 3.7)

We now look at some questions that relate to the combinatorial aspects of list decoding.

- For a rate R Reed-Solomon code, can one list decode more than $1 - \sqrt{R}$ fraction of errors in polynomial time? (Open Question 6.2)

- To get to within ε of list decoding capacity can one prove a lower bound on the worst case list size? For random codes it is known that list of size $O(1/\varepsilon)$ suffice but no general lower bound is known.[1]

[1] For high fraction of errors, tight bounds are known [65].

- For codes of rate R over fixed alphabet of size $q > 2$, can one show existence of linear codes that have $q^{o(1/\varepsilon)}$ many codewords in any Hamming ball of radius $1-H_q(R)-\varepsilon$? (See discussion in Section 2.2.1)

9.2.2 Property Testing

Here are some open questions concerning property testing of codes.

- The biggest open question in this area is to answer the following question. Are there codes of constant rate and linear distance that can be locally tested with constant many queries?

- A less ambitious (but perhaps still very challenging) goal is to show that the answer to the question above is no for 3 queries.

- Can one show that the answer to the first question (or even the second) is no, if one also puts in the extra requirement of tolerant testability? (Open Question 8.1)