Electronic Colloquium on Computational Complexity - Reports Series 1997 - available via:

ECCC
TR97-046

FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

syndrome trellis 73
test set 48*ff.*
time complexity 5
trellis complexity 74
zero neighbor 52

Index

algorithmic problem 5
almost all 7

bound
  Blokh–Zyablov 28
  Justesen 27
  Zyablov 27

boundary 51
channel 70
  additive 70
  discrete 70
  memoryless 70
  symmetric 70

code
  algorithmic problems for 8
  asymptotically good 11
  correcting defects 34
  correcting erasures 31
  correcting localized errors 37
  limit parameters of 11
  linear
    concatenated 26
    covering radius of 41
  low-density parity-check 13
  rate of 8
  regular-graph 17
  relative distance of 8
  systematic 7

check set 7
computation 5
covering 57
covering system 58

decision problem 5
defect 34

decoding
  bounded distance 8
  complete minimum distance 8
  covering set 56
  hard-decision 9, 46
  gradient-like 48
  information set 55
  iterated majority voting 13
    parallel 16
  generalized minimum distance 85
  list cascade 67
  maximum likelihood 70
  minimal-vectors 50
  soft-decision 9, 70*ff.*
  split syndrome 48
    punctured 62
  supercode 64*f.*
  syndrome 45
  syndrome trellis 72*ff.*
  weighted-partitions 83
  zero-neighbors 52

edge-vertex incidence graph 18
error-correcting pair 24
Evseev lemma 45
  generalized 77*f.*
expander 16

Gilbert–Varshamov distance 8
GUAVA 12
information set 7, 55*ff.*
large factors 101
logical circuit 6
  size 6
  depth 6
McEliece cryptosystem 58
minimal vector 49

problem
  algorithmic 5
  almost-NP-hard 100
  complete 6
    NP– 6, 94*ff.*
    Π_2^p– 98
  constructive 6
  easy 4
  difficult 4
  decision 4
  instance of 5
  intractable 6
  mass 5
  polynomially reducible 6

Random Access machine (RAM) 5
random matrix
  corank of 43*f.*
segmentation 80
space complexity 5
support 49

[157] M. A. Tsfasman and S. G. Vlăduţ, *Algebraic-Geometric Codes*, Dordrecht : Kluwer (1991).

[158] V. Uspensky and A. Semenov, *Algorithms: Main Ideas and Applications*, Kluwer (1993).

[159] A. Vardy, "The Nordstrom-Robinson code: representation over GF(4) and efficient decoding," *IEEE Trans. Inform. Theory*, **40** (5) (1994), 1686–1693.

[160] A. Vardy and Y. Be'ery, "More efficient soft decoding of the Golay codes," *IEEE Trans. Inform. Theory* **IT-37** (3) (1991), 667–672.

[161] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. Inform. Theory*, to appear. Preliminary version in *Proc. 29th ACM Annual Sympos. on the Theory of Computing* (STOC'97), ACM (1997), pp. 92–109.

[162] S. G. Vlăduţ and Yu. I. Manin, "Linear codes and modular curves," Современные проблемы математики, **25**, Moscow: VINITI (1984), 209–257, and *J. Soviet Math.* **30** (1985), 2611-2643.

[163] C. Voss and T. Høholdt, "An explicit construction of a sequence of codes attaining the Tsfasman–Vlăduţ–Zink bound: The first steps," *IEEE Trans. Inform. Theory*, **43** (1) (1997), 128–136.

[164] K. Wagner and G. Wechsung, *Computational Complexity*, Dordrecht: Reidel (1986).

[165] D. J. A. Welsh, *Matroid Theory*, London: Academic Press (1977).

[166] J. K. Wolf, "Efficient maximum likelihood decoding of linear codes using a trellis," *IEEE Trans. Inform. Theory*, **24** (1) (1978), 76–80.

[167] J. Wolfmann, "A permutation decoding of the $(24, 12, 8)$ Golay code," *IEEE Trans. Inf. Theory*, **29** (5) (1983), 748–751.

[168] V. V. Zyablov, "An estimate of complexity of constructing binary linear cascade codes," *Problems of Info. Trans.,* **7** (1) (1971), 3-10.

[169] V. V. Zyablov and M. S. Pinsker, "Estimation of the error-correcting complexity of Gallager low-density codes," *Problems of Info. Trans.,* **11** (1) (1975), 26–36 and 18–28.

[170] ———, "List cascade decoding," *Problems of Info. Trans.,* **17** (4) (1981), 29–33 and 236–240.

[171] V. V. Zyablov and V. R. Sidorenko, Bounds to the trellis decoding complexity of linear block codes," *Problems of Info. Trans.,* **29** (3) (1993), 3–9 and 202–207.

[141] I. E. Shparlinski, *Computational and Algorithmic Problems in Finite Fields*, Dordrecht: Kluwer (1992).

[142] L. J. Shulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," *Proc. 8th Annual ACM-SIAM Sympos. on Discrete Algorithms* (SODA '97), (1997) 669-674.

[143] V. M. Sidelnikov, "Weight spectrum of the binary Bose–Chaudhuri–Hocquenghem codes," *Problems of Info. Trans.,* **7** (1) (1971), 14–22 and 11–17.

[144] ———, "Decoding of Reed–Solomon codes beyond $(d-1)/2$ and zeros of multivariate polynomials," *Problems of Info. Trans*, **30** (1) (1994), 51–69 and 44–59. Errata in the English translation, *ibid.*, **30** (2) (1994).

[145] J. Simonis, "On generator matrices of codes," *IEEE Trans. Inform. Theory*, **IT-38** (2) (1992), p. 516.

[146] ———, "GUAVA: A computer algebra package for coding theory," Proc. 4th Intern. Workshop "*Algebraic and Combinatorial Coding Theory, Novgorod, Russia*" (ACCT4), (1994), pp. 165–166.

[147] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inform. Theory*, **IT-42** (6) (1996), 1710-1722. Preliminary version in *Proc. 35th Sympos. on the Foundations of Computer Science* (FOCS '94), IEEE Press (1994), pp. 566–576.

[148] J. Snyders, "Partial ordering of error patterns for maximum likelihood soft decoding," in G. Cohen et al., Eds, *Algebraic Coding*, Lect. Notes Comput. Science, **573**, New York: Springer (1992), pp. 120–125.

[149] U. K. Sorger, "A new Reed–Solomon code decoding algorithm based on Newton's interpolation," *IEEE Trans. Inform. Theory*, **IT-39** (2) (1993), 758–765.

[150] D. Spielman, "Linear-time encodable and decodable error-correcting codes," *IEEE Trans. Inform. Theory*, **IT-42** (6) (1996), 1723–1731. Preliminary version in *Proc. 27th ACM Annual Sympos. on the Theory of Computing* (STOC '95), ACM (1995), pp. 388–397.

[151] J. Stern, "Approximating the number of error locations within a constant ration is NP-complete," in G. Cohen, T. Mora, and O. Moreno, Eds., *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, (AAECC-10), Lect. Notes Comput. Science **673** (1993), pp. 325–331.

[152] ———, "A method for finding codewords of small weight," in G. Cohen and J. Wolfmann, Eds., *Coding Theory and Applications*, Lect. Notes Comput. Science **388**, New York: Springer (1989), pp. 106–113.

[153] M. Sudan, "Decoding of Reed–Solomon codes beyond the error-correcting bound," *J. Complexity*, **13** (1997) 180-193. Preliminary version in *Proc. 37th Sympos. on the Foundations of Computer Science* (FOCS '96), IEEE Press (1996).

[154] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "Superimposed concatenated codes," *IEEE Trans. Inform. Theory*, **IT-26** (6) (1980), 735–736.

[155] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, **IT-27** (5) (1981), 533–547.

[156] J. van Tilburg, "On the McEliece public-key cryptosystem," in S. Goldwasser, Ed., "*Advances in Cryptology*" (Crypto'88), Lect. Notes Comput. Science **403**, New-York: Springer (1990), pp. 119–131.

[122] A. McLoughlin, "The complexity of computing the covering radius of a code," *IEEE Trans. Inform. Theory* **IT-30** (6) (1984), 800–804.

[123] H. Miyakawa and T. Kaneko, "Decoding algorithms of error-correcting codes by use of analog weights," *Electron. Commun. Japan*, **58**-A (1975), 18–27.

[124] S. Ntafos and S. Hakimi, "On the complexity of some coding problems," *IEEE Trans. Inform. Theory*, **IT-27** (6) (1981), 794–796.

[125] C. Papadimitriou, *Computational Complexity*, Reading, MA: Addison-Wesley (1995).

[126] C. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *Journal of Computer and System Sciences*, **43** (1991), 425–440. Preliminary version in *Proc. 20th ACM Annual Sympos. on the Theory of Computing* (STOC'88), ACM (1988), pp. 229–234.

[127] R. Pellikaan, "On decoding by error location and dependent sets of error positions," *Discrete Math.*, **106/107** (1992), 369–381.

[128] ———, "On the existence of error-correcting pairs," *Journ. Stat. Plan. Inference*, **51** (1996), 229–242.

[129] E. Petrank and R. M. Roth, " Is code equivalence easy to decide?" *IEEE Trans. Inform. Theory*, **IT-43** (5) (1997), 1602–1604.

[130] V. Pless, "Decoding the Golay codes," *IEEE Trans. Inform. Theory*, **IT-32** (4) (1986), 561–567.

[131] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Trans.*, **IT-8** (1962), S5-S9.

[132] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, **36** (2) (1989), 335–348.

[133] H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw Hill, (1967).

[134] R. Roth, "Maximal-rank array codes and their application to crisscross error correction," *IEEE Trans. Inform. Theory*, **IT-37** (2) (1991), 328–336.

[135] R. Roth and G. Seroussi, "Location-correcting codes," *IEEE Trans. Inform. Theory*, **IT-42** (2) (1996), 554–565.

[136] J. E. Savage, "The complexity of decoders," I, II, *IEEE Trans. Inform. Theory*, **IT-15**, (6) (1969) 689–695 and **IT-17** (1) (1971), 77–85.

[137] J. E. Savage, *The Complexity of Computing*, New York: Wiley-Interscience Publications (1976).

[138] N. Sendrier, "Finding the permutation between equivalent binary codes," *Proc. IEEE Intern. Sympos. Inform. Theory*, Ulm (1997), p. 367.

[139] B.-Z. Shen, "A Justesen construction of binary concatenated codes which asymptotically meet Zyablov bound for low rate," *IEEE Trans. Inform. Theory*, **IT-39** (1) (1993), 239–242.

[140] P. W. Shor, "Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.* (to appear). Preliminary version in *Proc. 35th Annual Sympos. on Foundations of Computer Science* (FOCS'94), IEEE Press (1994), pp. 124–134.

[103] B. D. Kudryashov and T. G. Zakharova, "Block codes from convolutional codes," *Problems of Info. Trans.,* **25** (4) (1989), 98–102 and 336–339.

[104] A. V. Kuznetsov and B. S. Tsybakov, "Coding for memory with defective cells," *Problems of Info. Trans.,* **10** (2) (1974), 52–60 and 132–138.

[105] J. D. Lafferty and D. N. Rockmore, "Spectral techniques for expander codes," *Proc. 29th ACM Annual Sympos. on the Theory of Computing* (STOC'97), ACM (1997), pp. 160–167.

[106] A. Lafourcade and A. Vardy, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Inform. Theory,* **IT-41** (6) (1995) 1938–1954.

[107] J. Leon, "Computing automorphism groups of error-correcting codes," *IEEE Trans. Inform. Theory,* **IT-28** (3) (1982), 496–511.

[108] ———, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Trans. Inform. Theory,* **IT-34** (5) (1988), 1354–1359.

[109] L.A. Levin, "Average case complete problems," *SIAM J. Comput.,* **15** (1) (1986), 285–286.

[110] L. Levitin and C. R .P. Hartmann, "A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm," *IEEE Trans. Inform. Theory,* **IT-31** (3) (1985), 378–384.

[111] H.-A. Loeliger, "On the basic averaging arguments for linear codes," in R. Blahut et al., Eds., *Communications and Cryptography: Two Sides of One Tapestry,* Dordrecht: Kluwer (1994), pp. 251–261.

[112] B. Lopéz Jiménez, *Plane Models of Drinfeld Modular Curves,* Ph.Ḋ. Thesis, University of Complutense, Madrid (1996).

[113] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica,* **8** (3) (1988), 261–277.

[114] M. G. Luby, M. Mitzenmacher, M. Amin Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proc. 29th ACM Annual Sympos. on the Theory of Computing* (STOC'97), ACM (1997), pp. 150–159.

[115] Yu. I. Manin, *A Course in Mathematical Logic,* Berlin: Springer (1976).

[116] G. A. Margulis, "Explicit constructions of concentrators," *Problems of Info. Trans.,* **9** (4) (1973), 71–80 and 325–332.

[117] ———, "Explicit constructions of graphs without short cycles and low density codes," *Combinatorica,* **2** (1) (1982), 71–78.

[118] ———, "Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators," *Problems of Info. Trans.,* **24** (1) (1988), 51–60 and 39–46.

[119] J. L. Massey and T. Schaub, "Linear complexity in coding theory," in G. Cohen and P. Godlewski, Eds., *Coding Theory and Appl.,* Lect. Notes Comput. Science, **311**, New York: Springer (1988), pp.19–32.

[120] A. J. McAuley, "Reliable broadband communication using a burst erasure correcting code," *Proc. SIGCOMM* '90, Philadelphia: ACM Press (1990), pp. 287–306.

[121] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," DSN Progress Report 43-44, JPL, Pasadena, pp. 114–116 (1978).

[84] G. B. Horn and F. R. Kschischang, "On the intractability of permuting a block code to minimize trellis complexity," *IEEE Trans. Inform. Theory*, **IT-42** (6) (1996), 2042–2048.

[85] J. D. Horton, "A polynomial-time algorithm to find the shortest cycle basis of a graph," *SIAM J. Comput.*, **16** (2) (1987), 358–366.

[86] T.-Y. Hwang, "Decoding linear block codes for minimizing word error rate," IEEE Trans. Inform. Theory, **IT-25** (6) (1979), 733-737.

[87] ———, "Efficient optimal decoding of linear block codes," *IEEE Trans. Inform. Theory*, **IT-26** (5) (1980), 603–606.

[88] D. S. Johnson, "The NP-completeness column: An ongoing guide," editions 3, 18, and 22, *Journal of Algorithms*, **3** (1982), 182–195, **7** (1986), 584–601, **11** (1990), 144–151.

[89] ———, "A catalog of complexity classes," in J. van Leeuwen, Ed., *Handbook of Theoretical Computer Science*, vol. A: *Algorithms and Complexity*, North-Holland, (1990), pp. 67–161.

[90] D. Jungnickel, *Finite Fields*: *Structure and Arithmetics*, Mannheim: Wissenschaftsverlag (1992).

[91] J. Justesen, "A class of constructive, asymptotically good algebraic codes," *IEEE Trans. Inform. Theory*, **13** (Sept. 1972), 652–656.

[92] G. A. Kabatianski and E. A. Krouk, "Coding decreases delay of messages," *Proc. IEEE Intern. Sympos. Inform. Theory*, San-Antonio (1993), p.225.

[93] T. Kasami, "A decoding procedure for multiple-error-correcting cyclic codes," *IEEE Trans. Inform. Theory*, **10** (1964), 134–138.

[94] G. O. H. Katona, "The Hamming-sphere has minimum boundary," *Studia Sci. Math. Hungar.*, **10** (1-2) (1975), 131–140.

[95] M. Kiwi, "Testing and weight distributions of dual codes," *Electronic Colloquium on Computational Complexity*, TR97-010, `http://www.eccc.uni-trier.de/eccc/` .

[96] D. E. Knuth, *The Art of Computer Programming*, Vol. 3, Reading, MA: Addison-Wesley Publ. Co. (1973).

[97] R. Kötter, "Fast generalized minimum distance decoding of algebraic-geometry and Reed–Solomon codes," *IEEE Trans. Inform. Theory*, **IT-42** (3) (1996), 721–737.

[98] S. I. Kovalev, "Two classes of generalized minimum distance decoding algorithms," *Problems of Info. Trans.,* **22** (3) (1986), 35–42 and 186–192.

[99] ———, "Decoding of low-density codes," *Problems of Info. Trans.,* **27** (4) (1991), 51–56 and 317–321.

[100] J. Kratochvil, "Regular codes in regular graphs are difficult," *Discrete Mathematics*, **133** (1994), 191–205.

[101] E. A. Krouk, "Decoding complexity bound for linear block codes," *Problems of Info. Trans.,* **25** (3) (1989), 103–107 and 251–254.

[102] E. A. Krouk and S. V. Fedorenko, "Decoding by generalized information sets," *Problems of Info. Trans.,* **31** (2) (1995), 54–61 and 134–139.

[65] M. Frances and A. Litman, "On covering problems of codes," *Theory of Computing Systems*, **30** (2) (1997), 113–119.

[66] A. Frank, "Conservative weightings and ear-decompositions of graphs," *Combinatorica* **13** (1) (1993), 65–81.

[67] Z. Füredi, "Matchings and coverings in hypergraphs," *Graphs and Combinatorics*, **4** (1988), 115–206.

[68] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems of Info. Trans.,* **21** (1) (1985), 3–16 and 1–12.

[69] R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press (1963).

[70] ———, *Information Theory and Reliable Communication*, New York: John Wiley & Sons (1968).

[71] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San-Francisco: Freeman (1978).

[72] D. Gazelle and J. Snyders, "Reliability-based code-search algorithms for maximum-likelihood decoding of block codes," *IEEE Trans. Inform. Theory,* **IT-43** (1) (1997), 239–249.

[73] S. I. Gelfand and R. L. Dobrushin, "Complexity of asymptotically optimal code realization by constant depth schemes," *Problems of Control and Information Theory,* **1** (3/4) (1972), 197–215.

[74] S. I. Gelfand, R. L. Dobrushin, and M. S. Pinsker, "On the Complexity of Coding," in B. N. Petrov and F. Csaki, Eds., "*2nd International Sympos. on Inform. Theory, Tsahkadsor, Armenia*," Budapest: Akadémiai Kiadó (1973), pp. 177–184.

[75] F. Gerth, "Limit probabilities for coranks of matrices over $GF(q)$," *Linear and Multilinear Algebra*, **19** (1) (1986), 79–93.

[76] D. M. Gordon, "Minimal permutation sets for decoding the binary Golay code," *IEEE Trans. Inform. Theory,* **IT-28** (3) (1982), 541–543.

[77] D. M. Gordon, G. Kuperberg, O. Patashnik, and J. Spencer, "Asymptotically optimal covering designs," *J. Combin. Theory* A, **75** (2) (1996), 270–280.

[78] D. M. Gordon, O. Patashnik, and G. Kuperberg, "New constructions for covering designs," *J. Combin. Designs*, **3** (4) (1995), 269–284.

[79] D. A. Grable and K. T. Phelps, "Random methods in design theory: A survey," *J. Combin. Designs*, **4** (4) (1995), 255–273.

[80] Yu. Gurevich, "Average case complete problems," *J. Comput. Syst. Sciences*, **42** (3) (1991), 346–398.

[81] S. L. Hakimi and J. G. Bredeson, "Graph-theoretic error-correcting codes," *IEEE Trans. Inform. Theory,* **IT-14** (1968), 584–591.

[82] Y. S. Han, C. R. P. Hartmann, and C. C. Chen, "Efficient priority first search maximum-likelihood soft decision decoding of linear block codes," *IEEE Trans. Inform. Theory,* **IT-39** (5) (1993), 1514–1523.

[83] I. S. Honkala and P. R. J. Östergard, "Code design," in E. Aarts and J. K.Lenstra, Eds., *Local Search in Combinatorial Optimization*, New York: Wiley (1997), pp. 441–456.

[46] N. Cutland, *Computability*, Cambr. Univ. Press (1980).

[47] P. Diaconis and R. L. Graham, "The Radon transform on $\mathbf{Z}_2^k$," *Pacific J. Math.*, **118** (1985), 176–185.

[48] P. Delsarte, "Distance distributions of functions over Hamming spaces," *Philips Res. Repts.*, **30** (1975), 1–8.

[49] ———, "Bilinear forms over a finite field, with applications to coding theory," *J. Comb. Th.*, Ser. A, **25** (1978), 226–241.

[50] Y. Desaki, T. Fujiwara, and T. Kasami, "The weight distribution of extended binary primitive BCH codes of length 128," *IEEE Trans. Inform. Theory*, **IT-43** (4) (1997), 1364–1371.

[51] I. Dumer, "Two decoding algorithms for linear codes," *Problems of Info. Trans.,* **25** (1) (1989), 24–32 and 17–23.

[52] ———, "Asymptotically optimal linear codes correcting defects of linearly increasing multiplicity," *Problems of Info. Trans.,* **26** (2) (1990), 3–17 and 93–104.

[53] ———, "On minimum distance decoding of linear codes," *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, Moscow (1991), pp. 50–52.

[54] ———, "Suboptimal decoding of linear codes: Partition technique," *IEEE Trans. Inform. Theory*, **IT-42** (6) (1996), 1971–1986. Preliminary version in Proc. Eurocode '92, CISM Courses and Lectures, **339**, Berlin: Springer (1993), pp. 369–382.

[55] ———, "Maximum likelihood decoding with presorting," manuscript (1997), also *Proc. IEEE Intern. Sympos. Inform. Theory*, Ulm (1997), p. 396.

[56] I. M. Duursma and R. Kötter, "Error-locating pairs for cyclic codes," *IEEE Trans. Inform. Theory*, **IT-40** (4) (1994), 1108–1121.

[57] M. Elia, "Algebraic decoding of the $(23, 12, 7)$ Golay code," *IEEE Trans. Inform. Theory*, **IT-33** (1) (1987), 150–151.

[58] P. van Emde Boas, "Machine models and simulations," in J. van Leeuwen, Ed., *Handbook of Theoretical Computer Science*, vol. A: *Algorithms and Complexity*, Amsterdam: North-Holland (1990), pp. 1–66.

[59] P. Erdős and J. Spencer, *Probabilistic Methods in Combinatorics*, Budapest: Akadémiai Kiadó (1974).

[60] G. S. Evseev, "Complexity of decoding for linear codes," *Problems of Info. Trans.,* **19** (1) (1983), 3–8 and 1–6.

[61] S. V. Fedorenko, "Decoding algorithms of linear block codes," *Problems of Info. Trans.,* **29** (4) (1993), 18–23 and 313–317.

[62] J. Feigenbaum, "The use of coding theory in computational complexity," in R. Calderbank, Ed., *Different Aspects of Coding Theory*, Proc. Symposia in Applied Mathematics, vol. **50**, Providence: AMS (1995), pp. 207-233.

[63] G. D. Forney, "Generalized minimum distance decoding," *IEEE Trans. Inform. Theory*, **IT-12** (2) (1966), 125–131.

[64] M. P. C. Fossorier and S. Lin, "Computationally efficient soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inform. Theory*, **IT-42** (3) (1996), 738–750.

[26] L. A. Bassalygo, V. V. Zyablov and M. S. Pinsker, "Problems of complexity in the theory of error-correcting codes," *Problems of Info. Trans.,* **13** (3) (1977), 5–17 and 166–175.

[27] E. Berlekamp, "Bounded distance + 1 soft decision Reed–Solomon decoding," *IEEE Trans. Inform. Theory,* **IT-42** (3) (1996), 704–720.

[28] E. Berlekamp, R. J. McEliece and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform Theory,* **IT-29** (3) (1978), 384–386.

[29] M. Blaum and J. Bruck, "Simple combinatorial decoding of the [23,12,7] Golay code," in R. Capocelli, Ed. *Sequences*, Berlin: Springer (1990), pp. 433-448.

[30] V. M. Blinovskii, "Lower asymptotic bound on the number of linear code words in a sphere of given radius in F_q^n," *Problems of Info. Trans.,* **23** (2) (1987), 50–53 and 130–132.

[31] ———, "Asymptotically exact uniform bounds for spectra of cosets of linear codes," *Problems of Info. Trans.,* **26** (1) (1990), 99–103 and 83–86.

[32] ———, *Asymptotic Combinatorial Coding Theory*, Kluwer: Boston (1997).

[33] J. Bruck and M. Naor, "The hardness of decoding linear codes with preprocessing," *IEEE Trans. Inform. Theory,* **IT-36** (2) (1990), 381–385.

[34] L. Calabi, "A note on rank and nullity in coding theory," *Information and Control,* **4** (4) (1961), 359–363.

[35] R. Calderbank and P. Shor, "Good quantum error-correcting codes exist," *Physical Review,* Ser. A, **54** (2) (1996), 1098–1105.

[36] A. Canteaut and F. Chabaud, "A new algorithm for finding minimum-weight codewords in a linear code: application to primitive narrow-sense BCH codes of length 511," Rapport de recherche no. 2685, INRIA, Rocquencourt (1995).

[37] A. H. Chan and R. A. Games, "(n, k, t)-covering systems and error-trapping decoding," *IEEE Trans. Inform. Theory,* **IT-27** (1981), 643–646.

[38] P. Charpin, "Tools for coset weight enumerators of some codes," in G. L. Mullen and P. J.-S. Shiue, Eds., *Finite Fields: Theory and Applications*, Providence: AMS (1994), pp. 1–14.

[39] F. R. K. Chung, *Spectral Graph Theory*, Providence: Amer. Math. Society (1997).

[40] D. M. Chickering, D. Geiger, and D. Heckerman, "On finding a cycle basis with a shortest maximal cycle," *Information Processing Letters*, **54** (1) (1995), 55–58.

[41] G. Clark and J. Cain, *Error-Correcting Coding for Digital Communication*, London: Plenum Press (1981).

[42] J. T. Coffey and R. M. F. Goodman, "The complexity of information set decoding," *IEEE Trans. Inform. Theory,* **IT-35** (5) (1990), 1031–1037.

[43] J. T. Coffey, R. M. F. Goodman, and P. Farrell, "New approaches to reduced complexity decoding," *Discrete Applied Math.,* **33** (1991), 43–60.

[44] G. Cohen, S. Litsyn, and G. Zémor, "On greedy algorithms in coding theory," *IEEE Trans. Inform. Theory,* **IT-42** (6) (1996), 2053–2057.

[45] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, Cambridge, MA: MIT Press (1990).

[10] ———, "Reductions, codes, PCPs, and inapproximability," *Proc. 36th Annual Sympos. on Foundations of Computer Science* (FOCS'95), IEEE Press (1995), pp. 404–413.

[11] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximating optima in lattices, codes, and systems of equations," *Proc. 34th Annual Sympos. on Foundations of Computer Science* (FOCS'93), IEEE Press (1993), pp. 724–733.

[12] D. Augot, "Description of minimum weight codewords of cyclic codes by algebraic systems," *Finite Fields Appl.*, **2** (2) (1996), 138–152.

[13] A. Ashikhmin and A. Barg, "Minimal vectors in linear codes and sharing of secrets," Universität Bielefeld, SFB 343 Diskrete Strukturen in der Mathematik, Preprint 94-113 (1994), available from `ftp.uni-bielefeld.de`.

[14] J. J. Ashley, R. Karabed, and P. Siegel, "Complexity and sliding-block decodability," *IEEE Trans. Inform. Theory,* **IT-42** (6) (1996), 925–947.

[15] C. A. Asmuth and G. R. Blakley, "Pooling, splitting and restituting information to overcome total failure of some channels of communication," *Proc.* 1982 *Sympos. on Security and Privacy*, New York: IEEE Press (1982), pp. 156–169.

[16] A. Ashikhmin, A. Barg, G. Cohen, and L. Huguet, "Variations on minimal codewords in linear codes," in G. Cohen, M. Giusti and T. Mora, Eds., *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes* (AAECC-11), Lect. Notes Comput. Science, vol. **948** (1995), pp. 96–105.

[17] L. Babai, "Transparent proofs and limits to approximation," *Proc. First European Congress of Mathematics*, pt. 2, Basel: Birkhäuser (1994), pp. 31–92.

[18] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory,* **IT-20** (2) (1974), 284–287.

[19] A. Barg, "Some new NP-complete coding problems," *Problems of Info. Trans.,* **30** (3) (1994) 23–28 and 209–214.

[20] A. Barg and I. Dumer, "Concatenated decoding algorithm with incomplete inspection of code vectors," *Problems of Info. Trans.,* **22** (1) (1986), 3–8 and 1–7.

[21] ———, "On computing the weight spectrum of cyclic codes," *IEEE Trans. Inform. Theory,* **IT-38** (4) (1992), 1382–1386.

[22] A. Barg, E. Krouk and H. C. A. van Tilborg, "Remarks on the hard-decision decoding of linear codes," preprint (1996), also *Proc. IEEE Intern. Sympos. Inform. Theory,* Ulm (1997), p.331.

[23] A. Barg and S. Zhou, "Linear-time binary codes correcting localized erasures," *Proc. 35th Annual Allerton Conf. on Communication, Control and Computing*, Monticello, IL (1997).

[24] J.-P. Barthélemy, G. D. Cohen, and A. Lobstein, *Complexité algorithmique de problèmes de communication*, Paris: Masson (1992). English translation: *Algorithmic Complexity and Communication Problems*, London: University College of London (1996).

[25] L. A. Bassalygo, S. I. Gelfand, and M. S. Pinsker, "Codes for channels with localized errors," in: *Proc. 4th Swedish–Soviet Workshop on Information Theory,* (1989), pp. 95–99.

computations that operate with states of certain quantum systems. Shor [140] showed that using this hypothetical computing device it is possible to greatly expedite algorithmic solutions of some notoriously difficult problems. The existence of asymptotically good quantum codes has been proved in Calderbank and Shor [35]. Therefore, a most natural question is whether it is possible to construct simple quantum computations for the decoding of these quantum codes or of classical codes with a better performance than that of the classical algorithms discussed in this chapter.

Acknowledgment. A substantial part of this chapter was written during my stay at the Faculty of Mathematics and Computing Science of the Technical University of Eindhoven. I wish to thank Prof. H. C.A. van Tilborg for granting me an opportunity to spend the years 1995-96 at the faculty and for his support.

Discussions with Shiyu Zhou were helpful in improving the presentation of a number of results in Section 2.

References

[1] E. Agrell, "Voronoi regions for binary linear block codes," *IEEE Trans. Inform. Theory*, **IT-42** (1) (1996), 310–316.

[2] R. Ahlswede, L.A. Bassalygo, and M.S. Pinsker, "Asymptotically optimal binary codes of polynomial complexity correcting localized errors," *Problems of Info. Trans.,* **31** (2) (1995), 76–83 and 162–168 [4].

[3] A.V. Aho, J.E. Hoprcroft, and J.D. Ullman *The Design and Analysis of Computer Algorithms*, London: Addison-Wesley (1974).

[4] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inform. Theory*, **IT-42** (6) (1996), 1737–1747. Preliminary version in *Proc. 35th Annual Sympos. on Foundations of Computer Science* (FOCS'94), IEEE Press (1994), pp. 604–613.

[5] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth, "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs," *IEEE Trans. Inform. Theory,* **IT-38** (2) (1992), 509–516.

[6] N. Alon and F.R.K. Chung, "Explicit constructions of linear sized tolerant networks," *Discrete Mathematics*, **72** (1988), 15–19.

[7] N. Alon and M. Luby, "A linear time erasure resilient code with nearly optimal recovery," *IEEE Trans. Inform. Theory,* **IT-42** (6) (1996), 1732–1736. Preliminary version in N. Alon, J. Edmonds, and M. Luby, *Proc. 36th Annual Sympos. on Foundations of Computer Science* (FOCS'95), IEEE Press (1995), pp. 512–519.

[8] N. Alon and J. Spencer, *The Probabilistic Method*, Chichester: Wiley-Interscience (1992).

[9] S. Arora, *Probabilistic Checking of Proofs and Hardness of Approximation Problems,* Dept. of Computer Science, Princeton University, Techn. Rep. CS-TR-476-94 (1994), 158p., available from `http://www.eccc.uni-trier.de/eccc/`.

[4]The first page range refers to the Russian original (Проблемы передачи информации) published by the Russian Academy of Sciences and the second one to the English translation (Problems of Information Transmission) published by Plenum Press.

NP-completeness of deciding whether a binary code has a vector of weight $n/2$ has been proved by Calderbank and Shor (see [47]). The same result for ternary codes and weight n is from Barg [19].

For the NP-completeness of $\mathrm{P}_5, \mathrm{P}_6, \mathrm{P}_7$ see Frances and Litman [65], Horn and Kschischang [84], and Kratochvil [100], respectively. An easy reduction for P_6 follows from part (b) of Theorem 4.1; see Vardy [161].

The construction of codes from graphs was introduced in Calabi [34], Hakimi [81]. Properties of classical linear spaces in graphs and their combinatorial evolution form the subject of matroid theory; see Welsh [165]. Many problems that are difficult for general linear codes, are polynomial for cycle codes of graphs. Examples are decoding, see Ntafos and Hakimi [124], finding a basis of minimal total weight, see Chickering et al. [40], Horton [85], and computing the covering radius, see Frank [66].

Theorem 4.4 is due to McLoughlin [122]. The complexity of the Weight of error problem with preprocessing (P_9) was proved in Bruck and Naor [33]. Their reduction is from K_3. The proof that we give, following Lobstein [24, pp. 121-123], has the advantage of being valid for any fixed code alphabet.

The nonapproximability theorem (Theorem 4.6) and the surrounding discussion is from Arora et al. [11], Stern [151]. Stern [151] constructs a sequence of approximation preserving reductions from 3-SAT (cf. Problem K_2). Arora et al. [11] also prove similar nonapproximability results for integral lattices in $\mathbf{R}^n$. The general theory of problems that are hard to approximate is found in Papadimitriou and Yannakakis [126],

Some NP-complete problems related to constrained codes are found in Ashley et al. [14]. In [129] Petrank and Roth study the complexity of deciding whether two linear codes are equivalent. An algorithm for establishing equivalence of two binary codes with no nontrivial automorphisms is presented in Sendrier [138].

* * *

In this chapter we studied the use of complexity theory in coding. However, complexity theory itself also profits from coding methods. Since the technique is not related to codes and the proofs are long, we have decided not to include these results. They are covered in Arora [9], [10], Babai [17], Feigenbaum [62], Kiwi [95].

In conclusion we would like to mention two research areas related to our topic.

First, as is well known and partly shown above, most best results in coding theory are achieved by averaging over the ensemble of random codes. Therefore, it is natural to ask whether basic problems, first and foremost, the decoding, are NP-complete *on the average* in the sense of the definitions in Levin [109], Gurevich [80].

The second area is related to algorithms for a "quantum computer," i.e.,

$\mathsf{wt}(\boldsymbol{y}_1) = t$. This implies, as in the proof of Thm. 4.1, Part (a), that the nonzero coordinates of $\boldsymbol{y}_1$ equal one and as an incidence vector, it defines a matching $M \subset W^3$. Finally, the identity $\boldsymbol{y}_1 + \boldsymbol{y}_2 = \boldsymbol{z}_U$ locates the ones of $\boldsymbol{y}_1$ within the coordinates corresponding to U, that is, $M \subseteq U$. ∎

Remark 4.2 Note the difference between this reduction and the one in part (a) of Theorem 4.1. There H depended on the instance of K_1, here it is fixed, and the only influence of a particular instance of the matching problem is through $\boldsymbol{y}$ and w.

We conclude our discussion with another NP-completeness related result that deals with approximating the solution to Problem P_1. From the proof of Theorem 4.1(a), we see that this problem remains NP-complete even if in its statement we ask about exact equality.

Let $\beta > 1$. An algorithm is said to approximate the solution of a problem (in our case, the weight of error) if its output is within the interval $[w, \beta w]$, w being the exact answer. If β is a constant, one speaks on approximating the weight of error up to a constant factor. If $\beta = 2^{\log^{0.5-\epsilon} n}, \epsilon > 0$, then it is said that an algorithm approximates a solution up to a *large factor*.

Approximating the weight of an error even within any large factor is likely to be difficult. In the following theorem we use one technical concept. A function f is *almost*-NP-*hard* if using f as an oracle, we could solve any NP problem in deterministic *quasi-polynomial* time, i.e., in time $\mathcal{O}(n^{poly(\log n)})$.

Theorem 4.6 *Approximating the weight of an error within any constant factor is NP-hard and within any large factor is almost-NP-hard.*

A simple way to prove a weaker nonapproximability result is to reduce Problem K_3 to P_1. Note that this reduction preserves approximations. Therefore, if K_3 is NP-hard to approximate up to a certain constant, then so is P_1. In particular, finding a 98% approximation in the maximal cut problem is known to be NP-hard. Therefore, the same holds for Problem P_1.

4.1 Notes

A comprehensive account of logical foundations related to the complexity classes is given in Garey and Johnson [71]. The NP-completeness proofs of problems K_1–K_3 are also found there.

The first paper on NP-complete problems related to coding was Berlekamp et al. [28]. The results of this paper correspond to parts (a) and (c) of our Theorem 4.1. That Weight of error is NP-complete for arbitrary q and for multilevel codes was observed in Barg [19].

Part (b) is a recent breakthrough due to Vardy [161]. The NP-completeness of the Minimal weight problem has been conjectured in Berlekamp et al. [28] and has resisted all attacks for 19 years despite repeated calls for a proof; see Johnson [88]. Some partial results were proved in Ntafos and Hakimi [124]. The

mial time using a one-step oracle for problems from coNP. Clearly, coNP $\subseteq \Pi_2^p$ and a fundamental conjecture states that this inclusion is proper. Under this assumption, a Π_2^p-complete problem is more difficult that any problem from coNP.

We state without proof the following theorem.

Theorem 4.4 P_8 *is* Π_2^p*-complete.* ∎

By building longer quantified expressions of the form $\forall\exists\forall\exists\dots$ it is possible to form an hierarchy of complexity classes coNP $= \Pi_1^p \subseteq \Pi_2^p \subseteq \dots$. Then the above conjecture suggests that every inclusion in this chain is proper. The order of quantifiers in the alternating sequence is quite essential. We have illustrated this difference above for the classes NP and coNP, i.e., for problems which ask for the existence of an object (NP) and problems that claim something for all objects from a certain set (coNP). Reversing the order ($\exists\forall\exists\forall\dots$), we get an hierarchy of complementary classes NP $= \Sigma_1^p \subseteq \Sigma_2^p \subseteq \dots$. The general form of the fundamental conjecture is that every inclusion in either chain is proper. The next result that we are going to discuss is related to this conjecture.

In Problem P_1 we have assumed that the code is a part of the input, i.e., that the matrix H is supplied with every particular instance. Relaxing this assumption, let us suppose that it is given once and forever and we can perform an arbitrarily large preprocessing.

P_9 (Error weight with preprocessing). Given a vector $\boldsymbol{s}$ and a number w, find out whether there is a vector $\boldsymbol{y}$ of weight $\le w$ with this syndrome: $H\boldsymbol{y}^T = \boldsymbol{s}^T$.

Theorem 4.5 *A polynomial algorithm for* P_9 *would imply that* $\bigcup_{i\ge 1}\Sigma_i^p = \Sigma_2^p$.

Proof: We shall give a reduction from K_1 to P_9 with the property that every instance U of K_1 can be polynomially transformed to an instance of P_9 in such a way that U has a matching if and only if the question of P_9 can be answered in the affirmative. That this implies our claim will then follow from a general structure theorem on nonuniform complexity, discussed for instance in [33].

Let W be a t-set, $N = t^3$, and let us number the triples $u_j \in W^3$ in a certain fixed way. Form a $((3t+2N)\times 3N)$ matrix

$$H = \begin{bmatrix} & B & \\ & I_k & I_k \\ I_k & & I_k \end{bmatrix},$$

where B is a $3t \times N$ incidence matrix of the set W^3. Let $\boldsymbol{s} = (11\dots1, \boldsymbol{z}_U, \boldsymbol{z}_U)$ be a $(3t+2N)$-vector, where $\boldsymbol{z}_U = (\chi(u_j \in U), 1 \le j \le N)$ is the incidence vector of the set U. Finally, choose $w = t + |U|$.

If U contains a matching M and $\boldsymbol{z}_M$ is its incidence vector, then the vector $\boldsymbol{y} = (\boldsymbol{z}_M, \boldsymbol{z}_M, \boldsymbol{z}_{U\setminus M})$ satisfies the conditions of P_9. Conversely, if there is a vector $\boldsymbol{y} = (\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2)$ with $\mathsf{wt}(\boldsymbol{y}) \le w$ and $H\boldsymbol{y}^T = \boldsymbol{s}^T$, then $B\boldsymbol{y}_1^T = (1\dots1)^T$. However, $\mathsf{wt}(\boldsymbol{y}_0) + \mathsf{wt}(\boldsymbol{y}_2) \ge |U|$ whence $\mathsf{wt}(\boldsymbol{y}_1) \le w - |U| = t$ and therefore,

Lemma 4.3 Let $\boldsymbol{z}$ be a n-vector of ones and twos. Then $B\boldsymbol{z}^T = (11\dots1)^T$ over $\mathbf{Z}_3$ iff $\boldsymbol{z}$ provides a truth assignment with the properties required in K_2.

Proof: For the nonzero ternary numbers a_1, a_2, a_3, the only way that $a_1 + a_2 + a_3 = 1$ is that exactly one of them equals 2 while the two remaining equal 1. ∎

Form the matrix $H_2 = (B|\mathbf{1})$ by augmenting B with the all-one column and put $w = n + 1$. If there exists an assignment $\boldsymbol{z}$ satisfying K_2, then the vector $\boldsymbol{x} = (\boldsymbol{z}|\mathbf{2})$ is annihilated by H_2. Conversely, suppose that there exists a vector $\boldsymbol{x}$ such that $H_2\boldsymbol{x}^T = 0$ and $\mathsf{wt}\,(\boldsymbol{x}) = n + 1$. Then if $x_{n+1} = 2$, the necessary assignment is given by the first n coordinates of $\boldsymbol{x}$. If $x_{n+1} = 1$, then by the lemma, the necessary assignment is given by the vector $\boldsymbol{z} = (2x_1, \dots, 2x_n)$.

(d). Codes and graphs. In Sect. 2 we constructed codes from regular graphs. There is also another way of associating a binary linear code to a graph $G\{V, E\}$, which takes more account of classical linear spaces in graphs. Let $|V| = v$ and $|E| = e$. A *cut* in G is a set of edges such that removing them increases the number of connected components by 1. Characteristic vectors of cuts form a linear subspace E_2^e of the space of all binary words of length e. The dimension of this subspace (code) C is $v - 1$. The dual code $C^{\perp}$ has the parameters $[e, e - v + 1]$ and equals the linear space of *cycles* of G.

Then P_3 is NP-complete for $q = 2$ by an obvious reduction from K_3. For $q = 3$ its completeness follows from part (b).

Proofs of parts (e)–(f) are omitted. ∎

Lemma 4.2 implies one interesting corollary. Namely, it is NP-complete to find out whether a code over a given field $\mathbf{F}_{p^m}$ is MDS or not, i.e., whether its distance is one greater than its redundancy.

We would like to draw the reader's attention to part (e) of this theorem. Problem P_5 accepts as an input string the entire code M, in striking contrast to all other problems considered, where we deal with a concise representation of the code of total length $\mathcal{O}(\log^2 |M|)$. Therefore, a natural question is what is the complexity of computing the covering radius if we are only allowed to store an input of a similar size. Then intuition suggests that the covering radius should be a more difficult problem than, for instance, decoding. Indeed, a naïve way to decode is to take a given vector $\boldsymbol{y}$ and compare it with all codewords in order to find the closest one. To find the covering radius one has to take a vector $\boldsymbol{y}$, compute its distance to the code, and repeat this for *all* vectors outside the code in order to find the farthest-off one.

P_8 (Upper bound on covering radius). Given a linear code C and $w \ge 0$, is it true that

$$\forall_{\boldsymbol{y}\in E_q^n}\ \exists_{\boldsymbol{c}\in C}\ (\mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{c}) \le w)\,? \qquad (4.2)$$

Since the covering radius equals the maximum weight of a coset leader, an equivalent formulation is to ask whether for any given $\boldsymbol{s}$ there is a vector with this syndrome whose weight does not exceed w.

Decision problems involving quantified expressions of type (4.2) are joined into a class Π_2^p. A problem is in Π_2^p if it can be solved in nondeterministic polyno-

The definition of C implies that

$$d_{\square} \geq d_{\mathcal{C}} d_B.$$

Therefore, if $d_{\mathcal{C}} = w + 2$, we have

$$d_{\square} \geq (w+2) d_B.$$

We need to chose d_B so that the r.h.s. of this inequality is strictly greater than that of (4.1). This will be satisfied if

$$\frac{d_B}{n_B} > \frac{w+1}{w+2} \frac{q^{m-1}(q-1)}{q^m - 1} = \frac{q-1}{q} - \frac{q-1}{q} \frac{q^m - w - 2}{(w+2)(q^m-1)},$$

where $w \leq m$. Such codes exist and can be polynomially constructed. Indeed, if the length $n_{\mathcal{C}}$ of the code $\mathcal{C}$ is small, its distance can be found by exhaustive search. If $n_{\mathcal{C}}$ grows, then so do w and m. Then we need a polynomially constructible code with distance approaching $(q-1)/q$. An example is given by concatenating a q^m-ary Reed–Solomon code with a q-ary first-order Reed–Muller code; we omit the details.

(c). Let us reduce K_1 to P_2. Let $|U| = n$. As above, form a $3t \times n$ incidence matrix of U and construct the following $(3t(n+1) \times 3t(n+1) + n)$ matrix:

$$H = \begin{bmatrix} H_1 & \\ I_n & \\ \vdots & I_{3t(n+1)} \\ I_n & \end{bmatrix}.$$

Let $w = 3t^2 + 4t$. Let $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$, where $\boldsymbol{c}_1$ has length n and $\boldsymbol{c}_2$ has length $3t(n+1)$. If U contains a matching, then taking $\boldsymbol{c}_1$ as its incidence vector, we see that for the product Hc^T to vanish, $\boldsymbol{c}_2$ has to have weight $3t + t \cdot 3t$. Conversely, if $\mathsf{wt}\,(\boldsymbol{c}) = w$ and $H\boldsymbol{c}^T = 0$, then we have to prove that the product $\boldsymbol{s} = H_1 \boldsymbol{c}_1^T$ has weight $3t$:

$$\mathsf{wt}\,(\boldsymbol{c}_2) = \mathsf{wt}\,(\boldsymbol{s}) + 3t\mathsf{wt}\,(\boldsymbol{c}_1).$$

Adding $\mathsf{wt}\,(\boldsymbol{c}_1)$ to both parts, we obtain the equation

$$\mathsf{wt}\,(\boldsymbol{c}) = \mathsf{wt}\,(\boldsymbol{s}) + (3t+1)\mathsf{wt}\,(\boldsymbol{c}_1)$$

or

$$3t^2 + 4t = (3t+1)\mathsf{wt}\,(\boldsymbol{c}_1) + \mathsf{wt}\,(\boldsymbol{s}), \quad 0 \leq \mathsf{wt}\,(\boldsymbol{s}) \leq 3t,$$

which admits a unique solution $\mathsf{wt}\,(\boldsymbol{c}_1) = t, \mathsf{wt}\,(\boldsymbol{s}) = 3t$.

The reduction for $q = 2$, $w = n/2$ will be omitted.

If $q = 3$ and $w = n$, we can reduce K_2 to P_2 as follows.

Form the $r \times n$-incidence matrix B of the function $C = c_1 \& c_2 \& \ldots \& c_r$ by putting $b_{ij} = \chi(u_j \in c_i)$. Let $\boldsymbol{z} = (z_1, \ldots, z_n)$ be the vector corresponding to a truth assignment to the variables U, where $z_j = 1$ if u_j is set to *false* and $z_j = 2$ otherwise. The following simple fact is used in the proof.

If it includes the last column, then

$$\det H = \sigma \begin{vmatrix} 1 & 1 & \dots & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \dots & \alpha_{i_w} \\ \alpha_{i_1}^2 & \alpha_{i_2}^2 & \dots & \alpha_{i_w}^2 \\ \vdots & \vdots & & \vdots \\ \alpha_{i_1}^{w-2} & \alpha_{i_2}^{w-2} & \dots & \alpha_{i_w}^{w-2} \\ \alpha_{i_1}^{w-1} & \alpha_{i_2}^{w-1} & \dots & \alpha_{i_w}^{w-1} \end{vmatrix} - \begin{vmatrix} 1 & 1 & \dots & 1 \\ \alpha_{i_1} & \alpha_{i_2} & \dots & \alpha_{i_w} \\ \alpha_{i_1}^2 & \alpha_{i_2}^2 & \dots & \alpha_{i_w}^2 \\ \vdots & \vdots & & \vdots \\ \alpha_{i_1}^{w-2} & \alpha_{i_2}^{w-2} & \dots & \alpha_{i_w}^{w-2} \\ \alpha_{i_1}^{w} & \alpha_{i_2}^{w} & \dots & \alpha_{i_w}^{w} \end{vmatrix}.$$

The first term on the r.h.s. is the Vandermonde determinant $\Delta(i_1 \dots, i_w)$. Let us compute the second term. Changing α_{i_w} to x changes it into a polynomial in x. This polynomial has zeros $\alpha_{i_1} \dots, \alpha_{i_{w-1}}$ and thus, is divisible by $\prod_{j=1}^{w-1}(x - \alpha_{i_j})$. Continuing in this manner, we conclude that the necessary determinant is divisible by $\Delta(i_1 \dots, i_w)$. Comparing degrees, we see that it equals $\Delta(i_1 \dots, i_w)$ times a linear form of α's. Since the answer is symmetrical in α's, this linear form is $\sum_{j=1}^{w} \alpha_{i_j}$ times a constant. This constant must be one since the determinant in question does not have identical terms. Thus,

$$\det H = (\sigma - \sum_{j=1}^{w} \alpha_{i_j})\Delta(i_1 \dots, i_w),$$

which vanishes exactly when P_1'' has a positive answer. ∎

Thus, P_2 is shown to be NP-complete for q^m-ary codes. Note that in the course of this reduction we pass from $\mathbf{F}_q$ to its extension of degree m, where m is a part of the instance. It is known [141] that one can construct extensions (irreducible polynomials) in polynomial time.

The second part is, given an instance of P_2 for q^m-ary codes, to present a polynomial reduction to q-ary codes. This is easy since both codes have alphabets of the same characteristic. To build this reduction, we construct a q-ary concatenated code $C = \mathcal{C} \boxtimes B$ (see the definition in 2.1.2) with $\mathcal{C}$ the outer code. Our objective is, given C, to be able to determine the distance of $\mathcal{C}$ in polynomial time. This will be achieved by imposing restrictions on the parameters of B, which we denote by $[n_B, k_B, d_B]$. We use similar notation for the parameters of $\mathcal{C}$, namely, $[n_\mathcal{C}, k_\mathcal{C}, d_\mathcal{C}]$. Note that for C to be well-defined, k_B should be at least m. A double counting argument similar to Theorem 14.1 in Chapter 1 shows that the distance of C

$$d_\square \le n_B d_\mathcal{C} \frac{q^{m-1}(q-1)}{q^m - 1}.$$

Remember that we have to distinguish the cases $d_\mathcal{C} = w + 1$ and $d_\mathcal{C} = w + 2$. In the first case, we have

$$d_\square \le n_B(w+1)\frac{q^{m-1}(q-1)}{q^m - 1}. \tag{4.1}$$

P_1 that corresponds to the instance of K_1, put $\boldsymbol{s} = (11\dots1)$ and $w = t$. Then if U contains a sought subset and $\boldsymbol{y}$ is its characteristic vector, we have $H\boldsymbol{y}^T = \boldsymbol{s}^T$. Conversely, if for some $\boldsymbol{y}$ of weight $\le t$ this inequality holds true, then clearly $\mathsf{wt}\,(\boldsymbol{y}) = t$. Hence for every $i, 0 \le i \le 3t$, the sum $s_i = \sum_{j=1}^{|U|} y_j h_{ij}$ contains only one nonzero term, i_0 say, and $y_{i_0} = 1$, i.e., $\boldsymbol{y}$ defines a three-dimensional matching in U. To build a reduction for product codes, we can take H as the parity-check for the code B and choose A to be any nontrivial code that contains the all-one vector. We omit the details.

(b). We shall reduce P_1 to P_2. Let m be the number of rows and n the number of columns in H. The interesting case is $w \le m$ since any linear code contains a code vector of weight less or equal than its redundancy plus one. Observe that the reduction in (a) necessarily involves all $|U|$ columns of H; therefore, the proof is unscathed if we replace $\le$ by $=$ in the statement of P_1. We call this version Exact weight of error and denote it by P_1'.

Rephrasing P_1' as in Remark 4.1(iii) above, we conclude that the following problem P_1'' is NP-complete: given a set of elements $A = \{\alpha_1, \alpha_2, \dots, \alpha_n, \sigma\}$ of $\mathbf{F}_{q^m}$, $\sigma \ne 0$, and an integer $w \ge 0$, is there a nonempty subset of *distinct* elements $\{\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_w}\} \subset A$ such that $\sum_{j=1}^{w} \alpha_{i_j} = \sigma$?

Let us reduce P_1'' to P_2. The central part of the proof deals with constructing a code over $\mathbf{F}_{q^m}$ whose minimum distance is different according as the answer to P_2 is *yes* or *no*. This will prove that P_2 is NP-complete over $\mathbf{F}_{q^m}$. Then what remains is to prove that given a q^m-ary code $\mathcal{C}$ one can always construct a q-ary code C such that the distance of $\mathcal{C}$ can be computed in polynomial time given the distance of C.

Let us reduce P_1'' to P_2 for q^m-ary codes. Let $\{\alpha_1, \alpha_2, \dots, \alpha_n, \sigma\}$ be a given instance of P_1''. Form the matrix

$$H = \begin{bmatrix} 1 & 1 & \dots & 1 & 0 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n & 0 \\ \alpha_1^2 & \alpha_2^2 & \dots & \alpha_n^2 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ \alpha_1^{w-2} & \alpha_2^{w-2} & \dots & \alpha_n^{w-2} & 0 \\ \alpha_1^{w-1} & \alpha_2^{w-1} & \dots & \alpha_n^{w-1} & 1 \\ \alpha_1^{w} & \alpha_2^{w} & \dots & \alpha_n^{w} & \sigma \end{bmatrix}.$$

The interesting case is $w \le n$. Then H has the rank $w+1$ and the dimension of the code $\mathcal{C} = \mathsf{ker}\,(H)$ equals $n-w$. The following lemma is a crucial observation.

Lemma 4.2

$$d(\mathcal{C}) = \begin{cases} w+1 & \textit{if there is a subset } \{\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_w}\} \textit{ with} \sum \alpha_{i_j} = \sigma, \\ w+2 & \textit{otherwise.} \end{cases}$$

Proof: Recall that we have assumed that all α's are distinct. Therefore, any $w \times w$ square submatrix of H that does not include the last column, is nonsingular.

q. It remains NP*-complete even if $C = A \otimes B$ is a product code or a concatenated code $A \boxtimes B$ with nontrivial factors.*

(b) P_2 *is* NP*-complete for any given alphabet.*

(c) P_3 *is* NP*-complete for any given alphabet. If $q = 2$, it is NP-complete even if $w = n/2$. If $q = 3$, it is NP-complete even if $w = n$.*

(d) P_4 *is* NP*-complete for $q = 2, 3$.*

(e) Problems P_5–P_6 *are* NP*-complete for $q = 2$.*

(f) P_7 *is* NP*-complete even if $r = 1$.*

Remark 4.1 *(i)* Problem P_1 is closely related to minimum distance decoding. Indeed, any decoding algorithm can be used to solve P_1. Namely, given a syndrome $\boldsymbol{s}$, it is a matter of simple linear algebra to find a vector with this syndrome. Then running the decoding algorithm, we would obtain an answer to P_1. For this reason, one says that minimum distance decoding is NP-*hard*, meaning that it is not in the class NP itself but there is an NP-complete problem (P_1) that can be polynomially reduced to it.

Further evidence of the difficulty of Problem P_1 is given by Theorem 4.6, which says that even approximating the answer up to *any* given constant is NP-complete.

(ii) In the same fashion, Problem P_2 (P_3) is related to the minimum (resp., maximum weight) of the code, Problem P_5 to the covering radius of an (unrestricted) code, and Problem P_6 to the minimal trellis complexity (defined in Sect. 3.4.1).

(iii) For $q = 3$ and $w = n$, Problem P_2 can be also formulated as follows: given n elements of a fixed extension of the field $\mathbf{F}_3$, is it possible to place the signs in the sum $\sum_{i=1}^{n} \pm a_i$ so that it vanishes? This is the same as P_2 if we view columns of H as elements of an extension field. ∎

Turning to the proof of Theorem 4.1, we shall discuss only relatively simple reductions, omitting more difficult parts. The following problems are NP-complete, see Garey and Johnson [71].

K_1 (three-dimensional matching). Given a t-set W and a collection $U \subset W^3$ of triples find out whether there exists a subset $V \subset U$, $|V| = t$, such that for any two distinct triples $v_i, v_j \in V$ the corresponding components are different: $v_{i1} \neq v_{j1}, v_{i2} \neq v_{j2}, v_{i3} \neq v_{j3}$.

K_2 (monotone 1-in-3 SAT). Given a set of Boolean variables $U = \{u_1, \ldots, u_n\}$ and a set of clauses such that each clause contains three literals, none of which are negated, is there a truth assignment to U such that each clause contains exactly one true literal?

K_3 (maximal cut or cycle). Given a simple graph and $w \geq 0$, is there a cut (cycle) with at least w edges?

Proof of Theorem 4.1: (a). We reduce K_1 to P_1. Form a $3t \times |U|$ incidence matrix H of the set U. The columns of the matrix correspond to the triples $u \in U$. Let $u_i = (u_{i_1}, u_{i_2}, u_{i_3})$. Then the ith column of H contains $3t - 3$ zeros and 3 ones in positions $u_{i_1}, t + u_{i_2}, 2t + u_{i_3}$. In order to construct an instance of

4 Intractable Problems

In this section we present NP-complete coding problems and closely related results. Being NP-complete does not imply many practical consequences for a problem. Namely, even if P $\neq$ NP, this does not exclude the existence of simple probabilistic algorithms, algorithms that have a low average-case complexity, or a possibility to easily approximate the exact solution (a noteworthy exception to this being Theorem 4.6).

As discussed in Section 1.2, to prove that a problem $\mathcal{P}$ in NP is complete in it, we have to show that there is an NP-complete problem that can be polynomially reduced to $\mathcal{P}$.

The following is a list of some coding problems.

P_1 (Weight of error). Given a linear code C, an integer w and a vector $\boldsymbol{y}$, is it true that $\mathsf{dist}\,(\boldsymbol{y}, C) \leq w$? Since we want to encode instances of the problem as concisely as possible, we stick to the following formulation: Given a matrix H, a vector $\boldsymbol{s}$, and a nonnegative integer w, is there a vector $\boldsymbol{y}$ of weight $\mathsf{wt}\,(\boldsymbol{y}) \leq w$ such that $H\boldsymbol{y}^T = \boldsymbol{s}^T$?

P_2 (Minimal weight) Given a matrix H and an integer w, is there a nonzero vector $\boldsymbol{c}$ of weight $\leq w$ that belongs to the kernel of H: $H\boldsymbol{c}^T = 0$?

P_3 (Codeword of given weight). In the same setting, is there a vector $\boldsymbol{c} \in \mathsf{ker}\,(H)$ of weight w?

P_4 (Maxweight). In the same setting, is there a vector $\boldsymbol{c} \in \mathsf{ker}\,(H)$ with $\mathsf{wt}\,(\boldsymbol{c}) \geq w$?

P_5 (Farthest-off point). Given a subset $M \subset E_2^n$ and $w \geq 0$, is there a vector $\boldsymbol{y} \in E_2^n$ with $\mathsf{dist}\,(\boldsymbol{y}, M) \geq w$?

P_6 (Partition rank). Given a binary matrix H with n columns, and positive integers w and i, find out whether there is a partition $\mathcal{N} = I \cup (\mathcal{N} \setminus I)$, $|I| = i$, such that

$$\mathsf{rank}\, H(I) + \mathsf{rank}\, H(\mathcal{N} \setminus I) \leq w.$$

P_7 (Perfect code in regular graph). Given a planar 3-regular graph $G(V, E)$ with distance between any 2 vertices defined as the number of edges in a shortest path connecting them, find out whether G contains an r-perfect code, i.e., a subset $A \subset V$ such that any vertex $a \in V \setminus A$ is at a distance at most r of exactly one member of A and any two vertices in A are more that r edges apart.

It is easy to see that problems P_1–P_7 are in NP. As a side remark, we note that problems complementary to P_1–P_7 are not necessarily in NP. Indeed, looking at P_1, suppose we want to check the statement that *all* vectors $\boldsymbol{y}$ with syndrome $\boldsymbol{s}$ have weight $> w$. We are not aware of a way to check this in polynomial time. Problems complementary to the decision problems from the class NP form the class coNP.

Various results about the NP-completeness are collected in the following theorem.

Theorem 4.1 *(a)* P_1 *is* NP*-complete for any fixed size of the ground alphabet*

discussed in that paper, uses a bounded distance decoding algorithm to test vectors generated in the course of the search. This procedure has a smaller complexity for codes of high rate than the sparse-sets search.

The second method discussed in this section is due to Desaki et al. [50]. The main result of this work are tables of weight spectra for the extended BCH codes of length 128, which proves Theorem 3.52.

Other algebraic methods for the computation of the weight spectrum of cyclic codes are discussed in Charpin [38].

The idea is to single out a subcode of $C \cap R(m, r-1)$ and to reduce the computation to inspecting one of its cosets in C. Let $\Delta F = \{F_{\boldsymbol{a}},\ \boldsymbol{a} \in E_2^m\}$, where $F_{\boldsymbol{a}}$ is the polynomial obtained by deleting from $F(\boldsymbol{x}+\boldsymbol{a}) - F(\boldsymbol{x})$ all monomials of degree up to $r-2$. If $F \in C$, then clearly ΔF is a linear subcode of $C \cap R(r-1, m)$. Then the common part of C and $R(r-1, m)$ can be represented as a direct sum

$$C \cap R(r-1, m) = \Delta F \oplus C(F)$$

for some linear subcode $C(F) \subset C$.

An easy argument shows that for any $F \in C$, $\deg F = r$, the number of codewords of weight w in the coset $F + C \cap R(r-1, m) = F + C(F) + \Delta F$ equals $|\Delta F| A_w(F + C(F))$. This leads to the following theorem.

Theorem 3.51 *Let C be a code of order r equivalent to an affine-invariant extended cyclic code C' and $C_0 \subset C$ a code equivalent to an extended irreducible subcode of C' with nonzero α^u, $(u, n) = 1$. If $C = C_0 \oplus C \cap R(r-1, m)$, then*

$$A_w(C) = |\Delta F| n A_w(F + C(F)) + A_w(C \cap R(r-1, m))$$

where $F \in C$ is any m-variate polynomial of degree r and $C(F)$ is a linear subcode that satisfies $C \cap R(r-1, m) = C(F) \oplus \Delta F$. ▮

This general method was suggested by Desaki et al. [50] and applied to the extended BCH codes of length 128, leading to the following nice result.

Theorem 3.52 (Desaki et al. [50]) *The weight distribution of all binary primitive BCH codes of length 127 is known.* ▮

3.5.3 Notes

3.5.1. The first part of this section is adapted from Leon's paper [108], which was written independently of the "decoding line." Leon also points out that the implementation complexity of Algorithm A can be reduced if we know that if the code has one word of weight d, it has a certain number a_d of them (for instance, if we know the automorphism group of the code). The table of quadratic residue codes is also from [108]. The table of BCH codes of length 511 is from Canteaut and Chabaud [36]. Minimum-weight vectors in these codes were found using the technique similar to that of Example 3.4. There remain 6 codes of length 511 ($k = 268, 259, 238, 229, 202, 148$) which, because of the large search, cannot be analyzed by the algorithm. For all other BCH codes of this length the true distance is known [36].

The second algorithm and Theorem 3.44 are from Barg and Dumer [21]. Other algorithms for bounding the minimum distance of cyclic codes are discussed in Massey and Schaub [119], Augot [12].

3.5.2. The first part of this section (up to Theorem 3.47) is based on a paper by Barg and Dumer [21]. Another procedure based on similar ideas,

if there is a weight-preserving mapping that acts transitively on the cosets, we can compute the weight spectrum of C by finding the weight distribution in one of the cosets.

For instance, suppose that $C = C_0 \oplus C_1$, where C_1 is a cyclic subcode and

$$C_0 = \{0\} \cup \{n \text{ cyclic shifts of a vector } \boldsymbol{a}\}.$$

Then the quotient C/C_1 consists of $(n+1)$ cosets, namely C_1 and $T(\boldsymbol{a}) + C_1 = T(\boldsymbol{a} + C_1)$, where T is the cyclic permutation, $T = (n, 1, 2, \ldots, n-1)$. For C_0 we can take an irreducible subcode of C with nonzero α^u, $(u, n) = 1$. This gives us the following.

Lemma 3.49 *Let $C = C_0 \oplus C_1$, where C_0 is irreducible, and let $\boldsymbol{a}$ be any fixed codeword of C_0. Then for the number $A_w(C)$ of codewords of weight w in C we have*

$$A_w(C) = nA_w(C_1 + \boldsymbol{a}) + A_w(C_1).$$

∎

This lemma provides a reduction in complexity of computing A_w by about n times compared to exhaustive search. However, many extended primitive cyclic codes, for instance, the BCH codes, are invariant under a larger set of permutations than the powers of T, namely, the affine permutations given by the action of $AGL(2^m, 1)$ on coordinates. (see Chapter xx (Huffman)). Label the coordinates of C as

$$X_0 = 0,\, X_j = \alpha^j, 0 \le j \le 2^m - 1, \tag{3.22}$$

where α is an element of maximal order in $\mathbf{F}_{2^m}$. Then the code is affine-invariant if it is preserved as a subset of E_2^n by a set of $(2^m-1)2^m$ permutations $X_j \mapsto aX_j + b$ with $a \neq 0$. Therefore, we can further reduce the complexity of computing A_w if we manage to describe the partition of cosets according to the action of $\mathrm{AGL}(2^m, 1)$.

Every extended cyclic code is a subcode of the Reed–Muller code $R(r, m)$ for some r. Working with Reed–Muller codes, we shall also use the lexicographic arrangement of coordinates which is obtained if we fix a basis of $\mathbf{F}_{2^m}$ over $\mathbf{F}_2$ and represent every X_j as an m-vector. Any 2^m-vector $\boldsymbol{v} = (v_0, v_1, \ldots, v_n)$ can be represented as the set of values of a certain m-variate polynomial $F(\boldsymbol{x})$ with $\boldsymbol{x}$ ranging over E_2^m. Then r is the maximal degree of polynomials defining codewords of C, and $C \subseteq R(r, m)$, $C \not\subseteq R(r-1, m)$. For brevity we say that C has order r. If the vector $(F(\boldsymbol{x}),\, \boldsymbol{x} \in E_2^m) \in C$, we write simply $F \in C$.

The following obvious lemma forms a basis for the simpler computation of weight spectra.

Lemma 3.50 *Let C' be an extended cyclic code with coordinates ordered cyclically and C the same code with coordinates rearranged lexicographically. If C' is affine-invariant, then*

$$(F(\boldsymbol{x}) \in C) \;\Rightarrow\; \forall_{\boldsymbol{a} \in E_2^m} (F(\boldsymbol{x} + \boldsymbol{a}) \in C).$$

∎

Algorithm 3.12: Sparse-sets search

- Set $h_{ij} = 0$, $1 \le i, j \le n$.
- Choose a k-sparse information vector $\boldsymbol{m}$ and compute $\boldsymbol{a} = (\boldsymbol{m} \,|\, \boldsymbol{m}A)$.
- If $\mathsf{wt}(\boldsymbol{a}) = w$, find the period i of $\boldsymbol{a}$ and the number j of its cyclic shifts containing k-sparse information sets. Increase the number h_{ij} of such vectors by 1.
- Repeat the last two steps for all k-sparse vectors $\boldsymbol{m}$.
- Compute the number of vectors of weight w as

$$A_w = \sum_{i=1}^{n} i \sum_{j=1}^{n} h_{ij}/j.$$

Clearly, every vector that contains j distinct k-sparse information sets, will be generated j times. Therefore, h_{ij} equals j times the number of distinct cyclic representatives of period i.

The complexity of this algorithm is estimated in the following theorem. Its proof is technical and will not be included.

Theorem 3.47 *The sparse-sets search computes the weight spectrum of any binary cyclic code. For a given w it can be implemented by a sequential algorithm that finds A_w with time complexity $\mathcal{O}(nM(k,w))$, where*

$$\frac{1}{k}\sum_{i=1}^{s}\binom{k}{i} \le M(k,w) < \frac{\lambda}{\lambda-2}\left(\frac{1}{k} + \frac{1}{s+1}\frac{\lambda-1}{\lambda-2}\right)\binom{k}{s}, \qquad s = \lfloor kw/n \rfloor.$$

▮

Of course, if C contains the all-one vector, then $A_w = A_{n-w}$. The following lemma sometimes facilitates the computation of the weight spectrum for cyclic codes.

Lemma 3.48 *Suppose the extended cyclic binary code C^{ex} is invariant under a transitive group. Then*

$$A_{2w} = A_{2w-1}\frac{n-2w+1}{2w}, \qquad 1 \le j \le (n-1)/2.$$

Proof: See Chapter 1, Theorem 10.12. ▮

Therefore, the weight distribution of C is determined by that of C^{ex} and vice versa.

The second method that we plan to discuss makes a different use of the cyclic invariance. We restrict ourselves to *primitive* codes, i.e., those with $n = 2^m - 1$. The idea is to factor C into cosets with respect to a subcode $C_1 \subset C$. Then

the set of all subvectors $\{\boldsymbol{a}(I(i,m)),\ 0 \le i \le n-1\}$. Then

$$\sum_{i=0}^{n-1} \mathsf{wt}\,[\boldsymbol{a}(I(i,m))] = mw,$$

since every 1 of $\boldsymbol{a}$ is contained in m subvectors. Thus the average weight of subvectors of length m equals $mw/n = m/\lambda$.

For a vector $\boldsymbol{a}$, let us call an arbitrary segment $I(i,m)$ *heavy*, if the weight of the subvector $\boldsymbol{a}(I(i,m))$ is greater than the average weight m/λ. Let us call a segment $I(i,\ell_r)$ *good* if all the subvectors $\boldsymbol{a}(I(i,\ell_p)),\ 1 \le p \le r$, are light (not heavy). Otherwise, let us call it j-*bad*, where j is the smallest integer such that $I(i,\ell_j)$ is a heavy segment.

To prove the lemma, we describe a procedure that necessarily produces a good segment. Consider the segment $I(0,\ell_r)$ and suppose it is j_1-bad. Then consider the segment $I(\ell_{j_1},\ell_r)$. Suppose it is j_2-bad and consider the segment $I(\ell_{j_1}+\ell_{j_2},\ell_r)$. We continue inspecting the segments $I(\ell_{j_1}+\cdots+\ell_{j_s},\ell_r),\ s = 1,2,\ldots,$ until we find a good segment.

This procedure always stops. For if not, then we cover the ring a number of times with heavy segments of length $\ell_{j_1},\ell_{j_2},\ldots$. The multiset $U = (0,\ell_{j_1},\ell_{j_1}+\ell_{j_2},\ldots)$ of their starting points contains a finite number of different elements (chosen out of $(0,\ldots,n-1)$). Therefore after a number of revolutions (ℓ_r or less) the set U will contain identical elements. This means that we have covered integer number of circles with heavy segments, i.e., that $\mathrm{wt}(\boldsymbol{a}) > w$. This is absurd. ∎

Let $m_1,\ldots,m_w$ be the locations of the w ones in $\boldsymbol{a}$.

Corollary 3.46 *Every cyclic w-class contains a vector $\boldsymbol{a}$ with*

$$m_i \ge \lceil i\lambda \rceil, \quad 1 \le i \le w.$$

∎

Of the total of w restrictions in this corollary, $\lfloor k/\lambda \rfloor$ apply to the first k coordinates of $\boldsymbol{a}$. Namely, they imply that in $\boldsymbol{a}$ the first one is in or on the right of position $\lceil \lambda \rceil$, the second in or on the right of position $\lceil 2\lambda \rceil$ and so forth. This motivates the following definition. If a binary vector of length k satisfies $\lfloor k/\lambda \rfloor$ restrictions of this corollary, we call it k-*sparse*. Since every cyclic class contains a code vector whose first part is k-sparse, we can capture all cyclic w-classes by encoding all k-sparse vectors $\boldsymbol{m}$ of length k. We have assumed that the code is in the systematic form; therefore, encoding amounts to computing parity checks for every $\boldsymbol{m}$. Note that k-sparse vectors are easy to generate "on-line."

Theorem 3.44 *Let $C[n,k]$ be a linear code with a simple decoding algorithm of complexity* $\mathsf{dec}\,(n,k,e)$ *correcting e errors. Then it is possible to find its minimum distance by running Algorithm* 3.11 *not more than n times. The probability that the procedure gives a correct answer after $\ln\left(\binom{n}{d}/\epsilon\right)\binom{n}{d-e}/\binom{d}{e}$ iterations on each run is at least $1-\epsilon$. If $e \sim d/2$, then the asymptotic complexity of the procedure equals*

$$\mathcal{O}(2^{nH_2(\delta/2)}\mathsf{dec}\,(n,k,e)).$$

▮

Finally, we note that in view of Theorem 3.29 both algorithms of this section can be implemented by *deterministic* procedures.

3.5.2 Weight spectrum

Let C be a binary $[n,k,d]$ code and A_w the number of codewords of weight w in it. In this section we study the complexity of finding A_w first in general linear codes and then in cyclic codes.

First, note that since algorithms of the previous section construct a covering, they find all of the codewords of weight d, or, more generally, of any given weight w (if $w > n/2$, some obvious changes should be made). A corresponding reformulation is easy and is left to the reader.

In this section we present two algorithms for computing the weight spectrum of cyclic codes. For some cyclic codes this can be found using algebraic methods (see notes to this section). For long binary BCH codes the weight spectrum is known to be approximated by the quantity N_w from (1.7) with reasonable precision. Otherwise the problem is wide open.

For simplicity we work with binary cyclic codes. Suppose the generator matrix of C is represented in the form $G = [I_k|A]$. For a given vector $\boldsymbol{a}$, we consider the set of all its n cyclic shifts and call this a *cyclic class*. The number of distinct vectors in this set is called the *period* of $\boldsymbol{a}$. Moreover, if $\mathsf{wt}\,(\boldsymbol{a}) = w$, we speak of a cyclic w-class. Let $\lambda = n/w$ be the average distance between neighboring ones of the vector $\boldsymbol{a}$. By averaging, we see that for a given k-subset $I \subset \mathcal{N}$ of *consecutive* coordinates, every cyclic w-class contains a vector $\boldsymbol{a}$ with $\mathsf{wt}\,(\boldsymbol{a}(I)) = \lfloor k/\lambda \rfloor$. Therefore, if I is an information window, we can capture all codewords of weight w by encoding all possible combinations of $\lfloor k/\lambda \rfloor$ ones on it. The following lemma develops this idea. In its statement and proof we label the coordinate set with numbers $0, 1, \dots, n-1$. Let $I(i,j) = (i, i+1, \dots, i+j-1)$, where the indices are, if necessary, reduced modulo n.

Lemma 3.45 *For any r numbers $\ell_1, \dots, \ell_r$ with $1 \le \ell_1 < \cdots < \ell_r \le n$ every cyclic w-class $\mathcal{Z}$ contains a vector $\boldsymbol{a}$ with*

$$\mathsf{wt}\,[\boldsymbol{a}(I(0,\ell_j))] \le \lfloor \ell_j/\lambda \rfloor \quad \textit{for all } j = 1, 2, \dots, r.$$

Proof: Write the coordinates $a_0, \dots, a_{n-1}$ of $\boldsymbol{a}$ on a ring with n cells. Consider

Example 3.11 In Sect. 3.3.4 we discussed a simpler implementation of covering set decoding related to savings on diagonalizations of the generator matrix. Obviously, this method can be applied to finding low-weight vectors in any linear code. A challenging instance is determining the true distance of long primitive BCH codes. The starting value is the designed distance. The first length for which the code table is not complete is $n = 511$. Application of this method to these codes gives the following results.

k	385	358	340	331	304	193
d	29	37	41	43	51	87

For all six codes, the true minimum distance equals the designed distance. Therefore, though the algorithm is probabilistic, the answer is given with probability 1 (unlike the case with the QR codes in the previous example when the square root bound is far below the weight of the found codewords). ∎

The second procedure that we are going to discuss is well-suited for codes that admit a simple polynomial decoding algorithm $\psi(C)$ correcting up to e errors (for instance, BCH codes, Goppa codes, etc.). Let $\mathsf{dec}(n, k, e)$ be its complexity.

The idea is to apply $\psi(C)$ to vectors of weight $d - e$. If we successfully "correct" e errors, this results in a codeword of weight d. The major difference with the above approach is that we can work with coverings of subsets of smaller size, $d - e$ instead of d. Not to miss codewords, we would like to choose a family of $(d - e)$-sets such that every d-subset contains at least one of them. Let

$$T(n, m, t) = \{\mathcal{F} \subseteq \mathcal{N}^t : \forall_{E \in \mathcal{N}^m} \exists_{F \in \mathcal{F}} (F \subset E)\}. \tag{3.21}$$

In words: $T(n, m, t)$ is a family $\mathcal{F}$ of t-subsets of $\mathcal{N}$ such that every m-subset of $\mathcal{N}$ contains at least one of them.

Remark 3.12 Coverings of this type are called Turán designs. Comparing with the definition of $M(n, m, t)$ in Sect. 3.3.4, one can notice that complements of the sets in an $M(n, m, t)$ covering design form a Turán design $T(n, n-t, n-m)$.

We shall construct a covering $T(n, d, d - e)$ by a repeated random choice of $(d - e)$-sets.

> **Algorithm 3.11: Finding minimum-weight codewords, B**
>
> - Repeat $\ln\left(\binom{n}{d}/\epsilon\right)\binom{n}{d-e}/\binom{d}{e}$ times
> - Choose randomly a $(d - e)$-subset $W \subset \mathcal{N}$.
> - For every vector $\boldsymbol{y}$ with $\boldsymbol{y}(\overline{W}) = 0$, decode $\boldsymbol{y}$ with $\psi(C)$. If it outputs a codeword of weight d, STOP.

Its proof, which we wish to spare the reader, again builds on the argument familiar from Theorem 3.24.

Algorithm 3.10: Finding minimum-weight codewords, A

- Repeat $\ln(1/\epsilon)\binom{n}{s}/\binom{n-d}{s}$ times.
- Choose randomly an s-subset $W \subset \mathcal{N}$. Form a list of codewords $K(W) = \{\boldsymbol{c} \in C \mid \boldsymbol{c}(W) = 0\}$.
- If $K(W)$ contains a codeword of weight d, STOP.

The number of iterations of the algorithm is discussed in the next theorem. Observe that there is actually no difference between this algorithm and decoding. The complexity of this algorithm is again related to the probability of constructing a covering and to the size of the list $K(W)$. If $s = k-1$, then asymptotically this size is at most $q^{\sqrt{n}}$ (Corollary 3.8; in practical examples it is much smaller). Thus, we arrive at the following result.

Theorem 3.43 *Let $C[n,k]$ be a linear code. Then it is possible to find its minimum distance by running Algorithm* 3.10 *not more than n times. To guarantee an error probability of at most ϵ, it is sufficient to perform during each run* $\ln(1/\epsilon)\left[\binom{n}{s}/\binom{n-d}{s}\right]$ *iterations of the algorithm. The asymptotic complexity of the entire procedure is*

$$2^{n[H_2(\delta)-(1-R)H_2(\delta/(1-R))](1+o(1))}.$$

Proof: The proof is similar to that of Theorem 3.24. The probability of constructing a covering after the cited number of iterations equals $1 - e^{-\ln(1/\epsilon)} = 1 - \epsilon$. The asymptotic expression follows by taking $s \sim k$ since $\binom{n}{s}/\binom{n-d}{s} = \binom{n}{d}/\binom{n-s}{d}$. ∎

Example 3.10 Practically it may be advantageous to allow a few nonzero entries within the s coordinates. This yields a saving on the Gaussian elimination, which is then performed only once for large groups of codewords. If the number of these nonzero entries is ℓ, then we need to construct a covering system $N_2(n; s, n-s; \ell, d-\ell)$ (see Remark 3.8(ii)). This method has been applied to quadratic residue codes, yielding the following results.

n	d	ϵ	n	d	ϵ
311	36	10^{-100}	281	36	10^{-100}
359	40	10^{-100}	313	40	10^{-20}
367	48	10^{-20}	337	40	10^{-20}
383	48	10^{-10}	353	42	10^{-20}
431	48	10^{-10}	401	42	10^{-20}
439	48	10^{-10}	409	48	10^{-3}
463	$48-56$	10^{-3}	433	38	10^{-20}
479	$48-56$	10^{-3}	449	$48-56$	10^{-2}
487	$48-56$	10^{-3}	457	$48-56$	10^{-2}
503	$48-56$	10^{-3}	521	$48-54$	10^{-2}

and Be'ery [160], see also references in these papers.

Bounded distance decoding can also be extended to the soft-decision case, where it is known as generalized minimum distance decoding, see Berlekamp [27], Kötter [97], Kovalev [98], Sorger [149].

3.5 Computing numerical parameters of codes

Among many numerical parameters of the code, such as the minimum distance, weight spectrum, covering radius, the number of nonzero weights, higher distances, coset weight distribution etc., we choose to treat only the first two, the reason being that these already allow us to demonstrate applications of the known techniques and that the algorithms for other problems that we can construct are more or less trivial extensions of the ones being discussed.

3.5.1 Minimum distance

The only known way to compute the distance of a general linear code C is to look for codewords of small weight. In this form the problem is very close to bounded distance decoding up to half the minimum distance. Let C be a linear code.

Lemma 3.42 *Any algorithm that finds a minimum weight codeword in a general linear code, can also perform decoding up to $\lfloor (d-1)/2 \rfloor$ errors.*

Proof: Let $\boldsymbol{y} = \boldsymbol{c}+\boldsymbol{e}$ be a received vector, $c \in C$. Then $\boldsymbol{e}$ is linearly independent of C and is therefore a unique codeword of minimal weight in the code $\langle C, \boldsymbol{y} \rangle$. ∎

Errors of larger weight will also satisfy this lemma except that they need not to be unique. Therefore, to find them, one would need an algorithm that finds *all* minimum-weight codewords in the code.

The covering technique enables one to compute the distance of C with complexity comparable to that of covering set decoding. The general strategy is as follows. Let d_0 be a lower bound to the actual distance d of C. Then one should run the search for codewords of weight $d_0, d_0+1, \ldots$, until a codeword of weight d is found. If this search is done by a probabilistic algorithm with failure probability ϵ, we say that the distance of C equals d with probability $1-\epsilon$.

Since $d \le n-k+1$, every weight d codeword $\boldsymbol{c}$ vanishes on at least $k-1$ coordinates. The idea is therefore to fix some $s \le k-1$ coordinates of $\boldsymbol{c}$ to 0 and solve the system of linear equations $H\boldsymbol{c}^T = 0$ with respect to the remaining unknown coordinates. This can be done, for instance, by diagonalizing the generator matrix G in order to obtain only one nonzero element in the chosen s columns and taking all possible linear combinations of the remaining $k-s$ rows. If the repeated choice of the $\boldsymbol{s}$ coordinates exhausts a covering $M(n, n-d, s)$, we shall certainly find a codeword of weight d.

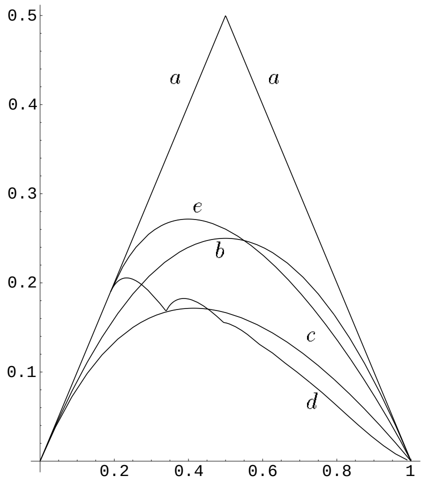


Figure 3.3: Complexity of soft-decision decoding algorithms for binary codes (x-axis—code rate, y-axis—complexity exponent):
(a) $\min(R, 1-R)$,
(b) weighted-partitions decoding, Theorem 3.40,
(c) Theorem 3.41.
(d) lower bound on complexity of syndrome trellis decoding, see Corollary 3.33 and the following discussion,
(e) lower bound on complexity of syndrome trellis decoding under the assumption that the binary asymptotic GV bound is tight.

path algorithm in graph theory. Theorem 3.31 is due to Zyablov and Sidorenko [171]. They also use the argument in its proof to derive a particular case ($m = 1$) of the asymptotic lower bound in Corollary 3.33. The general case (Theorem 3.32 and its corollary) is due to Lafourcade and Vardy [106].

3.4.2. This section follows rather closely two recent works by Dumer, [54], [55]. Paper [54] contains a more detailed discussion of sufficient conditions for a channel to yield the estimate (3.19) for p_{ϕ_N}. Algorithm 3.9 [54] is a generalization of a similar decoding for the case of (narrow-sense) q-ary symmetric channel due to Evseev [60]. Theorem 3.41 is from [55]. The material covered in this section presents an important development in coding theory and is currently a subject of ongoing research.

Different other decoding algorithms and examples for specific short codes are discussed in Gazelle and Snyders [72], Fossorier and Lin [64], Han et al. [82], Kudryashov and Zakharova [103], Miyakawa and Kaneko [123], Vardy [159], Vardy

Algorithm 3.9: Weighted-partitions decoding

- Set $\boldsymbol{c}_0 = 0$. Compute the $q \times n$ matrix $W = [w_i(a)], 0 \le i \le n-1, a \in X$.
- Run the Euclidean Division Algorithm on $s_0 = n, s_1 = s$ to find sequences $(s_1, \ldots, s_m), (a_1, \ldots, a_m)$, and the number m. Put $\mathfrak{S} = \mathcal{E}$ if m is even and $\mathfrak{S} = \mathcal{O}$ if m is odd.
- Take the segmentation $Z = \cup Z_u$ that corresponds to an element $\mathfrak{S}_j^{(i)} \in \mathfrak{S}$. Compute lists $X(Z_u)$ formed by $N^{|Z|_u/n}$ lightest subvectors on the segments Z_u.
- Form the list $X(Z) = X(Z_1) \times \cdots \times X(Z_u) \times \cdots$ by joining the lists constructed on the previous step.
- Use G to find an information set on s coordinates that correspond to the current segment Z. Form the list $L(X(Z))$ by encoding k coordinates of this information set for every vector in $(X(Z))$. If there is a vector $\boldsymbol{c} \in L(X(Z))$ with $w_{\boldsymbol{y}}(\boldsymbol{c}) < w_{\boldsymbol{y}}(\boldsymbol{c}_0)$, assign $\boldsymbol{c}_0 \leftarrow \boldsymbol{c}$.
- Repeat the previous 3 steps for every $\mathfrak{S}_j^{(i)} \in \mathfrak{S}$, $0 \le j \le n, 1 \le i \le \lfloor m/2 \rfloor + 1$. Output $\boldsymbol{c}_0$.

Finally, note that any cyclic code trivially satisfies Lemma 3.7. ∎

A further improvement of this result is based on two ideas, Theorem 3.36 and the (punctured) split syndrome decoding algorithm of the previous section. We formulate this result without further comments.

Theorem 3.41 *Maximum likelihood decoding for almost all linear codes used over any symmetric channel can be performed with both time and space complexity* $q^{n[R(1-R)/(1+R)](1+o(1))}$. ∎

Figure 3.3 shows the complexity exponents of the algorithms of this section ($q = 2$). We see that weighted-partitions decoding improves the complexity of syndrome trellis decoding for almost all codes and all code rates. Moreover, the complexity of this algorithm falls below the lower bound for trellis decoding for low code rates, for almost all linear codes and all cyclic codes. If the asymptotic GV bound is tight, the result of Theorem 3.41 ensures simpler decoding than the syndrome trellis algorithm for all linear codes and all code rates R, $0 < R < 1$.

3.4.3 Notes

Channels and maximum likelihood decoding are studied in Gallager [70].

3.4.1. Trellis decoding was introduced in Bahl et al.[18], Wolf [166] (see Chapter xx (Vardy) for the history and more results). Algorithm 3.8 is known in coding theory as the Viterbi algorithm and is very close to Dijkstra's shortest

(i). If there is a segment of length 5 such that the corresponding subvector of $\boldsymbol{x}$ is light, we are done.
(ii). Otherwise, if no segment in $\mathfrak{S}_j^{(1)}$ is good, consider segmentations of $\mathcal{N}' = \{0, \ldots, 17\}$ corresponding to the first step of the division in (3.20). By Lemma 3.38, for any $j \in [0, 17]$ the segment of length 3 starting at position j is good.
(iii). Now consider segmentations $\mathfrak{S}_j^{(3)}$. If there is a j such that the segment of length 2 is good, this yields a well-decomposable 5-segment.
(iv). Otherwise, according to the third step of the division, all 1-segments of $\mathcal{N}'$ are light, and we can pick a well-decomposable segment from $\mathfrak{S}_j^{(5)}$. ∎

Proof of Lemma 3.9: By now the proof should be clear from the example and can be left to the reader. The impact of the parity of m is understood if one considers stopping in any even step of the example provided that the corresponding division is exact. ∎

We are now ready to construct the algorithm. Its idea is quite in the spirit of algorithms in Section 3.3. Namely, we propose to find a well-decomposable s-segment, build lists of subvectors on subsegments of this segment and join them into one list. Every subvector in this list can be used to reconstruct a codeword according to Lemma 3.7. The algorithm outputs the lightest codeword among the inspected ones.

The input data of the algorithm are the received vector $\boldsymbol{y}$, $n, k, s = k + \lceil 2 \log_q n \rceil$, the number N, which controls the error probability, and the generator matrix G of C. For given i and j, the element $\mathfrak{S}_j^{(i)}$ defines a partition of the segment $Z = \{j, j+1, \ldots, j+s-1\}$, which we will again write in the generic form $Z = \cup Z_u$.

The algorithm (weighted-partitions decoding) appears below in a tabular form. Its properties are given by the following theorem.

Theorem 3.40 *For all long linear codes and any memoryless symmetric channel the decoding error probability of weighted-partitions decoding is equivalent to the error probability of ML decoding. For almost all linear codes and all cyclic codes the time complexity of its sequential implementation is at most* $q^{nR(1-R)(1+o(1))}$.

Proof: The first part of the statement follows from Theorem 3.35. The complexity of the algorithm is dominated by the size of the list $X(Z)$. This size equals
$$N^{(\sum_u |Z_u|)/n} = N^{s/n}.$$

The total number of segmentations examined is at most $n(\log_2 n + 1)$. The most time-consuming among nonexponential steps is finding the information set within the given s coordinates, which takes time $\mathcal{O}(n^3)$. Therefore, if we put $N = q^{n(1-R)\log n}$, the total time complexity is bounded from above as

$$\mathcal{O}\left(n^4 (\log_2 n) \left(q^{n(1-R)} \log n\right)^{k + 2\log_q k}\right) = q^{k(1-R)(1+o(1))}.$$

Euclidean Division Algorithm to find $s_m = gcd(s_0, s_1)$:

$$\begin{aligned} s_0 &= a_1 s_1 + s_2, \\ s_1 &= a_2 s_2 + s_3, \\ &\dots \\ s_{m-2} &= a_{m-1} s_{m-1} + s_m, \\ s_{m-1} &= a_m s_m. \end{aligned}$$

As is well known, if $\frac{n}{s} = \frac{s_m \nu}{s_m \sigma}$, then $m \le \log_2 \sigma + 1$.

Below we consider two sequences of segmentations depending on the parity of m. For m even let

$$\begin{aligned} &\mathcal{E} = \{\mathfrak{S}_j^{(1)}, \mathfrak{S}_j^{(3)}, \dots, \mathfrak{S}_j^{(m+1)}\}, \quad 0 \le j \le n-1, \\ &\mathfrak{S}_j^{(1)} = \big((1, s_1)\big), \\ &\mathfrak{S}_j^{(3)} = \big((1, s_3), (a_2, s_2)\big), \\ &\mathfrak{S}_j^{(5)} = \big((1, s_5), (a_4, s_4), (a_2, s_2)\big), \\ &\dots \\ &\mathfrak{S}_j^{(m-1)} = \big((1, s_{m-1}), (a_{m-2}, s_{m-2}), \dots, (a_2, s_2)\big), \\ &\mathfrak{S}_j^{(m+1)} = \big((a_m, s_m), (a_{m-2}, s_{m-2}), \dots, (a_2, s_2)\big). \end{aligned}$$

For m odd let $\mathcal{O} = \{\mathfrak{S}_j^{(1)}, \mathfrak{S}_j^{(3)}, \dots, \mathfrak{S}_j^{(m)}\}$, $0 \le j \le n-1$, where this time

$$\mathfrak{S}_j^{(m)} = \big((1, s_m), (a_{m-1}, s_{m-1}), \dots, (a_2, s_2)\big)$$

and all other segmentations are the same as above.

Lemma 3.39 *Let $\boldsymbol{y}$ be the received vector. For any s, n and $\boldsymbol{x} \in X^n$ there is at least one well-decomposable segment in the set $\mathcal{E}$ for m even and $\mathcal{O}$ for m odd.*

Though the notation may seem complicated, the idea is actually quite simple and beautiful. Before proving the lemma, let us consider an example.

Example 3.9 Let $n = 18$, $s = 5$, then $(a_1, a_2, a_3, a_4) = (3, 1, 1, 2)$ and $(s_1, s_2, s_3, s_4) = (5, 3, 2, 1)$ (i.e.,

$$\begin{aligned} 18 &= 3 \cdot 5 + 3 \\ 5 &= 1 \cdot 3 + 2 \\ 3 &= 1 \cdot 2 + 1 \\ 2 &= 2 \cdot 1.) \end{aligned} \tag{3.20}$$

Then $\mathcal{E} = \big(\mathfrak{S}_j^{(1)}, \mathfrak{S}_j^{(3)}, \mathfrak{S}_j^{(5)}\big)$, $0 \le j \le n$, where

$$\begin{aligned} \mathfrak{S}_j^{(1)} &= \big((1, 5)\big), \\ \mathfrak{S}_j^{(3)} &= \big((1, 2), (1, 3)\big), \\ \mathfrak{S}_j^{(5)} &= \big((2, 1), (1, 3)\big). \end{aligned}$$

Lemma 3.37 *Let $\boldsymbol{x} \in X^n$ be a vector and let $I \subseteq \mathcal{N}'$, $J \subseteq \mathcal{N}'$, $I \cap J = \emptyset$. Then*

$$\#_{\boldsymbol{y}}(\boldsymbol{x}(I \cup J)) \geq \#_{\boldsymbol{y}}(\boldsymbol{x}(I))\#_{\boldsymbol{y}}(\boldsymbol{x}(J)).$$

Proof: Let $\mathfrak{U} = \{(\boldsymbol{a}_1, \boldsymbol{a}_2) \in X^{|I|} \times X^{|J|} \mid \boldsymbol{a}_1 \preceq_I \boldsymbol{x}(I) \ \& \ \boldsymbol{a}_2 \preceq_J \boldsymbol{x}(J)\}$. Then $|\mathfrak{U}| = \#_{\boldsymbol{y}}(\boldsymbol{x}(I))\#_{\boldsymbol{y}}(\boldsymbol{x}(J))$. Considering $\mathfrak{U}$ as a subset of $X^{|I|+|J|}$, we observe that $(\boldsymbol{x}(I)|\boldsymbol{x}(J))$ is the last vector in it. Hence its number in the order $\preceq_{I \cup J}$ is at least the size of $|\mathfrak{U}|$. ∎

We shall look for light subvectors by averaging over certain partitions of $\mathcal{N}'$. Since ranks enjoy the property just proved, we shall take geometric means. This motivates the following definition. For a given $\boldsymbol{x} \in X^n$, and a segment $I \in \mathcal{N}'$, the subvector $\boldsymbol{x}(I)$ is called *light* if $\#_{\boldsymbol{y}}(\boldsymbol{x}(I)) \leq \#_{\boldsymbol{y}}(\boldsymbol{x})^{|I|/n}$. In this case we sometimes also call the segment I *good* (w.r.t. $\boldsymbol{x}$). The following lemma asserts that any vector $\boldsymbol{x}$ contains a light subvector.

Lemma 3.38 *Let $\mathcal{N}' = \cup Z_u$ be a partition of $\mathcal{N}'$ into pairwise disjoint segments. Then for any $\boldsymbol{x} \in X^n$ it contains a segment Z such that the subvector $\boldsymbol{x}(Z)$ is light.*

Proof: For suppose not. Then by the previous lemma and definition,

$$\#_{\boldsymbol{y}}(\boldsymbol{x}) \geq \prod_u \#_{\boldsymbol{y}}(\boldsymbol{x}(Z_u)) > \prod_u \#_{\boldsymbol{y}}(\boldsymbol{x})^{|Z_u|/n} = \#_{\boldsymbol{y}}(\boldsymbol{x}),$$

a contradiction. ∎

Let $\mathcal{N}' = \cup_u Z_u$ be a partition of the set of coordinates into pairwise disjoint segments. As we remarked above, we would like to find a good segment of length $s = k + \lceil 2 \log_q n \rceil$ on which it is sufficient to examine only $N^{s/n}$ subvectors $\boldsymbol{x}(Z)$ and then use them as message sets to recover codewords. Unfortunately, even though we are free to choose a partition of $\mathcal{N}'$ according to the last lemma, we cannot control the length of the good segment. However, we can prove that for any $s, 1 \leq s \leq n$, any vector $\boldsymbol{x}$ contains a subvector $\boldsymbol{x}(Z)$, $|Z| = s$, such that there is a partition of Z into pairwise disjoint segments $Z = \cup Z'_u$ in which every subvector $\boldsymbol{x}(Z'_u)$ is light. This is the second key idea (after Theorem 3.35) that leads to simpler decoding. Let us call a segment Z that has this property *well-decomposable*.

Suppose $s = a_1 s_1 + \cdots + a_m s_m$. This defines a partition of an s-segment into a_1 segments of length s_1 followed by a_2 segments of length s_2, and so on up to a_m segments of length s_m. Generally the indices should be reduced $\mod n$. Let us call this partition a $((a_1, s_1), \ldots, (a_m, s_m))$-*segmentation* and denote it by $\mathfrak{S}_j((a_1, s_1), \ldots, (a_m, s_m))$, where j is the first coordinate of the s-segment.

To construct a well-decomposable segment, let $s_0 = n, s_1 = s$, and run the

most $p_N = p_R$, and so

$$\mathsf{Pr}\,(E \setminus S_N) \le Q_\alpha p_N \le \frac{\mathsf{Pr}\,(S_R \setminus S)}{\frac{R}{T} - 1} \le \mathsf{Pr}\,(S_R \setminus S)\Big(1 + \frac{T}{N-T}\Big).$$

We conclude that the required inequality holds for every α, which completes the proof. ∎

Remark 3.11 If $p_N > p_{N+1}$, then it can be shown that p_{ϕ_N} admits a better estimate (3.19) even under the more general conditions of this theorem.

Just as in the case of hard-decision decoding, the Evseev lemma is supplemented in the asymptotic setting by Theorem 3.4, in the general case under discussion one can prove a similar asymptotic result. Again note the difference: while the estimates of the last two theorems are valid for all codes, the following holds true only for almost all linear codes. Recall that given a received vector $\boldsymbol{y}$ we can form a subset $X_N(\boldsymbol{y})$ of N most probable vectors of X^n.

Theorem 3.36 *There exists N with $\log_q N/(n-k) \to 1$ such that for almost all linear codes, every subset $X_N(\boldsymbol{y})$ contains a codeword, regardless of the channel used.* ∎

Hence for almost all long linear codes, inspecting N most probable vectors is equivalent to exact maximum likelihood decoding.

One of the main results of this section is a decoding algorithm for symmetric memoryless channels. By (3.15), we are looking for a vector $\boldsymbol{c} \in C$ that maximizes the *a posteriori* probability $\mathsf{Pr}\,(\boldsymbol{c}|\boldsymbol{y})$. For notational convenience we again introduce a $(q \times n)$ weight matrix $W(\boldsymbol{y}) = [w_i(a)]$, $w_i(a) = -\log \mathsf{Pr}\,(a|y_i)$, $a \in X$.

In the remaining part of this section code coordinates are numbered by the set $\mathcal{N}' = [0, 1, \ldots, n-1]$. Recall that by $\boldsymbol{x}(J)$ we denote a projection of $\boldsymbol{x} \in X^n$ on the coordinates in $J \in \mathcal{N}'$. We shall examine (cyclically) consecutive subsets of coordinates $I = (i, i+1, \ldots) \subseteq \mathcal{N}'$, where the indices are, if necessary, reduced modulo n. In order to distinguish them from other subsets we call them *segments*.

The general idea is to look for light subvectors of code vectors and then use Lemma 3.7 to retrieve codewords. Therefore, given a vector $\boldsymbol{x} \in X^n$, define a function on subsets $I \subseteq \mathcal{N}'$ as follows:

$$w(I) = w_{\boldsymbol{x},\boldsymbol{y}}(I) := \sum_{i \in I} w_i(x_i)$$

For a fixed subset I, these functions give a total order $\preceq_I$ on $X^{|I|}$ if we again agree to order vectors of equal weight lexicographically. Let us extend the definition of the rank function to subsets of $\mathcal{N}'$ by putting $\#_{\boldsymbol{y}}(\boldsymbol{x}(I))$ equal to the rank of the vector $\boldsymbol{x}(I)$ in the order $\preceq_I$. An important observation is that for a fixed $\boldsymbol{x} \in X^n$ the $\#$-function is log-supermodular.

Theorem 3.35 (Generalized Evseev lemma) *Let C be an arbitrary code of length n over a q-ary alphabet X. Suppose C is used over a symmetric channel with input alphabet X and output alphabet Y, defined by a probability distribution $\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{x})$, $\boldsymbol{y} \in Y^n, \boldsymbol{x} \in X^n$. Then the error probability of ϕ_N-decoding satisfies*

$$p_{\phi_N} \le p_{\phi_{ml}}\Big(1 + \frac{T}{N-T}\Big) \quad \textit{for any } N > T.$$

Proof: By the definition of symmetric channels, we can write $Y^n = \cup_\alpha Y^n_\alpha$, where Y^n_α, $Q_\alpha = |Y^n_\alpha|$ are disjoint finite subsets such that every column in every $(q^n \times Q_\alpha)$ matrix $P_\alpha = [\Pr(\boldsymbol{y}|\boldsymbol{x})]_{\boldsymbol{y}\in Y^n_\alpha}$ is a permutation of one and the same set of q^n numbers $\boldsymbol{p}_\alpha = \{p_1, \dots, p_{q^n}\}$. Below we assume w.l.o.g. that $p_1 \ge \dots \ge p_N \ge \dots \ge p_{q^n}$.

According to (3.17), we need to isolate N "first" entries in each column. Suppose that vectors $\boldsymbol{x}$ with equal probabilities $\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{x})$ are ordered lexicographically. This defines a total order in every column of P_α. Let S_N be the set of NQ_α pairs $(\boldsymbol{x}, \boldsymbol{y})$ that correspond to the first N entries in all columns.

Since the set $\boldsymbol{p}_\alpha$ can contain many equal entries, we will also need to consider the set $p_L = \dots = p_N = \dots = p_R$ defined as the maximal subset of ordered equal numbers that contains p_N. Each number p_i, $i \le R$, occurs in the matrix at least RQ_α times; therefore, there is a row that contains at least RQ_α/q^n numbers p_i, $i \le R$, and hence this is valid for each row since the channel is symmetric. Restricting our attention to rows that correspond to the codewords of C, we observe that they contain at least

$$\frac{|C|RQ_\alpha}{q^n} > \frac{RQ_\alpha}{T} \ge \frac{NQ_\alpha}{T}$$

such numbers. Denote by S_R the subset of pairs $(\boldsymbol{c}, \boldsymbol{y})$ that correspond to these entries in P_α.

Note that ML decoding is successful on the set $E = \{(\phi(\boldsymbol{y}), \boldsymbol{y})\}$ of Q_α entries in the code rows ("coset leaders"). By (3.11), (3.17), and (3.18),

$$|C|p_{\phi_N} \le \mathsf{Pr}\,(\overline{E}) + \mathsf{Pr}\,(E \setminus S_N).$$

We need to estimate the last term.

Parallelizing the proof of Lemma 3.9, consider also the subset of pairs $S \subseteq S_N$ that consists of exactly Q_α pairs $(\boldsymbol{c}, \boldsymbol{y})$ with the largest entries $\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{c})$ in the code rows. Since $|S| = |E|$ and S is formed by the most probable pairs, we conclude that $\mathsf{Pr}\,(S) \ge \mathsf{Pr}\,(E)$.

Let us relate $\mathsf{Pr}\,(E \setminus S_N)$ and $\mathsf{Pr}\,(S_R \setminus S)$. Since every term in the last probability is bounded from below by p_N, we have

$$\mathsf{Pr}\,(S_R \setminus S) \ge Q_\alpha\Big(\frac{R}{T} - 1\Big)p_N.$$

On the other hand, the subset $\mathsf{Pr}\,(E \setminus S_N)$ has size at most Q_α and is formed by pairs $(\boldsymbol{x}, \boldsymbol{y})$ on which ϕ_N-decoding fails. The probability of each pair is at

Finally, we briefly mention a further improvement of this algorithm achieved along the lines of the punctured split-syndrome decoding method in the hard-decision case (Algorithm 3.5). The reader already familiar with Section 3.3 can get a general impression of the current situation with soft-decision decoding algorithms by reviewing Fig. 3.3.

We use the notation introduced in the beginning of this section. Given a received vector $\boldsymbol{y}$, we can compute all q^n a posteriori probabilities $\mathsf{Pr}(\boldsymbol{x}|\boldsymbol{y})$, $\boldsymbol{x} \in E_q^n$. Using them, we can order the q^n vectors $\boldsymbol{x}$ by assigning number 1 to the vector with the largest a posteriori probability, number 2 to second largest, and so on. We only have to agree how to order vectors with equal probabilities. Suppose they are always ordered lexicographically. Then for a given $\boldsymbol{y}$ every vector $\boldsymbol{x} \in E_q^n$ has a well-defined rank in this order. Denote this rank by $\#_{\boldsymbol{y}}(\boldsymbol{x})$.

Recall that ϕ_{ml} denotes the maximum likelihood decoding algorithm, which for given $\boldsymbol{y}$ chooses a code vector with the smallest rank. Let $\boldsymbol{c}_0 = \phi_{ml}(\boldsymbol{y})$.

Suppose we agree to restrict our attention only to the first N most probable vectors, where N is a parameter. Denote such decoding by ϕ_N. More formally,

$$\phi_N(\boldsymbol{y}) = \begin{cases} \boldsymbol{c}_0, & \#_{\boldsymbol{y}}(\boldsymbol{c}) \le N, \\ 0, & \text{otherwise}. \end{cases} \tag{3.17}$$

The error probability of this decoding equals

$$p_{\phi_N} \le p_{\phi_{ml}} + \frac{1}{|C|} \sum_{\substack{\boldsymbol{c} \in C \\ \boldsymbol{y} \in \phi_{ml}^{-1}(\boldsymbol{c}),\ \#_{\boldsymbol{y}}(\boldsymbol{c}) > N}} \mathsf{Pr}(\boldsymbol{y}|\boldsymbol{c}) \tag{3.18}$$

(cf. (3.11)).

First, suppose that the channel is discrete, additive, and $Y = X$. The following theorem generalizes the Evseev lemma (Lemma 3.9) and shows that by increasing N one can trade error probability for decoding complexity. Let $T = \lceil q^n/|C| \rceil$.

Theorem 3.34 (Generalized Evseev lemma) *Let C be an arbitrary code of length n over a q-ary alphabet X. Suppose C is used over an additive discrete channel with input/output alphabet X defined by a probability distribution* $\Pr(\boldsymbol{y} - \boldsymbol{x})$, $\boldsymbol{x}, \boldsymbol{y} \in X^n$. *Then the error probability of ϕ_N-decoding satisfies*

$$p_{\phi_N} \le p_{\phi_{ml}} \left(1 + \frac{T}{N}\right) \quad \textit{for any } N \ge T. \tag{3.19}$$

∎

Remark 3.10 Taking here $N = T$, we again get Lemma 3.9 for linear codes over the q-ary (narrow-sense) symmetric channel.

The next theorem further generalizes this result and serves as a basis for the design of reduced-complexity ML decoding algorithms. Overloading the term, we also call this fact the generalized Evseev lemma.

Proof: The first part is straightforward by the previous theorem. For small rates $\alpha_q(\delta)$ is maximal for $m = 2$. However, $R^*(2\delta) = 0$ for $2\delta \geq \dfrac{q-1}{q}$ by the Plotkin bound. ∎

In Fig. 3.3 we draw this curve for a family of binary codes meeting the GV bound. Then $R = 1 - H_2(\delta)$ and α_2 can be written as a function of R. The two parts of the bound meet at the point $\delta = 1/4$ which corresponds to the point

$$R = 1 - H_2(1/4) = 0.1887$$

already familiar from Lemma 3.18. The best upper bound known is the McEliece–Rodemich–Ramsey–Welch bound (see Chapter xx (Levenshtein)). We can use this as R^* to compute the curve. Note that this bound is a maximum of two functions, the so-called "first" and "second" upper bounds. They coincide for $R > 0.273$ and for $0.188 \leq R \leq 0.273$ the second bound gives a marginal improvement of the function $\alpha_2(R)$. This lower bound is shown in Fig. 3.3 (curve d).

However, if this assumption is too optimistic, and the true upper bound on the size of binary codes is the GV one, this would give a much stronger lower bound on decoding complexity. This (conditional) lower bound is shown in Fig. 3.3 as curve e.

3.4.2 Maximum likelihood decoding with reduced complexity

Reducing the asymptotic complexity of decoding rests on generalizations of two results in the previous section, Lemma 3.9 and Theorem 3.4. We begin with the Evseev lemma and consider its generalizations in two settings, that of general symmetric channels and additive discrete channels, which gives two slightly different estimates of the error probability. Another group of results briefly mentioned at the end of this section is based on an important fact that in memoryless channels for almost all long codes any subset of $X^n = E_q^n$ of size $q^{(n-k)(1+o(1))}$ formed by most probable vectors with respect to any given point $\boldsymbol{y} \in Y^n$ contains at least one code vector (recall that X denotes the input alphabet and Y the output alphabet). Therefore finding the most probable code vector in this subset w.r.t. the received vector $\boldsymbol{y}$ for almost all long codes is equivalent to maximum likelihood decoding.

Next we formulate an algorithm that for any memoryless symmetric channel constructs a set of most probable ("lightest") error vectors of any given size N. In particular, for $N = q^{(n-k)(1+o(1))}$ it performs maximum likelihood decoding for almost all long linear codes and all cyclic codes with complexity of order $q^{k(1-R)(1+o(1))}$, which is better than $q^{\min(k,n-k)}$ in Theorem 3.31 for all code rates $R, 0 < R < 1$. For instance, for $R = 1/2$ we obtain in the exponent 0.25 instead of 0.5, i.e., for long codes the complexity of the algorithm in this section is only a square root of the time and space complexity of syndrome trellis decoding. Moreover, for low code rates, the complexity of this algorithm falls below the asymptotic *lower* bounds on the trellis complexity (Corollary 3.33).

Proof: Let $(v_{0i}, i = 0, \ldots, n)$ be the all-zero path in T. Let H_1 and H_2 be two submatrices of H formed by its columns 1 to i and $i+1$ to n, respectively, and let C_1 (C_2) be the code of length i (resp., $n-i$) orthogonal to H_1 (resp., H_2). Every codeword of C, whose first i symbols form a codeword of C_1, passes through vertex v_{0i} and so does every codeword whose last $n-i$ symbols form a codeword in C_2. Hence the total number of codewords that cross v_{0i} is at least $|C_1||C_2|$. Any other vertex v_{ij} corresponds to a nonzero syndrome s_i. Hence the above argument is valid with respect to cosets of C_1 and C_2 defined by this syndrome. Thus,

$$|V_i| \geq \frac{|C|}{|C_1||C_2|} = q^{k-(i-\mathsf{rank}\, H_1)-((n-i)-\mathsf{rank}\, H_2)}.$$

The rank of H_1 and H_2 for almost every choice of H follows from Lemma 3.6. We get

$$\mathsf{rank}\, H_1 \sim \min(n-k, i) \quad \mathsf{rank}\, H_2 \sim \min(n-k, n-i).$$

Taking the maximum over i completes the proof. ∎

We conclude this discussion by proving a lower bound on the trellis complexity of linear codes.

Theorem 3.32 *Let $A(n,d)$ be the maximum size of a code of length n and distance d. Then for any $m = 2, 3, \ldots$,*

$$\Big(\max_i |V_i|\Big)^{m-1} \geq \frac{|C|}{\Big(A(\frac{n}{m}, d)\Big)^m}.$$

Proof: We elaborate on the argument that led to the previous lemma. Partition the set $\mathcal{N}$ into m equal segments $\ell(i) = [(i-1)n/m, \ldots, in/m]$ (since n is going to be large, roundoff errors are irrelevant). Let $C_{i-1,i}$ be a subcode of C with zeros outside $\ell(i)$. Let $j_i = in/m, 1 \leq i \leq m-1$ be the division points other than the endpoints of $\mathcal{N}$.

As above, we easily see that

$$\prod_{i=1}^{m-1} |V_{j_i}| \geq \frac{|C|}{\prod_{i=1}^{m} |C_{i-1,i}|}.$$

The required estimate follows. ∎

Taking logarithms, we arrive at the following.

Corollary 3.33 *Let $R^*(\delta)$ be the maximal rate of a code with relative distance δ. Let C be a family of codes of growing length with limit rate $R(\delta), 0 < \delta < (q-1)/q$. Then both time and space complexity of trellis decoding for these codes are at least $q^{n\alpha_q(\delta)}$, where*

$$\alpha_q(\delta) = \begin{cases} \max_{m\geq 1} \dfrac{R(\delta) - R^*(m\delta)}{m-1}, & 0 \leq \delta < \dfrac{q-1}{2q}, \\ R(\delta), & \dfrac{q-1}{2q} \leq \delta \leq \dfrac{q-1}{q}. \end{cases} \tag{3.16}$$

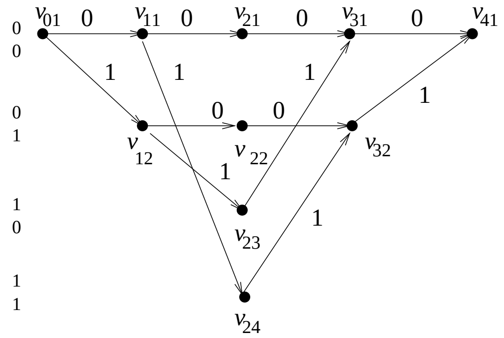


Figure 3.2: Trellis diagram for the $[4, 2, 2]$ code

Algorithm 3.8: Syndrome trellis decoding

- For every $v_{ij} \in V$ set $r_{ij} = 0$.
- Repeat the next step for $i = 1$ to n.
- For every j, $1 \le j \le |V_j|$, inspect all arcs entering v_{ij} and set $r_{ij} = \min_{\ell,\alpha}(r_{i-1,\ell} + v_i(\alpha))$. Store the values ℓ and α that furnish this minimum in the vertex v_{ij}.
- Retrieve the decoded codeword $(c_1, \ldots, c_n)$ moving from vertex v_{ij} to vertex $v_{i-1,\ell}$ against the arrows and taking $c_i = \alpha$.

Remark 3.9 In fact, this quantity is a parameter of the code. Note that it is not invariant to permutations of coordinates. Therefore, it is reasonable to study the minimum over all permutations of the maximal size of V_i. The base q logarithm of this quantity is called the *trellis complexity*. This term does not seem especially successful because it suggests links to the complexity of decoding. Lest the reader think that they exist, we note that while making a syndrome trellis is *sufficient* for maximum likelihood decoding, it is by no means *necessary*. Moreover, below we show that for decoding purposes the size of the trellis can be considerably reduced. This is the reason that we always refer to the syndrome trellis which points to the representation of the code as defined above.

In view of this it is no surprise that for most long codes trellis complexity meets the above bound with equality. Namely, the following is true.

Theorem 3.31 *For most long $[n, k]$ linear codes, both time and space complexity of syndrome trellis decoding equal $q^{\min(k,n-k)(1-o(1))}$.*

$(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$ and $\boldsymbol{c} \in C$ a vector. Computing the product $0 = H\boldsymbol{c}^T$ can be viewed as successively adding columns to the already accumulated vector $\boldsymbol{s}_i$. The initial value $\boldsymbol{s}_0$ is set to 0. In the first step, the vector $\boldsymbol{s}_1$ can take on q values depending on c_1, namely, $\boldsymbol{s}_1 = \boldsymbol{s}_0 + c_1\boldsymbol{h}_1$. Continuing in this manner, in step i, $\boldsymbol{s}_i$ can take on $q^i \le q^{n-k}$ values depending on the contents of $(c_1, \ldots, c_i)$. To these values we associate a subset V_i of vertices of T by making a vertex v_{ij} for every possible vector $\boldsymbol{s}_i$. Vertices $v_{i-1,j}$ and v_{ik} are connected with an arc $b^i_{jk}(\alpha)$ if there exists an $\alpha \in \mathbf{F}_q$ such that

$$\boldsymbol{s}_i = \boldsymbol{s}_{i-1} + \alpha\boldsymbol{h}_i.$$

All the arcs are directed away from the source and marked with the corresponding α's.

To qualify as a codeword, a vector must have a zero syndrome $\boldsymbol{s}_n = (00\ldots0)$. Therefore, $|V_n| = 1$ with v_{n1} corresponding to $\boldsymbol{s}_n$. Therefore, we do not draw arcs from vertices in V_{n-1} other than those leading to v_{n1} and keep in $V_{n-1}, V_{n-2}, \ldots$ only those vertices from which one can walk to v_{n1} along the arcs of T.

Definition. The graph $T(V_0 \cup \cdots \cup V_n, B)$ is called a *syndrome trellis* of the code C.

Actually this definition is but a way to store all q^k codewords in the memory. An interesting observation is that we sometimes can use a smaller size of memory (of order q^{n-k}) by storing common parts of the codewords only once.

Example 3.8 Let C be a binary $[4, 2, 2]$ code with the parity-check matrix $H = \begin{bmatrix} 0110 \\ 1101 \end{bmatrix}$. Its trellis is shown in Fig. 3.2. Vertices are arranged into 4 levels, each corresponding to a certain value of the accumulated syndrome, shown on the left. The code consists of 4 vectors, $0000, 1001, 1110, 0111$; therefore, the trellis has two outgoing forks, namely, at vertices v_{11} and v_{12}. The common part of vectors 1001 and 1110 is stored only once as is the common part of vectors 1001 and 0111, etc. Therefore the graph has only 12 edges (rather then 16).

Examining the trellis representation, we immediately arrive at a maximum likelihood decoding algorithm called syndrome trellis decoding (shown in the tabular form). Here $v_i(\alpha)$ is the weight function defined above. To save on the computation time, one should pre-compute and store the $q \times n$ log-likelihood matrix $W(\boldsymbol{y}) = [v_i(\alpha)]$.

Clearly, both the time and space complexity of this decoding algorithm are determined by the number $\max_i |V_i|$. To estimate it, first note that every path from v_0 to v_n in T corresponds to a codeword. Hence we have

$$\max_i |V_i| \le \min(q^k, q^{n-k}).$$

generalization of the minimal-vectors algorithm in Sect. 3.3.3 (Algorithm 3.7). Here C is a *binary* linear code. Recall that $\mathcal{M} \subset C$ is the set of minimal vectors in C.

Algorithm 3.7: Minimal-vectors ML decoding

- Set $\boldsymbol{c} = 0$.
- Find $\boldsymbol{m} \in \mathcal{M}$ such that $v_{\boldsymbol{y}}(\boldsymbol{c}+\boldsymbol{m}) < v_{\boldsymbol{y}}(\boldsymbol{c})$. Let $\boldsymbol{c} \leftarrow \boldsymbol{c}+\boldsymbol{m}$.
- Repeat until no such $\boldsymbol{m}$ is found. Output $\boldsymbol{c}$.

Proposition 3.30 *For any binary linear code C Algorithm 3.7 performs complete maximum likelihood decoding.*

Proof: Let $\boldsymbol{c}$ be the current approximation to the decoding result. Suppose there is a $\boldsymbol{c}' \in C$ such that $v_{\boldsymbol{y}}(\boldsymbol{c}+\boldsymbol{c}') < v_{\boldsymbol{y}}(c)$. By Lemma 3.12 it is possible to write $\boldsymbol{c}'$ as a sum of minimal vectors. Therefore, let

$$\boldsymbol{c}' = \sum \boldsymbol{m}_u, \quad \boldsymbol{m}_u \in \mathcal{M}.$$

Note that since the code is binary, the supports of different vectors in this expansion are disjoint. Since $\boldsymbol{c}'$ improves the current decision, so does at least one of the minimal vectors in its expansion. To prove that this process eventually converges, note that if $\epsilon = \min_{c \neq c'} |v_{\boldsymbol{y}}(\boldsymbol{c}) - v_{\boldsymbol{y}}(\boldsymbol{c}')|$, then every decoding iteration reduces the weight by at least ϵ. ∎

Lemmas 3.14 and 3.15 imply that most long codes have many minimal codewords; therefore, there is little hope on improving the decoding complexity with this algorithm. This is further complicated by the fact that though the previous theorem proves convergence of the algorithm, it is not clear that one and the same minimal vector does not appear in several different iterations. Therefore, we do not make any formal claims about the complexity of this decoding. A number of other heuristic methods appear in the references following this section. We move to decoding methods with *provable* complexity estimates.

3.4.1 Syndrome trellis decoding

As mentioned above, syndrome decoding can be implemented by building a code trellis. Trellises form a subject of Chapter xx (Vardy) in the present volume. We give a constructive definition and prove bounds for the complexity of syndrome trellis decoding. An interesting fact is that this decoding forms a so sharply restricted class of algorithms that they admit exponential *lower* bounds on their complexity, which is quite an exception in computer science.

A trellis $T(V,B)$ associated with a code C is a directed tree with a source v_0 and a sink v_n connected by q^k paths of length n. We give a constructive definition of this graph. Let H be the parity-check matrix of C with columns

If C is linear, this is equivalent to finding a code vector $\boldsymbol{c}$ such that

$$\mathsf{Pr}(\boldsymbol{y}|\boldsymbol{c}) \geq \mathsf{Pr}(\boldsymbol{y}|\boldsymbol{c}-\boldsymbol{c}') \quad \forall \boldsymbol{c}' \in C. \tag{3.13}$$

If the channel is memoryless, we can also write (3.12) as

$$\prod_{i=1}^{n} \mathsf{Pr}(y_i|c_i) \geq \prod_{i=1}^{n} \mathsf{Pr}(y_i|c_i') \quad \forall \boldsymbol{c}' \in C. \tag{3.14}$$

Still another formulation is obtained if we introduce the function ("analog weight")

$$v_i(\alpha) = -\log \mathsf{Pr}(y_i|\alpha), \quad \alpha \in \mathbf{F}_q,$$

and put $v_{\boldsymbol{y}}(\boldsymbol{c}) = \sum_{i=1}^{n} v_i(c_i)$. Then our problem is to find a codeword with

$$v_{\boldsymbol{y}}(\boldsymbol{c}) \leq v_{\boldsymbol{y}}(\boldsymbol{c}') \quad \forall \boldsymbol{c}' \in C.$$

In particular, in the binary case, if we put $\varphi_i = \log \dfrac{\mathsf{Pr}(y_i|0)}{\mathsf{Pr}(y_i|1)}$, then the goal of maximum likelihood decoding is to maximize the sum $\sum_{i=1}^{n} (-1)^{c_i}\varphi_i$.

ML decoding is also known to find a codeword $\boldsymbol{c}$ with the maximum *a posteriori* probability:

$$\mathsf{Pr}(\boldsymbol{c}|\boldsymbol{y}) \geq \mathsf{Pr}(\boldsymbol{c}'|\boldsymbol{y}) \quad \forall \boldsymbol{c}' \in C. \tag{3.15}$$

This simple remark allows us to transfer the study of maximum likelihood decoding algorithms into a purely combinatorial context and serves a basis for Sect. 3.4.2.

Example 3.7 (Gaussian channel) Suppose the binary data is transmitted with two antipodal signals $X = \{-1, +1\}$ (binary phase shift-keying) over a memoryless channel with additive white Gaussian noise (AWGN). Then the output alphabet $Y = \mathbf{R}$ can be written as $Y = \cup_{\alpha \geq 0}\{\pm\alpha\}$, which shows that the binary AWGN channel is symmetric. Likewise, the q-PSK used over a memoryless channel with 2-dimensional AWGN yields a symmetric channel.

A trivial implementation of ML decoding is to inspect all codewords (time complexity $\mathcal{O}(nq^k)$) [3].

As remarked before Lemma 3.9, hard-decision decoding is a particular case of this problem, so all the results of this section carry over to it. Moving in the reverse direction is a very difficult task. However, (3.13) prompts a

[3] Speaking of complexity one should take into account that we have to perform computations over an infinite domain (a typical example is real numbers). This complicates the foundations and we prefer to leave this question behind the scenes. Practically reals are replaced with rational approximations since the probability that this causes an error is remote. Another option is to count separately the number of operations with real numbers.

more narrow classes of codes that meet the GV bound (see Barg and Dumer [20], Kudryashov and Zakharova [103]).

Bounded distance decoding of Reed–Solomon codes beyond $d/2$ was considered in Dumer [51], Sidelnikov [144], Sudan [153]. List cascade decoding and Theorem 3.27 are due to Zyablov and Pinsker [170].

Constructing covering designs is discussed in Füredi [67], Gordon et al. [78], [77], Grable [79]. Paper [78] also lists tables of good covering designs for n up to 32. Deterministic construction of coverings with linearly growing k and t is discussed in Dumer [54], Fedorenko [61].

3.4 Soft-decision decoding

As we have mentioned in the introduction, soft-decision decoders use information about the reliability of the received signal. Let us specialize the problem setting. Let X be a set of q input signals and $Y \supseteq X$ a set of output signals (possibly, infinite). We assume that Y forms an additive group. A *channel* is a set of probability distributions on $(X^n \times Y^n)$, $n \geq 1$, known to the decoder. A channel is called *discrete* if Y is a discrete set. A channel is called *additive* if $\mathsf{Pr}\,(\boldsymbol{x}, \boldsymbol{y}) = \Pr(\boldsymbol{x}+\boldsymbol{z}, \boldsymbol{y}+\boldsymbol{z})$, $\boldsymbol{z} \in E_q^n$.

Often instead of joint we prefer to consider conditional probabilities. To simplify this transition, we assume for the rest of this section that messages (codewords) are equiprobable. Then additivity implies that $\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{x}) = \Pr(\boldsymbol{y}-\boldsymbol{x})$, i.e., that for additive channels the error process is independent of the transmitted message. Note that implicitly we assume that X and Y form additive groups. By abuse of speech we continue to call elements of X^n and Y^n vectors.

A channel is called *memoryless* if $\mathsf{Pr}\,(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{n} \mathsf{Pr}\,(x_i, y_i)$.

The last property of information transmission channels that we need is symmetricity. Suppose that Y^n can be decomposed into a disjoint union of finite subsets $Y^n = \cup_\alpha Y_\alpha^n$, where $|Y_\alpha^n| = Q_\alpha$. Then we can describe the channel by a set of $q^n \times Q_\alpha$ matrices $P_\alpha = [\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{x})]$. A channel is called *symmetric* if for every matrix P_α there is a row of Q_α numbers and a column of q^n numbers such that every row (column) of P_α is a permutation of this row (column).

Let $\phi : Y^n \to C$ be a decoding mapping. In particular, the *maximum likelihood* (ML) decoder is a mapping ϕ_{ml} such that

$$\boldsymbol{y} \mapsto \boldsymbol{c} \in C : \mathsf{Pr}\,(\boldsymbol{y}, \boldsymbol{c}) \geq \mathsf{Pr}\,(\boldsymbol{y}, \boldsymbol{c}') \quad \forall \boldsymbol{c}' \in C.$$

The error probability p_ϕ of a decoding ϕ equals the total probability that a transmitted code vector $\boldsymbol{c}$ is outside the preimage ϕ^{-1} of the received vector $\boldsymbol{y}$:

$$p_\phi = \frac{1}{|C|} \sum_{\boldsymbol{c} \in C} \mathsf{Pr}\,(\boldsymbol{c}, Y^n \setminus \phi^{-1}(\boldsymbol{c})). \tag{3.11}$$

Since codewords are equiprobable, one can say that ML decoding finds $\boldsymbol{c} \in C$ such that for all $\boldsymbol{c}' \in C$,

$$\mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{c}) \geq \mathsf{Pr}\,(\boldsymbol{y}|\boldsymbol{c}'). \tag{3.12}$$

codes including the Hamming and 2nd order Reed–Muller codes. Lemma 3.15 is from Ashikhmin et al. [16]. From the combinatorial point of view minimal supports of a code correspond to cycles in the matroid represented by this code, see Welsh [165].

The zero-neighbors algorithm for binary codes is due to Levitin and Hartmann [110]. Their definition includes the minimality condition in (3.8). Dropping it does not affect the result and simplifies the presentation. The boundary of a set A in the Hamming space can be defined in several ways. Under a more topological approach, the boundary is defined as A minus its interior points, see Katona [94]. Theorems 3.21 and 3.22 are from [13].

3.3.4. Information set decoding was introduced by Prange [131]. One of the first papers to mention the use of coverings for decoding was Chan et al. [37], though for cyclic codes coverings (covering polynomials) were discussed already in Kasami [93]. The minimal collection of information sets for the extended Golay code $\mathcal{G}_{24}$ (Example 3.3) is found in Gordon [77], Wolfmann [167]. An application of information set decoding to the bounded distance decoding problem is considered in Chapter xx (Huffman) under the name of *permutation decoding*.

The covering set decoding algorithm is discussed in numerous publications devoted to the security of the McEliece cryptosystem beginning with McEliece's original work, [121]. See Canteaut and Chabaud [36], van Tilburg [156] and references therein (see also Ch. xx (van Tilborg)). The asymptotic analysis of this algorithm was made possible by Lemma 3.9 and Theorem 3.4. It was performed in Coffey et al. [42], [43], Krouk [101].

This algorithm can be applied to finding minimum-weight codewords in a linear code (see Sect. 3.5.1). In this form it has been advanced by Leon [108]. A version of this procedure based on looking for certain covering systems has been proposed in Stern [152]. Observe that no algorithm based solely on coverings can be asymptotically better than covering set decoding since the bound for $M(n, m, t)$ is asymptotically tight (Remark 3.8(ii)).

The version of covering set decoding with bit swapping was suggested by Omura in 1969 (see [41, Sect. 3.2]) and rediscovered by van Tilburg [156]. Its applications were studied by van Tilburg [156] (cryptanalysis) and by Canteaut and Chabaud [36] (cryptanalysis; finding the minimum distance of BCH codes, see Sect. 3.5.1). The complexity reduction in Example 3.4 was obtained in [36], [156].

Example 3.5 is due to Krouk and Fedorenko [102]. This is probably not the simplest known decoder of this code. For other methods see Blaum and Bruck [29], Elia [57], Pless [130]. We chose to include this method because it is easy to describe and is applicable to other linear codes. The general punctured split syndrome decoding algorithm and Theorem 3.25 are due to Dumer [54]. Example 3.6 is due to E. Krouk, see [22]. The supercode decoding algorithm and Theorem 3.26 are due to Barg et al. [22].

In this section we studied the asymptotic behaviour of decoding algorithms applicable to all linear codes. Another approach is to study the decoding for

the same size as above can be constructed deterministically. As remarked after Theorem 3.24, the size of the minimal covering is of order $\binom{n}{t}/\binom{n-k}{t}$. This can be expressed more precisely as follows.

Lemma 3.28 $\min |M(n, m, t)| \leq \left[1 + \log \binom{m}{t}\right] \binom{n}{t} \Big/ \binom{m}{t}$.

Proof (outline): Compute the fraction of t-subsets that are not covered by a probabilistic procedure. Add one m-set per each uncovered t-set and compute the average size of the covering. Then there exists a covering of size at most the average. ▮

The following theorem can be proved by a recursive construction.

Theorem 3.29 *Let k and t grow linearly in n. A covering $M(n, n-k, t)$ of size $u = o(n)\binom{n}{t}/\binom{n-k}{t}$ can be constructed with time complexity $\mathcal{O}(u \log u)$.* ▮

Thus, it is possible to implement the covering set decoding by a deterministic algorithm with asymptotically the same complexity as the probabilistic one (Theorem 3.24).

3.3.5 Notes

3.3.1. Lemma 3.9 is from Evseev [60]. Actually for the lemma itself, as is readily seen from the proof, reference to the q-ary symmetric channel is quite irrelevant. Already in [60] it was proved for all *discrete* channels with additive noise. See Sect. 3.4.2 for formal definitions and generalizations of this result. Based on his lemma, Evseev [60] suggested a decoding algorithm with time complexity of order $q^{k(1-R)}$.

3.3.2. Split syndrome decoding was suggested by Dumer [53]. Sorting algorithms and networks are discussed in Knuth [96], Cormen et al. [45].

3.3.3. Terms like steepest descent and gradient-like decoding should be used with caution because they refer to optimization in *discrete* space. There have been numerous attempts to reduce the decoding complexity by embedding E_q^n into $\mathbf{R}^n$ and applying classical gradient methods. However, in this case local optima hinder the successful decoding and any significant results still remain out of reach.

Hwang [86], [87] was the first to introduce minimal words in binary codes in the decoding context. Moreover, he showed that minimal codewords can be used in the more general setting of soft-decision decoding. This is related to an observation of Agrell [1], which states that if a binary code is considered as a subset of points of the unit cube in $\mathbf{R}^n$, then zero neighbors (vectors whose Voronoi regions have a common $n-1$-dimensional facet with the Voronoi region of 0) are precisely the minimal vectors of this code. We comment further on this in the next section. Minimal supports are interesting in several other respects. They define access structures in a linear secret-sharing scheme associated with the code. Ashikhmin and Barg [13] found minimal supports for several classes of

We shall assume that as the length of both A and B grows, the code C meets the GV bound, i.e. $\mathsf{dist}(C) = n_\square \delta_0(R)$. Suppose $m = n = \sqrt{n_\square}$. Let $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ be the received vector, split into n parts of length m. Let $\epsilon > 0$.

Algorithm 3.6: List cascade decoding

- Set $\boldsymbol{c} = 0$.
- For every i, $1 \le i \le n$, form a list

$$L_i = \{\boldsymbol{a} \in A \mid \mathsf{dist}(\boldsymbol{a}, \boldsymbol{y}_i) \le d_0(m, \ell) - \epsilon m\}.$$

  Form a corresponding list $K_i = \{\omega \mid A(\omega) \in L_i\}$.
- Let $W \in \binom{\mathcal{N}}{k}$. Encode every k-vector

$$\omega = (\omega_1, \ldots, \omega_k),\ \omega_i \in K_i, i \in W,$$

  with C to form a vector $\boldsymbol{c}'$.
- If $\mathsf{dist}(\boldsymbol{y}, \boldsymbol{c}') < \mathsf{dist}(\boldsymbol{y}, \boldsymbol{c})$, assign $\boldsymbol{c} \leftarrow \boldsymbol{c}'$.
- Repeat the last two steps for all $W \in \binom{\mathcal{N}}{k}$ and every possible choice of ω. Output $\boldsymbol{c}$.

List cascade decoding corrects an error if there is a k-subset W with at most $d_0(m, \ell) - \epsilon m$ errors in each of the vectors $\boldsymbol{y}_i$. Hence it corrects all errors of weight less than

$$t = (1/2)(n-k)(d_0[m, \ell] - \epsilon m) = (1/2) n_\square (1 - k/n)(\delta_0(\ell/m) - \epsilon)$$

for any given $\epsilon > 0$. As shown in Chapter xx (Dumer), the parameters can be chosen in such a way that $2t > n_\square \delta_0(R)$ for small values of R, in the binary case for $0 \le R \le 0.02$.

Theorem 3.27 *Let $C = A \boxtimes B$ be a concatenated code of length $n_\square$ and rate R. The time complexity of list cascade decoding is $\mathcal{O}\big(n_\square^2 \exp(\sqrt{n_\square})\big)$. For small values of R there exist concatenated codes for which the algorithm performs bounded distance decoding up to $n_\square \delta_0(R)/2$.*

Proof: By Lemma 3.3, the size $|K_i|$ does not grow with $n_\square$. Then the complexity is determined by the number of choices of W, which is $\binom{n}{k} < \exp(\sqrt{n_\square})$. Encoding with the code C takes at most $\mathcal{O}(n_\square^2)$ operations. ∎

CONSTRUCTING COVERINGS

In the beginning of this section we said that it is possible to consider bounded distance decoding in a purely combinatorial context. Then the problem carries no inherent probabilistic element, and we would like to derandomize the decoding algorithm itself. The only "random" part of the algorithm is the probabilistic construction of the covering $M(n, n-k, t)$. However, coverings of exponentially

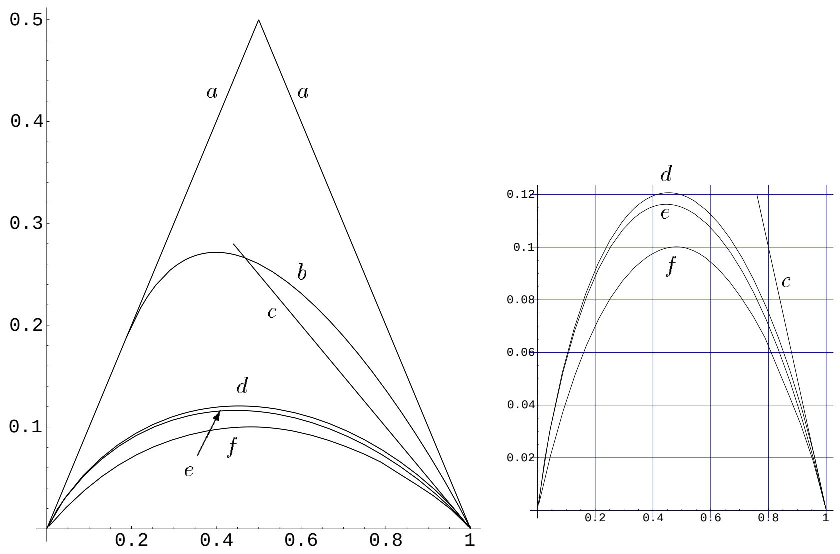


Figure 3.1: Complexity of hard-decision decoding algorithms for binary codes (x-axis—code rate, y-axis—complexity exponent):
(a) $\min(R, 1-R)$,
(b) zero-neighbors decoding, Theorem 3.19,
(c) split-syndrome decoding, Theorem 3.10,
(d) covering set decoding, Theorem 3.24,
(e) punctured split-syndrome decoding, Theorem 3.25,
(f) supercode decoding, Theorem 3.26.

that we are going to discuss. This algorithm applies to a class of linear codes called concatenated codes, which form the subject of Chapter xx (Dumer) of the present volume. We have briefly mentioned them in Sect. 2.1. As explained in Chapter xx (Dumer), this class contains codes meeting the GV bound. We are going to present a decoding algorithm in the sphere of a large radius t that depends on the rate R of the code. It will be seen that for a certain finite range of rates, the relative error weight exceeds $\delta_0(R)/2$. The complexity of this algorithm behaves as $\exp(\sqrt{n})$. This is the only known example of codes meeting the GV bound and decodable up to $n\delta_0/2$ errors with a *subexponential* complexity.

Let A be a q-ary linear $[m, \ell]$ code and B a Q-ary $[n, k]$ Reed–Solomon code, $Q = q^\ell$. Let $C = A \boxtimes B$ be their concatenation, i.e., a q-ary linear $[n_\square = mn, k_\square = \ell k]$ code. Each vector $\boldsymbol{a} \in A$ corresponds to a Q-ary symbol ω. Having fixed a basis of F_Q, we can write this correspondence as a function $\boldsymbol{a} = A(\omega)$.

Properties of the entire procedure, which we call *supercode decoding*, can be summarized as follows.

Theorem 3.26 *The supercode decoding algorithm for almost all long linear $[n,k]$ codes of rate $R = k/n$ has error probability equivalent to that of minimum distance decoding. The time complexity of the algorithm for almost all codes is at most $q^{n\zeta^{(q)}(R)(1+o(1))}$, where*

$$\zeta^{(q)}(R) = \min_{v,\alpha,\ell}\Big\{\epsilon_1(R,\alpha) + \max\Big[\frac{1}{2}\epsilon_2(R,\alpha,v,\ell) + v, \epsilon_2(R,\alpha,v,\ell) - \ell v\Big]\Big\}$$

and the functions $\epsilon_1(\cdot)$ and $\epsilon_2(\cdot)$ are defined in (3.10) *and* (3.9)*, respectively. The optimization parameters are restricted to*

$$\max(0, \delta_0 + R - 1) < \alpha < \min(\delta_0, R), \quad \ell v \le 1 - R.$$

The space complexity of the algorithm is estimated from above as

$$q^{(1/2)n\epsilon_2(R,v,\alpha,\ell)(1+o(1))}.$$

∎

Let us write out an explicit expression for the complexity exponent for binary codes:

$$\zeta^{(2)}(R) = \min_{v,\alpha,\ell}\Big\{(1-R)\Big(1 - H_2\Big(\frac{\delta_0-\alpha}{1-R}\Big)\Big) - \max\Big[\frac{1}{2}RH_2\Big(\frac{\alpha}{R}\Big) \\ - \frac{1}{2}vH_2\Big(\frac{\delta_0-\alpha}{1-R-(\ell-2)v}\Big) - v, \Big(\ell - H_2\Big(\frac{\delta_0-\alpha}{1-R-(\ell-2)v}\Big)\Big)v\Big]\Big\},$$

We collect the results of this section in Fig. 3.1.

Bounded distance decoding

We have formulated Algorithms 3.1, 3.4, and 3.5 for bounded distance decoding in a sphere of radius $n\delta_0(R)$. They can be adjusted to decoding in a sphere of any other radius t. If $t \le d_0$, they will correct any t or fewer errors for almost all codes. In this case we should simply discard a decoding candidate if the distance between it and the received vector $\boldsymbol{y}$ exceeds t.

The complexity estimates should be changed accordingly. For instance, the complexity of Algorithm 3.4 would be $n^4(\log n)q^{\alpha(t,R)n(1+o(1))}$, where

$$\alpha(t,R) = \log_q 2\Big[H_q(t/n) - (1-R)H_q\Big(\frac{t/n}{1-R}\Big)\Big]$$

and so on for the other algorithms.

Of particular interest is the case of $t = \lfloor (d-1)/2 \rfloor$. Some results for the $[24,12]$ Golay code $\mathcal{G}_{24}$ were already discussed in Examples 3.3, 3.5. Algorithmically this problem has few specific features if any; therefore, there are not many general results available. A notable exception is one decoding algorithm

less for which $\mathsf{dist}\,(A_i \boldsymbol{e}^T, \boldsymbol{s}_i) \le 1$.

The decoding is repeated for each of the two check sets. For a given check set, we compile a list of error patterns that appear in 3 out of 4 tables T_i in the record corresponding to the received syndrome $\boldsymbol{s}_i$. Each error pattern is subtracted from the 24 coordinates of $\boldsymbol{y}$ that correspond to the message part. The obtained message set is then encoded with the code. As usual, from the code vectors obtained in this way we choose the vector closest to $\boldsymbol{y}$.

The total size of memory used by tables T_i is 8Kbytes. The decoding requires about 3000 operations with binary vectors of length 24 and seems to be the simplest known for this code. ∎

We propose to do the same thing in the general case, namely, to partition the syndrome $\boldsymbol{s}$ into a number of consecutive segments of a certain length, say y, and combine lists of codewords obtained from check equations that involve the corresponding parts of the syndrome. The total number of segments (lists) is $s = \lceil \frac{n-k}{y} \rceil$. The final list of candidates will be formed by those codewords for which the corresponding message symbols appear in at least ℓ of s lists, where ℓ is another parameter.

Thus, in the first stage of the algorithm we want to decode a number of $[k+y, k]$ codes $C(y)$ each of which is a supercode of the original code C. We shall assume that the weight of the error on the information positions is e_1. To ensure this, we perform $L_n(k, e_1) = (n \log n)[\binom{n}{d_0} / \binom{k}{e_1}\binom{n-k}{d_0-e_1}]$ independent choices of k positions, aiming at constructing a covering system $N_2(n; e_1, d_0 - e_1; k, n-k)$.

Each $[k+y, k]$ code $C(y)$ will be decoded using split syndrome decoding. To apply it we not only need to bound the weight of the error on the message part, but also on the check part y. Suppose this weight is e_2. Let us compute the minimal value of e_2 such that $s - \ell + 1$ y-segments of the error vector cannot all be heavier than e_2 assuming that the total weight of the error on the $n-k$ check coordinates is $d_0 - e_1$. Therefore, we obtain for e_2 the inequality

$$e_2 > \frac{d_0 - e_1}{(s - \ell + 1)y}.$$

Thus, we apply the split syndrome decoding algorithm to decode $s = \mathcal{O}(n)$ codes $C(y)$, where the weight of the error is e_1 on the k-part and e_2 on the y-part of the received vector. Let $y = vn, e_1 = \alpha n$. The (time and space) complexity of each decoding is at most $q^{(1/2)n\epsilon_2(R,v,\alpha,\ell)(1+o(1))}$, where

$$\epsilon_2(R, v, \alpha, \ell) = \left[RH_q\Big(\frac{\alpha}{R}\Big) + vH_q\left(\frac{\delta_0 - \alpha}{1 - R - (\ell - 2)v}\right)\right]. \tag{3.9}$$

The entire procedure is performed $L_n(k, e_1) \le q^{n\epsilon_1(R,\alpha)(1+o(1))}$ times, where

$$\epsilon_1(R, \alpha) = (\log_q 2)\left[H_2(\delta_0) - RH_2\Big(\frac{\alpha}{R}\Big) - (1 - R)H_2\Big(\frac{\delta_0 - \alpha}{1 - R}\Big)\right]. \tag{3.10}$$

The formal description of the algorithm is too technical to be included.

$$\begin{array}{c|c|c|c|c|c|l}
\hline
H_1 & \begin{smallmatrix}1&&&\\&1&&\\&&\ddots&\\&&&1\end{smallmatrix} & 0 & 0 & 0 & A_1 & \boldsymbol{s}_1 \\
\hline
H_2 & 0 & \begin{smallmatrix}1&&&\\&1&&\\&&\ddots&\\&&&1\end{smallmatrix} & 0 & 0 & A_2 & \boldsymbol{s}_2 \\
\hline
H_3 & 0 & 0 & \begin{smallmatrix}1&&&\\&1&&\\&&\ddots&\\&&&1\end{smallmatrix} & 0 & A_3 & \boldsymbol{s}_3 \\
\hline
H_4 & 0 & 0 & 0 & \begin{smallmatrix}1&&&\\&1&&\\&&\ddots&\\&&&1\end{smallmatrix} & A_4 & \boldsymbol{s}_4 \\
\hline
\multicolumn{1}{c}{} & \multicolumn{1}{c}{\mathcal{N}_1} & \multicolumn{1}{c}{\mathcal{N}_2} & \multicolumn{1}{c}{\mathcal{N}_3} & \multicolumn{1}{c}{\mathcal{N}_4} & \multicolumn{1}{c}{} &
\end{array}$$

Let $\boldsymbol{y}$ be a received vector and $\boldsymbol{s} = H\boldsymbol{y}^T$ be its syndrome, where $H = [I_{24}|A]$ is the parity-check matrix that corresponds to the chosen check set. Let H_i, $1 \le i \le 4$, be the submatrix of H formed by rows $6(i-1)+j$, $1 \le j \le 6$, and let $\boldsymbol{s}_i = H_i\boldsymbol{y}^T$ be the corresponding part of the syndrome. Denote by A_i the corresponding 6 rows of the matrix A. The partition into submatrices defines a partition of the first 24 coordinates of the code into 4 parts, $\mathcal{N}_i = [6(i-1), 6(i-1)+1, \ldots, 6(i-1)+5]$, $1 \le i \le 4$. The syndrome $\boldsymbol{s}$ is divided into four parts $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_4$, where part $\boldsymbol{s}_i$ is formed by the sum of the columns in H_i that correspond to the positions of errors. If a part, say $\mathcal{N}_1$, is error-free, then there is an error pattern $\boldsymbol{e}$ of weight $\mathsf{wt}(\boldsymbol{e}) \le 2$ in the information part such that $A_1\boldsymbol{e}^T = \boldsymbol{s}_1$. Any single error in $\mathcal{N}_1$ affects one coordinate in the syndrome. Therefore, if $\mathcal{N}_1$ contains one error, by inspecting all error patterns $\boldsymbol{e}$ of weight ≤ 2 in the information part we shall find a syndrome $\boldsymbol{s}' = A_1\boldsymbol{e}^T$ at a distance one from $\boldsymbol{s}_1$.

Therefore, the decoder can be constructed as follows. For each i, $1 \le i \le 4$, we make a list of error patterns $\boldsymbol{e}$ with $\mathsf{wt}(\boldsymbol{e}) \le 2$ that yield a syndrome $\boldsymbol{s}' = A_i\boldsymbol{e}$ at a distance ≤ 1 from $\boldsymbol{s}_i$. An error pattern is a plausible candidate if it appears in 3 out of four lists. This covers all possible cases. Indeed, if a check set contains 3 (4) errors, then 3 of the 4 sets $\mathcal{N}_i$ contain at most one error. The complement of the check set (the message set) contains 2 errors (resp., 1 error). Thus, this error pattern of weight 2 (resp., 1) will appear in 3 lists. If a check set contains 5 errors, then the message set is error-free, and the empty error pattern will appear in all 4 lists.

Thus, to construct a decoder, we need to store 4 tables of error patterns for each of the 2 check sets. Each table consists of 64 records, one for each possible value of $\boldsymbol{s}_i$. Finally, each record is formed by all error patterns $\boldsymbol{e}$ of weight 2 or

We shall choose the values of u and α that minimize the complexity. Let

$$\beta_q(R) = \min_{u,\alpha}((\log_q L_n(k,u,\alpha)) + \epsilon_q(R,u,\alpha)),$$

with $R \le u \le 1$ and $\max(0, \delta_0 + u - 1) < \alpha < \min(\delta_0, u)$, and let $\alpha_0 = \alpha_0(R)$ and $u_0 = u_0(R)$ be the values that furnish this minimum. The properties of *punctured split syndrome decoding* can be summarized as follows.

Theorem 3.25 *For most long codes the punctured split syndrome algorithm has error probability equivalent to that of complete minimum distance decoding. Its sequential implementation for a code of rate R has time complexity $q^{n\beta_q(R)(1+o(1))}$ and space complexity $q^{(1/2)u_0 n H_q(\alpha_0/u_0)(1+o(1))}$.* ∎

For binary codes the function $\beta_2(R)$ has the following form:

$$\beta_2(R) = \min_{u,\alpha}\max\Big\{(1-u)\Big[1 - H_2\Big(\frac{\delta-\alpha}{1-u}\Big)\Big],$$
$$1 - R - \frac{1}{2}uH_2\Big(\frac{\alpha}{u}\Big) - (1-u)\Big[1 - H_2\Big(\frac{\delta-\alpha}{1-u}\Big)\Big]\Big\}.$$

Algorithm 3.5: Punctured split syndrome decoding

- For a given R find the values u and α that minimize the function $\beta_q(R)$. Set $\boldsymbol{a} = 0$.
- Choose randomly a u-subset $W \subset \mathcal{N}$. Put $C' = C(W)$ and $\boldsymbol{y}' = \boldsymbol{y}(W)$. Perform split syndrome decoding of C' w.r.t. $\boldsymbol{y}'$. Form a list $L(W) = \{\boldsymbol{c}' \in C' \mid \mathsf{dist}(\boldsymbol{c}', \boldsymbol{y}') \le \alpha n\}$.
- Form a list $K(W) = \{\boldsymbol{c} \in C \mid \exists_{\boldsymbol{c}' \in L(W)}\, \boldsymbol{c}(W) = \boldsymbol{c}'\}$.
- If there is a $\boldsymbol{c} \in K(W)$ such that $\mathsf{dist}(\boldsymbol{c}, \boldsymbol{y}) < \mathsf{dist}(\boldsymbol{a}, \boldsymbol{y})$, assign $\boldsymbol{a} \leftarrow \boldsymbol{c}$.
- Repeat the last three steps $L_n(k,u,\alpha)$ times. Output $\boldsymbol{a}$.

Theorem 3.25 can be further improved by performing repeated decoding attempts based on a part of the received syndrome least affected by errors. Decoding based on a part of the syndrome amounts to restricting oneself to a part of parity checks defining the original code C, i.e., decoding in a supercode of C. The following example captures some features of this algorithm and shows the ways of generalizing information set decoding and split syndrome decoding.

Example 3.6 (The $[48, 24, 12]$ binary extended QR code). We aim at constructing a decoder that corrects 5 errors. Suppose the first 24 coordinates form a check set of the code. Since the code is self-dual, the last 24 coordinates also form a check set. At least one of these sets will contain no fewer than 3 errors, in which case the remaining information part contains at most 2 errors.

algorithm discussed. For instance, for $q = 2$ it is better than covering set decoding for $R \geq 0.954$. Therefore, if we puncture the code C (i.e., cast aside some of its parity symbols) so that its rate becomes large, we then can gain in complexity by applying split syndrome decoding to the punctured code C'. By Lemma 3.5 we may regard any symbols as parity ones provided that their number is less than $n - k$.

Let us elaborate on this idea. Assume that the length of C' equals $n' = un$ and the number of errors on these coordinates is αn. Let $\boldsymbol{y}'$ be the projection of the received vector on the coordinates of C'. Since the distance of C' is small, the number of its codewords in a sphere of radius αn around y' equals with high probability

$$\binom{n'}{\alpha n}(q-1)^{-n(u-R)} = q^{n(uH_q(\alpha/u)-(u-R))(1+o(1))}$$

by Lemma 3.2. By Remark 3.5, we can use split syndrome decoding to find these codewords. For each of them, recover the appropriate codeword of C (strictly speaking, a list of at most $q^{\sqrt{n}}$ such codewords, see Cor. 3.8) and choose the one closest to $\boldsymbol{y}$.

The time complexity of the described procedure is the sum of the complexity of split syndrome decoding and the inspection of this list,

$$\mathcal{O}\left(n^4\binom{n'}{\alpha n}^{1/2} + n\binom{n'}{\alpha n}(q-1)^{-n(u-R)}\right).$$

To minimize the sum of two exponential functions we have to equate exponents. Therefore, the time complexity of this algorithm equals

$$q^{n\,\epsilon_q(R,u,\alpha)(1+o(1))},$$

where

$$\epsilon_q(R,u,\alpha) = \max\left\{\frac{1}{2}uH_q\left(\frac{\alpha}{u}\right), uH_q\left(\frac{\alpha}{u}\right) - (u-R)\right\},$$

and the space complexity equals that of split syndrome decoding.

As usual, we are interested in the correction of up to d_0 errors. In order to control the weight of y', we construct a covering system

$$N_2(n; un, (1-u)n; \alpha n, (\delta_0 - \alpha)n).$$

This can be accomplished with high probability in

$$\begin{aligned} L_n(k,u,\alpha) &= (n\log n)\left[\binom{n}{d_0}\Big/\binom{un}{\alpha n}\binom{(1-u)n}{d_0-\alpha n}\right] \\ &= q^{n(\log_q 2)\left[H_2(\delta_0)-uH_2\left(\frac{\alpha}{u}\right)-(1-u)H_2\left(\frac{\delta_0-\alpha}{1-u}\right)\right](1+o(1))} \end{aligned}$$

independent attempts, which is proved exactly in the same way as Theorem 3.24.

Then we can use the residual code $C' = C(\overline{S})$ to correct one error. This decoding yields a codeword of $\mathcal{C}'$, say $\boldsymbol{c}'$. Since $\dim C' = \dim C - 1$, there are 2 codewords of C whose projection on $\overline{S}$ equals $\boldsymbol{c}'$. Since we do not know which residual code to decode, we need to repeat this for all $|\mathcal{A}|$ of them. Each decoding yields a pair of codewords of C or reports a decoding failure. Finally, out of all codewords of C obtained in this way we choose the one closest to received word.

Thus, we need to construct the set $\mathcal{A}$. This set is constructed using the following artifice. Suppose the set of coordinates is split in two halves of size 12. Then if on each half we have a covering $M(12, 8, 2)$ formed by codewords of weight 8, then out of every 3 coordinates there is a pair covered by one of these codewords.

A covering $M(12, 8, 2)$ is formed by 3 codewords of weight 8 constructed as follows. Take any 4 coordinates of the code. Adding a fifth coordinate to them defines a unique codeword of weight 8 (see Ch. 1, Example 11.10). Denote this codeword by $\boldsymbol{c}_1$. Next, take another quintuple formed by the same 4 coordinates and a fifth one outside the support of $\boldsymbol{c}_1$. This again isolates a codeword of weight 8, say $\boldsymbol{c}_2$. Finally take $\boldsymbol{c}_3 = \boldsymbol{c}_1 + \boldsymbol{c}_2$. This gives a covering $M(12, 8, 2)$ of size 3 which is responsible for any two errors in the 12 coordinates. Repeat the same procedure for the remaining 12 coordinates. This gives a total of 6 codewords of C.

Therefore, we need to decode six $[16, 11, 4]$ codes. The 6 parity-check matrices of C can be stored in memory, which is about half the space needed for information set decoding (14 matrices). The time complexity of both methods is roughly the same.

For instance, if we define code C by the generator matrix $G = [I_{12} \,|\, A]$, where A is a circulant matrix with the first row $(1\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0\,0)$, then one can take the 6 codewords in the following form:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$\boldsymbol{c}_1$	1												1	1		1	1	1	1		1			
$\boldsymbol{c}_2$	1								1	1	1		1		1	1	1							
$\boldsymbol{c}_3$									1	1	1			1	1			1	1		1			
$\boldsymbol{c}_4$		1	1	1	1		1	1				1								1				
$\boldsymbol{c}_5$				1	1	1		1												1		1	1	1
$\boldsymbol{c}_6$		1	1			1	1					1										1	1	1

This algorithm is fairly general and can be worked out for other codes, which may be less well studied than the code $\mathcal{G}_{24}$. For instance, decoding of the extended $[48, 24, 12]$ quadratic residue code can be reduced to 28 decodings of $[36, 23, 6]$ codes, which also yields a considerable saving on the complexity (either time or space, depending on the implementation chosen). Another decoder for this code is given in the next example. ∎

Let us turn to asymptotics. To improve on the complexity of covering set decoding we need an algorithm that performs better than this on shortened codes. Looking at the derivatives of complexity exponents for $R \to 1$, we see that the time complexity of split syndrome decoding is better than any other

any given number of errors. The correctness of this algorithm rests on a number of facts that hold with large probability for very long codes. For codes of a fixed length its application is not fully mathematically justified. Cryptanalysts nonchalantly assume that these properties hold for the code generated by G. This assumption is supported by the presence of the random matrix S.

As follows from Remark 3.3, the average corank of a large square binary matrix is 0.85. Therefore, we can assume that the number of codewords that project identically on a given k-set does not exceed 2 (the dimension of the space of solutions of the linear system of equations does not exceed 1). Performing $\binom{n}{t}/\binom{n-k}{t}$ independent choices of k out of n coordinates, we see that the probability of not constructing a covering $M(n, n-k, t)$ is about $1/e = 0.37$. Therefore, the probability of success is 0.63, which is a sufficiently serious threat to the security of the system. Then the average number of computer operations needed to perform the decoding is at most $2(nk^2 + k^2)\binom{n}{t}/\binom{n-k}{t} \approx 2^{82.7}$ (we allow ourselves the order of k^2 operations for each encoding)[2].

An improvement of this estimate is based on an already mentioned observation that choosing successive information windows independently requires independent diagonalizations of the generator matrix. We can save on this by swapping a coordinate in the current information window and a coordinate outside it. This newly chosen subset is also likely to form an information window. In this version of the algorithm the successive choices are not independent and the proof of Theorem 3.24 has to be replaced by a more accurate argument. On the other hand, reducing the matrix to diagonal form now becomes at least n times easier. Calculations show that the complexity is brought down to at most 2^{70} operations. For further details see Chapter xx (van Tilborg). ∎

Shortening and then decoding

Instead of decoding the code C directly, we can make several attempts at decoding shortened codes and choose the closest codeword in the resulting list. We begin with an example.

Example 3.5 In Example 3.3 we discussed the information set decoding of the extended binary Golay code $C = \mathcal{G}_{24}$. Here we present another decoder that combines information-set decoding and shortenings. Let $\boldsymbol{c}$ be a codeword of weight $\le 2d-1$ in the code and let $S = \mathsf{supp}(\boldsymbol{c})$. Then the code $C' = C(\overline{S})$, i.e., the original code projected on the zero coordinates of $\boldsymbol{c}$, has the parameters

$$[n - \mathsf{wt}(\boldsymbol{c}), k-1, d - \lfloor \mathsf{wt}(\boldsymbol{c})/2 \rfloor]$$

(called the residual code, see Ch. 1, Sect. 3). In particular, if $\mathsf{wt}(\boldsymbol{c}) = 8$, then the code C' has $n = 16, k = 11$, distance at least 4, and will be used to correct one error.

Suppose we can find a set $\mathcal{A}$ of codewords of weight 8 in C such that any triple error hits the support of at least one vector $\boldsymbol{c} \in \mathcal{A}$ in at least two coordinates.

[2] By varying the parameters of the code it is possible to raise this complexity up to 2^{85} operations.

(see Chapter xx (Huffman)). For $\mathcal{G}_{24}$ this yields $|M(24,12,3)| \geq 14$; therefore, the collection of information sets constructed in the previous example has the minimal possible size.

More generally, one can define a *covering system* $N_r(n; m_1, m_2, \dots, m_r; t_1, t_2, \dots, t_r)$ as a collection $\mathcal{F}$ of r-tuples of pairwise disjoint subsets $(F_1, F_2, \dots, F_r)$ of respective size m_i with $\cup F_i = \mathcal{N}$ such that every r-tuple of pairwise disjoint subsets $(E_1, E_2, \dots, E_r)$, $|E_i| = t_i$, is contained in at least one element of $\mathcal{F}$ in the sense that $E_1 \subseteq F_1, \dots, E_r \subseteq F_r$. Then

$$M(n, m, t) = N_2(n; m, n-m; t, 0).$$

(*iii*) Algorithm 3.4 has a variety of different applications. In Sections 3.5.1, 3.5.2 we explain that the problems of computing the minimum distance and weight spectrum can be approached with in essence the same techniques as decoding. Covering set decoding can be reformulated as a probabilistic algorithm for these problems.

(*iv*) Here is another version of covering set decoding that uses the parity-check matrix H. Pick a random $(n - k + \mathcal{O}(\sqrt{n}))$-subset E of $\mathcal{N}$. By Lemma 3.5, for almost all matrices H, submatrix $H(E)$ will be nonsingular, and H can be transformed to the form $H' = [B \mid I_{n-k}]$, where the identity matrix is located within E. Performing the order of $L_n(k)$ independent attempts, we ensure, with high probability, that in one of them all d_0 (or fewer) errors are located within E. Therefore, any attempt that yields the syndrome $\boldsymbol{s}' = H'\boldsymbol{y}^T$ of weight at most d_0, supplies us with a plausible error vector $\boldsymbol{e}'$. This error vector should be formed by taking zeros on $\mathcal{N} \setminus E$ and nonzero symbols in those coordinates of E that correspond to nonzeros in the syndrome (cf. Remark 3.4(*iii*)).

Example 3.4 (CRYPTANALYSIS.) One convenient setting for examples is that of public-key encryption schemes based on computationally difficult coding problems. This application is particularly attractive because these schemes use relatively long codes that sustain brute force attacks.

The McEliece cryptosystem is based on the use of an $[n = 1024, k = 524]$ binary linear code $\mathcal{A}$ with a relatively simple decoding algorithm correcting $t = 50$ errors (for instance, a Goppa code). Let A be a generator matrix of $\mathcal{A}$ and $\boldsymbol{x}$ a plaintext of k bits. The ciphertext is calculated as

$$\boldsymbol{z} = \boldsymbol{x}G + \boldsymbol{e},$$

where $G = SAP$ with S a $k \times k$ nondegenerate scrambler matrix and P an $n \times n$ permutation matrix, and $\boldsymbol{e}$ is an n-vector with t ones. Matrix G is a public key and its decomposition is kept secret. A legal user calculates $\boldsymbol{y} = \boldsymbol{z}P^{-1} = \boldsymbol{x}SA + \boldsymbol{e}P^{-1}$ and decodes $\boldsymbol{y}$ using the decoding algorithm of $\mathcal{A}$ in order to find $\boldsymbol{u} = \boldsymbol{x}S$. Then $\boldsymbol{z}$ is found as $\boldsymbol{u}S^{-1}$. An unauthorized person presumably has to perform the decoding of a random-looking linear code with the generator matrix G.

Let us analyze the complexity of this decoding. We shall discuss only the covering set decoding algorithm. Clearly, it can be adjusted to the correction of

Theorem 3.24 *Covering set decoding for almost all codes has error probability equivalent to that of complete minimum distance decoding. The decoding can be implemented by a sequential algorithm with time complexity at most* $\mathcal{O}\big(n^4(\log n)q^{n\alpha_q(R)+\sqrt{nH_2(R)\log_q 2}}\big) = q^{n\alpha_q(R)(1+o(1))}$, *where*

$$\alpha_q(R) = (\log_q 2)\Big[H_2(\delta_0) - (1-R)H_2\Big(\frac{\delta_0}{1-R}\Big)\Big].$$

Proof: A decoding error can occur if the transmitted codeword is not the closest one in the code to the received word, which happens with probability p_c, or if the repeated choice fails to find an error-free k-set. The probability that a randomly chosen k-set W is not error-free equals

$$1 - \binom{n-k}{d_0}\Big/\binom{n}{d_0}.$$

Performing the choice independently $L_n(k)$ times we observe that the probability of this falls as $e^{-n\log n}$. This term declines faster than the error probability of minimum distance decoding p_c; hence, its contribution to the overall error rate is negligible. This proves the first part of our claim.

The complexity of each independent decoding attempt is formed by the time needed to diagonalize the matrix G with respect to W, which takes at most n^3 operations, and a number of linear-time subroutines. Hence the overall complexity is at most $\mathcal{O}(n^3 L_n(k)|L(W)|)$. The required expression for the exponent follows by (1.1). ∎

Remark 3.8 (i) In implementations, the independent diagonalization on each step is quite impractical. It is possible to pass from one information window to the other by swapping a pair of columns in the matrix. For codes of length 1000 this allows to speed up the computations by several thousand times; see the next example.

(ii) A *covering (covering design)* $M(n,m,t)$ is a collection of m-subsets of$\mathcal{N}$ defined as follows

$$M(n,m,t) = \big\{\mathcal{F} \subseteq \mathcal{N}^m\colon\ \forall_{E\in\mathcal{N}^t}\,\exists_{F\in\mathcal{F}}(E\subseteq F)\big\}.$$

In words: $M(n,m,t)$ is a collection $\mathcal{F}$ of m-subsets of $\mathcal{N}$ such that every t-subset is contained in some $F\in\mathcal{F}$. Coverings are studied in combinatorics. It is known that

$$\min|M(n,n-k,d_0)| \sim \binom{n}{d_0}\Big/\binom{n-k}{d_0}.$$

The upper bound in this equality (Erdős and Spencer [59, Thm. 13.4]) is proved by a probabilistic argument of which Theorem 3.24 is derived.

It is easy to bound the number $|M(n,k,t)|$ from below. Namely, a standard double counting argument shows that $k|M(n,k,t)| \ge n|M(n-1,k-1,t-1)|$, which combined with $|M(n,k,1)| = \lceil n/k\rceil$ yields the following lower bound:

$$|M(n,k,t)| \ge \left\lceil \frac{n}{k}\left\lceil \frac{n-1}{k-1}\cdots\left\lceil\frac{n-t+1}{k-t+1}\right\rceil\cdots\right\rceil\right\rceil$$

$$
G = \left[\begin{array}{cccccccccccc|cccccccccccc}
1 &&&&&&&&&&&& 1&0&0&1&1&1&1&1&0&0&0&1 \\
& 1 &&&&&&&&&&& 0&1&0&1&0&0&1&1&1&0&1&1 \\
&& 1 &&&&&&&&&& 0&0&1&1&0&1&0&1&0&1&1&1 \\
&&& 1 &&&&&&&&& 1&1&1&1&0&0&0&0&1&1&1&0 \\
&&&& 1 &&&&&&&& 1&0&0&0&1&0&0&1&1&1&1&1 \\
&&&&& 1 &&&&&&& 1&0&1&0&0&1&1&1&1&0&1&0 \\
&&&&&& 1 &&&&&& 1&1&0&0&0&1&1&0&0&1&1&1 \\
&&&&&&& 1 &&&&& 1&1&1&0&1&1&0&1&0&1&0&0 \\
&&&&&&&& 1 &&&& 0&1&0&1&1&1&0&0&1&1&0&1 \\
&&&&&&&&& 1 &&& 0&0&1&1&1&0&1&1&1&1&0&0 \\
&&&&&&&&&& 1 && 0&1&1&1&1&1&1&0&0&0&1&0 \\
&&&&&&&&&&& 1 & 1&1&1&0&1&0&1&0&1&0&0&1
\end{array}\right].
$$

Form a collection of 14 information sets by applying the following permutations on the set $\{1, 2, \ldots, 24\}$:

$$\sigma_1^i \circ \sigma_2^j, \quad 0 \le i \le 1,\ 0 \le j \le 6,$$

$$\sigma_1 \text{ swaps halves: } \{1, 2, \ldots, 12\} \leftrightarrow \{13, 14, \ldots, 24\},$$

$$\sigma_2 = (4, 7, 16, 10, 22, 19, 13)(5, 8, 17, 11, 23, 20, 14)(6, 9, 18, 12, 24, 21, 15).$$

It can be shown that these information sets are sufficient to correct any 3 errors. Furthermore, in Remark 3.8(ii) we argue that 14 is the minimal possible number of sets. If the 14 generator matrices are stored in the memory, then the decoding amounts to 14 encodings and comparisons, i.e., about 180 operations with vectors of length 12. In greater detail this example is considered in Chapter xx (Huffman). ∎

To implement the general algorithm, we have to specify a way of choosing the information sets. One obvious suggestion is to take random uniformly distributed k-subsets of $\mathcal{N}$. We call the next algorithm *covering set decoding* (the name will be clear in a while). Let

$$L_n(k) = (n \log n)\binom{n}{d_0}\bigg/\binom{n-k}{d_0}.$$

Algorithm 3.4: Covering set decoding

- Set $\boldsymbol{c} = 0$.
- Choose randomly a k-subset W. Form a list of codewords $L(W) = \{\boldsymbol{c} \in C \mid \boldsymbol{c}(W) = \boldsymbol{y}(W)\}$.
- If there is a $\boldsymbol{c}' \in L(W)$ such that $\mathsf{dist}\,(\boldsymbol{c}', \boldsymbol{y}) < \mathsf{dist}\,(\boldsymbol{c}, \boldsymbol{y})$, assign $\boldsymbol{c} \leftarrow \boldsymbol{c}'$.
- Repeat the last two steps $L_n(k)$ times. Output $\boldsymbol{c}$.

We know from the previous section (Corollary 3.8) that for most codes, for any choice of W the list $L(W)$ will be of small size. Therefore, the time complexity of this algorithm is determined by the quantity $L_n(k)$. Let us formulate the properties of this algorithm as a theorem.

3.3.4 Information set decoding

Algorithms considered below in this section restrict themselves to decoding in the sphere of radius $d_0 = n\delta_0(R)$. By Theorem 3.4, for most long codes this is sufficient for complete decoding. Denote by p_c the error probability of complete minimum distance decoding.

Let G be a generator matrix and H a parity-check matrix of C. Let $\mathcal{W}$ be the collection of all information sets of C. Given a message $\boldsymbol{a}$ we find the codeword corresponding to it as $\boldsymbol{c} = \boldsymbol{a}G$. If for a certain k-subset $W \subset \mathcal{N}$ the submatrix $G(W) = I_k$, the symbols of $\boldsymbol{a}$ will appear on this subset of coordinates in $\boldsymbol{c}$. Using our notation,

$$\boldsymbol{a}(W) = \boldsymbol{c}(W).$$

If the error vector $\boldsymbol{e}$ has zeros on W, we shall be able to find the transmitted word $\boldsymbol{c}$ from the received word $\boldsymbol{y} = \boldsymbol{e} + \boldsymbol{c}$. This enables us to formulate the following general decoding method.

Algorithm 3.3: Information set decoding

- Set $\boldsymbol{c} = 0$.
- Choose an information set W. Compute the codeword $\boldsymbol{c}' = \boldsymbol{y}(W)G$. If $\mathsf{dist}\,(\boldsymbol{c}', \boldsymbol{y}) < \mathsf{dist}\,(\boldsymbol{c}, \boldsymbol{y})$, assign $\boldsymbol{c} \leftarrow \boldsymbol{c}'$.
- Repeat for all information windows. Output $\boldsymbol{c}$.

Recall that if W is an information set, the remaining subset $B = \mathcal{N} \setminus W$ is called a check set. Clearly, $\mathsf{rank}\,(H(B)) = n - k$.

Theorem 3.23 *The information set decoding algorithm performs complete minimum distance decoding.*

Proof: Let $\boldsymbol{s}$ be a nonzero syndrome and $\boldsymbol{l}$ a leader of the coset defined by it. Let $E = \mathsf{supp}\,(\boldsymbol{l})$. We have to prove that $\mathcal{N} \setminus E$ contains an information set or that E is contained in a check set. Indeed, for any nonzero $\boldsymbol{s}$ and any check set B there is a vector $\boldsymbol{e}$ such that

$$H\boldsymbol{e}^T = \boldsymbol{s}, \quad \mathsf{supp}\,(\boldsymbol{e}) \subseteq B.$$

Hence $|E| \le n - k$. Moreover, $\boldsymbol{l}$ is a vector of minimal weight in the coset and therefore, in particular, for no other vector $\boldsymbol{e}$ with the same syndrome, $\mathsf{supp}\,(\boldsymbol{e}) \subseteq E$. This implies that $\mathsf{rank}\,H(E) = |E|$; hence E can be augmented to form a check set. ∎

Example 3.3 Let $C = \mathcal{G}_{24}$ be the $[24, 12, 8]$ extended Golay code given by a generator matrix G. We want to use information set decoding for the correction of 3 errors. Let

Theorem 3.21 *Let C be a binary linear code all of whose codewords have even weight and let $\mathcal{T} \subseteq C$ be a test set. Then $|\mathcal{T}| \geq Z_{\min}$.* ∎

Therefore, general improvements of zero-neighbors decoding within the gradient-like approach are impossible.

Theorem 3.21 also implies that for even codes $|\mathcal{M}| \geq Z_{\min}$. However, for these codes the following stronger result holds true.

Theorem 3.22 *Let C be as in the previous theorem. Then the set $\mathcal{Z}_{\min}$ can be chosen so that $\mathcal{Z}_{\min} \subseteq \mathcal{M}$.* ∎

Remark 3.7 Generally, zero neighbors need not to be minimal codewords. Indeed, consider the code $C = \{0000, 1100, 0011, 1111\}$. Vector 0110 is equally far from all the codewords meaning that they all, except 0, are zero neighbors. However, 1111 is not minimal.

Note that gradient-like algorithms require a nontrivial preprocessing for the construction of the test set of codewords. For simplicity we have included proportional codewords into the test set. In implementations there is of course no need to store them.

Finding zero neighbors and minimal codewords usually is a difficult task. However, note a difference: whereas checking whether a given vector is a zero neighbor seems computationally hard, the question of a vector being minimal is solved immediately by Property (1) of minimal vectors.

Example 3.1 Binary $[n = 2^m - 1, k = n - m, 3]$ Hamming codes. The code is perfect; therefore, its zero neighbors are exactly $(1/3)\binom{n}{2}$ words of weight 3. Generally, there are

$$A_w = \frac{1}{2^m}\left[\binom{n}{w} + (-1)^{\lceil w/2 \rceil} n \binom{2^{m-1} - 1}{\lfloor w/2 \rfloor}\right]$$

vectors of weight w in the code; it can be shown that

$$M_w = \frac{1}{w!}\prod_{i=0}^{w-2}(2^m - 2^i)$$

among them are minimal. For instance, for $m = 6$ this gives $|\mathcal{M}| = 4994493 = 2^{22.3}$, $Z_{\min} = 651$.

Of course, the simplest way to decode in this case is syndrome decoding. To perform it, we only need to compute the syndrome of the received vector.

Example 3.2 Binary $[n = 2^m, k = 1 + m + m(m-1)/2, 2^{m-2}]$ second order Reed–Muller codes. The number of zero neighbors in the code can be estimated by the number of all code vectors of weight $< 2d - 1 = n/2 - 1$. For instance, let $m = 6$, then the code has at most 1183084 zero neighbors. The number of minimal codewords equals 3821804. The entire code has $2^{22} = 4194304$ vectors. The number of cosets is much higher; therefore, the table of zero neighbors has much smaller size than other possible decoding tables. ∎

Let us formulate properties of the zero-neighbors decoding.

Theorem 3.19 *Let C be an $[n, k = Rn]$ linear code. For any $\boldsymbol{y} \in E_q^n$, zero-neighbors decoding always finds a closest codeword. For almost all codes it can be implemented by a sequential algorithm with both time and space complexity $q^{n\alpha_q(R)}$.* ∎

For instance, for $q = 2$, the complexity of this decoding is exponentially smaller than that of exhaustive search for $R \geq 1 - H_2(1/4) = 0.189$, and thus also smaller than the complexity of minimal-vector decoding.

The only property of the set $\mathcal{Z}$ that is essential for successful decoding is formulated in (3.7):

$$\mathcal{X}(D(0)) \subset \bigcup_{\boldsymbol{z} \in \mathcal{Z}} D(\boldsymbol{z}). \tag{3.8}$$

Thus, we may further restrict the test set of vectors by choosing a *smallest* subset of $\mathcal{Z}$ with this property. Denote this subset by $\mathcal{Z}_{\min}$. Note that though the set $\mathcal{Z}_{\min}$ may be not unique, its size is well defined. Therefore, let $Z_{\min} = |\mathcal{Z}_{\min}|$.

The set $\mathcal{Z}_{\min}$ has the following property that confirms an intuitive picture of zero neighbors.

Lemma 3.20 *Let $\boldsymbol{z} \in \mathcal{Z}_{\min}$ and let $\boldsymbol{y}_i \in \mathcal{X}(D(0)) \cap D(\boldsymbol{z})$. Then there exists a chain of immediate descendants $\{0 = \boldsymbol{y}_0 \prec \cdots \prec \boldsymbol{y}_{i-1} \prec \boldsymbol{y}_i \prec \cdots \prec \boldsymbol{z}\}$ such that every member satisfies $\boldsymbol{y}_i \in D(0)$ or $\boldsymbol{y}_i \in D(\boldsymbol{z})$ or $\boldsymbol{y}_i \in D(0) \cap D(\boldsymbol{z})$. If furthermore $\boldsymbol{y}_i \in D(\boldsymbol{z})$ then $\boldsymbol{y}_i \notin D(\boldsymbol{z}'), \boldsymbol{z}' \in \mathcal{Z}_{\min}, \boldsymbol{z}' \neq \boldsymbol{z}$.*

Proof: Let $\boldsymbol{z} \in \mathcal{Z}_{\min}$. Since $\mathcal{Z}_{\min}$ is minimal, there is a $\boldsymbol{y} \in \mathcal{X}(D(0)) \cap D(\boldsymbol{z})$ such that $\boldsymbol{y} \neq D(\boldsymbol{z}')$ for any other $\boldsymbol{z}' \in \mathcal{Z}_{\min}$. Let $\mathsf{wt}\,(\boldsymbol{y}) = w$ and let $\boldsymbol{x} \prec \boldsymbol{y}$ be a point in $D(0)$ of weight $\mathsf{wt}\,(\boldsymbol{x}) = w - 1$. Form a chain

$$0 = \boldsymbol{y}_0 \prec \cdots \prec \boldsymbol{y}_{w-1} = \boldsymbol{x} \prec \boldsymbol{y}_w = \boldsymbol{y} \prec \cdots \prec \boldsymbol{z},$$

where each member $\boldsymbol{y}_i$ is an immediate descendant of $\boldsymbol{y}_{i+1}$. It can be seen that $\boldsymbol{y}_i \in D(0), 0 \leq i \leq w - 1$. If $w \leq i \leq \mathsf{wt}\,(\boldsymbol{z})$, then

$$\begin{aligned}
\mathsf{dist}\,(\boldsymbol{y}_i, \boldsymbol{z}) &= \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{z}) - \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{y}_i) \leq \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{z}') - \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{y}_i) \\
&\leq \mathsf{dist}\,(\boldsymbol{y}_i, \boldsymbol{z}') \quad \forall \boldsymbol{z}' \in C, \\
\mathsf{dist}\,(\boldsymbol{y}_i, \boldsymbol{z}) &< \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{z}') - \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{y}_i) \leq \mathsf{dist}\,(\boldsymbol{y}_i, \boldsymbol{z}'), \quad \forall \boldsymbol{z}' \in \mathcal{Z}_{\min}, \boldsymbol{z}' \neq \boldsymbol{z}.
\end{aligned}$$

Thus, $\boldsymbol{y}_i \in D(\boldsymbol{z})$ and $\boldsymbol{y}_i \notin D(\boldsymbol{z}'), \boldsymbol{z}' \in \mathcal{Z}_{\min}, \boldsymbol{z}' \neq \boldsymbol{z}$. ∎

For the set $\mathcal{Z}_{\min}$ the upper bound in Lemma 3.17 can be made one less. Indeed, minimality of the subset $\mathcal{Z}_{min}$ implies that $\boldsymbol{x}$ in the proof of the lemma must be contained in $D(z)$. Therefore, $\mathsf{dist}\,(\boldsymbol{x}, \boldsymbol{z}) \leq t$. Unfortunately, better estimates of $Z_{\min}$ than those in Lemma 3.18 seem difficult to achieve. However, as is shown by the following theorem, stated here without proof, the size of $\mathcal{Z}_{\min}$ gives, in certain cases, a *lower* estimate on the size of *any* test set.

Remark 3.6 This definition is motivated by the following example. Let $C = \{\boldsymbol{c}_0 = 000, \boldsymbol{c}_1 = 111\}$, then $D(\boldsymbol{c}_0)$ is formed by $\boldsymbol{c}_0$ and the vectors of weight one and $D(\boldsymbol{c}_1)$ by $\boldsymbol{c}_1$ and the vectors of weight two. Thus, if points outside $D(\cdot)$ are not included in the boundary, $\partial D(\boldsymbol{c}_0)$ and $\partial D(\boldsymbol{c}_1)$ would be disjoint, which seems unreasonable.

Definition. A nonzero codeword $\boldsymbol{z} \in C$ is called a *zero neighbor* if its Voronoi region shares a common boundary with $D(0)$, i.e., if $\partial D(\boldsymbol{z}) \cap \partial D(0) \neq \emptyset$.

Clearly, if $\boldsymbol{z}$ is a zero neighbor, then so are all of its scalar multiples. Let $\mathcal{Z}$ be the set of zero neighbors. The definition has the following simple consequence:

$$(\,\mathcal{X}(D(0)) \cap D(\boldsymbol{z}) \neq \emptyset\,) \quad \Rightarrow \quad \boldsymbol{z} \in \mathcal{Z}. \tag{3.7}$$

Indeed, $\boldsymbol{x} \in \mathcal{X}(D(0)) \cap D(\boldsymbol{z})$ implies that there is a $\boldsymbol{y} \in D(0)$ at a distance 1 from $\boldsymbol{x}$. Hence $\boldsymbol{y} \in \partial D(0) \cap \partial D(\boldsymbol{z})$.

Decoding with zero neighbors proceeds in the same way as with minimal supports except that now we choose the test set $\mathcal{T}$ in Algorithm 3.2 equal to $\mathcal{Z}$. This version of the algorithm is called *zero-neighbors decoding*.

The zero-neighbors decoding always converges to the closest codeword. To justify this we again verify that $\mathcal{Z}$ is a test set.

Theorem 3.16 *The zero-neighbors algorithm performs complete minimum distance decoding.*

Proof: Let $\boldsymbol{y} \notin D(0)$. Consider a chain of inclusions $0 = \boldsymbol{y}_0 \prec \boldsymbol{y}_1 \cdots \prec \boldsymbol{y}_{i-1} \prec \boldsymbol{y}_i \prec \cdots \prec \boldsymbol{y}$, where $\mathsf{wt}\,(\boldsymbol{y}_i) = i$. Clearly there exists a number i such that $\boldsymbol{y}_{i-1} \in D(0)$ and $\boldsymbol{y}_i \in \partial D(0) \setminus D(0)$. Then $\boldsymbol{y}_i \in D(\boldsymbol{z})$ for some $\boldsymbol{z} \in \mathcal{Z}$. We have

$$\begin{aligned} \mathsf{wt}\,(\boldsymbol{y} - \boldsymbol{z}) = \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{z}) &\leq \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{y}_i) + \mathsf{dist}\,(\boldsymbol{y}_i, \boldsymbol{z}) \\ &< \mathsf{dist}\,(\boldsymbol{y}, \boldsymbol{y}_i) + \mathsf{dist}\,(\boldsymbol{y}_i, 0) = \mathsf{wt}\,(\boldsymbol{y}). \end{aligned}$$

Hence $\mathcal{Z}$ is a test set and the theorem follows. ∎

The complexity of zero-neighbors decoding is determined by the size of $\mathcal{Z}$. This can be easily estimated from the following lemma.

Lemma 3.17 *For all $\boldsymbol{z} \in \mathcal{Z}$, $\mathsf{wt}\,(\boldsymbol{z}) \leq 2t + 2$, where t is the covering radius of C.*

Proof: Let $\boldsymbol{x}$ be a point in $\partial D(\boldsymbol{z}) \cap \partial D(0)$. Then $\mathsf{dist}\,(\boldsymbol{z}, 0) \leq \mathsf{dist}\,(\boldsymbol{z}, \boldsymbol{x}) + \mathsf{dist}\,(\boldsymbol{x}, 0) \leq (t+1) + (t+1)$. ∎

Hence, combining Lemma 3.2 and Theorem 3.4 we get the following.

Lemma 3.18 *For almost all codes, $|\mathcal{Z}| \leq q^{n\alpha_q(R)}$, where*

$$\alpha_q(R) = \begin{cases} R, & 0 \leq R \leq 1 - H_q\Big(\dfrac{q-1}{2q}\Big), \\ (H_q(2\delta_0) - (1-R))(1 + o(1)), & 1 - H_q\Big(\dfrac{q-1}{2q}\Big) < R \leq 1. \end{cases}$$

∎

Lemma 3.15 $\quad \mathsf{Var}\, M_w \le \mathsf{E} M_w(1+2^{-d/2}\mathsf{E} M_w)$. ∎

Passing to the exponential form, we see that

$$2^{-d/2}\mathsf{E} M_w < 2^{n(H_2(w/n)-(1-R)-(d/2n))}.$$

By property (2) below, $w/n \le 1-R+\frac{1}{n}$. Plugging this in and taking codes meeting the GV bound, we see that this exponent is negative for $0<R\le 0.26$. Thus, at least for this range of code rates, $\mathsf{Var}\, M_w \le \mathsf{E} M_w(1+o(1))$. Therefore by (1.5), the worst-case complexity of minimal-vectors decoding for most binary codes has the same order of magnitude as the average-case complexity.

We conclude this part by stating some further properties of minimal supports in binary linear codes.

Properties of minimal supports.

(1). Let $E \subset \mathcal{N}$ be a support of a codeword in C. Then E is minimal if and only if $\mathsf{rk}\,(H(E)) = |E|-1$, where H is the parity-check matrix of C.

(2). (E is minimal) $\Rightarrow$ ($|E| \le n-k+1$).

(3). Every support of size $|E| \le 2d-1$ is minimal.

Proof: The *only if* part of Property (1) is obvious as is Property (3).

(1), $\Leftarrow$. Let $\boldsymbol{h}_i$ be the ith column of $H(E)$. By the assumption, there exist $w=|E|$ nonzero numbers λ_i such that

$$\sum_{i=1}^{w} \lambda_i \boldsymbol{h}_i = 0$$

and some $w-1$ of these columns, say the first, are linearly independent. Suppose there exists $\boldsymbol{c}', \boldsymbol{c}' \prec \boldsymbol{c}$, i.e., there exists a vanishing linear combination of columns that does not involve at least one of the first $w-1$ columns, for instance,

$$\sum_{i=2}^{w} \mu_i \boldsymbol{h}_i = 0$$

with $\mu_w \ne 0$. Multiply this sum by λ_w/μ_w and subtract from the first one. This gives a linear dependence between the first $w-1$ columns, a contradiction.

Property (2) is implied by (1).

Examples are deferred until the end of the section.

Zero neighbors

Let us show that it is possible to reduce the number of codewords inspected by the gradient-like search while preserving the decoding performance.

Let $A \subseteq E_q^n$ and let $\mathcal{X}(A)$ be formed by all the points of E_q^n at distance 1 from A:

$$\mathcal{X}(A) = \{\boldsymbol{x} \mid \mathsf{dist}\,(\boldsymbol{x}, A) = 1\}.$$

Define the *boundary* of A as follows:

$$\partial A = \mathcal{X}(A) \cup \mathcal{X}(E_q^n \setminus A).$$

case, minimal supports define a set of lines in the code. Note that no minimal codeword covers a nonzero codeword with a smaller support (thus justifying the name).

From here until the end of this part we consider only binary codes. Set $\mathcal{T} = \mathcal{M}$ in the gradient-like decoding algorithm. We call this version *minimal-vectors decoding.*

Theorem 3.13 *The minimal-vectors algorithm for binary codes performs complete minimum distance decoding.*

Proof: Let $\boldsymbol{y} \notin D(0)$. Then there is a codeword $\boldsymbol{c}$ such that

$$\mathsf{wt}\,(\boldsymbol{y}+\boldsymbol{c}) < \mathsf{wt}\,(\boldsymbol{y}). \tag{3.6}$$

Expand $\boldsymbol{c}$ into a sum of minimal vectors according to Lemma 3.12. The supports of these vectors do not intersect. Therefore, at least one of them satisfies Eq. (3.6). The proof now follows from Theorem 3.11. ∎

The complexity of this algorithm is determined by the size of the set $\mathcal{M}$. Let us determine it for long linear codes. Let C be a random linear code and N_w (resp., M_w) be the number of vectors (resp., minimal vectors) of weight w in it.

Lemma 3.14 *Let $w = (n-k+1) - \ell$, $\ell \to \infty$. Then*

$$\mathsf{E}M_w = \binom{n}{w} 2^{-(n-k)} \prod_{i=0}^{w-2} (1 - 2^{-(n-k-i)}) \sim \mathsf{E}N_w.$$

Proof: Let H be the parity-check matrix of C. Let $\pi_{n,k}(w)$ be the probability that a given support is minimal, then $\mathsf{E}M_w = \binom{n}{w}\pi_{n,k}(w)$. The event considered is that some (say, first) $w-1$ columns of H among the chosen w columns are linearly independent and the remaining column is their linear combination with $w-1$ nonzero coefficients. The number of collections of w columns that satisfy the above conditions equals

$$(2^{n-k}-1)(2^{n-k}-2)\dots(2^{n-k}-2^{w-2})$$

and the total number of choices is $2^{w(n-k)}$. The probability $\pi_{n,k}(w)$ equals the quotient of these quantities.

The asymptotic equality follows by (1.7). ∎

For ℓ fixed, the quotient $\mathsf{E}M_w/\mathsf{E}N_w$ tends to a positive constant less than one. For small ℓ its values are shown in Remark 3.3.

This enables us to conclude that *on average* the time complexity of minimal-vectors decoding for long codes does not improve the complexity of examining all codewords or syndromes. In addition this algorithm requires a space of size $|\mathcal{M}|$. To bound the worst-case complexity one should find the variance of M_w. We discuss it only for the binary case. The following lemma can be proved along the lines of the proof of Lemma 1.1.

Let us prove that this algorithm always converges to the nearest codeword.

Theorem 3.11 *For any set of codewords satisfying* (3.5) *the gradient-like algorithm performs a complete minimum-distance decoding. The time complexity of the algorithm equals* $\mathcal{O}(n^2|\mathcal{T}|)$*. The space complexity equals* $\mathcal{O}(n|\mathcal{T}|)$*.*

Proof: Let $\boldsymbol{y} \notin D(0)$. The algorithm expands $\boldsymbol{y}$ into a sum of test vectors. Suppose that after m steps no further test vectors satisfying (3.5) are found. This means that we managed to bring $\boldsymbol{y}$ "down" to $D(0)$:

$$\boldsymbol{e} = \boldsymbol{y} - \sum_{u=1}^{m} \boldsymbol{z}_u \in D(0).$$

However this means that $\boldsymbol{y} \in D(\sum_{u=1}^{m} \boldsymbol{z}_u)$. ∎

Submitting a codeword $\boldsymbol{c} \neq 0$ to this algorithm, we observe that it constructs a decomposition of zero in the form

$$0 = \boldsymbol{c} - \sum_{u} \boldsymbol{z}_u.$$

In addition we can observe that in each step the algorithm produces a vector of a strictly smaller weight. Let us formulate this as a lemma.

Lemma 3.12 *Let* $\mathcal{T} \subseteq C$ *be a test set. Then any codeword* $\boldsymbol{c} \neq 0$ *can be decomposed into a sum*

$$\boldsymbol{c} = \sum_{u=1}^{m} \boldsymbol{z}_u, \quad \boldsymbol{z}_u \in \mathcal{T},\ m \geq 1$$

where

$$\mathsf{wt}(\boldsymbol{c}) > \mathsf{wt}(\boldsymbol{c} - \boldsymbol{z}_1) > \mathsf{wt}(\boldsymbol{c} - (\boldsymbol{z}_1 + \boldsymbol{z}_2)) > \cdots \geq 0$$ ∎

Thus, the linear hull of $\mathcal{T}$ equals the entire code C. We discuss two ways of constructing the test set $\mathcal{T}$.

Minimal vectors

Let $\mathsf{supp}(\boldsymbol{x}) = \{i \in \mathcal{N} \mid x_i \neq 0\}$ be the *support* of $\boldsymbol{x}$. If $\mathsf{supp}(\boldsymbol{x}) \subset \mathsf{supp}(\boldsymbol{y})$ (resp., $\subseteq$), we also write $\boldsymbol{x} \prec \boldsymbol{y}$ (resp., $\preceq$).

The above lemma suggests to try to construct $\mathcal{T}$ in such a way that every codeword $\boldsymbol{c}$ outside $\mathcal{T}$ covers a certain codeword $\boldsymbol{z}$ from this set, i.e., $\boldsymbol{z} \prec \boldsymbol{c}$. Then $\mathcal{T}$ can be viewed as a set of minimal codewords. Let us give a definition.

Definition. A codeword $\boldsymbol{c} \in C$ is called *minimal* if $0 \neq \boldsymbol{c}' \preceq \boldsymbol{c}$ implies that $\boldsymbol{c}' = a\boldsymbol{c}$ for a nonzero constant a. The support of a minimal codeword is called minimal with respect to C.

Let $\mathcal{M}$ be the set of minimal codewords of a given code C. For binary codes, $\mathcal{M}$ can be also viewed as the set of minimal supports. In the general

Algorithm 3.1: Split syndrome decoding

Precomputation stage: For every weight $t, 1 \le t \le d_0$, find the point m such that the tables X_ℓ and X_r have an (almost) equal size. Store the pair (m, u) in the set $E(t)$.

- Compute $\boldsymbol{s} = H\boldsymbol{y}^T$ and set $t \leftarrow 1$.
- For every entry of $E(t)$, form the tables X_ℓ and X_r as described.
- Order X_ℓ with respect to the entries $\boldsymbol{s}_\ell$.
- For every entry of X_r check whether X_ℓ contains the vector $\boldsymbol{s}_\ell = \boldsymbol{s} - \boldsymbol{s}_r$. If this is found, then output $\boldsymbol{c} = \boldsymbol{y} - (\boldsymbol{e}_\ell|\boldsymbol{e}_r)$ and STOP.
- Otherwise set $t \leftarrow t + 1$ and repeat the last three steps while $t \le d_0$.

In the remaining part of this section we are going to discuss two principles of minimum distance decoding with reduced complexity, namely, *steepest descent* and *information set decoding*.

Remark 3.5 Note that generally this algorithm is a way of generating the list of solutions of the equation $\boldsymbol{s} = H\boldsymbol{e}^T$ of weight $\mathsf{wt}(\boldsymbol{e}) \le t$ for a given t. The complexity of this algorithm is for high code rates exponentially smaller than that of the exhaustive search.

3.3.3 Gradient-like decoding

Methods of steepest descent are harder to implement in Hamming space than in real space. A general principle of decoding is to construct a set $\mathcal{T}$ of codewords in such a way that every vector $\boldsymbol{y}$ either lies in $D(0)$ or there exists a $\boldsymbol{z} \in \mathcal{T}$ such that

$$\mathsf{wt}(\boldsymbol{y} - \boldsymbol{z}) < \mathsf{wt}(\boldsymbol{y}), \tag{3.5}$$

where $D(0)$ is the Voronoi region of the all-zero vector. Any set $\mathcal{T} \subseteq C$ satisfying this property will be called a *test set*.

This suggests that decoding of a vector $\boldsymbol{y}$ can be accomplished by recursively inspecting the test set for the existence of such a $\boldsymbol{z}$ and subtracting it from the current vector. Let us formulate the algorithm. It assumes that the test set has been precomputed and stored in the memory.

Algorithm 3.2: Gradient-like decoding

- Set $\boldsymbol{c} = 0$.
- Find $\boldsymbol{z} \in \mathcal{T}$ such that $\mathsf{wt}(\boldsymbol{y} - \boldsymbol{z}) < \mathsf{wt}\,\boldsymbol{y}$. Let $\boldsymbol{c} \leftarrow \boldsymbol{c} + \boldsymbol{z}$, $\boldsymbol{y} \leftarrow \boldsymbol{y} - \boldsymbol{z}$.
- Repeat until no such $\boldsymbol{z}$ is found. Output $\boldsymbol{c}$.

product $\boldsymbol{s}_\ell = H_\ell \boldsymbol{e}_\ell^T$ and store it, together with the vector $\boldsymbol{e}_\ell$, as an entry of the table X_ℓ. The total size of X_ℓ thus will be

$$\mathcal{O}\Big(n\binom{m}{u}(q-1)^u\Big).$$

Likewise, form the table X_r of size

$$\mathcal{O}\Big(n\binom{n-m}{t-u}(q-1)^{t-u}\Big)$$

and look for a pair of entries $(\boldsymbol{s}_\ell, \boldsymbol{s}_r)$ that add up to the received syndrome $\boldsymbol{s}$. Therefore, for every given $\boldsymbol{s}_r$ occurring in X_r, we should inspect X_ℓ for the occurrence of $\boldsymbol{s} - \boldsymbol{s}_r$. One practical way to do this is to order X_ℓ with respect to the entries $\boldsymbol{s}_\ell$ whereupon its lookup can be accomplished by binary search in $\mathcal{O}(n)$ steps. Sorting an array of $N = \exp^{\mathcal{O}(n)}$ vectors can be accomplished with both time and space complexity $\mathcal{O}(Nn)$. Therefore, sequential implementations of this algorithm and its generalizations in Sect. 3.3.4 involve exponential memory. Alternatively, sorting can be accomplished by a logical circuit (in this case called a "network") of size $\mathcal{O}(Nn)$ and depth at most $\mathcal{O}(n^2)$.

However, in reality we do not know the number of errors nor their distribution. Therefore, we have to repeat the described procedure for several choices of m and u. In doing so, we may optimize on the choice in order to reduce the total size of the memory used for the tables X_ℓ and X_r. Since this size is a sum of two exponential functions, for every error distribution we must choose the point m so that both tables are (roughly) equally populated. This is possible since the size of X_ℓ grows and the size of X_r falls with the increase of m. The size of the memory is then bounded as $\mathcal{O}(L(t))$, where

$$L(t) = n\Big(\binom{n}{t}(q-1)^t\Big)^{1/2} \le nq^{nH_q(t/n)/2}.$$

For every choice of m there are not more than t different options for the choice of u. Hence repeatedly building the tables not more than nt times, we shall capture any distribution of t errors and on each iteration use not more than $\mathcal{O}(L(t))$ memory bits.

Finally, the entire procedure should be repeated for all $t = 1, 2, \ldots, d_0$ until we find the error vector that has the "received" syndrome $\boldsymbol{s}$.

Below we state the algorithm in tabular form. Let us summarize its properties in a theorem whose proof can be seen to follow from the above discussion.

Theorem 3.10 *For most long codes the split syndrome decoding algorithm has the error probability equivalent to that of complete minimum distance decoding. Its sequential implementation has, for any code of rate R, time complexity*

$$\mathcal{O}(nd_0^2 L(d_0)) = q^{(1/2)n(1-R)(1+o(1))}$$

and space complexity $\mathcal{O}(nL(d_0))$. The algorithm can be implemented by a Boolean circuit of size $\mathcal{O}(nd_0^2 L(d_0))$ and depth $\mathcal{O}(n^2)$. ∎

contributes to the error probability p_c:

$$p_c = \Pr\{\boldsymbol{e} \in E_q^n \setminus L\} \tag{3.4}$$

For the error probability of bounded distance decoding we have

$$\begin{aligned} p_b = \Pr\{\boldsymbol{e} \in E_q^n \setminus (B \cap L)\} &= \Pr\{\boldsymbol{e} \in E_q^n \setminus L\} + \Pr\{\boldsymbol{e} \in L \setminus (B \cap L)\} \\ &\le p_c + \Pr\{\boldsymbol{e} \in B \setminus (B \cap L)\} \end{aligned}$$

The last inequality follows because $|B| = |L|$ and B is formed by the most probable vectors. Finally, observe that the last term describes a part of the event in (3.4), so its probability does not exceed p_c. ∎

Remark 3.4 (i) Theorem 3.4 and this lemma suggest another implementation of syndrome decoding that trades space complexity for time complexity. Namely, inspect all error patterns in a sphere of radius $d_0[n,k]$ around the received word $\boldsymbol{y}$. Their number for long codes is $q^{n(1-R)}$ and they can be easily generated in the course of the decoding.

(ii) In view of this lemma, we can forget about transmission of information and think of the decoding in the following purely combinatorial setting:

Hard-decision decoding. Given a vector $\boldsymbol{y} \in E_q^n$ with $\mathsf{dist}(\boldsymbol{y}, C) \le d_0$ find the closest (joint closest) code vector $\boldsymbol{c}$ to $\boldsymbol{y}$.

Thus, we do not care whether $\boldsymbol{c}$ is the transmitted vector and allow ourselves the assumption that such a vector exists within a distance d_0 of $\boldsymbol{y}$.

(iii) Putting the matrix H in systematic form $\widetilde{H} = [I_{n-k} \,|\, A]$, we observe that if the syndrome $\boldsymbol{s} = \widetilde{H}\boldsymbol{y}^T$ has weight $u < d/2$, then the nonzero coordinates in $\boldsymbol{s}$ locate u errors in the check part (first $n-k$ bits). Indeed, every coset contains at most one vector of weight $< d/2$; on the other hand, we can form such a vector by placing u ones in the check part. Thus, syndromes of weight $< d/2$ need not be decoded.

3.3.2 Split syndrome decoding

We want to reduce the complexity of syndrome decoding by a better arrangement of tables. The idea is to split the syndrome into several parts and build up a list of candidates for decoding. Let $\boldsymbol{y}$ be a received vector and $\boldsymbol{s} = H\boldsymbol{y}^T$ its syndrome. Suppose for a while that the actual number of errors is t. Let us partition $\mathcal{N}$ into two parts, $L = \{1, \ldots, m\}$ and $R = \{m+1, \ldots, n\}$ and let $[H_\ell | H_r]$ be the corresponding partition of the parity-check matrix H. Any error vector $\boldsymbol{e} = (\boldsymbol{e}_\ell | \boldsymbol{e}_r)$ with

$$H\boldsymbol{e}^T = H_\ell \boldsymbol{e}_\ell^T + H_r \boldsymbol{e}_r^T = \boldsymbol{s}$$

is a plausible candidate for the decoding result.

Assume in addition that the number of errors within the subset L equals u, where the numbers m and u are chosen in accordance with the natural restrictions $u \le m$, $t - u \le n - m$. For every possible m-vector $\boldsymbol{e}_\ell$, compute the

3.3 Hard-decision decoding

3.3.1 General remarks

Let C be an $[n, k, d]$ q-ary linear code. We are interested in complete minimum distance decoding in E_q^n with respect to C, i.e., a mapping that given a vector $\boldsymbol{y} \in E_q^n$ finds (one of) the closest codeword(s) in C. Consider a partition of E_q^n into Voronoi regions $D(\boldsymbol{c})$, where

$$D(\boldsymbol{c}) = \{\boldsymbol{x} \in E_q^n \,|\, \mathsf{dist}\,(\boldsymbol{x}, \boldsymbol{c}) \le \mathsf{dist}\,(\boldsymbol{x}, \boldsymbol{c}'), \boldsymbol{c}' \ne \boldsymbol{c}\}.$$

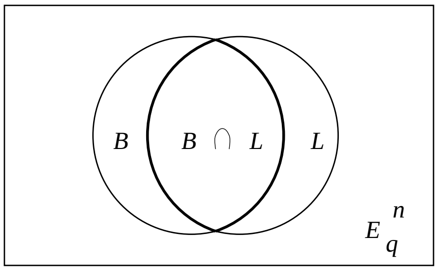
Thus, some points of E_q^n may be contained in several regions. The decoding problem is given a vector $\boldsymbol{y}$ to find in which region it is contained. An obvious way to do this is to inspect all q^k codewords. The time complexity of this is $\mathcal{O}(nq^k)$. Another way is to keep the table of q^{n-k} syndromes and the corresponding coset leaders. Let H be the parity-check matrix of C. *Syndrome decoding* of $\boldsymbol{y}$ is accomplished by computing the syndrome $\boldsymbol{s} = H\boldsymbol{y}^T$ and subtracting the corresponding coset leader $\boldsymbol{e}$ from $\boldsymbol{y}$. The space complexity of this is $\mathcal{O}(nq^{n-k})$. Another implementation of syndrome decoding with the same space complexity suggests constructing a code trellis. This version is discussed in Chapter xx (Vardy), and in Section 3.4.1.

By Theorem 3.4, for most long codes inspecting all error patterns of weight $\le d_0[n, k] + o(n)$ is sufficient for complete decoding. However, the following lemma ensures a very good performance of such decoding even for codes of finite length.

The transmission model that we consider is a q-ary symmetric channel, in which the error patterns are distributed binomially. Let B be the set (maybe not unique) of q^{n-k} most probable error vectors. By definition, $B \subset \{\boldsymbol{e} \in E_q^n \,|\, \mathsf{wt}\,(\boldsymbol{e}) \le d_0\}$.

Lemma 3.9 (Evseev lemma) *Bounded distance decoding in the sphere of radius d_0 at most doubles the error probability p_c of complete decoding.*

Proof:



Let L be the set of q^{n-k} coset leaders. Every error pattern $\boldsymbol{e}$ outside L

The last sum is maximal for $q = 2$. It can be checked not to exceed 0.2. Therefore, the required probability is at most $\binom{n}{k} q^{-(k-\ell)^2}$, which falls exponentially if $\mathsf{corank}\, B > \sqrt{\log_q \binom{n}{k}}$. ∎

Remark 3.3 The probability $\lim_{k\to\infty} \pi(k, k-c)$ that a large square matrix over $\mathbf{F}_q$ has corank c falls very rapidly as c grows. The following table shows this probability for $0 \le c \le 5$:

q \ c	0	1	2	3	4	5
2	.2888	.5776	.1284	.0052	4.7×10^{-5}	9.7×10^{-8}
3	.5601	.4201	.0197	8.7×10^{-5}	4.1×10^{-8}	2.1×10^{-12}
5	.7603	.2376	.0021	6.7×10^{-7}	8.6×10^{-12}	4.4×10^{-18}

Variation on the above argument gives the following results.

Lemma 3.6 *For almost all $k \times n$ matrices over $\mathbf{F}_q$ the corank of every $\ell \times m$ submatrix B, $1 \le \ell, m \le k$, is*

$$\mathsf{corank} B \le \sqrt{\log_q \binom{n+k}{k}}.$$

∎

Lemma 3.7 *For almost all $k \times n$ matrices, every submatrix formed by $k + 2\log_q k$ (cyclically) consecutive columns has rank k.* ∎

The following corollary of Lemma 3.5, which follows by (1.1), is repeatedly used in this section.

Corollary 3.8 *Let C be an $[n, k]$ linear code. The number of codewords that project identically on any given k-subset of $\mathcal{N}$ is at most*

$$\exp_q \sqrt{\log_q \binom{n}{k}} < \exp_q \sqrt{H_2(k/n) \log_q 2}. \tag{3.3}$$

for almost all codes. ∎

3.2.1 Notes

Lemma 3.3 is due to Zyablov and Pinsker [170]. Theorem 3.4 is due to Blinovskii [30]. He also proves in [31] that the lower estimate in Lemma 3.2 holds uniformly for all q^n spheres in E_q^n and is valid for all long codes except a $q^{o(n)\cdot n}$ fraction of them, $o(n) \to \infty$. This again implies Theorem 3.4. Barg and Dumer in [20] prove that the upper bound in Lemma 3.2 holds uniformly for all spheres. Lemma 3.5 is largely a folk lore. In coding literature it was mentioned in Coffey and Goodman [42], Dumer [52], Krouk [101]. The proof in [42] uses Kolmogorov complexity. The rank of large random matrices was also studied in Gerth [75].

An important consequence of this theorem is that for most codes, to perform complete minimum distance decoding, we have to search for the closest code vector in the sphere of radius $d_0 + o(n)$. A closer look at the proof shows that $o(n) = \mathcal{O}(\log n)$.

Even for codes of finite length, bounded distance decoding in the sphere of radius $d_0 = d_0[n, k]$ provides nearly optimal performance. This will follow from Lemma 3.9.

The following lemma forms a basis for information set decoding.

Lemma 3.5 *Let A be a random $k \times n$ matrix over $\mathbf{F}_q$, $k < n$, $k, n \to \infty$. For almost all matrices, the corank of every square $k \times k$ submatrix B is*

$$\mathsf{corank}\, B \leq \sqrt{\log_q \binom{n}{k}}.$$

Proof: Let us estimate the probability $\pi(k, u)$ that a random $k \times k$ matrix has rank u. Every $k \times k$-matrix of rank u corresponds to a linear mapping that takes E_q^k to a u-dimensional space F that can be viewed as a subspace of E_q^k. The number of subspaces is $\left[{k \atop u}\right] = \prod_{j=0}^{u-1}(q^k - q^j)/(q^u - q^j)$. For every choice of the basis in a subspace $F \subset E_q^k$ we have a different linear mapping from E_q^k onto F. The number of bases in F equals $\prod_{j=0}^{u-1}(q^k - q^j)$. Therefore, the number of square matrices of order k and rank u equals $\prod_{j=0}^{u-1}(q^k - q^j)^2/(q^u - q^j)$. Thus,

$$\pi(k, u) = q^{-k^2} \prod_{j=0}^{u-1} \frac{(q^k - q^j)^2}{(q^u - q^j)} < q^{-k^2+ku} \prod_{i=0}^{u-1} \frac{q^{k-i} - 1}{q^{u-i} - 1},$$

where we have bounded $\prod_{j=0}^{u-1}(q^k - q^j) = q^{(1/2)u(u-1)} \prod(q^{k-j} - 1)$ by q^{ku}. Estimating the Gaussian binomial, we obtain

$$\prod_{i=0}^{u-1} \frac{q^{k-i} - 1}{q^{u-i} - 1} < q^{u(k-u)} \prod_{i=0}^{u-1} \frac{q^{u-i}}{q^{u-i} - 1} < q^{u(k-u)} \prod_{i=1}^{u-1} \Big(1 + \frac{1}{q^i - 1}\Big)$$
$$= q^{u(k-u)} \frac{q}{q-1} \Big(1 + \frac{1}{q^2 - 1} + \dots\Big).$$

For $q \geq 2$, the constant factor here is less than 5 since the omitted terms are always less than 1. Thus,

$$\pi(k, u) < 5q^{-(k-u)^2}.$$

Since there are $\binom{n}{k}$ possibilities for B, the probability that there is a submatrix of A with corank $> k - \ell$ does not exceed

$$\sum_{u=0}^{\ell-1} \binom{n}{k} 5q^{-(k-u)^2} < 5\binom{n}{k} q^{-(k-\ell)^2} \sum_{u=0}^{\ell-1} q^{-(k-u)^2+(k-\ell)^2}$$
$$< 5\binom{n}{k} q^{-(k-\ell)^2} \sum_{i=1}^{\ell} q^{-(i+1)^2+1}.$$

Using (1.4), we deduce that the chance for C_0 to have $s_0 \geq q^{\alpha}(q^{n-2\alpha})$ empty spheres is at most $q^{-\alpha}$. Let us restrict our attention to the remaining $(1-q^{-\alpha})$ fraction of $[n, k_0]$ codes with $s_0 < q^{n-\alpha}$.

Hereafter the proof is just first-moment analysis. It proceeds by successively adding cosets to $C_0, C_1, \ldots$ in such a manner that the fraction of empty spheres falls and at the same time the fraction of codes under consideration remains large. After $\log n$ steps we end up with a $(1-o(1))$ fraction of $[n,k]$ codes for which the number of empty spheres is less than one. Thus, in these codes every sphere contains a codeword, and w is an upper bound on their covering radius.

The argument builds on the fact that the average over all $\boldsymbol{x} \in E_q^n$ fraction of empty spheres in $\boldsymbol{x} + C_0$ is s_0/q^n. Thus if we make a code C_1 by augmenting C_0 with $\boldsymbol{x}$ and taking the linear hull, the average over $\boldsymbol{x}$ fraction of empty spheres in it will be at most $(s_0/q^n)^2$ (and their average *number* at most s_0^2/q^n). By (1.4), the fraction of $\boldsymbol{x}$ for which the number of empty spheres in C_1 exceeds $q^{\beta}(s_0^2/q^n)$, is at most $q^{-\beta}$. Hence for a $(1-q^{-\beta})$ fraction of vectors $\boldsymbol{x}$, the code C_1 has at most

$$s_1 = q^{\beta} s_0^2/q^n \leq q^{\beta-2\alpha} \cdot q^n \tag{3.1}$$

empty spheres. Since we can take any of codes C_0 within the $(1-q^{-\alpha})$ fraction of them, we end up with a $(1-q^{-\beta})(1-q^{-\alpha})$ fraction of (k_0+1)-dimensional codes C_1 with few empty spheres.

We proceed as described basing ourselves now on this subset of codes C_1. The recursion stops after $m = \log_2 n$ steps when the average over $\boldsymbol{x}$ number $\overline{s}_m$ of empty spheres falls below one. Then by (3.1),

$$\overline{s}_m = s_{m-1}^2/q^n = q^n q^{2^m(\beta-\alpha)-2\beta} < 1. \tag{3.2}$$

With $m = \log_2 n$, this inequality is satisfied if $\alpha > \beta - 1$ and holds for a $(1-q^{-\alpha})(1-q^{-\beta})^m$ fraction of codes.

Each codeword in a linear code has the same relation to all the other words in the code; hence, if there is one empty sphere w.r.t. C_m, there are at least q^k of them.

$$\Pr\{s_m \geq 1\} = \Pr\{s_m \geq q^k\} \overset{(3.2)}{\leq} \Pr\{s_m \geq q^k \overline{s}_m\} \leq q^{-k}.$$

This leaves us with a $(1-q^{-\alpha})(1-q^{-\beta})^m(1-q^{-k})$ fraction of codes with no empty spheres. If both α and β grow slowly in n, this fraction approaches 1. In each of these $[n,k]$ codes every sphere of radius w is not empty, i.e., contains at least

$$q^{-(n-k)} \sum_{i=0}^{w} \binom{n}{w} (q-1)^w (1-o(1))$$

codewords. By (1.9), this number is positive for large n if $w = d_0(1+o(1))$.

A lower bound for the covering radius follows from the fact that the q^k spheres around codewords must contain all q^n points of E_q^n. Then by definition, the radius of these spheres is at least d_0. ∎

linearly independent s-tuples belonging to the code equals

$$\alpha_n(k,s) = \prod_{i=0}^{s-1}(q^k - q^i) \Big/ \prod_{i=0}^{s-1}(q^n - q^i) \le q^{-(1-R)ns}.$$

Taking the union bound, the probability that an l.i. $(s+1)$-tuple of vectors of weight $\le t$ is contained in a coset is at most $\alpha_n(k,s)$ times the total number of $(s+1)$-tuples in a sphere of radius t:

$$\binom{L_t}{s+1} q^{-(1-R)ns} < (L_t)^{s+1} q^{-(1-R)ns} < q^{n((s+1)H_q(\delta_0-\epsilon)-s(1-R))},$$

where L_t is the volume of the sphere. By concavity,

$$H_q(\delta_0-\epsilon) < H_q(\delta_0) - \epsilon H_q'(\delta_0).$$

From this, the probability that an l.i. s-tuple of codewords is in a sphere of radius t, is at most

$$q^{n((1-R)-(s+1)\epsilon H_q'(\delta_0))}$$

and falls exponentially if

$$s \ge \frac{1-R}{\epsilon H_q'(\delta_0)}.$$

This implies that for most codes the sphere contains no more than q^s words. ∎

The next question that arises is which radius t of the decoding sphere ensures minimum distance decoding or a similar performance. Roughly the answer is that one should take t equal to $d_0 = n\delta_0(R)$. The following result explains this in the asymptotic setting. It is proved by a fine analysis of the ensemble of random linear codes.

Theorem 3.4 *The covering radius of almost all long $[n,k]$ linear codes equals $d_0(1+o(1))$.*

Proof: We begin with a random code C_0 of dimension $k_0 = k - \log_2 n$. Let U_w be the number of codewords of C_0 in a given sphere of radius w, i.e., codewords at a distance $\le w$ from its center. By (1.5) the probability that U_w is small is bounded as follows:

$$\Pr\{U_w \le \mathsf{E}U_w - a\} \le \Pr\{|U_w - \mathsf{E}U_w| \ge a\} \le \frac{\mathsf{Var}\, U_w}{a^2}.$$

Taking $a = q^{\alpha+1/2}\sqrt{\mathsf{E}U_w}$ and using Lemma 1.1 (ii), we get $q^{-2\alpha}$ on the right-hand side. If a sphere contains fewer than $(\mathsf{E}U_w - a)$ codewords, for our purposes we call it *empty* (w.r.t. C_0). Thus for the average, over the choice of C_0, number s_0 of empty spheres we have

$$\mathsf{E}s_0 \le q^{n-2\alpha}.$$

3.2 General properties of linear codes, II

In this section we prove three results of general interest (3.3 to 3.5), which will be used below to reduce the complexity of decoding. Some proofs are difficult, but to understand the decoding algorithms in the next section it is sufficient to know the statements of the results.

A number of hard-decision decoding algorithms search for a closest vector of C within a sphere of a large radius around the received word. Therefore, it is important to estimate the number of codewords in this sphere. Note that counting the number of codewords in a sphere of radius w with center at $\boldsymbol{a}$ is the same as counting the number of words of weight $\le w$ in the coset $C + \boldsymbol{a}$ of the code C.

We begin where we stopped in Sect. 1.3. Let us formulate its last statement as a lemma.

Lemma 3.2 *The number U_w of codewords of a long $[n, Rn]$ linear code in a sphere of radius $w > \delta_0(R)n$ for most codes is*

$$U_w = \begin{cases} q^{n[H_q(w/n)-(1-R)](1+o(1))}, & \delta_0(R) < \dfrac{w}{n} < \dfrac{q-1}{q}, \\ q^{nR(1-o(1))}, & \dfrac{q-1}{q} \le \dfrac{w}{n} \le 1. \end{cases}$$

Remark 3.1 Taking $m = \log^2 n$ in (1.10), it is possible to prove that the fraction of codes that do not satisfy this lemma decays as n^{-n}.

The proof is immediate from (1.7), (1.8) and (1.5). The following is a more surprising result.

Lemma 3.3 *Let $\delta_0 = \delta_0(R)$ be the relative GV distance for the rate R. For almost all $[n, Rn]$ linear codes the number of codewords in a sphere of radius t, where $t = n(\delta_0 - \epsilon)$, is at most*

$$q^{(1-R)/\epsilon H_q'(\delta_0)}$$

and does not depend on n.

Remark 3.2 For δ_0 not too large, i.e., R not too small, $H_q'(\delta_0) \ge H_q(\delta_0)$, and the number of codewords is at most $q^{1/\epsilon}$. For $q = 2$ this is true for $R \ge 0.345$. (Here $H_q'(\delta_0) = \frac{d}{dx}H_q(x)|_{x=\delta_0}$.)

Proof: According to the remark before Lemma 3.2, we have to estimate the number of words of weight $\le t$ in a coset of C. We shall count the number of *linearly independent* (l.i.) words in a coset.

A coset $(C + \boldsymbol{a})$ contains $(s+1)$ l.i. words of weight $\le t$ iff there are s l.i. codewords contained in the sphere of radius $\le t$ centered at $\boldsymbol{a}$. Any $(s+1)$-tuple of l.i. words in a coset gives an s-tuple of l.i. codewords. The fraction of

3 Difficult Problems

In this section, we group algorithmic problems whose solutions involve a large amount of backtracking and have exponential complexity.

The central part of the section deals with decoding. From the definitions in Sect. 1.2 we see that complete decoding always outputs a codeword. If the transmitted codeword was distorted by many errors, the decoder may not be able to find the codeword, i.e., a decoding error occurs. The probability of this is known to fall exponentially with the length n of the code. This error is inherent to any decoding algorithm. In other words, decoding algorithms are inherently probabilistic.

Incomplete decoding may fail to find a codeword if the received signal is outside its domain. In this case we may pick an arbitrary codeword as the decoding result. This event adds an *algorithmic* error. If the algorithmic error rate does not exceed the inherent error rate, the overall error probability will still have the same behaviour as that of complete decoding. The complexity, however, may be reduced.

Algorithms that we discuss examine a list of plausible candidates for the decoder output. Our sole concern will be that the transmitted codeword appear in this list. If later it fails to be chosen, this is part of the inherent error event rather than a fault of the specific decoder. Note that in practice we often do not need to store the whole list, keeping only the most plausible candidate so far.

3.1 Code construction

For small alphabets, the best known families of codes meet the asymptotic GV bound. Unfortunately, the only known construction "method" is exhaustive search.

Proposition 3.1 *The complexity of constructing an $[n, k]$ linear code meeting the GV bound is $\mathcal{O}(n^3 q^{n-k})$.*

Proof: The construction is accomplished by adding columns to the parity-check matrix so that for every i, $1 \leq i \leq n$, the ith column is outside the linear subspace spanned by any $d - 2$ among the first $i - 1$ columns. ∎

Asymptotically good codes of polynomial construction complexity are discussed in Sect. 2.1. These codes have parameters inferior to the GV bound. Contrary to the decoding algorithms discussed below, no results "in between" (that is, constructions with complexity higher than polynomial but lower than exponential whose parameters are better than the bounds of Theorem 2.14) are known.

of errors on each of these segments is encoded as a message in the part of the codeword on the coordinates within I_1.

Then apply the described procedure recursively to I_1, I_2, and so forth until for some m both encoding and decoding on I_m can be accomplished by exhaustive search in polynomial time.

This solves, in principle, the original problem, but creates another one, namely, of communicating the locations of $I_j, j \geq 1$, to the decoder. This can be done by viewing the codeword output by step $j-1$ as an error vector and constructing a code that corrects *known* errors and localized errors at the same time. An essential property of the required codes is that the decoding into the nearest codeword reconstructs both the message (which in our case carries the location of I_j) and the transmitted codeword. It turns out that there exist codes with these properties and the redundancy essentially the same as in Theorem 2.25. This implies the asymptotic optimality of the resulting code. An actual implementation of this plan requires a long and careful analysis.

Thus, both for defects and localized errors, substantially fewer check bits than for usual errors ensure a unique recovery of the transmitted message. Asymptotically optimal codes can be constructed, encoded, and decoded with simple polynomial algorithms.

2.2.4 Notes

2.2.1. The construction of linear-time codes correcting erasures (Theorem 2.21) is due to Alon et al. [7]. In [7] it is also shown that introducing a small probability of decoding error allows one to reduce the complexity to $\mathcal{O}(n \log(1/\epsilon)/\epsilon)$ and the packet length b to $\mathcal{O}(\log^2(1/\epsilon))$. Codes correcting a linear fraction of erasures with high probability and linear complexity are constructed in Luby et al. [114]. Applications of codes correcting erasures to enhancing the throughput of global networks operating in datagram transmission mode, mentioned in [7], were addressed in Albanese et al. [4], Asmuth and Blakley [15], Kabatianski and Krouk [92], McAuley [120], Rabin [132].

2.2.2. Codes for memories with defective cells were introduced by Kuznetsov and Tsybakov [104]. In the same paper they proved Theorem 2.22. Theorems 2.23 and 2.24 are from Dumer [52].

2.2.3. Studying codes correcting localized errors was suggested by Bassalygo, Gelfand, and Pinsker [25]. A problem complementary to correcting erasures is decoding given the set of error values, i.e., finding error locations. This has been studied by Roth and Seroussi [135]. Linear-time binary codes correcting localized erasures were constructed in [20].

Another well-studied model of transmission assumes that symbols of a codeword are arranged in a matrix and errors are confined to a prescribed number of its rows or columns. Polynomial-time codes for this problem were suggested by Delsarte [49], Gabidulin [68], and Roth [134], see Chapter xx (Blaum, Farrell, van Tilborg). Asymptotically good codes correcting insertions, deletions, and transpositions were constructed in [142].

segment M. However, this segment also contains some defects; therefore, some bits of the segment M are used as parity checks of a code correcting d_M defects. Finally, the number of the segment M, which consists of $\mathcal{O}(\log\log n)$ bits, is transmitted to the decoder using the segment L. Thus, the problem of finding a code of length n that corrects $t = \tau n$ defects is reduced to finding a code of length $n/\log n$ that corrects the same fraction of defects. Now the induction process can be continued until finally we reach the length $\log n$ for which we again use an asymptotically optimal code with encoding/decoding complexity of order poly $\log(n)$. ∎

2.2.3 Codes correcting localized errors

Under this model, the encoder is informed about the positions of possible errors. We discuss only the binary case.

Definition. Let $\boldsymbol{c}$ be a codeword and $E \subset \mathcal{N}$ a subset known to the encoder. Submitting $\boldsymbol{c}$ to the channel with errors localized at E means that the received symbols outside E are correct and symbols within E may contain or not contain errors. The number $t = |E|$ is called the error weight.

This model is halfway between defects and Hamming errors. The maximum number $M(n,t)$ of messages that can be transmitted is determined by the following theorem.

Theorem 2.25 (Bassalygo et al. [25].)

$$\frac{1}{2n}\frac{2^n}{\sum_{i=0}^{t}\binom{n}{i}} \le M(n,t) \le \frac{2^n}{\sum_{i=0}^{t}\binom{n}{i}}.$$

∎

The following theorem is the main result of this part.

Theorem 2.26 (Ahlswede et al. [2].) *For every $\epsilon > 0$ and $0 \le t/n \le 1/2$ there exists a code of length n with $n(H_2(t/n) + \epsilon)$ check bits that corrects t localized errors. The construction, encoding, and decoding of this code can be implemented by sequential algorithms of complexity $\mathcal{O}(n^4)$.* ∎

We shall not attempt to give a proof of this theorem. Some explanations of it may still be in order.

The construction of codes is recurrent. The idea is to successively reduce the encoding on the length n to a number of encodings of shorter codes correcting localized errors until finally both the encoding and decoding for each of them can be done by exhaustive search with polynomial complexity. The decoder of these shorter codes needs to know the number of possible errors on each of these subsets. Therefore, we split $\mathcal{N}$ into consecutive segments of length δn and isolate the one, I_1, with the fewest number of possible errors. Then the set $\mathcal{N} \setminus I_1$ is split into consecutive segments of length $\ell = \log\log n$. The weight

except maybe a small part of these coordinates. The entire $\boldsymbol{c}$ can be found from its last t bits. They are chosen as follows.

Assume that of the t defects s appear in the last t bits of the n-word. Fix the corresponding s bits of $\boldsymbol{c}$ to the values of the defect. The remaining $t-s$ defects are in the check part of $\boldsymbol{c}$. Choose the remaining information bits in $\boldsymbol{c}$ in such a way that these check bits except possibly e of them satisfy condition (b). This is possible because the $(t-s)\times k$ submatrix of A corresponding to these check bits has corank at most e (Lemma 3.6). Thus, the entire information set is determined and we can compute the remaining check bits.

We conclude the first step by the following result.

Theorem 2.23 *For almost all codes C, the encoding described uses at most*

$$r = t + \sqrt{nH_2(t/n)}\log_2(n-t)$$

check bits. The n bits read from the memory ensure a unique recovery of $n-r$ message bits.

Proof: The number of checks equals t plus the redundancy of the e-error-correcting BCH code of length $n-t$. Since writing $\boldsymbol{a}$ into the memory with defect $E_0 \cup E_1$ leaves the last t bits of $\boldsymbol{a}$ intact, they can be used to compute the first $n-t$ checks. By condition (b), this determines the vector $\boldsymbol{b}$ up to at most e wrong bits. Then we can decode $\boldsymbol{b}$ with code B, correcting e errors. ∎

Though these codes have optimal redundancy, their construction is difficult. Therefore, in the second step one applies recursion with respect to the length.

Theorem 2.24 *There exist asymptotically optimal codes with redundancy*

$$t + \epsilon\mathcal{O}\Big(\frac{n\log\log n}{\log n}\Big)$$

that correct $t=\tau n$ defects. They can be constructed with complexity $n^{2/\epsilon}$. The encoding and decoding can be implemented by sequential algorithms with complexity $n(\log_2 n/\epsilon)^3$ and $n\log_2 n/\epsilon$, respectively.

Proof (outline): First, the set $\mathcal{N}$ is split into segments of length roughly $n/\log n$. One of these segments, say M, has not more than

$$d_M = \tau\frac{n}{\log n}$$

defects. On the remaining coordinates we isolate a segment, say L, of length $\mathcal{O}(\log\log n)$ with at most $\mathcal{O}((\log\log n)t/(n-\log n))$ defects. Finally, the remaining coordinates are split into segments ℓ_i of length $\log n$. Suppose d_i is the number of defects on the segment ℓ_i. By Theorem 2.22 we can use this segment to transmit about $(\log n - d_i - o(\log n))$ bits of the message. The decoder will be able to reconstruct them if it knows d_i. Therefore, the numbers d_i have to be communicated to the decoder. This is done by writing them on the

The number $t = |E_0 \cup E_1|$ is called the *weight of defect.* The set of defects may vary for different messages and becomes known to the user before each individual encoding. A code C is said to correct defects of weight t if for any pair (E_0, E_1) of weight t the message $\boldsymbol{m} = (m_1, \ldots, m_k)$ can be encoded with a codeword $\boldsymbol{c}$ such that the vector $\boldsymbol{x}$ read out of the memory allows a unique recovery of $\boldsymbol{m}$.

Let $r(n,t)$ be the minimal redundancy of a length-n-code C correcting t defects,

$$r(n,t) = \min_{C \subset E_2^n} (n - \log_2 |C|).$$

The following theorem gives bounds for codes correcting defects.

Theorem 2.22

$$t \leq r(n,t) \leq t + \left\lceil \log_2 \ln \left(\binom{n}{t} 2^t \right) \right\rceil$$

Proof: Given a set of t defective positions, the actual bits appearing in the memory may take on any of the 2^t values. The encoding mapping, which depends on these values, has to be injective, otherwise a unique decoding would be impossible. This implies the lower bound.

A slightly weaker upper bound will follow from Theorem 2.23 below. ∎

Our goal will be to present asymptotically optimal (up to the main term) codes with simple encoding and decoding. The description consists of two steps. First, we prove that a code of length n correcting t defects can be constructed from a random linear code and a code B correcting relatively few errors.

This step is accomplished in two stages. Let

$$e = \sqrt{\log_2 \binom{n}{t}} < \sqrt{n H_2(t/n)}.$$

We begin with $n - t - e \log_2(n-t)$ message bits and stretch them into $n-t$ bits of encoding. For this we use a binary systematic BCH code B of length $n-t$ correcting e errors. Let $\boldsymbol{b} = (b_1, \ldots, b_{n-t})$ be the resulting codeword.

The codeword written into the memory has the form

$$\boldsymbol{a} = (c_1 + b_1, \ldots, c_{n-t} + b_{n-t}, c_{n-t+1}, \ldots, c_n),$$

where $\boldsymbol{c} = (c_1, \ldots, c_n)$ is a vector in a linear systematic code C with generator matrix $G = [A|I_t]$, where A is a random matrix. Vector $\boldsymbol{c}$ should satisfy the following conditions:

(a), in the message part $(n - t + 1 \leq i \leq n)$,

$$c_i = 0 \text{ if } i \in E_0 \text{ and } c_i = 1 \text{ if } i \in E_1,$$

(b), in the check part $(1 \leq i \leq n-t)$,

$$c_i = b_i \text{ if } i \in E_0 \text{ and } c_i = 1 + b_i \text{ if } i \in E_1,$$

blocks. For this purpose we permute symbols of a codeword in C_3, changing the contents of the blocks in a pseudo-random fashion. More specifically, take a cb-regular graph on n_2/b vertices with the expanding properties similar to those discussed above. We make n_2/b packets from n_2/b blocks as follows. Number the vertices of the graph and let $(i, u_1), \ldots, (i, u_{cb})$ be the edges incident to a vertex i. Then the jth symbol of block i is placed into packet P_{u_j}.

The proof of the following lemma will not be included.

Lemma 2.20 *Every set $I \subseteq \{1, \ldots, n_2/b\}$ with $|I| \geq k(1+\epsilon)/cb$ has the following property:*
For at least a fraction of $(1 - \frac{1}{8}\frac{\alpha^2}{1+4\alpha})$ of $i \in \{1, \ldots, n_2/b\}$, at least b symbols of block i are contained in the set of packets indexed by I.

Both the described mapping and its inverse have complexity $\mathcal{O}(k)$. ∎

By combining these four lemmas, we immediately arrive at the main result of this section.

Theorem 2.21 *For a certain $\epsilon > 0$ and any R, $0 < R < 1/(1+4\alpha)$, there exists an easily constructible $[n, Rn]$ systematic q-ary code, $q = 2^{17}/R\epsilon^4$, such that*
(i), codewords are formed by packets of length $\mathcal{O}(\epsilon^{-4})$ each,
(ii), the message can be recovered from any set of packets containing the total of at least $(1+\epsilon)k$ q-ary symbols, and
(iii), the complexity of both encoding and decoding is $\mathcal{O}(n\epsilon^{-4})$. ∎

2.2.2 Codes for memories with defective cells

In this and the next subsections we shall study transmission models under which the encoder is informed about the actual errors in the channel. The problem is to use this information to reduce the code redundancy.

We discuss only the binary case. Suppose that the n bits of the encoded message are stored in memory with defective cells, which always contain 0 or 1 irrespective of what was written into them.

Definition. Let $\boldsymbol{c} = (c_1, \ldots, c_n)$ be a codeword and (E_0, E_1) a pair of disjoint subsets of $\mathcal{N}$ known to the encoder. The word $\boldsymbol{x}$ read out of the memory has the form $(x_1, \ldots, x_n)$, where

$$x_i = \begin{cases} c_i & i \in \mathcal{N} \setminus (E_0 \cup E_1), \\ 0 & i \in E_0, \\ 1 & i \in E_1. \end{cases}$$

Thus, writing into memory with defect (E_0, E_1) has the effect of changing coordinates of $\boldsymbol{c}$ within $E_0\,(E_1)$ to $0\,(1)$ and leaving the remaining coordinates intact.

the number of edges in F does not exceed

$$k(\gamma^2 + \frac{\lambda}{\ell}(\gamma - \gamma^2)),$$

which should be greater than $\gamma v \frac{\alpha\ell}{2}$. An easy computation shows that with our assumptions on ℓ and λ, this would lead to a contradiction.

The overall decoding complexity is $\mathcal{O}(v\ell^2) = \mathcal{O}(k/\epsilon^2)$. ∎

(b). In this step, we encode the $r = \alpha k$ check symbols ($\alpha k \log_2 q$ check bits) of a codeword of C_1 with a $[4r, r]$ code of Theorem 2.11, which corrects θr erasures, where θ is a certain positive number. This produces a linear-complexity code C_2 correcting a linear fraction of erasures (however, much smaller than the optimum). In the sequel we assume that $\epsilon \le 96\theta$, i.e., $\alpha \le 8\theta$ or

$$k\alpha^2/8 \le \theta r.$$

Lemma 2.18 *By encoding the check symbols of C_1 we obtain a $[n_2 = k(1 + 4\alpha), k]$ q-ary systematic code C_2 that corrects $k\alpha^2/8$ erasures. The encoding and decoding can be implemented with time complexity $\mathcal{O}(k/\epsilon)$ and $\mathcal{O}(k/\epsilon^2)$, respectively.*

Proof: According to the condition before the lemma, The code recovers $k\alpha^2/8$ erasures in the last $4r = 4\alpha k$ symbols. The same number of erasures in the first k symbols can be corrected by the preceding lemma. ∎

(c). This step serves to enhance the erasure recovery of code C_2. Let $b = 4/\alpha^4$. A codeword $\mathbf{c} \in C_2$ is split into n_2/b blocks of b symbols each. Each block is then encoded with a $[cb, b]$ MDS code, $(1/c) = R(1 + 4\alpha)$, whose existence is ensured by our choice of q since

$$cb = \frac{1}{R}\frac{1}{1+4\alpha}\frac{4}{\alpha^4} = \frac{1}{R}\frac{3}{3+\epsilon}\frac{4 \cdot 12^4}{\epsilon^4} < \frac{1}{R\epsilon^4}2^{10}3^4 < q.$$

This gives a codeword in a code C_3 of length $n_3 = cn_2$.

Lemma 2.19 *Suppose a fraction of at least $(1 - \frac{1}{8}\frac{\alpha^2}{1+4\alpha})$ blocks of a codeword in C_3 contains no fewer than b not erased symbols each. Then we can recover the message $(m_1, \ldots, m_k)$ with time complexity $\mathcal{O}(kb)$. The encoding complexity of C_3 also equals $\mathcal{O}(kb)$.*

Proof: At least the claimed fraction of blocks will be recovered due to the properties of the MDS code. Then by the preceding lemmas, we shall be able to correct all possible

$$\frac{1}{8}\frac{\alpha^2 n_2}{1+4\alpha} = \frac{k\alpha^2}{8}$$

erasures. The time of both encoding and decoding using a quadratic-time MDS code is $(n_2/b) \cdot \mathcal{O}(b^2) = \mathcal{O}(kb)$. ∎

(d). In the final step we achieve the goal of the complete message recovery *irrespective* of how the claimed $n - k(1 + \epsilon)$ erasures are distributed among the

erasures, which is the best possible erasure recovery. The major example of MDS codes are the Reed–Solomon codes for which practical decoding methods have complexity of order n^2 (faster methods are competitive only for very large n). The aim of this section is to present a slightly less powerful coding method for very large alphabets with *linear* decoding complexity.

Note that in view of Lemma 3.5, long random $[n, k]$ codes ensure almost optimal erasure recovery. Namely, a codeword can be reconstructed with high probability from any $k(1+o(1))$ of its symbols. However, this requires inversion of $k \times k$ matrices and has complexity greater than order n^2.

Let $\epsilon < 1$ be a fixed constant and suppose $k = nR$. We shall construct q-ary systematic $[n, k]$ codes, $q = 2^{17}/(R\epsilon^4)$, with the property that the codeword is partitioned into packets of a certain length b and the decoder can recover the entire message from any set of packets that in total contain $k(1+\epsilon)$ unerased symbols. The decoding complexity is $\mathcal{O}(n\epsilon^{-4})$. The construction is based on two ingredients, explicit families of expanding graphs (Lubotzky et al. [113], Margulis [118]) and linear-time codes of Theorem 2.11, which also make use of these graphs. It is accomplished in several steps.

(a). Let $\alpha = \epsilon/12$. The goal of this step is to present systematic codes correcting erasures in the message part. Let $m_1, \dots, m_k$ be the message symbols.

Lemma 2.17 *There exists an easily constructible* $[n_1 = k(1+\alpha), k]$ *q-ary systematic code* C_1 *that corrects* $k\alpha^2/8$ *erasures in the message part. The encoding and decoding can be implemented by sequential algorithms with time complexity* $\mathcal{O}(k/\epsilon)$ *and* $\mathcal{O}(k/\epsilon^2)$*, respectively.*

Proof: Fix an integer ℓ, $64/\alpha^2 < \ell \le 128/\alpha^2$, and take an ℓ-regular graph G on $v = 2k/\ell$ vertices whose second largest eigenvalue has absolute value λ. By [113], [118], there exist families of such graphs with $\lambda = 2\sqrt{\ell - 1}$. Message symbols will be identified with the edges of G.

Let $\mathcal{C}$ be a q-ary systematic $[\ell(1+\frac{\alpha}{2}), \ell]$ MDS code[1]. Let $m_{u,1}, \dots, m_{u,\ell}$ be the symbols incident with a vertex u. Encode these symbols with $\mathcal{C}$ using $\ell \cdot \alpha\ell/2 = \mathcal{O}(\ell/\epsilon)$ time units and repeat this independently for all vertices u of G. Any vertex gives rise to $\ell(1+\frac{\alpha}{2})$ symbols, ℓ message symbols and $\frac{1}{2}\alpha\ell$ check symbols. Every message symbol is associated with 2 vertices. Thus, the total number of symbols is $v(\frac{\ell}{2} + \frac{\ell\alpha}{2}) = k(1+\alpha)$, as claimed. The overall encoding time is $\mathcal{O}(v\ell/\epsilon) = \mathcal{O}(k/\epsilon)$. The resulting string forms a codeword of C_1.

Clearly, any $\alpha\ell/2 = \mathsf{dist}(\mathcal{C}) - 1$ erasures in the message part of a codeword of $\mathcal{C}$ can be recovered from the remaining part of the codeword. Therefore, to decode in C_1, we successively inspect vertices of G and, having found a vertex with at most $\alpha\ell/2$ edges incident with it erased, recover them. This is repeated until there are no more vertices with $\le \alpha\ell/2$ adjacent erasures left. If at the end some erasures are not corrected, this means that there is a subgraph F of G with minimum degree greater than $\alpha\ell/2$ all of whose edges correspond to erasures. Let the number of vertices of F be γv; then Lemma 2.8 implies that

[1] Such a code exists since $\ell < q$.

[157]. This book also gives tables of the bounds for $q = 2, 4, 16, 64, 256$, and lists the asymptotic behaviour of these and many other bounds in coding theory for $\delta \to 0$ and $\delta \to 1 - (1/q)$.

Heuristic algorithms of constructing good short codes are addressed in Honkala and Östergard [83].

2.1.3. Theorem 2.15 is a rephrasing of a result by Simonis [145]. Finding a basis formed by minimum-weight codewords seems to be a difficult problem. A related algorithmic problem of finding a basis with a minimal total weight of all vectors in it was studied by Chickering et al. [40], Horton [85]. Theorem 2.16 is due to Gelfand et al. [74]. In an earlier related work [73] Gelfand and Dobrushin proved that there are codes for which the encoding circuit has size $\mathcal{O}(n \log n)$ and depth $\mathcal{O}(\log n)$.

2.2 Other models of noise

We intend to take a brief look at codes correcting other types of errors. Of numerous models of noise we chose to treat codes correcting erasures, localized errors, and codes for defective memory cells. In each case it is assumed that the part of the codeword affected by noise has size that grows *linearly* in n. We give bounds on the size of codes and outline polynomial-time constructions. The construction of erasure-correcting codes requires a large alphabet size. In the last two parts we treat only binary codes.

Note that in principle we are interested in recovering symbols of the message rather than the transmitted codeword. This plays no role for codes correcting Hamming errors but becomes important for codes correcting errors of other types.

The main idea behind the constructions is actually the same. It suggests reducing the encoding on the length n to encoding on a growing number of intervals of smaller length.

2.2.1 Codes correcting erasures

In this section we assume familiarity with regular-graph codes of Section 2.1.1, particularly, Theorem 2.11. The problem is, given a part of the codeword with some symbols missing, to reconstruct the transmitted message. In practical situations, a received symbol is declared erased if the a posteriori probability for all letters of the code alphabet is roughly the same, so no reasonable decision can be taken. Another practical setting for this problem is transmitting the message over a global network dividing it into a number of relatively short packets. Different groups of packets may take different routes and in general the overall delay of the message may be difficult to control. However, if packets correspond to groups of symbols of a codeword, then treating a part of them as erasures enables the receiving party to reduce the average decoding delay.

First, let C be a q-ary $[n, k, d]$ linear MDS code. Then every $k \times k$ submatrix of its generator matrix is nonsingular. This implies that C corrects any $n - k$

Lemma 2.6 is also from [147], though our proof is simpler. As remarked in [147], it is possible to concatenate codes from Theorem 2.10 with short codes meeting the GV bound to obtain linear time decodable asymptotically good codes. This idea is the same as discussed in Theorem 2.14(a) and Remark 2.4(i). For the spectral theory of expanders see Chung [39]. The complexity of encoding for expander codes is studied in [105].

Asymptotically good codes of rate $1/4$ with linear-time encoding and decoding in Theorem 2.11 were constructed by Spielman [150].

Decoding with error-correcting pairs in its general form it has been formulated by Pellikaan [127]. For short cyclic codes error-correcting pairs were found by Duursma and Kötter [56]. They list error-correcting pairs for all cyclic codes of length up to 63. For all but four of these codes the constructed pairs provide decoding up to their *true* minimum distance. Examples 2.2, 2.3 are also borrowed from [56]. This paper also discusses relations to the Berlekamp–Massey algorithm and its generalizations.

2.1.2. Theorem 2.14 is a summary of developments that took place in coding theory during the last 30 years. Forney's discovery of concatenated codes led to the Zyablov bound in part (a). The complexity of constructing codes meeting this bound is not uniform in the parameters of the code (the degree of the polynomial grows as δ approaches $1-(1/q)$). Justesen's codes have uniform construction complexity of order n^2 but are inferior to the Zyablov bound for large values of δ. The last statement of part (a) was proved by Shen [139]. He used the Justesen construction with code A constructed from a Hermitian curve rather than a Reed–Solomon code. This accounts for codes meeting the Zyablov bound for large δ, and the whole result follows by combining this fact and the classical Justesen bound.

Alon et al. in [5] use expanders to improve the Zyablov bound for very low code rates for arbitrary q. For $q = 2$ the improvement holds for $R \le 2 \cdot 10^{-6}$ ($\delta \ge 0.4991$). The complexity of their construction is uniform in δ. Apart from this, the bound by Sugiyama et al. in (2.10) is the best known result for uniformly polynomial nonbinary codes. This has been achieved by replacing the Reed-Solomon code A by a longer code that is inferior to the Singleton bound.

The last two parts of this theorem are related to codes from algebraic curves. A very detailed discussion of these results is carried out in the monograph by Tsfasman and Vlăduţ [157]. In particular, a polynomial-time construction of codes in part (c) (due to Vlăduţ [162]) is given in Chapter 4.3 of that book. The order n^{30} has been recently reduced to n^{17} in Lopéz Jiménez [112]. At the time of writing this there is ongoing research towards reducing this complexity to n^2 or n^3 (see Voss and Høholdt [163]). Part (d) simply uses these codes in the standard concatenated construction (2.7). To compute a lower estimate of this bound, one takes a number of good binary linear codes. This gives a number of points. Gaps between them are filled by shortening or puncturing, The bound in part (d) can be further improved by using multilevel concatenations, but this has never been done because of apparently little interest. The best known bounds for asymptotically good *polynomially decodable* codes are also found in

Theorem 2.15 *If there is an* $[n, k, d]$ *binary linear code* C*, then there also is an* $[n, k, d]$ *binary linear code* C' *that can be encoded with* kd *additions.*

Proof: We shall bound the number of nonzero entries in the generator matrix G of C'. Let $\mathcal{M}$ be a maximal set of independent vectors of weight d in C and suppose $|\mathcal{M}| < k$. Pick a code vector $\boldsymbol{a}$ outside $\mathsf{span}(\mathcal{M})$ of lowest weight, say w, and extend the set $\{\mathcal{M}, \boldsymbol{a}\}$ to a basis $\mathcal{B}$ of C. Now change $\boldsymbol{a}$ into a vector $\boldsymbol{a}'$ of weight d by changing any of its $w - d$ coordinates into zeros. The new set, $\mathcal{B}'$, generates a code C' with distance d since $\mathcal{M}$ is unaltered and the vectors in $C \setminus \mathsf{span}(\mathcal{M})$ have changed in at most $w - d$ coordinates. The dimension of C' is k since otherwise the vector $\boldsymbol{a}'$ would be spanned by $\mathcal{B}' \setminus \boldsymbol{a}$. Since the weight of both $\boldsymbol{a}'$ and $\boldsymbol{a} - \boldsymbol{a}'$ is less than w, they would be spanned by $\mathcal{M}$ and so would be their sum, $\boldsymbol{a}$, a contradiction. Thus, we have constructed a new code with the same parameters, whose maximal set of independent minimal-weight vectors is one greater than that of C. By induction we conclude that there exists a code C' with the same parameters as C generated by vectors of weight d. The number of additions to encode it is not more than $n \times (\#\text{ nonzero entries in a column of } G) = kd$. ∎

Note that this theorem also gives a way of constructing C' given C. A way to do encoding even more economically is related to the fact that when we compute different symbols of the codeword independently, one and the same addition is sometimes performed several times. The following theorem, stated here without proof, guarantees the existence of good codes with a very simple encoding circuit.

Theorem 2.16 *There exist codes meeting the GV bound that can be encoded with a circuit of both size and depth* $\mathcal{O}(n)$*.* ∎

The proof involves constructing a special ensemble of linear codes. Note that averaging over the ensemble of all linear codes gives complexity of order $n^2/\log^2 n$.

2.1.4 Notes

2.1.1. Low-density parity-check codes were introduced by Gallager [69]. He studied the error probability of maximum likelihood decoding and the minimum distance of these codes. In [69] Gallager also suggested a sequential decoding algorithm of l.-d. p.-c. codes but stopped short of estimating the number of errors corrected by it. Zyablov and Pinsker [169] introduced the "iterated majority voting" decoding technique and studied its performance for the random ensemble of l.-d. p.-c. codes. Theorem 2.2 has been proved by Kovalev [99]. The construction of codes from bipartite graphs is due to Tanner [155]. Margulis also used graphs and bipartite graphs with expanding properties to construct codes, see [116] and [117], respectively. However, their analysis relied on the girth (length of the shortest cycle) of graphs, which proves insufficient to secure asymptotically good behaviour.

The analysis of regular graph codes, including Algorithm 2.3 and Theorem 2.10 is by Sipser and Spielman [147]. The bound on the level of expansion in

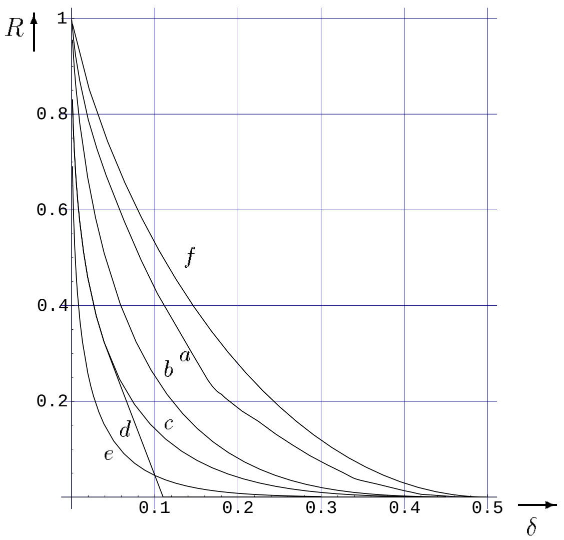


Figure 2.4: Lower bounds for polynomially constructible binary codes: (a) Bound (2.9), (b) R_{BZ}, (c) R_Z, (d) R_J, (e) Bound (2.10); (f) GV bound.

This is better that the Justesen bound (but worse than the Zyablov bound) for large δ. We collect the bounds in Fig. 2.4 ($q = 2$).

(*ii*) For any *fixed* s, there exists a family of codes (concatenations of order s), whose parameters lie above the Zyablov bound but below the bound in part (b) of the theorem. The function in part (b) is called the Blokh–Zyablov bound.

(*iii*) For $q \geq 49$ there exists an interval of values of δ on which the bound in part (c) is better than the GV bound. However, the construction complexity is still very high and the algorithm can be only nominally regarded as easy. The bound in part (d) is better than the Blokh–Zyablov bound for all $0 \leq \delta \leq 1 - (1/q)$ for any $q \geq 2$. ∎

Concatenated codes form a subject of Chapter xx (Dumer). The reader is referred to this chapter for an extensive discussion of this concept as well as for the proof of parts (a) and (b) of the theorem. For codes from algebraic curves see Chapter xx (Hoeholdt et al.).

2.1.3 Encoding complexity

The following simple result bounds the encoding complexity of binary linear codes.

and construction complexity $\mathcal{O}(n^{2s/(1-H_q(\delta))})$.

(c) (codes from algebraic curves) *Let q be an even power of a prime. It is possible to construct a family of codes C with*

$$R = 1 - \delta - (\sqrt{q} - 1)^{-1}.$$

The time complexity of this construction is $\mathcal{O}(n^{30})$.

(d) (concatenations with codes from algebraic curves) *There exist codes C with*

$$R = \max\Big\{(1 - (q^{\ell/2} - 1)^{-1})\frac{\ell}{m} - \frac{\ell}{d}\delta\Big\} \tag{2.9}$$

where the maximum is taken over all q-ary linear $[m, \ell, d]$ codes such that q^ℓ is a square. The construction complexity of the codes is $\mathcal{O}(n^{30})$. ∎

Remark 2.4 (i) In part (a) we have combined two code constructions, which have complexity estimates given as the two cases for $\gamma(u)$. From the statement we see that there exist q-ary linear codes with

$$R_Z = \max_{\delta \le u \le 1-(1/q)} (1 - H_q(u))\Big(1 - \frac{\delta}{u}\Big).$$

This expression is usually called the Zyablov lower bound. The construction complexity of these codes is given as the first case of part (a). A certain dissatisfaction caused by this result is that the construction complexity is very high for large code distance (low rate). The complexity is greater than order n^2 if the maximum in the expression for R_Z is attained for $u \ge H_q^{-1}(2/3)$. For instance, for $q = 2$ this happens for $0.1 \le \delta < 0.5$.

Another version of the concatenated construction, called Justesen's codes, has construction complexity bounded uniformly in δ. The achieved code rate behaves as

$$R_J = \max_{\delta \le u \le \delta_0(1/2)} (1 - H_q(u))\Big(1 - \frac{\delta}{u}\Big).$$

The construction complexity of these codes in $\mathcal{O}(n^2)$. This gives the second condition on $\gamma(u)$ in part (a) of the theorem. This bound is inferior to the Zyablov bound when the maximum in the expression for R_J is attained for $u > \delta_0(1/2)$. For $\delta > \delta_0(1/2)$ the function R_J is negative and hence the Justesen bound is vacuous. The construction complexity of the Justesen codes is less or equal than that of the Zyablov codes. However, for $q = 2$ these complexities are equal for $0 \le \delta < 0.1$ and for $\delta \ge \delta_0(1/2) = 0.11$ Justesen codes do not exist. Moreover, in the interval $0.1 \le \delta \le 0.11$ their parameters are inferior to the Zyablov bound. Hence in the binary case the latter codes are uniformly better.

Both constructions use a Reed–Solomon code as code A in (2.7).

Still another version is due to Sugiyama et al. [154]. This paper gives concatenated codes with uniform construction complexity of order n^2 that meet the bound

$$R = \max_{\delta \le u \le 1-(1/q)} (1 - H_q(u))\Big(1 - \frac{\delta}{u}\Big(1 + \ln\frac{u}{\delta}\Big)\Big). \tag{2.10}$$

it is possible to notably improve the guaranteed error correction of the codes. The major steps that account for the improvement of the bounds, have been
• construction of Reed-Solomon codes with simple decoding;
• concatenations and multilevel concatenations of codes;
• the discovery of asymptotically good codes from algebraic curves;
• the discovery of a simple decoding algorithm for codes from plane curves.

Definition. Let $Q = q^\ell$ and suppose $A : E_Q^k \to E_Q^n$ and $B : E_q^\ell \to E_q^m$ are two linear codes. A *linear concatenated code* C, sometimes denoted $A \boxtimes B$, is defined as the image of the mapping

$$E_Q^k \xrightarrow{A} E_Q^n \hookrightarrow (E_q^\ell)^n \xrightarrow{B} (E_q^m)^n. \tag{2.7}$$

The basic idea is to combine a very good short code B and a long Q-ary code A to come up with a family of asymptotically good codes C. The construction complexity of good long codes over large alphabets is polynomial in their length n. Therefore, if we choose the parameters so that the overall length nm is dominated by the quantity n, the complexity of its construction/decoding will also be polynomial.

Unfortunately, the derivation of the bounds involves many interlaced arguments and the whole picture seems far from being definitive. For referential purposes, we shall quote the bounds in the theorem that follows. The bounds are compared in the remarks after it. Some comments and references appear in the notes following this section. A comprehensive account of these and other results related to multilevel codes is given in Chapter xx (Dumer).

Theorem 2.14 *Let C be a family of q-ary linear codes with parameters $[n, k, d]$ such that $\lim_{n\to\infty} \frac{k}{n} = R$ and $\lim_{n\to\infty} \frac{d}{n} = \delta$.*

(*a*) (concatenated codes) *Let u be such that $\delta \le u \le 1 - \frac{1}{q}$. There exist codes C with*

$$R = (1 - H_q(u))\Big(1 - \frac{\delta}{u}\Big). \tag{2.8}$$

The codes can be constructed by a sequential algorithm with time complexity $\mathcal{O}(n^{\gamma(u)})$, where

$$\gamma(u) = \begin{cases} \max\Big\{\Big(\dfrac{1}{1 - H_q(u)} - 1\Big), 2\Big\}, & \delta \le u \le 1 - \frac{1}{q}; \\ 2, & \delta \le u \le \delta_0(1/2). \end{cases}$$

For $q = 2$ codes meeting the bound (2.8) can be constructed with time complexity $\mathcal{O}(n^2)$ for any u.

(*b*) (multilevel concatenations) *Let $s \to \infty$. There exist codes C with*

$$R_{BZ} = 1 - H_q(\delta) - \delta \int_0^{1-H_q(\delta)} du/\delta_0(u)$$

We have $\operatorname{dim} U = 3$, $\operatorname{dist} U = 5$, $\operatorname{dist} V^{\perp} = 3$; therefore, conditions (2.2)-(2.5) are satisfied. Suppose the received vector is $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} = (1411211)$. Computing the matrix $S(\boldsymbol{y}) = G_V Y G_U^T$, we arrive at the following system of linear equations:

$$\begin{pmatrix} 4 & 0 & 5 \\ 0 & 5 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = 0.$$

Solutions to this system include, for instance, $\boldsymbol{m}_1 = (421)$ and $\boldsymbol{m}_2 = (214)$. They yield two error-locating vectors, $\boldsymbol{u}_1 = \boldsymbol{m}_1 G_U = (4055043)$ and $\boldsymbol{u}_2 = \boldsymbol{m}_2 G_U = (2066025)$. Suppose we want to use $\boldsymbol{u}_1$ to find $\boldsymbol{e}$ and $\boldsymbol{c}$. Denote by a_1, a_2, a_3 the message symbols of $\boldsymbol{c}$, unknown to the decoder. To find them, we form a system $(\boldsymbol{y} - (a_1, a_2, a_3)G_C) * \boldsymbol{u}_1 = 0$. This yields the equations

$$\begin{aligned} a_1 &= 1, \\ a_1 + a_2 + a_3 &= 1, \\ a_1 + 3a_3 &= 1, \end{aligned}$$

whence we finally get $(a_1 a_2 a_3) = (100)$. The corresponding code vector is $\boldsymbol{c} = (1111111)$ and the error vector is $\boldsymbol{e} = (0300100)$. ∎

Example 2.3 Let C be a binary cyclic code of length n not divisible by 3 with locator field $K = \mathbf{F}_2^m$. Suppose C has zeros $a, (1/a)$, where a is a generating element of K. Then the distance of C is known to be 5 (for instance, $[n = 2^{2m-1} - 1, n - 2m, 5]$ Melas codes or $[2^{2m} + 1, n - 4m, 5]$ Zetterberg codes, $m \geq 2$). Let U and V be two cyclic codes with generator matrices

$$G_U = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & a^{-3} & a^{-3 \cdot 2} & \dots & a^{-3(n-1)} \\ 1 & a^3 & a^{3 \cdot 2} & \dots & a^{3(n-1)} \end{pmatrix} \qquad G_V = \begin{pmatrix} 1 & a & a^2 & \dots & a^{n-1} \\ 1 & a^{-1} & a^{-2} & \dots & a^{-(n-1)} \end{pmatrix}.$$

Conditions (2.2)-(2.5) are checked immediately. ∎

Decoding with error-correcting pairs is an outgrowth of decoding algorithms for codes from algebraic curves and the classical BCH decoding. Not every code has an error-correcting pair as is shown by the following theorem, cited without proof.

Theorem 2.13 (Pellikaan [128]) *An $[n, n-4, 5]$ code over $\mathbf{F}_q$ has a 2-error-correcting pair over a finite extension of $\mathbf{F}_q$ if and only if it is an extended generalized Reed–Solomon code.* ∎

2.1.2 Construction complexity

So far we have considered asymptotically good codes with very low construction and decoding complexity. By passing from this to polynomials of larger degrees

The vector $\boldsymbol{u} = \boldsymbol{m}G_U$ *locates the nonzero coordinates of* $\boldsymbol{e}$: $e_i \neq 0$ *implies* $u_i = 0$. *Moreover, the condition* $\boldsymbol{e} * \boldsymbol{u} = 0$ *determines* $\boldsymbol{e}$ *in a unique fashion.*

Proof: Put $\boldsymbol{u} = \boldsymbol{m}G_U$ and rewrite Eq. (2.6) as

$$(\boldsymbol{y}, \boldsymbol{u} * \boldsymbol{v}) \stackrel{(2.2)}{=} (\boldsymbol{e}, \boldsymbol{u} * \boldsymbol{v}) = 0.$$

Hence any vector $\boldsymbol{u}$ with $\boldsymbol{u} * \boldsymbol{e} = 0$ yields a solution to (2.6). By (2.3), there is a nonzero solution $\boldsymbol{u}$. Again rewrite the last equation as $0 = (\boldsymbol{e} * \boldsymbol{u}, \boldsymbol{v})$ implying $\boldsymbol{e} * \boldsymbol{u} \in V^{\perp}$. Then by (2.4), $u_i = 0$ whenever $e_i \neq 0$. This proves the first part of the theorem.

To prove uniqueness, suppose that $\boldsymbol{y} = \boldsymbol{e}_1 + \boldsymbol{c}_1 = \boldsymbol{e}_2 + \boldsymbol{c}_2$ with $\boldsymbol{e}_1 * \boldsymbol{u} = \boldsymbol{e}_2 * \boldsymbol{u} = 0$. This would imply that $\boldsymbol{c}_2 - \boldsymbol{c}_1 = \boldsymbol{e}_1 - \boldsymbol{e}_2 \in C$ and

$$(\boldsymbol{e}_1 - \boldsymbol{e}_2) * \boldsymbol{u} = 0.$$

This implies that $\mathsf{supp}\,(\boldsymbol{e}_1 - \boldsymbol{e}_2) \cap \mathsf{supp}\,(\boldsymbol{u}) = \emptyset$, whence by (2.5) $\boldsymbol{e}_1 - \boldsymbol{e}_2 = 0$. ∎

Observe that conditions (2.2)-(2.4) are sufficient to *locate* the errors with the help of the vector $\boldsymbol{u}$. Therefore, a pair of codes (U, V) satisfying these three conditions is called an *error-locating pair* for the code C. If it also satisfies (2.5), it is called an *error-correcting pair*. Note that codes U and V may be defined over an extension field of $\mathbf{F}_q$.

The actual decoding, i.e., finding the error vector $\boldsymbol{e}$ given $\boldsymbol{y}$ and an error-correcting pair amounts to (1) solving the system of linear equations (2.6) which gives a nonzero locator vector $\boldsymbol{u} = \boldsymbol{m}G_U$, and (2) solving a system

$$\boldsymbol{e} * \boldsymbol{u} = (\boldsymbol{y} - \boldsymbol{c}) * \boldsymbol{u} = 0$$

with respect to the unknown message symbols of the code vector $\boldsymbol{c}$. The complexity obviously does not exceed $\mathcal{O}(n^3)$. If codes U and V are defined over $\mathbf{F}_{q^m}$, the complexity is $\mathcal{O}((nm)^3)$.

The most important application of this technique is decoding cyclic codes up to the true distance.

Example 2.2 Let C be an extended RS code over $\mathbf{F}_7$ with parity-check and generator matrices given by

$$H_C = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 0\,1\,2\,3\,4\,5\,6 \\ 0\,1\,4\,2\,2\,4\,1 \\ 0\,1\,1\,6\,1\,6\,6 \end{pmatrix} \qquad G_C = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 0\,1\,1\,0\,5\,2\,5 \\ 0\,0\,1\,3\,6\,3\,1 \end{pmatrix}$$

and minimum distance $d = 2t + 1 = 5$. Define codes U and V by generator matrices

$$G_U = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 0\,1\,2\,3\,4\,5\,6 \\ 0\,1\,4\,2\,2\,4\,1 \end{pmatrix} \qquad G_V = \begin{pmatrix} 1\,1\,1\,1\,1\,1\,1 \\ 0\,1\,2\,3\,4\,5\,6 \end{pmatrix}.$$

random low-density parity-check codes are known to possess very good error-correcting properties. This belief was further confirmed by experiments carried out in [147], in which codes of length 40000 and rate 1/2 constructed from randomly chosen graphs corrected 1720 random errors with a very high probability. The GV bound for this length guarantees the correction of 2200 errors.

We have seen in the proof of Lemma 2.9 that one decoding round removes a constant fraction of errors. This can be used to construct asymptotically good binary codes with *linear* decoding complexity. The construction is multistage and applies regular-graph codes codes to recursively encode parts of the codeword. To give full details would take us too long. Let us formulate the main result.

Theorem 2.11 (Spielman [150]) *There exists a polynomial-time constructible family of binary codes of rate 1/4 that can correct any $\epsilon < 4 \cdot 10^{-7}$ fraction of errors. Both encoding and decoding of these codes have complexity $\mathcal{O}(n)$. A parallel implementation requires a circuit of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$ for encoding and a circuit of size $\mathcal{O}(n \log n)$ and depth $\mathcal{O}(\log n)$ for decoding.* ∎

In the next section we shall see an application of this result to the construction of low-complexity codes correcting erasures.

Error-correcting pairs

We end this part by discussing a general decoding technique applicable to any $[n, k, d]$ linear code C. We are interested in the bounded distance decoding of up to $t = \lfloor (d-1)/2 \rfloor$ errors. The decoding algorithm of Theorem 2.12 has complexity of order $\mathcal{O}(n^3)$. However, to build the algorithm one needs to perform much preprocessing work and though this technique is shown to be applicable to many short codes, given an arbitrary linear code, it is not at all clear whether it has an error-correcting pair..

For any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ let $\boldsymbol{u} * \boldsymbol{v} = (u_1 v_1, \dots, u_n v_n)$ be their componentwise product. Suppose U and V are two linear codes that satisfy the following conditions

$$\begin{aligned} U * V &\subseteq C^{\perp} && (2.2)\\ \mathsf{dim}\, U &> t && (2.3)\\ \mathsf{dist}\, V^{\perp} &> t && (2.4)\\ \mathsf{dist}\, C + \mathsf{dist}\, U &> n. && (2.5) \end{aligned}$$

Let $\boldsymbol{y}$ be at a distance at most t from C, say $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, $\mathsf{wt}\,(\boldsymbol{e}) \le t$. Form an $n \times n$ diagonal matrix $Y = [y_i \delta_{ij}]$.

Theorem 2.12 *Let G_U and G_V be generator matrices for U and V. There exists a vector $\boldsymbol{m} \in E_q^{\mathsf{dim}\, U}$, $\boldsymbol{m} \ne 0$, such that*

$$(G_V Y G_U^T) \boldsymbol{m}^T = 0. \tag{2.6}$$

Let us compute the parameters of the code C. As remarked above, its rate is at least $R = 2r - 1 = 1 - 2H_2(\beta)$. The distance of C is easily estimated from Remark 2.3(ii). Indeed, take a set M of αn 2-regular vertices in V_2. The set L of its neighbors in V_1 has size at least γv; hence the average number of neighbors of a vertex in L is no fewer than $2\alpha n/\gamma v$. Recalling that $n = \ell v/2$, we obtain

$$\frac{2\alpha n}{\gamma v} = \frac{\ell v(\gamma + (\lambda/\ell)(\gamma - \gamma^2))}{\gamma v} = \ell\Big(\gamma + \frac{\lambda}{\ell}(1-\gamma)\Big).$$

This number cannot be smaller than the minimum distance of $\mathcal{C}$. Thus if γ is such that

$$(\gamma + (\lambda/\ell)(1-\gamma)) < \beta,$$

i.e., $\gamma < (\beta - (\lambda/\ell))/(1-(\lambda/\ell))$, the code C does not contain a vector of weight αn. Since $\alpha > \gamma^2$, we conclude that

$$\delta > \left(\frac{\beta - \frac{\lambda}{\ell}}{1 - \frac{\lambda}{\ell}}\right)^2,$$

i.e., δ is close to β^2. This gives for the parameters (R,δ) of C_n the bound $R = 1 - 2H_2(\sqrt{\delta})$ and the estimate $n\delta/48$ for the error-correction capacity of C_n. ∎

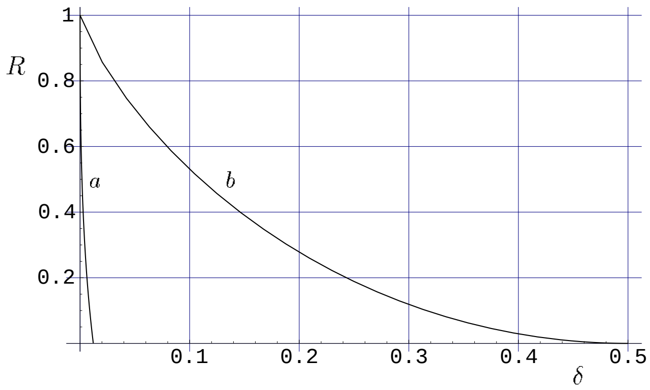


Figure 2.3: (a) Bound for asymptotically good codes from Theorem 2.10, (b) GV bound.

Thus, for codes C_n we have $R \geq 0$ for $0 \leq \delta \leq 0.0121$. The bound from this theorem is shown in Fig. 2.3 together with the GV bound.

Note that the guaranteed error correction of codes in Theorem 2.10 is much smaller than that of random codes in Theorem 2.3. However, generally, long

decoding round. By Remark 2.3(i), the number of such errors (vertices) is not more than

$$n\left(\left(\frac{4\tau}{\beta}\right)^2+\left(\frac{4\tau}{\beta}\right)\frac{\lambda}{\ell}\right).$$

A decoding round adds errors if there exists a subset X_i with $|X_i \cap E| \geq \dfrac{3}{4}\beta\ell$. This again happens for at most $2\tau n/(\frac{3}{4}d)$ subsets. Each of them is responsible for not more than $d/4$ false alarms, which makes a total of at most $2n\tau/3$ miscorrections. Thus after one decoding round the number of remaining errors is not greater than $n\left(\frac{2}{3}\tau+(\frac{4\tau}{\beta})^2+\frac{4\tau}{\beta}\frac{\lambda}{\ell}\right)$. Let us compute the difference between the number of errors before and after one decoding round:

$$n\left(\tau-\frac{2}{3}\tau-\left(\frac{4\tau}{\beta}\right)^2-\left(\frac{4\tau}{\beta}\right)\frac{\lambda}{\ell}\right)=n\tau\left[\left(\frac{1}{3}-\frac{4}{\beta}\right)\frac{\lambda}{\ell}-\frac{16\tau}{\beta^2}\right].$$

If τ satisfies the assumption of the lemma, this is positive, which proves our claim. ∎

To construct an infinite family of good codes we need an explicit family of graphs G_v for which the quotient λ/ℓ vanishes as v grows. Such families were discovered in Lubotzky et al. [113], Margulis [118].

Theorem 2.10 *There exists an easily constructible family of binary linear codes C_n of length n, of minimum distance arbitrarily close to $n\delta$ and rate $R = 1 - 2H_2(\sqrt{\delta})$ for which Algorithm 2.3 corrects up to $n\delta/48$ errors in $\mathcal{O}(\log n)$ rounds. The decoding circuit has depth $\mathcal{O}(\log n)$ and size $\mathcal{O}(n \log n)$. The time complexity of its sequential implementation is $\mathcal{O}(n \log n)$.*

Proof: Let C_n be a regular-graph code $C(G, \mathcal{C})$, where G is an ℓ-regular graph with n edges and $\mathcal{C}$ an $[\ell, r\ell, \beta\ell]$ binary linear code. Suppose we want to correct a $\tau = \frac{\beta^2}{48} - \epsilon$ fraction of errors. Let us show that we can choose G and $\mathcal{C}$ so that this is possible and compute the parameters of the code C_n.

Choose ℓ so that $\frac{\beta}{4}\frac{\lambda}{\ell} < \epsilon$. This is possible since in the construction of expander graphs that we are going to use, λ is known to behave as $2\sqrt{\ell}$. From the previous lemma we know that one parallel decoding round applied on the code C_n removes a constant fraction of errors, and thus, $\mathcal{O}(\log n)$ rounds are sufficient to correct any error of weight τn. Since ℓ is a constant, we can allow ourselves to choose a code $\mathcal{C}$ of length ℓ whose rate r and relative distance β satisfy the GV bound $r = 1 - H_2(\beta)$. The complexity of its decoding does not grow with n. Thus the decoding circuit has depth $\mathcal{O}(\log n)$ and size $\mathcal{O}(n \log n)$. The complexity of a sequential implementation of the algorithm equals the complexity of each round times the number of rounds, i.e., $\mathcal{O}(n \log n)$.

Note that by our choice of the code $\mathcal{C}$, its relative distance β is a much larger fraction than $\lambda/\ell \approx 2/\sqrt{\ell}$. Therefore, the condition of Lemma 2.9 is not vacuous.

Lemma 2.8 (Alon–Chung [6].) *Suppose all eigenvalues of G other than ℓ have absolute values at most λ. Then the number of edges in a subgraph of G induced by a subset F of vertices of size γv is at most αn, $\alpha = \gamma^2 + \frac{\lambda}{\ell}(\gamma - \gamma^2)$.* ∎

Remark 2.3 (i) The degree of every vertex in V_2 is 2. Therefore, this lemma gives a bound on the number of vertices in V_2 for which both edges incident with them have their other ends in F.

(ii) This lemma also implies that any subset of αn 2-regular vertices in V_2 has at least γv neighbors in V_1, i.e., that Γ is an $(2, \ell, \alpha, \gamma v/\alpha n)$-expander. Indeed, suppose a set M of αn 2-regular vertices has few, say ϵv, neighbors, $\epsilon < \gamma$. Denote the set of these neighbors by L. Take any vertex in V_2 outside M, add it to M and add its two neighbors to L. Continue in this manner until the size of L achieves γn. Then the number of neighbors of L connected to L by both edges violates the bound in Lemma 2.8. ∎

In the following lemma we prove that if the number of errors is relatively small, one decoding round of Algorithm 2.3 reduces the weight of error in the received vector by a constant factor. The proof goes along the lines of the proof of Theorem 2.4.

Lemma 2.9 *Let G be an ℓ-regular graph on v vertices such that all its eigenvalues other than ℓ have absolute values at most λ, Γ its edge-vertex incidence graph, and $\mathcal{C}$ a linear $[\ell, m, d = \beta\ell]$ code. Suppose $\boldsymbol{y}$ is a received vector, $\boldsymbol{c} \in C(G, \mathcal{C})$ is the code vector closest to $\boldsymbol{y}$ and $\mathsf{dist}\,(\boldsymbol{c}, \boldsymbol{y}) \le \tau n$, where*

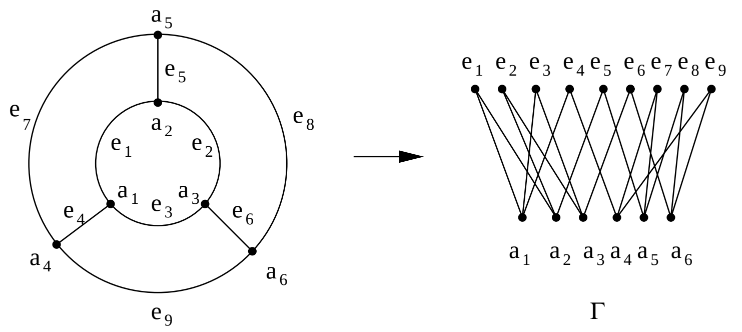
$$\tau < \frac{\beta^2}{48} - \frac{\beta}{4}\frac{\lambda}{\ell}.$$

Then after one parallel decoding round the distance $\mathsf{dist}\,(\boldsymbol{c}, \boldsymbol{y})$ is reduced by a constant factor.

Proof: Let $\boldsymbol{e}$ be the error vector of weight τn and let $\mathcal{E} = \mathsf{supp}\,(\boldsymbol{e})$, $\mathcal{E} \subset V_2$. Fix a coordinate $e \in \mathcal{E}$ and let e_i be the part of the ith column of H contained in the rows from the subset X_i. If this part is not all-zero, we say that e *intersects* X_i and that X_i and $\mathcal{E}$ have a common coordinate. Any coordinate $e \in \mathcal{E}$ intersects two subsets, X_1 and X_2 say. A decoding round removes some of the errors, failing to notice the remaining ones, and adds some errors. It will fail to correct an error e if both X_1 and X_2 have more than $d/4$ common coordinates with $\mathcal{E}$. The inequality $|X_i \cap \mathcal{E}| \ge d/4$ will hold for at most

$$\frac{2|\mathcal{E}|}{\frac{1}{4}d} = \frac{4\tau}{\beta}v$$

subsets X_i. This collection of subsets corresponds to a group F of ℓ-regular vertices in V_1. Every 2-regular vertex in $\mathcal{E}$ for which both edges incident with it have their other ends in F, corresponds to an error not corrected after one



G

$$
\begin{array}{c|ccccccccc}
 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \\
a_1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
a_3 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
a_4 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
a_5 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
a_6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad
H = \left[\begin{array}{ccccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \hline
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{array}\right]
$$

Figure 2.2: Codes from regular graphs

that this definition also gives us a way to construct codes from arbitrary regular bipartite graphs. For instance, if Γ is an (ℓ, h)-regular bipartite graph with the parts of size v and $n = (\ell/h)v$, then the corresponding code has length n and rate at least $h\frac{m}{\ell} - (h-1)$.

Knowing the eigenvalues of G one can estimate the expansion level of Γ. Combining this with the fact that every error affects only two blocks of $\ell - m$ checks in H, it is not difficult to estimate the error correction of $C(G, \mathcal{C})$. In total, there are v blocks of checks, each involving ℓ coordinates of the received vector. Denote these subsets of coordinates by X_i, $1 \le i \le v$.

Example 2.1 Let G be a 3-regular graph on $v = 6$ vertices and $\mathcal{C}$ a $[3, 1, 3]$ code with the parity-check matrix $\mathcal{H} = \begin{bmatrix} 1\,1\,0 \\ 1\,0\,1 \end{bmatrix}$. Figure 2.2 shows the construction of the parity-check matrix H of $C(G, \mathcal{C})$ from the incidence matrix of G. The six sets X_i are formed by subsets of edges incident with a_i in G: $X_1 = \{1, 3, 4\}, X_2 = \{1, 2, 5\}$, etc. Thus, a subset X_i is determined by vertex a_i. ∎

Let G be an ℓ-regular graph on v vertices and Γ its edge-vertex incidence graph. Γ has the vertex set $V_1 \cup V_2$, $|V_1| = v$, $|V_2| = n = v\ell/2$.

Proof: Let $U \subseteq V_2$ be a subset of vertices of size αn. The probability that a vertex in V_1 is connected to a given subset $U \subseteq V_2$ of size αn is at least $1-(1-\alpha)^\ell$. ∎

To present an explicit family, let us modify the code construction. The idea of this modification is as follows. In low-density parity-check codes we wanted that each group of ℓ coordinates corresponding to ones in check equations satisfy an overall parity check. In the following definition we require that each group of ℓ coordinates satisfies check equations of a certain binary code.

Definition. (*Regular-graph codes*) Let $\mathcal{H}$ be a parity-check matrix of a binary $\mathcal{C}[\ell, m, \beta\ell]$ code and A the $v \times n$ vertices vs. edges incidence matrix of an ℓ-regular graph G with v vertices and n edges. The parity-check matrix H of the regular-graph code $C(G, \mathcal{C})$ is obtained by replacing the ith row in A by its $\ell - m$ copies and then replacing the ℓ all-one columns in these $\ell - m$ rows by ℓ columns of $\mathcal{H}$, $1 \leq i \leq v$.

The code $C(G, \mathcal{C})$ has length n and rate at least $2\frac{m}{\ell} - 1$. If $\mathcal{C}$ has parameters $[\ell, \ell-1, 2]$, then these codes form a subclass of low-density parity-check codes with $h = 2$.

Regular-graph codes can be decoded by the following algorithm, which will be shown to correct a linear fraction of errors.

Algorithm 2.3: Parallel decoding

- In parallel, for each of the v subsets, if the current setting of coordinates is within distance $\beta\ell/4$ of a codeword of $\mathcal{C}$, mark all the coordinates that should be flipped to obtain this codeword. Invert the marked coordinates.
- Repeat $\mathcal{O}(\log n)$ times.

We remark that the qualitative properties of the algorithm do not change if we replace $1/4$ with another fraction of $\beta\ell$.

Our plan is to begin with explicit constructions of regular graphs that have the expanding property, meaning that every subset of vertices of a given size has "many" neighbors, i.e., vertices outside this set adjacent to a vertex in it. One practical way to prove that a graph has the expanding property is to bound from above the number of edges with both ends *inside* any given subset of vertices of fixed size. This gives a lower bound on the number of neighbors in this subset.

Instead of working directly with the incidence matrix A of a given regular graph $G(V, E)$, we shall relate a bipartite graph Γ to it by associating V and E with the two parts of Γ.

Definition. Let $G(V, E)$ be a graph. Its *edge-vertex incidence graph* Γ is a bipartite graph with the vertex set $V \cup E$. Two vertices $u \in V$ and $e \in E$ are connected if u is incident with e in G.

The relevance of this definition will follow from Remark 2.3(ii) below. Note

A to be the parity-check matrix of a code, which in this case is denoted $C(G)$. The rate of $C(G)$ is $R \geq 1 - h/\ell$.

Lemma 2.6 *Suppose every group of $e \leq a = \alpha n$ h-regular vertices has at least $(\frac{3}{4} + \epsilon)he$ neighbors, $\epsilon > 0$. Then Algorithm 2.1 applied to $C(G)$ corrects $\alpha n/2$ errors.*

The key property of G is that any not too large subset of h-regular vertices has not too few neighbors. Generally, a bipartite graph is called an $(\ell, h, \alpha, \gamma)$-*expander* if every set $U \subset V_1$ of at most an α fraction of h-regular vertices has at least $\gamma|U|$ neighbors:

$$|U| \leq \alpha n \quad \Rightarrow \quad \left|\{u' \in V_2 \,:\, \exists_{u \in U} \text{ such that } (u, u') \in E\}\right| \geq \gamma|U|.$$

Proof of Lemma 2.6: We shall prove that choosing G with the claimed expansion ensures $e < e_0$. Indeed, the total number of checks containing errors equals the weight $\mathsf{wt}\,(\boldsymbol{s})$ of the syndrome plus the number, x, of satisfied checks that contain errors. By the assumption of the lemma,

$$\mathsf{wt}\,(\boldsymbol{s}) + x > (3/4)he.$$

Every unsatisfied check contains at least one error. Every satisfied check with an error connects at least two errors. Hence

$$\mathsf{wt}\,(\boldsymbol{s}) + 2x \leq he.$$

These two inequalities imply $\mathsf{wt}\,(\boldsymbol{s}) > eh/2$. Since this argument holds for any error vector of weight up to e, we conclude that $e < e_0$ by the definition of e_0. ∎

Note that this lemma corresponds to simply choosing the graph so that $w_*(e)$ is (almost) equal to $eh/2$. Unfortunately, the level of expansion required in it is by far greater than that of the known explicit constructions, though on the average random bipartite graphs probably have a good expansion. To choose a random (ℓ, h)-regular graph one can choose a random matching between two sets of nh vertices and then glue together consecutive sets of h vertices in one set to get n h-regular vertices and glue together consecutive sets of ℓ vertices in the other set to get nh/ℓ ℓ-regular vertices. Occasional multiple edges are discarded. Averaging over this ensemble, we get the following theorem.

Proposition 2.7 *Let $G = (V_1 \cup V_2, E)$ be a randomly chosen (ℓ, h)-regular graph with $|V_1| = nh/\ell$ and $|V_2| = n$. Then for all $0 < \alpha < 1$, a set of αn vertices in V_2 will, on the average, have at least*

$$n\frac{h}{\ell}(1 - (1 - \alpha)^{\ell})$$

neighbors.

A coordinate, say i, appears in h checks. Every such check involves $\ell-1$ other coordinates. Let us call these coordinates *connected* with the ith coordinate. Flipping the ith coordinate changes the value of h checks and affects at most $h(\ell-1)$ other coordinates connected with it. Suppose that before this flipping they all were contained in more than $h/2$ unsatisfied checks and should have been inverted during the current decoding round. Suppose also that inverting the ith coordinate changes their status so that inverting them does not reduce the weight of the syndrome, and the algorithm leaves them unchanged. Thus, if we mark all the coordinates that need inverting during the current round, a single flipping can remove at most $h(\ell-1)+1$ marks. Therefore, the number of coordinates inverted during one decoding round is no fewer than $c/(h(\ell-1)+1)$. Inverting one coordinate reduces the weight of the syndrome by at least one. Thus, the weight of the syndrome after this round is not greater than

$$s - \frac{c}{h(\ell-1)+1} \leq s\Big(1 - \frac{2s-eh}{sh(h(\ell-1)+1)}\Big).$$

The amount in the parentheses is less than 1, and we are done. ∎

Remark 2.2 Whereas the lower bound on the weight of correctable errors in Theorem 2.3 is proved only for codes from Gallager's ensemble, the decoding complexity estimate is valid for *any* low-density code.

The iterated majority voting algorithm has the following parallel analog.

Algorithm 2.2: Parallel iterated majority voting

- In parallel, mark all coordinates contained in more than $h/2$ unsatisfied checks and invert them.
- Repeat until no coordinates can be marked.

Theorem 2.5 *Parallel iterated majority voting corrects a linear fraction of errors. It can be implemented with a Boolean circuit of size* $\mathcal{O}(n \log n)$ *and depth* $\mathcal{O}(\log n)$.

To prove this is easy and we shall not stop here to do so.

Powerful results from algebraic graph theory enable us to present an explicit, i.e, easily constructible family of low-density codes for which the value e_0 is also linear in n. Let us formulate the condition $e < e_0$ in terms of a bipartite graph associated with the code. Let $G = (V_1 \cup V_2, E)$ be such a graph. G is called (ℓ, h)-regular if the degree of every vertex in V_1 is ℓ and the degree of every vertex in V_2 is h, $h < \ell$. Let $|V_2| = n$, then by counting edges we see that the number of ℓ-regular vertices is $|V_1| = nh/\ell$. Form the $(h/\ell)n \times n$ "adjacency" matrix A of G with rows numbered by vertices from V_1 and columns with vertices from V_2. Let $u_i \in V_1$ and $u'_j \in V_2$, then $a_{ij} = 1$ if $(u_i, u'_j) \in E$ and 0 otherwise. Take

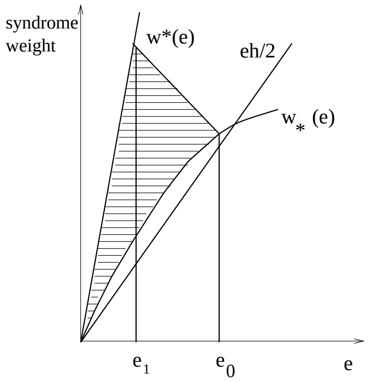


Figure 2.1: Visualizing the iterated majority voting

randomly its columns. Repeat this independently h times to form an $(mh \times m\ell)$ parity-check matrix of the low-density code. As $m \to \infty$, this procedure defines an ensemble of low-density parity-check codes of growing length.

Suppose a code is chosen randomly with uniform distribution from this ensemble. The following theorem characterizes the typical performance of the algorithm and relates the fraction of correctable errors to the Gilbert-Varshamov distance $\delta_0(R)$.

Theorem 2.3 (Zyablov–Pinsker [169].) *For almost all codes in Gallager's ensemble of low-density codes, the value e_0/n is strictly positive. If $R \to 1$, the fraction of errors corrected by the iterated majority voting algorithm is not less than $\delta_0(R)/22$.* ∎

The proof is difficult and will be omitted. Although the decoding of codes is easy, their construction would involve a large search. Theorem 2.10 provides easily constructible codes with a similar error protection and in this sense supersedes this result.

Let us estimate the complexity of the algorithm assuming that ℓ and h are constants.

Theorem 2.4 *The iterated majority voting algorithm has complexity $\mathcal{O}(n \log n)$.*

Proof: Every check involves only ℓ coordinates, therefore, computing it has complexity $\mathcal{O}(1)$. Then clearly every decoding round can be accomplished in $\mathcal{O}(n)$ time units. The proof will be complete if we show that every round reduces the syndrome weight by a finite fraction.

Let e be the number of errors and suppose that c of them are contained in more than $h/2$ unsatisfied checks. Let $s = \mathsf{wt}(\boldsymbol{s})$, then $s \le ch + (e-c)h/2$ and so,

$$c \ge \frac{2s - eh}{h}.$$

Let us give a lower bound to the number of errors corrected by the algorithm. Suppose $\boldsymbol{e}$ is an error vector of weight e and let $\boldsymbol{s} = H\boldsymbol{e}^T$ be its syndrome. We assume that there exists a function $w_*(e)$ bounding the weight of the syndrome from below that depends only on the weight of $\boldsymbol{e}$. The existence of this function is a crucial ingredient of the proof. Thus,

$$w_*(e) \le \mathsf{wt}\,(\boldsymbol{s}) \le w^*(e) := eh. \tag{2.1}$$

Lemma 2.1 *If* $\mathsf{wt}\,(\boldsymbol{s}) > eh/2$*, there exists a coordinate,* i*, such that* $\mathsf{wt}\,(\boldsymbol{s} + \boldsymbol{h}_i) < \mathsf{wt}\,(\boldsymbol{s})$*,* $\boldsymbol{h}_i$ *being the ith column of* H*.*

Proof: The average number of unsatisfied checks per coordinate is $\mathsf{wt}\,(\boldsymbol{s})/e > h/2$. ∎

The assumption of the lemma will be satisfied if $w_*(e) > eh/2$. This inequality is valid for $e = 1$. Hence, there exists a nonempty region $0 < e \le e_0$ where it holds true for any error vector of weight e. Furthermore, suppose e_0 is the maximal number with this property. Thus, there may be errors of weight $> e_0$ that are not corrected by the algorithm.

For any error vector of weight $e \le e_0$, there exists a coordinate that is contained in more than $h/2$ unsatisfied checks. Inverting it, we reduce the weight of the syndrome and either add or remove an error. In the worst case, the number of errors becomes one greater and the weight of syndrome one less. To visualize the decoding process, let us represent it as a trajectory in the "phase plane" showing the weight of syndrome vs the multiplicity of errors (see Fig. 2.1). Starting from an (integer) point in the plane, in the worst case we advance to its south-east neighbor. Thus, for successful decoding, it is sufficient that moving along the worst-case trajectory, we do not leave the region $\mathsf{wt}\,(\boldsymbol{e}) < e_0$. We conclude that for the algorithm to converge, it is sufficient that the initial point is below the line drawn from the point $(e_0, e_0h/2)$ with slope -1. Denote by e_1 the value of e for which this line and the upper bound $w^*(e)$ intersect. This is a lower bound on the number of correctable errors.

Theorem 2.2 *For any error vector* $\boldsymbol{e}$ *of weight*

$$\mathsf{wt}\,(\boldsymbol{e}) < \left\lfloor \frac{e_0}{2}\frac{h+2}{h+1} \right\rfloor,$$

the algorithm performs a successful decoding.

Proof: Any decoding trajectory that originates in the stroked region in Fig. 2.1 cannot leave this region throughout the decoding. Indeed, by the above argument, the trajectory cannot cross the straight line connecting points e_0 and e_1 and inequalities (2.1) bound it from above and below. ∎

Remark 2.1 In particular, if $e < e_0$, the algorithm corrects any $e/2$ errors.

Gallager's ensemble of low-density parity-check codes is specified as follows. Take ℓ copies of the identity matrix I_m and form an $(m \times m\ell)$ matrix. Permute

2 Easy Problems

This section is devoted to problems of constructing asymptotically good codes and their decoding that have polynomial complexity in the length n of the code. For a problem to be really easy, i.e., practical, we want the degree of this polynomial to be small, for instance, 2 or 3. A part of the best known results for error correcting codes is achieved using concatenations of good codes of smaller length. In this case we refer to Chapter xx (Dumer) for the proofs.

The main results of Section 2.1 are Theorems 2.3, 2.4, 2.10, and 2.11. The accent here is on decoding complexity. Theorem 2.14 outlines the best known results with respect to construction complexity. Section 2.2 consists of 3 parts, each discussing one construction.

2.1 Error-correcting codes

2.1.1 Decoding

We begin with asymptotically good *binary* codes with a very low complexity of decoding. The idea is to assume that every row of the parity-check matrix H contains ℓ ones and every column h ones, $\ell > h$. If n grows and ℓ and h are small, then every check (row) connects a vanishing fraction of coordinates and every coordinate appears in a vanishing fraction of checks. For this reason, such codes are called *low-density parity-check codes.* This property allows one to bound the number of computations of their decoding. Gallager [69] proved that a code from a random ensemble of low-density codes probably comes close to the asymptotic Gilbert–Varshamov bound. Unfortunately, due to the lack of structure, a simple decoding realizing this distance seems difficult to achieve. However, it is possible to decode a received word by successively flipping coordinates if doing so reduces the weight of the syndrome. More specifically, we propose to inspect coordinates from left to right and invert the first coordinate that appears in more than $h/2$ unsatisfied checks. Then we recompute the part of the syndrome affected by this inversion and continue the inspection. We call one complete pass a decoding round. Our goal will be to prove that after $\log n$ rounds we can correct a linear fraction of errors. Though this fraction is much smaller than the distance of the code, it still provides an algorithmic lower estimate of the distance and ensures asymptotically good behavior.

Let us formulate the decoding algorithm.

Algorithm 2.1: Iterated majority voting

- Find a coordinate contained in more unsatisfied than satisfied checks. Invert it and re-compute the syndrome.
- Repeat until no such coordinates are found or stop after $\mathcal{O}(\log n)$ decoding rounds.

1.4 Notes

1.2.1. An expanded informal discussion of models and complexity classes for an expert is given in the first two chapters of the Handbook of Theoretical Computer Science, Van Emde Boas [58] and Johnson [89]. Books by Cutland [46] and Savage [137] provide an easy introduction to computability. The logical foundations are treated in Manin [115], Rogers [133]. Uspensky and Semenov in [158] give a detailed overview of the literature. The monograph by Wagner and Wechsung [164] is an up-to-date mathematical treatise of the subject. The book by Papadimitriou [125] reflects a computer scientist's viewpoint and discusses some fundamental recent discoveries briefly mentioned in Sections 4 and concluding remarks.

1.2.2. Inequalities cited in this subsection form a particular case of estimates of large deviations for the binomial distribution (see Gallager [69]). A more recent reference is Alon and Spencer [8] which also contains many other similar results.

1.2.3. The following books and papers treat systematically general algorithmic problems for codes: Barthélemy et al. [24], Bassalygo et al. [26], Savage [136].

We study Problems 1–4 and 6. Computing automorphism groups of codes has been studied by Leon [107]. Leon's algorithm has been implemented in the computer algebra package GUAVA, see Simonis [146]. This package also solves a number of algorithmic problems for codes of small and moderate length such as computing the minimum distance, weight distribution, covering radius, and so on, and is useful for constructing examples of codes. GUAVA is available from `ftp.math.rwth.aachen` as a part of the GAP package.

Some other algorithmic problems that we have chosen not to include deal with the construction of combinatorial arrays closely related to coding theory, see Cohen et al. [44], Füredi [67], Gordon et al. [78],[77], Grable and Phelps [79]. For the construction of covering codes see Chapter xx (Pless, Litsyn, Brualdi) of this volume. Complexity of computations in finite fields is covered in Jungnickel [90], Shparlinski [141].

1.3. The general technique of this section is due to Gallager [69]. More refined results on the average behavior of the weight spectra of codes were proved by Blinovskii [30]-[32], Delsarte [48], Loeliger [111], Sidelnikov [143]. Higher moments of the number of codewords in a sphere are estimated in Barg and Dumer [23] as follows:

$$\mathsf{E}(U_w)^m \le (\mathsf{E}U_w)^m(1+q^m/\mathsf{E}U_w)^m \tag{1.10}$$

provided that $\mathsf{E}(U_w) > q^m$. Blinovskii [31] studied *lower* estimates on the number of code vectors of a random code in a sphere (or any set of a given size).

Note that unrestricted codes on the average are worse than linear codes. Namely, on the average the parameters (R, δ) of a randomly chosen code with q^{Rn} vectors approach the bound $2R = 1 - H_q(\delta)$.

Note that if $w/n < \delta_0(R)$, the exponent is negative. Therefore, the probability that C contains a codeword of weight w, $1 \le w \le (\delta_0(R) - \epsilon)n$, declines exponentially in n for any fixed $\epsilon > 0$. Thus, if we consider the ensemble of random linear codes of *growing* length and fixed rate R, then for any $\epsilon > 0$ the fraction of codes with relative distance

$$\delta \le \delta_0(R) - \epsilon$$

approaches zero. We have proved the following.

Lemma 1.2 *Almost all linear codes meet the asymptotic GV bound.*

This is the reason that complexity estimates of different algorithms are usually compared for codes meeting the GV bound. For small q this is also the best choice of codes available. However, generally dealing with codes meeting the GV bound is computationally difficult. Often it is possible to reduce the complexity of algorithms by choosing codes with somewhat inferior properties, but still asymptotically good.

Definition. Suppose a family of $[n, k_n, d_n]$ codes of growing length has the limit parameters $R = \lim\limits_{n\to\infty} k_n/n$ and $\delta = \lim\limits_{n\to\infty} d_n/n$. Then the codes are called *asymptotically good* if $R\delta > 0$.

Since the variance (1.8) of the number of codewords is small, (1.5) implies the following estimate:

$$\mathsf{Pr}\{|N_w - \mathsf{E}N_w| \ge \sqrt{n\,\mathsf{E}N_w}\} \le \frac{q-1}{n}.$$

Thus, for all linear codes except for a small fraction, the number of codewords of a given weight cannot be too small. This is a heuristic reason that most algorithms for typical codes tend to have a large complexity. Replacing $\sqrt{n\,\mathsf{E}N_w}$ by $\mathsf{E}N_w$, we observe that the fraction of codes with $N_w \ge 2\mathsf{E}N_w$ falls exponentially in n.

Let $\mathcal{S}$ be the interior of the sphere of radius w in E_q^n, then $L = \sum_{i=0}^{w} \binom{n}{w}(q-1)^w$. Let $U_w := N(\mathcal{S})$. By Lemma 1.1 and (1.3), the average number of codewords in $\mathcal{S}$ for long codes is

$$\mathsf{E}U_w = q^{n[H_q(w/n)-(1-R)](1+o(1))}. \tag{1.9}$$

Thus, if $w > n\delta_0(R)$, this value grows exponentially. Therefore, for almost all codes, the number of codewords inside a *given* sphere of radius $w > n\delta_0(R)$ in E_q^n does not deviate sharply from this average value. For w very large, only a trivial estimate is possible. Namely, if $w \ge w_*$, where $w_* = n(q-1)/q$ is a root of

$$q^{n(H_q(w/n)-(1-R))} = q^{Rn},$$

then the sphere of radius w tends to contain most codewords, and the best possible estimate is q^k.

Some further results of this kind are delayed until Section 3.2.

Let $\mathcal{S} \subseteq E_q^n$ be an arbitrary subset of vectors, $|\mathcal{S}| = L$. The above formula enables us to compute the moments of the random number $N(\mathcal{S})$ of code vectors of C contained in this set.

Lemma 1.1 *Suppose Lq^{-r} grows exponentially in n. Then*

(i) $\mathsf{E}N(\mathcal{S}) = \begin{cases} Lq^{-r}, & 0 \notin \mathcal{S}, \\ (L-1)q^{-r}+1, & 0 \in \mathcal{S}; \end{cases}$

(ii) $\mathsf{Var}\,N(\mathcal{S}) \le \left\{ \begin{array}{ll} (q-1)Lq^{-r}, & 0 \notin \mathcal{S} \\ (q-1)(L-1)q^{-r}, & 0 \in \mathcal{S} \end{array} \right\} \le (q-1)\mathsf{E}N(\mathcal{S}).$

Proof: Number all the vectors in $\mathcal{S}$ from 1 to L and let $\nu_i = 1$ or 0 according as the ith vector is contained in C or not. Then $\mathsf{E}\nu_i = q^{-r}$ if the ith vector is nonzero and 1 otherwise. Part (i) follows.

Let us compute the variance.

$$\mathsf{Var}\,N(\mathcal{S}) = \mathsf{E}(\sum_{i=1}^{L} \nu_i)^2 - (\mathsf{E}\sum_{i=1}^{L} \nu_i)^2.$$

Suppose that $0 \notin \mathcal{S}$; then

$$\begin{aligned}
\mathsf{Var}\,N(\mathcal{S}) &= \mathsf{E}\sum_{i=1}^{L} \nu_i + \mathsf{E}\sum_i \sum_{j \ne i} \nu_i \nu_j - L^2 q^{-2r} \\
&= \mathsf{E}\sum_{i=1}^{L} \nu_i + \sum_i \sum_{j \ne i} \mathsf{Pr}\{\nu_i = 1\}\mathsf{Pr}\{\nu_j = 1 | \nu_i = 1\} - L^2 q^{-2r} \\
&\le Lq^{-r} + Lq^{-r}((L-(q-2))q^{-r} + (q-2)) - L^2 q^{-2r} \\
&\le (q-1)Lq^{-r}.
\end{aligned}$$

In the third line we have assumed that with the ith vector the set $\mathcal{S}$ contains all of its nonzero proportional vectors, which certainly are codewords if the ith vector itself is.

The remaining case is treated similarly. ∎

In particular, if $\mathcal{S} = \{\boldsymbol{x} \in E_q^n \,|\, \mathsf{wt}\,(\boldsymbol{x}) = w\}$ is a sphere of radius w around 0 (that is, $0 \notin \mathcal{S}$), then $N(\mathcal{S}) = N_w$ is the random number of code vectors of weight w in C, and the moments become

$$\mathsf{E}N_w = q^{-(n-k)} \binom{n}{w} (q-1)^w, \tag{1.7}$$

$$\mathsf{Var}\,N_w \le q^{-(n-k)} \binom{n}{w} (q-1)^{w+1}. \tag{1.8}$$

Combining (1.7), (1.4), and (1.1), we obtain

$$\Pr\{N_w \ge 1\} \le \mathsf{E}N_w \le q^{-n((1-R)-H_q(w/n))}.$$

This combinatorial setting corresponds to transmission over the q-ary symmetric channel, i.e., a discrete memoryless channel in which a transmitted letter is received correctly with large probability $1-\epsilon$ and substituted by any of the other $q-1$ letters with probability $\epsilon/(q-1)$. This group of problems is also called *hard-decision decoding* as opposed to soft-decision decoding.

Soft-decision decoding takes into account reliability information of the code symbols, supplied by the demodulator. Let $y=(y_1,\dots,y_n)$ be the received signal, whose components are no longer restricted to the code alphabet. Mathematically this means that the transmission defines a random mapping from E_q^n on a certain space Y (a common example is $\mathbf{R}^n$). Let $\boldsymbol{y}\in Y$ be the received signal and suppose we can compute the joint probability $\mathsf{Pr}\,(\boldsymbol{y},\boldsymbol{c})$ for all $\boldsymbol{c}\in C$. Then the value of the decoding mapping $\phi(\boldsymbol{y})$ is the codeword $\boldsymbol{c}$ that maximizes this probability.

4. Numerical parameters. Given a linear code C, compute its minimum distance d, covering radius ρ, number of code vectors of weight w, ...

5. Permutation group. Find the set of permutations that preserve C as a subset of vectors of E_q^n.

Generally, all the above problems except encoding are difficult. For encoding, there is a trivial upper bound $\mathcal{O}(kn)$.

6. Codes for non-Hamming errors. Study the complexity of Problems 1–3 under other models of noise. (This sometimes requires the use of unrestricted codes).

1.3 General properties of linear codes, I

Most algorithms below will deal with long codes. Therefore we treat here the typical behavior of linear $[n,k,d]$ codes of large length, $0<k<n$. By a "long code" we mean a sequence C_n of codes of growing length and dimension. We are interested in the number of codewords of a given weight and in the number of codewords in a sphere of given radius (this is important for the study of decoding).

A code C will be specified by a parity-check matrix H whose entries are chosen from the underlying alphabet independently with uniform distribution. This defines the ensemble of all linear codes with uniform distribution. Let $\boldsymbol{h}$ be a row of H and $\boldsymbol{c}\neq 0$ a vector of E_q^n. Then

$$\mathsf{Pr}\,\{(\boldsymbol{h},\boldsymbol{c})=0\}=q^{-1}.$$

Since all $r=n-k$ check equations are independent, the probability $\Pr\{\boldsymbol{c}\in C\}$ equals

$$\mathsf{Pr}\,\{\boldsymbol{c}\in C\}=\mathsf{Pr}\,\{H\boldsymbol{c}^T=0\}=q^{-r}. \tag{1.6}$$

Remark. More generally, this is the probability that $\boldsymbol{c}$ has a given syndrome $\boldsymbol{s}$, i.e., is contained in a given coset of C.

Given a q-ary code of length n, size M and distance d, $R = \log_q M/n$ is its *rate* and $\delta = d/n$ is its *relative distance.*

Definition. The *Gilbert-Varshamov distance* d_0 is defined as the maximal number such that

$$|C| \sum_{i=0}^{d_0-1} \binom{n}{i} (q-1)^i \leq q^n.$$

The *relative* Gilbert-Varshamov distance $\delta_0(R)$ is the smallest positive root of the equation

$$R = 1 - H_q(x)$$

(cf. Theorem 3.5 in Ch. 1).

If $\boldsymbol{a} \in E_q^n$ and $X \subseteq E_q^n$, then the distance from $\boldsymbol{a}$ to X is defined in the usual way:

$$\mathsf{dist}\,(\boldsymbol{a}, X) = \min_{\boldsymbol{x} \in X} \mathsf{dist}\,(\boldsymbol{a}, \boldsymbol{x}).$$

Notation $\subset$ always means a proper inclusion in contrast to $\subseteq$. If the base of log is not given, this means that it is irrelevant. The symbol 0 means zero or the all-zero vector as appropriate. The symbol $o(1)$ always denotes a *positive* infinitesimal. We say that two infinitely small (large) quantities are *equivalent* if their quotient tends to 1.

3. Algorithmic problems for codes

Let us formulate the main algorithmic problems for linear codes.

1. Code construction. Given a set of parameters q, n, k, d, find a generator matrix G of a q-ary $[n, k, d]$ code C.

2. Encoding. A code may be viewed as a linear injective mapping $C : E_q^k \to E_q^n$ defined by the matrix G. The problem is then to implement this mapping.

3. Decoding. The decoding is a mapping $f : E_q^n \to C$ such that

$$\forall_{\boldsymbol{x} \in E_q^n} \ \mathsf{dist}\,(\boldsymbol{x}, f(\boldsymbol{x})) = \mathsf{dist}\,(\boldsymbol{x}, C).$$

If for a certain $\boldsymbol{x}$, this is satisfied for many code vectors, the value of $f(\boldsymbol{x})$ is chosen arbitrarily from them. This procedure is called (*complete*) *minimum distance decoding.*

If the domain of complete decoding is restricted to those vectors in E_q^n that lie within a given distance t from the code, this partial mapping f_t is called *bounded distance decoding*:

$$f_t : \{\boldsymbol{x} \in E_q^n : \mathsf{dist}\,(\boldsymbol{x}, C) \leq t\} \to C.$$

Of particular interest is the case of t equal to half the distance of C, $t = \lfloor (d-1)/2 \rfloor$, because this is the largest value of t for which, for every vector in the domain, the decoding result is unique.

2. Notation and useful facts

By E_q^n we denote the set of all n-words over the q-ary alphabet. Speaking of linear codes, we assume that E_q^n is a linear space over the field $\mathbf{F}_q$. Let G be a $k \times n$ generator matrix of a linear k-dimensional code C and H its $(n-k) \times n$ parity-check matrix.

We use a short notation $\mathcal{N}$ for the set $\{1, 2, \ldots, n\}$. Let $W \subseteq \mathcal{N}$ and let A be a matrix with n columns. By $A(W)$ we denote the submatrix of A formed by the columns of A labeled with indices from W. Thus, $\boldsymbol{y}(W)$ is a projection of a vector $\boldsymbol{y}$ on its coordinates in W.

Definition. Let C be a linear code. An *information set* (message set) is a k-set $W \subset \mathcal{N}$ such that the corresponding $k \times k$ submatrix $G(W)$ is nonsingular. The remaining $n-k$ coordinates are called a *check set.*

Knowing the information (message) symbols, we may uniquely compute the codeword. If the message symbols form a part of the codeword, the code is said to be represented in a *systematic* form.

Below we often consider the probability that a randomly chosen code in E_q^n satisfies a certain property. If this probability tends to 1 as $n \to \infty$, we say that this property is satisfied for *almost all* codes (or simply for *most* codes). Speaking of random codes, we call the corresponding sample space an *ensemble.*

The entropy function is defined by

$$H_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x).$$

In the asymptotic setting, we always assume that n grows and q is fixed. We shall make frequent use of the following inequalities:

$$\binom{n}{k} < \frac{1}{\sqrt{2\pi k(1-(k/n))}} 2^{nH_2(k/n)}, \tag{1.1}$$

$$\frac{1}{\sqrt{8n\mu(1-\mu)}} q^{nH_q(\mu)} \le \sum_{i=0}^{\mu n} \binom{n}{i} (q-1)^i \le q^{nH_q(\mu)}, \quad 0 < \mu \le (q-1)/q, \tag{1.2}$$

and their asymptotic corollary

$$\log_q \sum_{i=0}^{\mu n} \binom{n}{i} (q-1)^i \sim nH_q(\mu), \quad n \to \infty. \tag{1.3}$$

The Chebyshev inequality in its various forms is used in the proofs of most facts about the ensemble of random codes. For a nonnegative-valued random variable X,

$$\begin{aligned} \Pr\{X \ge a\} &\le \mathsf{E}X/a, & (1.4)\\ \Pr\{|X - \mathsf{E}X| \ge a\} &\le \mathsf{Var}\, X/a^2. & (1.5) \end{aligned}$$

environment is provided by *Random Access Machines* (RAMs), which perform certain Boolean operations and additions and some flow control instructions. It is assumed that the RAM has an unrestricted number of registers with instant access. This model is realistic if the space complexity is not too high and fits into the main memory of a computer. In order to preserve the formalism, each instruction of the RAM is simulated by a Turing machine program, thus relating the time complexity of the two machines. We assume that an operation of the Turing machine corresponds to a binary operation of the RAM (*logarithmic cost model*). Transforming informal descriptions of algorithms into RAM programs usually does not cause principal difficulties.

A usual abstraction of *parallel computations* is a *logical circuit*. This is a directed tree with nodes (gates) corresponding to Boolean operations from a chosen basis. An input string is fed to the input nodes of the circuit and the result of the computation appears on the output nodes. Complexity is measured by the number of gates (*size*) and the length of the longest path from an input to an output (*depth*).

Decision problems are joined in complexity classes using the concept of reducibility. A decision problem $\mathcal{A}$ may be viewed as a subset of $\{0,1\}^*$ formed by instances with the answer *yes* under $\mathcal{A}$. Thus, a decision problem is, in essence, a string relation. To solve $\mathcal{A}$ means to construct a machine recognizing this relation. Let $\mathcal{B}$ be another decision problem. Suppose that for every instance $A \in \mathcal{A}$ of length ℓ we can construct an instance $B \in \mathcal{B}$ such that A has the answer *yes* under $\mathcal{A}$ if and only if B has the answer *yes* under $\mathcal{B}$, and the time required for this construction is at most $f(\ell)$, where f is a certain fixed polynomial. Then one says that $\mathcal{A}$ is *polynomially reducible* to $\mathcal{B}$.

Given a class of problems $\mathcal{A}$, it is sometimes possible to show that every problem in it is polynomially reducible to a certain problem, $\mathcal{A} \in \mathcal{A}$. Then $\mathcal{A}$ is said to be *complete* in $\mathcal{A}$. This means that the worst-case complexity of any problem in $\mathcal{A}$ is bounded by a polynomial in the complexity of $\mathcal{A}$. Therefore, the complexity of $\mathcal{A}$ is universal for the whole class of problems $\mathcal{A}$.

Problems of polynomial complexity are joined in the class P. They are usually deemed easy. Many combinatorial problems heuristically known to be difficult are contained in the class NP (and are complete in it). A problem in this class can be solved in polynomial time by a nondeterministic machine, which at each step checks in parallel all the existing possibilities of the computation. A widely accepted conjecture is that (deterministic) polynomial algorithms for the class NP do not exist. Therefore, NP-complete problems are often termed *intractable*.

In computability theory a problem is called *constructive* if there exists an algorithm solving it. In the vernacular of discrete mathematics, however, *constructive* or *explicit* has come to mean soluble with polynomial complexity in contrast to *non-constructive* which just means involving exhaustive search or table lookup.

about the polynomial hierarchy of complexity classes). The most important result is the NP-hardness of minimum distance decoding of general linear codes. This shows that, basically, algorithms better than exhaustive search are highly unlikely to be found. This belief is further supported by the fact that even approximating the number of errors up to any constant is NP-complete (Theorem 4.6). We also present a recent proof [161] of the long-standing conjecture about the NP-hardness of computing the minimum distance of a code. The results in this section provide conditional lower bounds on complexity. Unconditional lower bounds are not known except for a rather restrictive setting of syndrome trellis decoding (see Sect. 3.4).

We have attempted to include as many proofs as the chosen format could allow. Often we rewrote the existing proofs and sometimes replaced them by new ones.

Historical remarks and attributions are given in notes following the sections.

1.2 Conventions

1. Algorithms and complexity

We are used to say and think that complexity of algorithms is measured by the time of the computation and the size of memory used for it. Though the definitions of computation and complexity will not be given, some hints of possible formalizations may be in order.

An informal notion of an *algorithmic problem* includes a class of objects and a property that some of these objects may enjoy. The goal of the computation is to produce objects from this class with this property or to check whether a given object has it. Often the problem depends on varying certain numerical parameters from which one produces an infinite series of individual instances of the problem. The class of instances is then called an (algorithmic) *mass problem*. These concepts are formalized by introducing a suitable encoding of objects which turns an instance into a binary string. An answer to the problem may be an object, a number, or simply *yes* or *no*. In the latter case, the mass problem is called a *decision problem*.

A *computation* is a way to solve a given algorithmic problem $\mathcal{A}$. To formalize this notion, one introduces an abstract computing device (machine) which, given a string corresponding to an instance of the problem, produces an answer to it. Consider all instances that can be written as strings of a certain length ℓ. The (worst-case) *time complexity* of $\mathcal{A}$ is a function of ℓ defined as the maximal, over all instances of length ℓ, time of the computation. Depending on the assumptions, it may be necessary to consider separately the *space complexity* of $\mathcal{A}$, i.e., the number of registers used in the course of the computation. This is not essential for single-tape Turing machines, whose running time combines the time and space complexity. The omnipresence of Turing machines relies on the Church thesis which asserts that any computation can be modeled by a Turing machine program in a reasonable fashion. However, writing such a program for a complicated algorithm is a notably time-consuming task. A less restrictive

check matrices. The entries of the matrices are chosen independently and with uniform distribution; therefore, all the probabilities that appear below have a purely combinatorial sense of the *fraction* of favorable cases out of the total number of cases.

As far as coding is concerned, the order of complexity can be either polynomial in the length n of the code, which is considered *easy*, or exponential, which is *difficult*. We shall mostly concentrate on *linear* codes, the reason being that they admit a simple description in terms of the generator matrix. The size of this matrix is at most n^2, which makes it sensible to speak of polynomial complexity. On the contrary, to define a nonlinear code one has to explicitly specify all of its codewords. In this case the input data has size commensurate with the total size of the code M. Most algorithmic problems of coding theory allow an exhaustive search with complexity of order M, i.e., linear in the size of the input. Therefore, studying complexity problems for unrestricted codes usually attracts less attention (except when the complexity grows essentially faster than the size of the code; see Theorem 4.4).

We begin with easy problems. They are grouped in Section 2. There is a developed theory of codes with algebraic structure, which have a low construction complexity and admit a simple decoding. This theory is covered in other chapters of this Handbook. We focus our attention on codes without apparent structure, often codes chosen randomly from a conveniently defined ensemble. Though these results are not always practical, they provide a frame of reference in the field of complexity. We first study error-correcting codes and then review results related to other models of noise, namely, erasures, defective memory cells, and localized errors.

However, the majority of coding problems are computationally difficult. One of the most important problems of this type is decoding, which forms the subject of Sections 3.3, 3.4. We begin with hard decision decoding. This group of problems has a relatively established history. We cover most general results up to very recent developments. We make an essential use of general properties of linear codes, most of which we prove in Sections 1.3, 3.2. Then we proceed to soft decision decoding. This is a much more difficult problem and until recently general results have been scanty. However, now the general theory of hard decision decoding finds it parallels in the field of soft decision, which makes our exposition satisfactorily complete.

Another group of problems deals with computing important numerical parameters of codes such as the minimum distance (Sect. 3.5.1) and the weight spectrum (Sect. 3.5.2). General algorithms for these problems are closely related to those for minimum distance decoding.

These two sections form the main part of the chapter. For some problems we present a number of solutions that reflect different ideas. However, for each problem we always present an algorithm that has the best, to our knowledge, asymptotic complexity.

Section 4 lists algorithmic problems related to codes that are presumed to be computationally difficult (i.e. based on conjectures from complexity theory

1 Introduction

1.1 Outline

Coding theory is motivated by a practical problem of transmission over noisy channels. Therefore, along with the question "Do good codes exist?" there is always another question, namely, "How to construct and decode them?" The words *construct* and *decode* hint of algorithms and those suggest studying complexity. However, defined in this way, the topic would be too broad for a chapter since almost every problem in coding theory is linked to the algorithmic complexity of actually presenting a solution to it. We intend to study primarily problems related to the construction of block codes, their decoding, and computing important numerical parameters of codes.

The topic of complexity involves a number of conventions related to the computation models admitted and particular details of implementation of algorithms. This makes the comparison of results related to specific short codes complicated if not impossible at all. To make the exposition mathematically sound, we shall mostly focus on the *asymptotic* behavior of the algorithms considered, which seems to be the best theoretical framework available. We shall also mostly discuss algorithms with provable performance and complexity estimates. For this reason, many heuristics, especially dealing with decoding, for which only simulation results are known, are left out. This decision also helps to bring down the number of references which otherwise would occupy scores of pages. For instance, an electronic search for papers on decoding in the *Mathematical Reviews* database in July 1996 brought a response of 599 titles. With keyword search available electronically, the reader can produce a comprehensive bibliography himself.

In Section 1.2 we discuss models of computation. Formal definitions of computations, algorithms, and complexity can be given, in particular, in terms of Turing machines. However, descriptions of specific algorithms as Turing machine programs may prove unwieldy. A step towards more practical descriptions is taken by defining more liberal computation abstractions and simulating them by Turing machines. A sufficiently convenient model is provided by the Random Access Machine (RAM), which better corresponds to "real-life" computers. Accepting this model roughly means that the complexity of a (sequential) algorithm is measured as the number of basic operations needed for its implementation. For parallel computations, logical circuits seem to be a convenient formalization. Their complexity is measured by the size (number of gates) and depth (computation time).

The general framework of studying specific coding problems is complexity vs performance. In other words, we shall be interested in the least known (or possible) complexity of constructing codes with given properties or the best known parameters of codes with a specified order of complexity. In Section 1.3 we outline the typical properties of long codes. By this we mean the average behavior of the parameters of codes from the random ensemble given by parity-

Contents

1 Introduction **3**
1.1 Outline . . . 3
1.2 Conventions . . . 5
1.3 General properties of linear codes, I . . . 9
1.4 Notes . . . 12

2 Easy Problems **13**
2.1 Error-correcting codes . . . 13
2.1.1 Decoding . . . 13
2.1.2 Construction complexity . . . 25
2.1.3 Encoding complexity . . . 28
2.1.4 Notes . . . 29
2.2 Other models of noise . . . 31
2.2.1 Codes correcting erasures . . . 31
2.2.2 Codes for memories with defective cells . . . 34
2.2.3 Codes correcting localized errors . . . 37
2.2.4 Notes . . . 38

3 Difficult Problems **39**
3.1 Code construction . . . 39
3.2 General properties of linear codes, II . . . 40
3.2.1 Notes . . . 44
3.3 Hard-decision decoding . . . 45
3.3.1 General remarks . . . 45
3.3.2 Split syndrome decoding . . . 46
3.3.3 Gradient-like decoding . . . 48
3.3.4 Information set decoding . . . 55
3.3.5 Notes . . . 68
3.4 Soft-decision decoding . . . 70
3.4.1 Syndrome trellis decoding . . . 72
3.4.2 Maximum likelihood decoding with reduced complexity . . . 76
3.4.3 Notes . . . 83
3.5 Computing numerical parameters of codes . . . 85
3.5.1 Minimum distance . . . 85
3.5.2 Weight spectrum . . . 88
3.5.3 Notes . . . 92

4 Intractable Problems **94**
4.1 Notes . . . 101

COMPLEXITY ISSUES IN CODING THEORY

Alexander Barg

Bell Laboratories, Lucent Technologies
600 Mountain Avenue, Room 2C-375
Murray Hill, NJ 07974
abarg@research.bell-labs.com

Abstract. This paper deals with complexity issues in the theory of linear error-correcting codes. Algorithmic problems that we study are constructing good codes, encoding and decoding them. According to their complexity. problems are divided into easy, i.e., polynomial in the length n of the code, and difficult, i.e., exponential ones. The first part deals with easy problems. We present a construction of codes that correct a linear fraction of errors with complexity $n \log n$. The construction is based on well-known since the late 80ies explicit constructions of good expanding graphs. Another group of problems in this part is related to codes for non-Hamming errors, namely, erasures, defects (codes for memories with defective cells), and localized errors.
The second part, which forms the core of this paper, deals with difficult problems, first and foremost, maximum likelihood decoding of linear codes. We study separately the complexity of hard-decision and soft-decision decoding. For the hard-decision decoding case we present algorithms grouped in two classes, gradient-like decoding and information-set decoding. It turns out that this general approach is sufficient to study most if not all known general decoding methods. In the soft-decision decoding context, we first discuss possible problem settings and then implementations of decoding with reduced complexity.
The last part of the paper overviews most known NP-hard decoding problems including some recent nonapproximability results.
The supporting material includes many general properties of linear codes from well-known to rather sophisticated, and a brief discussion of models of computations and relevant settings for the study of complexity issues in coding theory. We also give examples of many methods studied. Sometimes they just illustrate concepts and definitions, but sometimes capture the most essential features of the proofs and on occasion even replace them.
Generally we give complete and self-contained proofs of the results.
The coverage is extended from classical algorithms up to very recent developments. We thoroughly study and compare different algorithms, especially those applicable to several seemingly non-related problems. This unified approach to algorithmic coding problems enables us to organize previously independent results in a self-contained part of coding theory.
This paper will appear as a chapter in Handbook of Coding Theory, V. Pless, W. Cary Huffman and R. Brualdi, Eds., Elsevier Science, to be published.

Keywords: linear code, complexity, encoding, regular-graph codes, maximum likelihood decoding, Gilbert–Varshamov bound, soft-decision decoding, NP-hard problems.