Interior-Point Algorithms for Linear-Programming Decoding

Pascal O. Vontobel
Hewlett–Packard Laboratories
Palo Alto, CA 94304, USA
Email: pascal.vontobel@ieee.org

***Abstract*— Interior-point algorithms constitute a very interesting class of algorithms for solving linear-programming problems. In this paper we study efficient implementations of such algorithms for solving the linear program that appears in the linear-programming decoder formulation.**

I. Introduction

Consider a binary linear code $\mathcal{C}$ of length n that is used for data transmission over a binary-input discrete memoryless channel. As was observed by Feldman *et al.* [1], [2], the ML decoder for this setup can be written as

$$\hat{\mathbf{x}}_{\mathrm{ML}} = \arg\max_{\mathbf{x}\in\mathcal{C}} \langle \boldsymbol{\gamma}, \mathbf{x} \rangle,$$

where $\boldsymbol{\gamma}$ is a length-n vector that contains the log-likelihood ratios and where $\langle \boldsymbol{\gamma}, \mathbf{x} \rangle$ is the inner product (in $\mathbb{R}$) of the vector $\boldsymbol{\gamma}$ with the vector $\mathbf{x}$. Because the cost function in this maximization problem is linear, this is essentially equivalent to the solution of

$$\hat{\mathbf{x}}'_{\mathrm{ML}} \triangleq \arg \max_{\mathbf{x}\in\mathrm{conv}(\mathcal{C})} \langle \boldsymbol{\gamma}, \mathbf{x} \rangle,$$

where $\mathrm{conv}(\mathcal{C})$ denotes the convex hull of $\mathcal{C}$ when $\mathcal{C}$ is embedded in $\mathbb{R}^n$. (We say "essentially equivalent" because in the case where there is a unique optimal codeword then the two maximization problems yield the same solution. However, when there are multiple optimal codewords then $\hat{\mathbf{x}}_{\mathrm{ML}}$ and $\hat{\mathbf{x}}'_{\mathrm{ML}}$ are non-singlet sets and it holds that $\mathrm{conv}(\hat{\mathbf{x}}_{\mathrm{ML}}) = \hat{\mathbf{x}}'_{\mathrm{ML}}$.)

Because the above two optimization problems are usually practically intractable, Feldman *et al.* [1], [2] proposed to solve a relaxation of the above problem. Namely, for a code $\mathcal{C}$ that can be written as the intersection of m binary linear codes of length n, i.e., $\mathcal{C} \triangleq \cap_{j=1}^{m} \mathcal{C}_j$, they introduced the so-called linear programming (LP) decoder

$$\hat{\mathbf{x}}_{\mathrm{LP}} \triangleq \arg\max_{\mathbf{x}\in\mathcal{P}} \langle \boldsymbol{\gamma}, \mathbf{x} \rangle, \tag{1}$$

with the relaxed polytope

$$\mathcal{P} \triangleq \bigcap_{j=1}^{m} \mathrm{conv}(\mathcal{C}_j) \supseteq \mathrm{conv}(\mathcal{C}) \supseteq \mathcal{C}, \tag{2}$$

for which it can easily be shown that all codewords in $\mathcal{C}$ are vertices of $\mathcal{P}$.

The same polytope $\mathcal{P}$ appeared also in papers by Koetter and Vontobel [3], [4], [5], where message-passing iterative (MPI) decoders were analyzed and where this polytope $\mathcal{P}$ was called the fundamental polytope. The appearance of the same object in these two different contexts suggests that there is a tight connection between LP decoding and MPI decoding.

The above codes $\mathcal{C}_j$ can be any codes of length n, however, in the following we will focus on the case where these codes are codes of dimension $n-1$. For example, let $\mathbf{H}$ be an $m \times n$ parity-check matrix for the code $\mathcal{C}$ and let $\mathbf{h}_j^{\mathsf{T}}$ be the j-th row of $\mathbf{H}$.[1] Then, defining

$$\mathcal{C}_j = \big\{ \mathbf{x} \in \{0,1\}^n \;\big|\; \langle \mathbf{h}_j, \mathbf{x} \rangle = 0 \pmod 2 \big\}$$

for $j = 1, \ldots, m$, we obtain $\mathcal{C} = \cap_{j=1}^{m} \mathcal{C}_j$.

Of course, the reason why the decoder in (1) is called LP decoder is because the optimization problem in that equation is a linear program (LP).[2] There are two standard forms for LPs, namely

$$\begin{aligned} \text{minimize} \quad & \langle \mathbf{c}, \mathbf{x} \rangle \\ \text{subj. to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3}$$

and

$$\begin{aligned} \text{maximize} \quad & \langle \mathbf{b}, \boldsymbol{\lambda} \rangle \\ \text{subj. to} \quad & \mathbf{A}^{\mathsf{T}}\boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ & \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{4}$$

Any LP can be reformulated (by introducing suitable auxiliary variables, by reformulating equalities as two inequalities, etc.) so that it looks like the first standard form. Any LP can also be reformulated so that it looks like the second standard form. Moreover, the first and second standard form are tightly related in the sense that they are dual convex programs. Usually, the LP in (3) is called the primal LP and the LP in (4) is called the dual LP. (As it is to be expected from the expression "duality," the primal LP is the dual of the dual LP.)

Not unexpectedly, there are many ways to express the LP that appears in (1) in either the first or the second standard form, and each of these reformulations has its advantages (and disadvantages). Once it is expressed in one of the standard forms, any general-purpose LP solver can basically be used to obtain the LP decoder output. However, the LP at hand has a lot of structure and one should take advantage of it in order to

[1] Note that in this paper all vectors are *column* vectors.

[2] We use LP to denote both "linear programming" and "linear program."

obtain very fast algorithms that can compete complexity- and time-wise with MPI decoders.

Several ideas have been presented in the past in this direction, e.g., by Feldman *et al.* [6] briefly mention the use of sub-gradient methods for solving the LP of an early version of the LP decoder (namely for turbo-like codes), by Yang et al. [7], [8] on efficiently solvable variations of the LP decoder, by Taghavi and Siegel [9] on cutting-hyperplane-type approaches, by Vontobel and Koetter [10] on coordinate-ascent-type approaches, by Dimakis and Wainwright [11] and by Draper *et al.* [12] on improvements upon the LP decoder solution, and by Taghavi and Siegel [13] and by Wadayama [14] on using variations of LP decoding (together with efficient implementations) for intersymbol-interference channels.

In this paper our focus will be on so-called interior-point algorithms, a type of LP solvers that has become popular with the seminal work of Karmarkar [15]. (After the publication of [15] in 1984, earlier work on interior-point-type algorithms by Dikin [16] and others became more widely known). We present some initial thoughts on how to use this class of algorithms in the context of LP decoding. So far, with the notable exception of [14], interior-point-type algorithms that are especially targeted to the LP in (1) do not seem to have been considered. One of our goals by pursuing these type of methods is that we can potentially obtain algorithms that are better analyzable than MPI decoders, especially when it comes to finite-length codes. (Wadayama [14] discusses some efficient interior-point-type methods, however, he is trying to minimize a quadratic cost function, and the final solution is obtained through the use of the sum-product algorithm that is initialized by the result of the interior-point search. Although [14] presents some very interesting approaches that are worthwhile pursuing, it is not quite clear if these algorithms are better analyzable than MPI decoders.)

There are some interesting facts about interior-point-type algorithms that make them worthwhile study objects. First of all, there are variants for which one can prove polynomial-time convergence (even in the worst case, which is in contrast to the simplex algorithm). Secondly, we can round an intermediate result to the next vector with only 0 / $\frac{1}{2}$ / 1 entries and check if it is a codeword.[3] (This is very similar to the stopping criterion that is used for MPI algorithms.) Note that a similar approach will probably not work well for simplex-type algorithms that typically wander from vertex to vertex of the fundamental polytope. The reason is that rounding the coordinates of a vertex yields only a codeword if the vertex was a codeword.[4] Thirdly, interior-point-type algorithms are also interesting because they are less sensitive than simplex-type algorithms to degenerate vertices of the feasible region; this is important because the fundamental polytope has many degenerate vertices.

The present paper is structured as follows. In Secs. II and III we discuss two classes of interior-point algorithms, namely affine-scaling algorithms and primal-dual interior-point algorithms, respectively. As we will see, the bottleneck step of the algorithms in these two sections is to repeatedly find the solution to a certain type of system of linear equations. Therefore, we will address this issue, and efficient solutions to it, in Sec. IV. Finally, we briefly mention some approaches for potential algorithm simplifications in Sec. V and we conclude the paper in Sec. VI.

II. Affine Scaling Algorithms

An interesting class of interior-point-type algorithms are so-called affine scaling algorithms which were introduced by Dikin [16] and re-invented many times afterwards. Good introductions to this class of algorithms can be found in [17], [18].

Fig. 1 gives an intuitive picture of the workings of one instance of an affine-scaling algorithm. Consider the LP in (3) and assume that the set of all feasible points, i.e., the set of all $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$, is a triangle. For the vector $\mathbf{c}$ shown in Fig. 1, the optimal solution will be the corner in the lower left part. The algorithm works as follows:

1) Select an initial point that is in the interior of the set of all feasible points, cf. Fig. 1(b), and let the current point be equal to this initial point.
2) Minimizing $\langle \mathbf{c}, \mathbf{x} \rangle$ over the triangle is difficult (in fact, it is the problem we are trying to solve); therefore, we replace the triangle constraint by an ellipsoidal constraint that is centered around the current point. Such an ellipsoid is shown in Fig. 1(c). Its skewness depends on the closeness to the different boundaries.
3) We then minimize the function $\langle \mathbf{c}, \mathbf{x} \rangle$ over this ellipsoid. The difference vector between this minimizing point and the center of the ellipsoid (see the little vector in Fig. 1(d)) points in the direction in which the next step will be taken.
4) Depending on what strategy is pursued, a shorter or a longer step is taken in the above-found direction. This results in a new current point. (Whatever step size is taken, we always impose the constraint that the step size is such that the new current point lies in the interior of the set of feasible points.)
5) If the current point is "close enough" to some vertex then stop, otherwise go to Step 2. ("Closeness" is determined according to some criterion.)

Not surprisingly, when short (long) steps are taken in Step 4, the resulting algorithm is called the short-step (long-step) affine scaling algorithm. Convergence proofs for different cases can be found in [19], [20], [21].

Of course, an affine-scaling algorithm can also be formulated for the LP in (4). Moreover, instead of the above-

[3] To be precise, by rounding we mean that coordinates below $\frac{1}{2}$ are mapped to 0, that coordinates above $\frac{1}{2}$ are mapped to 1, and that coordinates equal to $\frac{1}{2}$ are mapped to $\frac{1}{2}$.

[4] Proof: in an LDPC code where all checks have degree at least two, the largest coordinate of any nonzero-vector vertex is at least $\frac{1}{2}$. Therefore, there is no nonzero-vector vertex that is rounded to the all-zero codeword. The proof is finished by using the symmetry of the fundamental polytope, i.e., the fact that the fundamental polytope "looks" the same from any codeword.

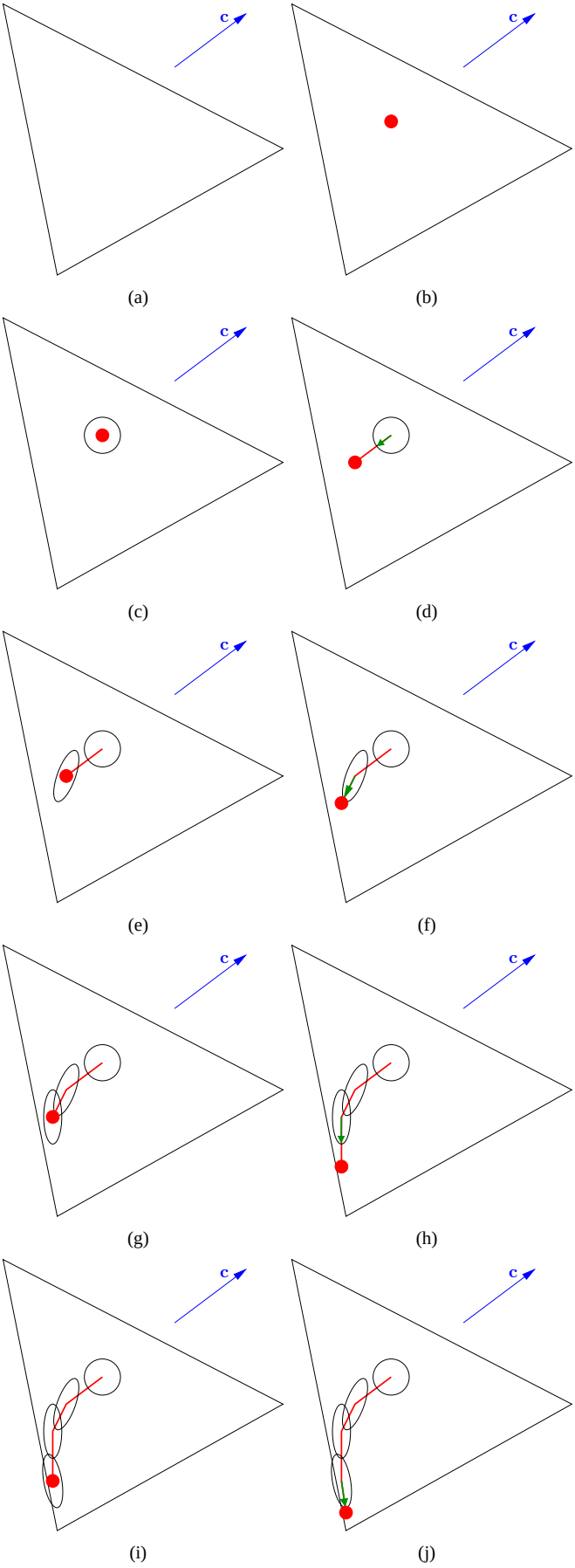

Fig. 1. Some iterations of the affine-scaling algorithm. (See text for details.)

described discrete-time version, one can easily come up with a continuous-time version, see e.g. [22]. The latter type of algorithms might actually be interesting for decoders that are implemented in analog VLSI.

The bottleneck step in the affine-scaling algorithm is to find the new direction, which amounts to solving a linear system of equations of the form $\mathbf{Pu} = \mathbf{v}$, where $\mathbf{P}$ is a given (iteration-dependent) positive definite matrix, $\mathbf{v}$ is a given vector, and $\mathbf{u}$ is the direction vector that needs to be found. We will comment on efficient approaches for solving such systems of equations in Sec. IV.

III. Primal-Dual Interior Point Algorithms

In contrast to affine-scaling algorithms, which either work only with the primal LP or only with the dual LP, primal-dual interior point algorithms – as the name suggests – work simultaneously on obtaining a primal *and* a dual optimal solution. A very readable and detailed introduction to this topic can be found in [23]. As in the case of the affine-scaling algorithm there are many different variations: short-step, long-step, predictor-corrector, path-following, etc.

Again, the bottleneck step is to find the solution to a linear system of equations $\mathbf{Pu} = \mathbf{v}$, where $\mathbf{P}$ is a given (iteration-dependent) positive definite matrix, $\mathbf{v}$ is a given (iteration-dependent) vector, and $\mathbf{u}$ is the sought quantity. We will comment in Sec. IV on how such systems of linear equations can be solved efficiently.

A variant that is worthwhile to be mentioned, is the class of so-called infeasible-interior-point algorithms. The reason is that very often it is easy to find a primal feasible initial point or it is easy to find a dual feasible initial point but not both at the same time. Therefore, one starts the algorithm with a primal/dual point pair where the primal and/or the dual point are infeasible points; the algorithm then tries to decrease the amount of "infeasibility" (a quantity that we will not define here) at every iteration, besides of course optimizing the cost function.

One of the most intriguing aspects of primal-dual interior-point algorithms is the polynomial-time worst-case bounds that can be stated. Of course, these bounds say mostly something about the behavior when the algorithm is already close to the solution vertex. It remains to be seen if these results are useful for implementations of the LP decoder where it is desirable that the initial iterations are as aggressive as possible and where the behavior close to a vertex is not that crucial. (We remind the reader of the rounding-procedure that was discussed at the end of Sec. I, a procedure that took advantage of some special properties of the fundamental polytope.)

IV. Efficient Approaches for Solving $\mathbf{Pu} = \mathbf{v}$ where $\mathbf{P}$ is a Positive Definite Matrix

In Secs. II and III we saw that the crucial part in the discussed algorithms was to repeatedly and efficiently solve a system of linear equations that looks like

$$\mathbf{Pu} = \mathbf{v},$$

where $\mathbf{P}$ is an iteration-dependent positive definite matrix and where $\mathbf{v}$ is an iteration-dependent vector. The fact that $\mathbf{P}$ is positive definite helps because $\mathbf{u}$ can also be seen to be the solution of the quadratic unconstrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{u}^{\mathsf{T}}\mathbf{P}\mathbf{u} - \langle \mathbf{v}, \mathbf{u} \rangle \\ \text{subj. to} \quad & \mathbf{u} \in \mathbb{R}^h \end{aligned} \tag{5}$$

where we assumed that $\mathbf{P}$ is an $h \times h$-matrix. It is important to remark that for the algorithms in Secs. II and III the vector $\mathbf{u}$ usually does not have to be found perfectly. It is good enough to find an approximation of $\mathbf{u}$ that is close enough to the correct $\mathbf{u}$. (For more details, see e.g. [18, Ch. 9].)

Using a standard gradient-type algorithm to find $\mathbf{u}$ might work. However, the matrix $\mathbf{P}$ is often ill-conditioned, i.e., the ratio of the largest to the smallest eigenvalue can be quite big (especially towards the final iterations), and so the convergence speed of a gradient-type algorithm might suffer considerably.

Therefore, more sophisticated approaches are desirable. Such an approach is the conjugate-gradient algorithm which was introduced by Hestenes and Stiefel [24]. (See Shewchuk's paper [25] for a very readable introduction to this topic and for some historical comments.) This method is especially attractive when $\mathbf{P}$ is sparse or when $\mathbf{P}$ can be written as a product of sparse matrices, the latter being the case for LP decoding of LDPC codes.

In the context of the affine scaling, e.g. Resende and Veiga [26] used the conjugate-gradient algorithm to efficiently solve the relevant equation systems and studied the behavior of the conjugate-gradient algorithm with different preconditioners.

A quite different, yet interesting variant to solve the minimization problem in (5) is by using graphical models. Namely, one can represent the cost function in (5) by an *additive* factor graph [27], [28], [29]. Of course, there are a variety of factor graph representations for the this cost function, however, probably the most reasonable choice in the context of LP decoding is to choose the factor graph that looks topologically like the factor graph that is usually used for sum-product or min-sum algorithm decoding of LDPC codes. One can then try to find the solution with the help of the min-sum algorithm.

[Equivalently, one can look at the maximization problem

$$\begin{aligned} \text{maximize} \quad & \exp\left(-\frac{1}{2}\mathbf{u}^{\mathsf{T}}\mathbf{P}\mathbf{u} + \langle \mathbf{v}, \mathbf{u} \rangle\right) \\ \text{subj. to} \quad & \mathbf{u} \in \mathbb{R}^h. \end{aligned} \tag{6}$$

Here the function to be optimized is proportional to a Gaussian density and can be represented with a Gaussian factor graph [27], [28], [29]. (Which in contrast to the above factor graph is a *multiplicative* factor graph.) One can then try to find the solution with the help of the max-product algorithm, which in the case of Gaussian graphical models is equivalent (up to proportionality constants) to the sum-product algorithm.]

The reason for this being an interesting approach is that the behavior of the min-sum algorithm applied to a quadratic-cost-function factor graphs is much better understood than for other factor graphs. E.g., it is known that if the algorithm converges then the solution vector is correct. Moreover, by now there are also practically verifiable sufficient conditions for convergence [30], [31], [32]. However, the quadratic-cost-function factor graphs needed for the above problem are more general than the special class of quadratic-cost-function factor graphs considered in the cited papers. Of course, one could represent the cost function in (6) by a factor graph within this special class (so that the above-mentioned results are applicable), however, and quite interestingly, when this cost function is represented by a factor graph that is not in this special class, then the convergence conditions seem to be (judging from some empirical evidence) less stringent. In fact, we obtained some very interesting behavior in the context of the short-step affine-scaling algorithm where only one iteration of the min-sum algorithm was performed per iteration of the affine-scaling algorithm. (The min-sum algorithm was initialized with the messages obtained in the previous affine-scaling algorithm iteration.)

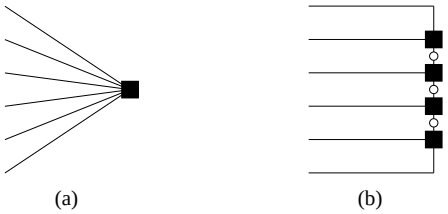

Fig. 2. Replacement of a partial factor graph representing a degree-k check function node by another partial factor graph with $k-2$ check nodes of degree three and with $k-3$ new auxiliary variable nodes. (Here $k = 6$.)

V. Other Simplifications

Depending on the used algorithm, there are many small (but very useful) variations that can – when properly applied – lead to considerable simplifications. E.g., one can replace the partial factor graph in Fig. 2(a) by the partial factor graph in Fig. 2(b) that contains new auxiliary variable nodes but contains only check nodes of degree three [33], [34]. Or, one can adaptively modify the set of inequalities that are included in the LP formulation [13], [14].

VI. Conclusion

We have presented some initial considerations towards using interior-point algorithms for obtaining efficient LP decoders. Encouraging preliminary results have been obtained but more research is needed to fully understand and exploit the potential of these algorithms.

Acknowledgment

This research was partly supported by NSF Grant CCF-0514801.

References

[1] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2003, available online under `http://www.columbia.edu/~jf2189/pubs.html`.

[2] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. on Inform. Theory*, vol. IT–51, no. 3, pp. 954–972, May 2005.

[3] R. Koetter and P. O. Vontobel, "Graph covers and iterative decoding of finite-length codes," in *Proc. 3rd Intern. Symp. on Turbo Codes and Related Topics*, Brest, France, Sept. 1–5 2003, pp. 75–82.

[4] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *accepted for IEEE Trans. Inform. Theory, available online under* `http://www.arxiv.org/abs/cs.IT/0512078`, 2007.

[5] ——, "On the relationship between linear programming decoding and min-sum algorithm decoding," in *Proc. Intern. Symp. on Inform. Theory and its Applications (ISITA)*, Parma, Italy, Oct. 10–13 2004, pp. 991–996.

[6] J. Feldman, D. R. Karger, and M. J. Wainwright, "Linear programming-based decoding of turbo-like codes and its relation to iterative approaches," in *Proc. 40th Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, Illinois, USA, October 2–4 2002, available online under `http://www.columbia.edu/~jf2189/pubs.html`.

[7] K. Yang, X. Wang, and J. Feldman, "Non-linear programming approaches to decoding low-density parity-check codes," in *Proc. 43rd Allerton Conf. on Communications, Control, and Computing*, Allerton House, Monticello, Illinois, USA, Sep. 28–30 2005.

[8] ——, "Fast ML decoding of SPC product code by linear programming decoding," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, Nov. 26–30 2007, pp. 1577–1581.

[9] M. H. Taghavi and P. H. Siegel, "Adaptive linear programming decoding," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Seattle, WA, USA, July 9–14 2006, pp. 1374–1378.

[10] P. O. Vontobel and R. Koetter, "On low-complexity linear-programming decoding of LDPC codes," *Europ. Trans. on Telecomm.*, vol. 5, pp. 509–517, Aug. 2007.

[11] A. G. Dimakis and M. J. Wainwright, "Guessing facets: polytope structure and improved LP decoder," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Seattle, WA, USA, July 9–14 2006, pp. 1369–1373.

[12] S. C. Draper, J. S. Yedidia, and Y. Wang, "ML decoding via mixed-integer adaptive linear programming," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Nice, France, June 24–29 2007, pp. 1656–1660.

[13] M. H. Taghavi and P. H. Siegel, "Equalization on graphs: linear programming and message passing," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Nice, France, June 24–29 2007, pp. 2551–2555.

[14] T. Wadayama, "Interior point decoding for linear vector channels," *submitted, available online under* `http://arxiv.org/abs/0705.3990`, May 2007.

[15] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, Dec. 1984.

[16] I. I. Dikin, "Iterative solution of problems of linear and quadratic programming," *Soviet Math. Doklady*, vol. 8, pp. 674–675, 1967.

[17] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[18] D. Bertsimas and J. N. Tsitsiklis, *Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[19] T. Tsuchiya, "Global convergence of the affine scaling methods for degenerate linear programming problems," *Math. Progr.*, vol. 52, no. 3, pp. 377–404, May 1991.

[20] T. Tsuchiya and M. Muramatsu, "Global convergence of a long-step affine scaling algorithm for degenerate linear programming problems," *SIAM J. Opt.*, vol. 5, no. 3, pp. 525–551, Aug. 1995.

[21] R. Saigal, "A simple proof of a primal affine scaling method," *Ann. Oper. Res.*, vol. 62, pp. 303–324, 1996.

[22] I. Adler and R. D. C. Monteiro, "Limiting behavior of the affine scaling continuous trajectories for linear programming problems," *Math. Progr.*, vol. 50, no. 1, pp. 29–51, Mar. 1991.

[23] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1997.

[24] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, 1952.

[25] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*. Technical Report, Carnegie Mellon University, Pittsburgh, PA, 1994.

[26] M. G. C. Resende and G. Veiga, "An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks," *SIAM Journal on Optimization*, vol. 3, no. 3, pp. 516–537, 1993.

[27] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 498–519, Feb. 2001.

[28] G. D. Forney, Jr., "Codes on graphs: normal realizations," *IEEE Trans. on Inform. Theory*, vol. IT–47, no. 2, pp. 520–548, Feb. 2001.

[29] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Sig. Proc. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[30] D. M. Malioutov and J. K. Johnson and A. S. Willsky, "Walk-sums and belief propagation in Gaussian graphical models," *J. Mach. Learn. Res.*, vol. 7, pp. 2031–2064, Dec. 2006.

[31] C. C. Moallemi and B. Van Roy, "Convergence of the min-sum message passing algorithm for quadratic optimization," *submitted*, Mar. 2006.

[32] ——, "Convergence of the min-sum algorithm for convex optimization," *submitted*, May 2007.

[33] M. Chertkov and M. Stepanov, "Pseudo-codeword landscape," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Nice, France, June 24–29 2007, pp. 1546–1550.

[34] K. Yang, X. Wang, and J. Feldman, "Cascaded formulation of the fundamental polytope of general linear block codes," in *Proc. IEEE Intern. Symp. on Inform. Theory*, Nice, France, June 24–29 2007, pp. 1361–1365.