Nonconvex Global Optimization for Latent-Variable Models[*]

Matthew R. Gormley **Jason Eisner**
Department of Computer Science
Johns Hopkins University, Baltimore, MD
`{mrg,jason}@cs.jhu.edu`

Abstract

Many models in NLP involve latent variables, such as unknown parses, tags, or alignments. Finding the optimal model parameters is then usually a difficult nonconvex optimization problem. The usual practice is to settle for *local* optimization methods such as EM or gradient ascent.

We explore how one might instead search for a *global* optimum in parameter space, using branch-and-bound. Our method would eventually find the global maximum (up to a user-specified ϵ) if run for long enough, but at any point can return a suboptimal solution together with an upper bound on the global maximum.

As an illustrative case, we study a generative model for dependency parsing. We search for the maximum-likelihood model parameters and corpus parse, subject to posterior constraints. We show how to formulate this as a mixed integer quadratic programming problem with nonlinear constraints. We use the Reformulation Linearization Technique to produce convex relaxations during branch-and-bound. Although these techniques do not yet provide a practical solution to our instance of this NP-hard problem, they sometimes find better solutions than Viterbi EM with random restarts, in the same time.

1 Introduction

Rich models with latent linguistic variables are popular in computational linguistics, but in general it is not known how to find their optimal parameters. In this paper, we present some "new" attacks for this common optimization setting, drawn from the mathematical programming toolbox.

We focus on the well-studied but unsolved task of unsupervised dependency parsing (i.e., dependency grammar induction). This may be a particularly hard case, but its structure is typical. Many parameter estimation techniques have been attempted, including expectation-maximization (EM) (Klein and Manning, 2004; Spitkovsky et al., 2010a), contrastive estimation (Smith and Eisner, 2006; Smith, 2006), Viterbi EM (Spitkovsky et al., 2010b), and variational EM (Naseem et al., 2010; Cohen et al., 2009; Cohen and Smith, 2009). These are all *local search* techniques, which improve the parameters by hill-climbing.

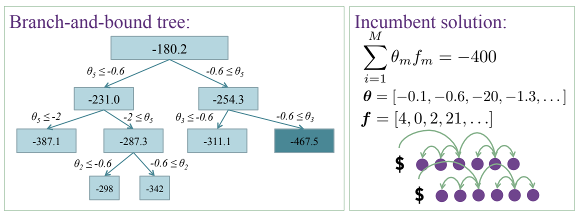


Figure 1: Each node contains a local upper bound for its subspace, computed by a relaxation. The node branches on a single model parameter θ_m to partition its subspace. The lower bound, -400, is given by the best solution seen so far, the incumbent. The upper bound, -298, is the min of all remaining leaf nodes. The node with a local bound of -467.5 can be pruned because no solution within its subspace could be better than the incumbent.

The problem with local search is that it gets stuck in local optima. This is evident for grammar induction. An algorithm such as EM will find numerous different solutions when randomly initialized to different points (Charniak, 1993; Smith, 2006). A variety of ways to find better local optima have been explored, including heuristic initialization of the model parameters (Spitkovsky et al., 2010a), random restarts (Smith, 2006), and annealing (Smith and Eisner, 2006; Smith, 2006). Others have achieved accuracy improvements by enforcing linguistically motivated *posterior constraints* on the parameters (Gillenwater et al., 2010; Naseem et al., 2010), such as requiring most sentences to have verbs or encouraging nouns to be children of verbs or prepositions.

We introduce a method that performs *global*

[*]This research was partially funded by the JHU Human Language Technology Center of Excellence.

search with certificates of ϵ-optimality for both the corpus parse and the model parameters. Our search objective is log-likelihood. We can also impose posterior constraints on the latent structure.

As we show, maximizing the joint log-likelihood of the parses and the parameters can be formulated as a mathematical program (MP) with a nonconvex quadratic objective and with integer linear and nonlinear constraints. Note that this objective is that of hard (Viterbi) EM—we do not marginalize over the parses as in classical EM.[1]

To globally optimize the objective function, we employ a branch-and-bound algorithm that searches the continuous space of the model parameters by branching on individual parameters (see Figure 1). Thus, our branch-and-bound tree serves to recursively subdivide the global parameter hypercube. Each node represents a search problem over one of the resulting boxes (i.e., orthotopes).

The crucial step is to prune nodes high in the tree by determining that their boxes *cannot* contain the global maximum. We compute an upper bound at each node by solving a relaxed maximization problem tailored to its box. If this upper bound is worse than our current best solution, we can prune the node. If not, we split the box again via another branching decision and retry on the two halves.

At each node, our relaxation derives a linear programming problem (LP) that can be efficiently solved by the dual simplex method. First, we linearly relax the constraints that grammar rule probabilities sum to 1—these constraints are nonlinear in our parameters, which are *log*-probabilities. Second, we linearize the quadratic objective by applying the Reformulation Linearization Technique (RLT) (Sherali and Adams, 1990), a method of forming tight linear relaxations of various types of MPs: the *reformulation* step multiplies together pairs of the original linear constraints to generate new quadratic constraints, and then the *linearization* step replaces quadratic terms in the new constraints with auxiliary variables.

Finally, if the node is not pruned, we search for a better incumbent solution under that node by projecting the solution of the RLT relaxation back onto the feasible region. In the relaxation, the model parameters might sum to slightly more than one and the parses can consist of fractional dependency edges. We project in order to compute the true objective and compare with other solutions.

Our results demonstrate that our method can obtain higher likelihoods than Viterbi EM with random restarts. Furthermore, we show how posterior constraints inspired by Gillenwater et al. (2010) and Naseem et al. (2010) can easily be applied in our framework to obtain competitive accuracies using a simple model, the Dependency Model with Valence (Klein and Manning, 2004). We also obtain an ϵ-optimal solution on a toy dataset.

We caution that the linear relaxations are very loose on larger boxes. Since we have many dimensions, the binary branch-and-bound tree may have to grow quite deep before the boxes become small enough to prune. This is why nonconvex quadratic optimization by LP-based branch-and-bound usually fails with more than 80 variables (Burer and Vandenbussche, 2009). Even our smallest (toy) problems have hundreds of variables, so our experimental results mainly just illuminate the method's behavior. Nonetheless, we offer the method as a new tool which, just as for local search, might be combined with other forms of problem-specific guidance to produce more practical results.

2 The Constrained Optimization Task

We begin by describing how for our typical model, the Viterbi EM objective can be formulated as a **mixed integer quadratic programming** (MIQP) problem with **nonlinear constraints** (Figure 2).

Other locally normalized log-linear generative models (Berg-Kirkpatrick et al., 2010) would have a similar formulation. In such models, the log-likelihood objective is simply a linear function of the feature counts. However, the objective becomes **quadratic** in unsupervised learning, because the feature counts are themselves unknown variables to be optimized. The feature counts are constrained to be derived from the latent variables (e.g., parses), which are unknown discrete structures that must be encoded with **integer** variables. The **nonlinear** constraints ensure that the model parameters are true log-probabilities.

Concretely, (1) specifies the Viterbi EM objective: the total log-probability of the *best* parse trees under the parameters $\boldsymbol{\theta}$, given by a sum of log-probabilities θ_m of the individual steps needed to generate the tree, as encoded by the features f_m. The (nonlinear) sum-to-one constraints on the

[1]This objective might not be a great sacrifice: Spitkovsky et al. (2010b) present evidence that hard EM can outperform soft EM for grammar induction in a hill-climbing setting. We use it because it is a quadratic objective. However, maximizing it remains NP-hard (Cohen and Smith, 2010).

Variables:

θ_m	Log-probability for feature m
f_m	Corpus-wide feature count for m
e_{sij}	Indicator of an arc from i to j in tree s

Indices and constants:

m	Feature / model parameter index
s	Sentence index
c	Conditional distribution index
M	Number of model parameters
C	Number of conditional distributions
$\mathcal{M}_c$	c^{th} Set of feature indices that sum to 1.0
S	Number of sentences
N_s	Number of words in the s^{th} sentence

Objective and constraints:

$$\max \sum_m \theta_m f_m \quad (1)$$

$$\text{s.t.} \sum_{m \in \mathcal{M}_c} \exp(\theta_m) = 1, \forall c \quad (2)$$

$$A \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{e} \end{bmatrix} \leq b \qquad \textit{(Model constraints)} \quad (3)$$

$$\theta_m \leq 0, \quad f_m, e_{sij} \in \mathbb{Z}, \forall m, s, i, j \quad (4)$$

Figure 2: Viterbi EM as a mathematical program

probabilities are in (2). The linear constraints in (3) will ensure that the arc variables for each sentence $\boldsymbol{e}_s$ encode a valid latent dependency tree, and that the $\boldsymbol{f}$ variables count up the features of these trees. The final constraints (4) simply specify the range of possible values for the model parameters and their integer count variables.

Our experiments use the dependency model with valence (DMV) (Klein and Manning, 2004). This generative model defines a joint distribution over the sentences and their dependency trees.

We encode the DMV using integer linear constraints on the arc variables $\boldsymbol{e}$ and feature counts $\boldsymbol{f}$. These will constitute the model constraints in (3). The constraints must declaratively specify that the arcs form a valid dependency tree and that the resulting feature values are as defined by the DMV.

Tree Constraints To ensure that our arc variables, $\boldsymbol{e}_s$, form a dependency tree, we employ the same single-commodity flow constraints of Magnanti and Wolsey (1994) as adapted by Martins et al. (2009) for parsing. We also use the projectivity constraints of Martins et al. (2009).

The single-commodity flow constraints simultaneously enforce that each node has exactly one parent, the special root node (position 0) has no incoming arcs, and the arcs form a connected graph.

For each sentence, s, the variable ϕ_{sij} indicates the amount of flow traversing the arc from i to j in sentence s. The constraints below specify that the root node emits N_s units of flow (5), that one unit of flow is consumed by each each node (6), that the flow is zero on each disabled arc (7), and that the arcs are binary variables (8).

Single-commodity flow (Magnanti & Wolsey, 1994)

$$\sum_{j=1}^{N_s} \phi_{s0j} = N_s, \forall j \quad (5)$$

$$\sum_{i=0}^{N_s} \phi_{sij} - \sum_{k=1}^{N_s} \phi_{sjk} = 1, \forall j \quad (6)$$

$$\phi_{sij} \leq N_s e_{sij}, \forall i, j \quad (7)$$

$$e_{sij} \in \{0, 1\}, \forall i, j \quad (8)$$

Projectivity is enforced by adding a constraint (9) for each arc ensuring that no edges will cross that arc if it is enabled. $\mathcal{X}_{ij}$ is the set of arcs (k, l) that cross the arc (i, j).

Projectivity (Martins et al., 2009)

$$\sum_{(k,l) \in \mathcal{X}_{ij}} e_{skl} \leq N_s(1 - e_{sij}), \forall s, i, j \quad (9)$$

DMV Feature Counts The DMV generates a dependency tree recursively as follows. First the head word of the sentence is generated, $t \sim \text{Discrete}(\boldsymbol{\theta}_{\text{root}})$, where $\boldsymbol{\theta}_{\text{root}}$ is a subvector of $\boldsymbol{\theta}$. To generate its children on the **left** side, we flip a coin to decide whether an adjacent child is generated, $d \sim \text{Bernoulli}(\boldsymbol{\theta}_{\text{dec.L.0},t})$. If the coin flip d comes up `continue`, we sample the word of that child as $t' \sim \text{Discrete}(\boldsymbol{\theta}_{\text{child.L},t})$. We continue generating non-adjacent children in this way, using coin weights $\boldsymbol{\theta}_{\text{dec.L.}\geq 1,t}$ until the coin comes up `stop`. We repeat this procedure to generate children on the **right** side, using the model parameters $\boldsymbol{\theta}_{\text{dec.R.0},t}$, $\boldsymbol{\theta}_{\text{child.R},t}$, and $\boldsymbol{\theta}_{\text{dec.R.}\geq 1,t}$. For each new child, we apply this process recursively to generate its descendants.

The feature count variables for the DMV are encoded in our MP as various sums over the edge variables. We begin with the **root/child feature counts**. The constraint (10) defines the feature count for model parameter $\theta_{\text{root},t}$ as the number of all enabled arcs connecting the root node to a word of type t, summing over all sentences s. The constraint in (11) similarly defines $f_{\text{child.L},t,t'}$ to be the number of enabled arcs connecting a parent of type t to a left child of type t'. $\mathcal{W}_{st}$ is the index set of tokens in sentences s with word type t.

DMV root/child feature counts

$$f_{\text{root},t} = \sum_{s=1}^{N_s} \sum_{j \in \mathcal{W}_{st}} e_{s0j}, \forall t \tag{10}$$

$$f_{\text{child.L},t,t'} = \sum_{s=1}^{N_s} \sum_{j<i} \delta \begin{bmatrix} i \in \mathcal{W}_{st} \wedge \\ j \in \mathcal{W}_{st'} \end{bmatrix} e_{sij}, \forall t, t' \tag{11}$$

The **decision feature counts** require the addition of an auxiliary count variables $f_m^{(si)} \in \mathcal{Z}$ indicating how many times decision feature m fired at some position in the corpus s, i. We then need only add a constraint that the corpus wide feature count is the sum of these token-level feature counts $f_m = \sum_{s=1}^{S} \sum_{i=1}^{N_s} f_m^{(si)}, \forall m$.

Below we define these auxiliary variables for $1 \leq s \leq S$ and $1 \leq i \leq N_s$. The helper variable $n_{s,i,l}$ counts the number of enabled arcs to the left of token i in sentence s. Let t denote the word type of token i in sentence s. Constraints (11) - (16) are defined analogously for the *right side* feature counts.

DMV decision feature counts

$$n_{s,i,l} = \sum_{j=1}^{i-1} e_{sij} \tag{12}$$

$$n_{s,i,l}/N_s \leq f^{(s,i)}_{\text{dec.L.0},t,\texttt{cont}} \leq 1 \tag{13}$$

$$f^{(s,i)}_{\text{dec.L.0},t,\texttt{stop}} = 1 - f^{(s,i)}_{\text{dec.L.0},t,\texttt{cont}} \tag{14}$$

$$f^{(s,i)}_{\text{dec.L.}\geq 1,t,\texttt{stop}} = f^{(s,i)}_{\text{dec.L.0},t,\texttt{cont}} \tag{15}$$

$$f^{(s,i)}_{\text{dec.L.}\geq 1,t,\texttt{cont}} = n_{s,i,l} - f^{(s,i)}_{\text{dec.L.0},t,\texttt{cont}} \tag{16}$$

3 A Branch-and-Bound Algorithm

The mixed integer quadratic program with nonlinear constraints, given in the previous section, maximizes the nonconvex Viterbi EM objective and is NP-hard to solve (Cohen and Smith, 2010). The standard approach to optimizing this program is local search by the hard (Viterbi) EM algorithm. Yet local search can only provide a lower (pessimistic) bound on the global maximum.

We propose a branch-and-bound algorithm, which will iteratively tighten both pessimistic and optimistic bounds on the optimal solution. This algorithm may be halted at any time, to obtain the best current solution and a bound on how much better the global optimum could be.

A *feasible solution* is an assignment to all the variables—both model parameters and corpus parse—that satisfies all constraints. Our branch-and-bound algorithm maintains an *incumbent solution*: the best known feasible solution according to the objective function. This is updated as better feasible solutions are found.

Our algorithm implicitly defines a search tree in which each node corresponds to a region of model parameter space. Our search procedure begins with only the root node, which represents the full model parameter space. At each node we perform three steps: bounding, projecting, and branching.

In the **bounding** step, we solve a relaxation of the original problem to provide an upper bound on the objective achievable within that node's subregion. A node is pruned when $L_{\text{global}} + \epsilon|L_{\text{global}}| \geq U_{\text{local}}$, where L_{global} is the incumbent score, U_{local} is the upper bound for the node, and $\epsilon > 0$. This ensures that its entire subregion will not yield a ϵ-better solution than the current incumbent.

The overall optimistic bound is given by the worst optimistic bound of all current leaf nodes.

The **projecting** step, if the node is not pruned, projects the solution of the relaxation back to the feasible region, replacing the current incumbent if this projection provides a better lower bound.

In the **branching** step, we choose a variable θ_m on which to divide. Each of the child nodes receives a lower θ_m^{min} and upper θ_m^{max} bound for θ_m. The child subspaces partition the parent subspace.

The search tree is defined by a variable ordering and the splitting procedure. We do binary branching on the variable θ_m with the highest regret, defined as $z_m - \theta_m f_m$, where z_m is the auxiliary objective variable we will introduce in § 4.2. Since θ_m is a log-probability, we split its current range at the midpoint in probability space, $\log((\exp \theta_m^{\text{min}} + \exp \theta_m^{\text{max}})/2)$.

We perform best-first search, ordering the nodes by the the optimistic bound of their parent. We also use the LP-guided rule (Martin, 2000; Achterberg, 2007, section 6.1) to perform depth-first plunges in search of better incumbents.

4 Relaxations

The relaxation in the bounding step computes an optimistic bound for a subspace of the model parameters. This upper bound would ideally be not much greater than the true maximum achievable on that region, but looser upper bounds are generally faster to compute.

We present successive relaxations to the original nonconvex mixed integer quadratic program with nonlinear constraints from (1)–(4). First, we show how the nonlinear sum-to-one constraints can be relaxed into linear constraints and tightened. Second, we apply a classic approach to bound the nonconvex quadratic objective by a linear concave envelope. Finally, we present our full relaxation based on the Reformulation Linearization Technique (RLT) (Sherali and Adams, 1990). We solve these LPs by the dual simplex algorithm.

4.1 Relaxing the sum-to-one constraint

In this section, we use cutting planes to create a linear relaxation for the sum-to-one constraint (2). When relaxing a constraint, we must ensure that any assignment of the variables that was feasible (i.e. respected the constraints) in the original problem must also be feasible in the relaxation. In most cases, the relaxation is not perfectly tight and so will have an enlarged space of feasible solutions.

We begin by weakening constraint (2) to

$$\sum_{m \in \mathcal{M}_c} \exp(\theta_m) \leq 1 \tag{17}$$

The optimal solution under (17) still satisfies the original equality constraint (2) because of the maximization. We now relax (17) by approximating the surface $z = \sum_{m \in \mathcal{M}_c} \exp(\theta_m)$ by the max of N lower-bounding linear functions on $\mathbb{R}^{|\mathcal{M}_c|}$. Instead of requiring $z \leq 1$, we only require each of these lower bounds to be ≤ 1, slightly enlarging the feasible space into a convex polytope. Figure 3a shows the feasible region constructed from N=3 linear functions on two log-probabilities θ_1, θ_2.

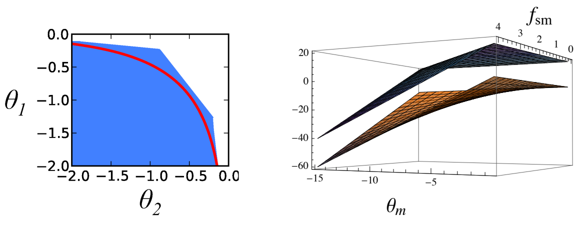
Formally, for each c, we define the i^{th} linear lower bound ($i = 1, \dots, N$) to be the tangent hyperplane at some point $\hat{\boldsymbol{\theta}}_c^{(i)} = [\hat{\theta}_{c,1}^{(i)}, \dots, \hat{\theta}_{c,|\mathcal{M}_c|}^{(i)}] \in \mathbb{R}^{|\mathcal{M}_c|}$, where each coordinate is a log-probability $\hat{\theta}_{c,m}^{(i)} < 0$. We require each of these linear functions to be ≤ 1:

Sum-to-one Relaxation

$$\sum_{m \in \mathcal{M}_c} \left(\theta_m + 1 - \hat{\theta}_{c,m}^{(i)}\right) \exp\left(\hat{\theta}_{c,m}^{(i)}\right) \leq 1, \; \forall i, \; \forall c \tag{18}$$

4.2 "Relaxing" the objective

Our true maximization objective $\sum_m \theta_m f_m$ in (1) is a sum of quadratic terms. If the parameters $\boldsymbol{\theta}$ were fixed, the objective would become linear in the latent features. Although the parameters are *not* fixed, the branch-and-bound algorithm does box them into a small region, where the quadratic objective is "more linear."



(a) (b)

Figure 3: In (a), the area under the curve corresponds to those points (θ_1, θ_2) that satisfy (17) ($z \leq 1$), with equality (2) achieved along the curve ($z = 1$). The shaded area shows the enlarged feasible region under the linear relaxation. In (b), the curved lower surface represents a single product term in the objective. The piecewise-linear upper surface is its concave envelope (raised by 20 for illustration; in reality they touch).

Since it is easy to maximize a concave function, we will maximize the *concave envelope*—the concave function that most tightly upper-bounds our objective over the region. This turns out to be *piecewise linear* and can be maximized with an LP solver. Smaller regions yield tighter bounds.

Each node of the branch-and-bound tree specifies a region via bounds constraints $\theta_m^{\min} < \theta_m < \theta_m^{\max}$, $\forall m$. In addition, we have known bounds $f_m^{\min} \leq f_m \leq f_m^{\max}$, $\forall m$ for the count variables.

McCormick (1976) described the concave envelope for a *single* quadratic term subject to bounds constraints (Figure 3b). In our case:

$$\theta_m f_m \leq \min[f_m^{\max}\theta_m + \theta_m^{\min} f_m - \theta_m^{\min} f_m^{\max}, \\ f_m^{\min}\theta_m + \theta_m^{\max} f_m - \theta_m^{\max} f_m^{\min}]$$

We replace our objective $\sum_m \theta_m f_m$ with $\sum_m z_m$, where we would like to constrain each auxiliary variable z_m to be $= \theta_m f_m$ or (equivalently) $\leq \theta_m f_m$, but instead settle for making it $\leq$ the concave envelope—a linear programming problem:

Concave Envelope Objective

$$\max \sum_m z_m \tag{19}$$

$$\text{s.t. } z_m \leq f_m^{\max}\theta_m + \theta_m^{\min} f_m - \theta_m^{\min} f_m^{\max} \tag{20}$$

$$z_m \leq f_m^{\min}\theta_m + \theta_m^{\max} f_m - \theta_m^{\max} f_m^{\min} \tag{21}$$

4.3 Reformulation Linearization Technique

The *Reformulation Linearization Technique* (RLT)[2] (Sherali and Adams, 1990) is a method of forming tighter relaxations of various types of MPs. The basic method reformulates the problem by adding products of existing constraints. The quadratic terms in the objective and in these new constraints are redefined as auxiliary variables, thereby linearizing the program.

In this section, we will show how the RLT can be applied to our grammar induction problem and contrast it with the concave envelope relaxation presented in section 4.2.

Consider the original MP in equations (1) - (4), with the nonlinear sum-to-one constraints in (2) replaced by our linear constraints proposed in (18). If we remove the integer constraints in (4), the result is a quadratic program with *purely linear constraints*. Such problems have the form

$$\max\ x^T Q x \tag{22}$$

$$\text{s.t. } Ax \le b \tag{23}$$

$$-\infty < L_i \le x_i \le U_i < \infty, \forall i \tag{24}$$

where the variables are $x \in \mathbb{R}^n$, A is an $m \times n$ matrix, and $b \in \mathbb{R}^m$, and Q is an $n \times n$ indefinite[3] matrix. Without loss of generality we assume Q is symmetric. The application of the RLT here was first considered by Sherali and Tuncbilek (1995).

For convenience of presentation, we represent both the linear inequality constraints and the bounds constraints, under a different parameterization using the matrix G and vector g.

$$\begin{bmatrix} (b_i - A_i x) \ge 0, & 1 \le i \le m \\ (U_k - x_k) \ge 0, & 1 \le k \le n \\ (-L_k + x_k) \ge 0, & 1 \le k \le n \end{bmatrix} \equiv \begin{bmatrix} (g_i - G_i x) \ge 0, \\ 1 \le i \le m + 2n \end{bmatrix}$$

The *reformulation* step forms all possible products of these linear constraints and then adds them to the original quadratic program.

$$(g_i - G_i x)(g_j - G_j x) \ge 0, \forall 1 \le i \le j \le m + 2n$$

In the *linearization* step, we replace all quadratic terms in the quadratic objective and new quadratic constraints with auxiliary variables:

$$w_{ij} \equiv x_i x_j, \forall 1 \le i \le j \le n$$

[2]The key idea underlying the RLT was originally introduced in Adams and Sherali (1986) for 0-1 quadratic programming. It has since been extended to various other settings; see Sherali and Liberti (2008) for a complete survey.

[3]In the general case, that Q is indefinite causes this program to be nonconvex, making this problem NP-hard to solve (Vavasis, 1991; Pardalos, 1991).

This yields the following RLT relaxation:

RLT Relaxation

$$\max \sum_{1 \le i \le j \le n} Q_{ij} w_{ij} \tag{25}$$

$$\text{s.t. } g_i g_j - \sum_{k=1}^{n} g_j G_{ik} x_k - \sum_{k=1}^{n} g_i G_{jk} x_k + \sum_{k=1}^{n} \sum_{l=1}^{n} G_{ik} G_{jl} w_{kl} \ge 0, \quad \forall 1 \le i \le j \le m + 2n \tag{26}$$

Notice above that we have omitted the original inequality constraints (23) and bounds (24), because they are fully enforced by the new RLT constraints (26) from the reformulation step (Sherali and Tuncbilek, 1995). In our experiments, we keep the original constraints and instead explore subsets of the RLT constraints.

If the original QP contains equality constraints of the form $G_e x = g_e$, then we can form constraints by multiplying this one by each variable x_i. This gives us the following new set of constraints, for each equality constraint e: $g_e x_i + \sum_{j=1}^{n} -G_{ej} w_{ij} = 0, \forall 1 \le i \le n$.

Theoretical Properties The new constraints in eq. (26) will impose the concave envelope constraints (20)–(21) (Anstreicher, 2009).

The constraints presented above are considered to be *first-level* constraints corresponding to the first-level variables w_{ij}. However, the same technique can be applied repeatedly to produce polynomial constraints of higher degree. These higher level constraints/variables have been shown to provide increasingly tighter relaxations (Sherali and Adams, 1990) at the cost of a large number of variables and constraints. In the case where $x \in \{0, 1\}^n$ the degree-n RLT constraints will restrict to the convex hull of the feasible solutions (Sherali and Adams, 1990).

This is in direct contrast to the concave envelope relaxation presented in section 4.2 which relaxes to the convex hull of each quadratic term independently. This demonstrates the key intuition of the RLT relaxation: The products of constraints are implied (and unnecessary) in the original variable space. Yet when we project to a higher-dimentional space by including the auxiliary variables, the linearized constraints cut off portions of the feasible region given by only the concave envelope relaxation in eqs. (20)-(21) .

4.4 Adding Posterior Constraints

It is a simple extension to impose posterior constraints within our framework. Here we emphasize constraints that are analogous to the universal linguistic constraints from Naseem et al. (2010). Since we optimize the Viterbi EM objective, we directly constrain the counts in the single corpus parse rather than expected counts from a distribution over parses. Let $\mathcal{E}$ be the index set of model parameters corresponding to edge types from Table 1 of Naseem et al. (2010), and N_s be the number of words in the sth sentence. We impose the constraint that 75% of edges come from $\mathcal{E}$: $\sum_{m \in \mathcal{E}} f_m \geq 0.75 \left(\sum_{s=1}^{S} N_s \right)$.

5 Projections

A pessimistic bound, from the projecting step, will correspond to a feasible but not necessarily optimal solution to the original problem. We propose several methods for obtaining pessimistic bounds during the branch-and-bound search, by projecting and improving the solutions found by the relaxation. A solution to the relaxation may be infeasible in the original problem for two reasons: the model parameters might not sum to one, and/or the parse may contain fractional edges.

Model Parameters For each set of model parameters $\mathcal{M}_c$ that should sum-to-one, we project the model parameters onto the $\mathcal{M}_c - 1$ simplex by one of two methods: (1) normalize the infeasible parameters or (2) find the point on the simplex that has minimum Euclidean distance to the infeasible parameters using the algorithm of Chen and Ye (2011). For both methods, we can optionally apply add-λ smoothing before projecting.

Parses Since we are interested in projecting the fractional parse onto the space of projective spanning trees, we can simply employ a dynamic programming parsing algorithm (Eisner and Satta, 1999) where the weight of each edge is given as the fraction of the edge variable.

Only one of these projection techniques is needed. We then either use *parsing* to fill in the optimal parse trees given the projected model parameters, or use *supervised parameter estimation* to fill in the optimal model parameters given the projected parses. These correspond to the Viterbi E step and M step, respectively. We can locally improve the projected solution by continuing with a few additional iterations of Viterbi EM.

Related models could use very similar projection techniques. Given a relaxed joint solution to the parameters and the latent variables, one must be able to project it to a nearby feasible one, by projecting either the fractional parameters or the fractional latent variables into the feasible space and then solving exactly for the other.

6 Related Work

The goal of this work was to better understand and address the non-convexity of maximum-likelihood training with latent variables, especially parses.

Gimpel and Smith (2012) proposed a concave model for unsupervised dependency parsing using IBM Model 1. This model did not include a tree constraint, but instead initialized EM on the DMV. By contrast, our approach incorporates the tree constraints directly into our convex relaxation and embeds the relaxation in a branch-and-bound algorithm capable of solving the original DMV maximum-likelihood estimation problem.

Spectral learning constitutes a wholly different family of consistent estimators, which achieve efficiency because they sidestep maximizing the nonconvex likelihood function. Hsu et al. (2009) introduced a spectral learner for a large class of HMMs. For supervised parsing, spectral learning has been used to learn latent variable PCFGs (Cohen et al., 2012) and hidden-state dependency grammars (Luque et al., 2012). Alas, there are not yet any spectral learning methods that recover latent tree *structure*, as in grammar induction.

Several integer linear programming (ILP) formulations of dependency parsing (Riedel and Clarke, 2006; Martins et al., 2009; Riedel et al., 2012) inspired our definition of grammar induction as a MP. Recent work uses branch-and-bound for decoding with non-local features (Qian and Liu, 2013). These differ from our work by treating the model parameters as constants, thereby yielding a linear objective.

For semi-supervised dependency parsing, Wang et al. (2008) used a convex objective, combining unsupervised least squares loss and a supervised large margin loss, This does not apply to our unsupervised setting. Branch-and-bound has also been applied to semi-supervised SVM training, a nonconvex search problem (Chapelle et al., 2007), with a relaxation derived from the dual.

7 Experiments

We first analyze the behavior of our method on a toy synthetic dataset. Next, we compare various parameter settings for branch-and-bound by estimating the total solution time. Finally, we compare our search method to Viterbi EM on a small subset of the Penn Treebank.

All our experiments use the DMV for unsupervised dependency parsing of part-of-speech (POS) tag sequences. For Viterbi EM we initialize the parameters of the model uniformly, breaking parser ties randomly in the first E-step (Spitkovsky et al., 2010b). This initializer is state-of-the-art for Viterbi EM. We also apply add-one smoothing during each M-step. We use random restarts, and select the model with the highest likelihood.

We add posterior constraints to Viterbi EM's E-step. First, we run a relaxed linear programming (LP) parser, then project the (possibly fractional) parses back to the feasible region. If the resulting parse does not respect the posterior constraints, we discard it. The posterior constraint in the LP parser is tighter[4] than the one used in the true optimization problem, so the projections tends to be feasible under the true (looser) posterior constraints. In our experiments, all but one projection respected the constraints. We solve all LPs with CPLEX.

7.1 Synthetic Data

For our toy example, we generate sentences from a synthetic DMV over three POS tags (Verb, Noun, Adjective) with parameters chosen to favor short sentences with English word order.

In Figure 4 we show that the quality of the root relaxation increases as we approach the full set of RLT constraints. That the number of possible RLT constraints increases quadratically with the length of the corpus poses a serious challenge. For just 20 sentences from this synthetic model, the RLT generates 4,056,498 constraints.

For a single run of branch-and-bound, Figure 5 shows the global upper and lower bounds over time.[5] We consider five relaxations, each using only a subset of the RLT constraints. *Max.0k* uses only the concave envelope (20)-(21). *Max.1k* uses the concave envelope and also randomly samples 1,000 other RLT constraints, and so on for *Max.10k* and *Max.100k*. *Obj.Filter* includes all constraints with a nonzero coefficient for one of the RLT variables z_m from the linearized objective. The rightmost lines correspond to *RLT Max.10k*: despite providing the tightest (local) bound at each node, it processed only 110 nodes in the time it took *RLT Max.1k* to process 1164. *RLT Max.0k* achieves the best balance of tight bounds and speed per node.

[4]80% of edges must come from $\mathcal{E}$ as opposed to 75%.

[5]The initial incumbent solution for branch-and-bound is obtained by running Viterbi EM with 10 random restarts.

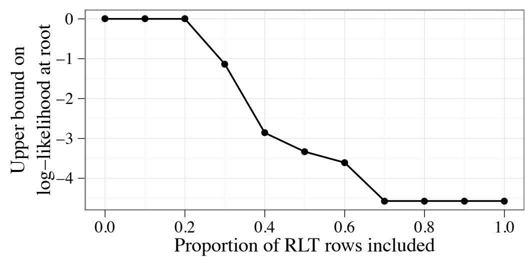


Figure 4: The bound quality at the root improves as the proportion of RLT constraints increases, on 5 synthetic sentences. A random subset of 70% of the 320,126 possible RLT constraints matches the relaxation quality of the full set. This bound is very tight: the relaxations in Figure 5 solve hundreds of nodes before such a bound is achieved.

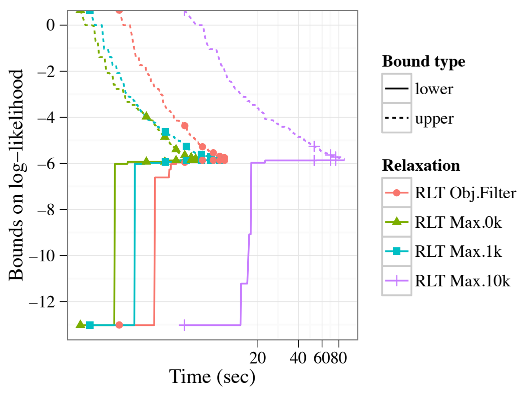


Figure 5: The global upper and lower bounds improve over time for branch-and-bound using different subsets of RLT constraints on 5 synthetic sentences. Each solves the problem to ϵ-optimality for $\epsilon = 0.01$. A point marks every 200 nodes processed. (The time axis is log-scaled.)

7.2 Comparing branch-and-bound strategies

It is prohibitively expensive to repeatedly run our algorithm to completion with a variety of parameter settings. Instead, we estimate the size of the branch-and-bound tree and the solution time using a high-variance estimate that is effective for comparisons (Lobjois and Lemaître, 1998).

Given a fixed set of parameters for our algorithm and an ϵ-optimality stopping criterion, we

RLT Relaxation	*Avg. ms per node*	*# Samples*	*Est. # Nodes*	*Est. # Hours*
Obj.Filter	63	10000	3.2E+08	4.6E+09
Max.0k	6	10000	1.7E+10	7.8E+10
Max.1k	15	10000	3.5E+08	4.2E+09
Max.10k	161	10000	1.3E+09	3.4E+10
Max.100k	232259	5	1.7E+09	9.7E+13

Table 1: Branch-and-bound node count and completion time estimates. Each standard deviation was close in magnitude to the estimate itself. We ran for 8 hours, stopping at 10,000 samples on 8 synthetic sentences.

can view the branch-and-bound tree T as fixed and finite in size. We wish to estimate some cost associated with the tree $C(T) = \sum_{\alpha \in \text{nodes}(T)} f(\alpha)$. Letting $f(\alpha) = 1$ estimates the number of nodes; if $f(\alpha)$ is the time to solve a node, then we estimate the total solution time using the Monte Carlo method of Knuth (1975). Table 1 gives these estimates, for the same five RLT relaxations. *Obj.Filter* yields the smallest estimated tree size.

7.3 Real Data

In this section, we compare our global search method to Viterbi EM with random restarts each with or without posterior constraints. We use 200 sentences of no more than 10 tokens from the WSJ portion of the Penn Treebank. We reduce the treebank's gold part-of-speech (POS) tags to a universal set of 12 tags (Petrov et al., 2012) plus a tag for auxiliaries, ignoring punctuation. Each search method is run for 8 hours. We obtain the initial incumbent solution for branch-and-bound by running Viterbi EM for 45 minutes. The average time to solve a node's relaxation ranges from 3 seconds for *RLT Max.0k* to 42 seconds for *RLT Max.100k*.

Figure 6a shows the log-likelihood of the incumbent solution over time. In our global search method, like Viterbi EM, the posterior constraints lead to lower log-likelihoods. *RLT Max.0k* finds the highest log-likelihood solution.

Figure 6b compares the *unlabeled directed dependency accuracy* of the incumbent solution. In both global and local search, the posterior constraints lead to higher accuracies. Viterbi EM with posterior constraints demonstrates the oscillation of incumbent accuracy: starting at 58.02% accuracy, it finds several high accuracy solutions early on (61.02%), but quickly abandons them to increase likelihood, yielding a final accuracy of 60.65%. *RLT Max.0k* with posterior constraints obtains the highest overall accuracy of 61.09% at

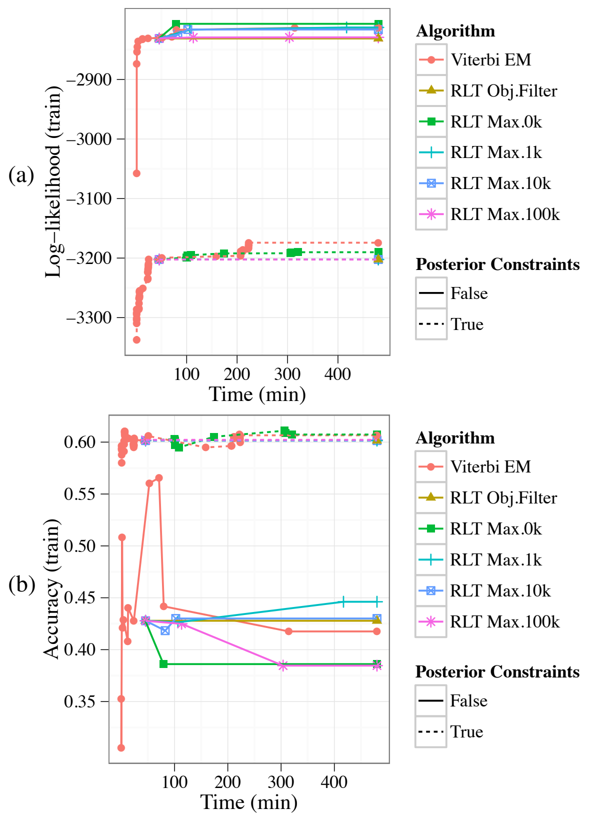


Figure 6: Likelihood (a) and accuracy (b) of incumbent solution so far, on a small real dataset.

306 min and the highest final accuracy 60.73%.

8 Discussion

In principle, our branch-and-bound method can approach ϵ-optimal solutions to Viterbi training of locally normalized generative models, including the NP-hard case of grammar induction with the DMV. The method can also be used with posterior constraints or a regularized objective.

Future work includes algorithmic improvements for solving the relaxation and the development of tighter relaxations. The Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) or Lagrangian Relaxation (Held and Karp, 1970) might satisfy both of these goals by pushing the integer tree constraints into a subproblem solved by a dynamic programming parser. Recent work on semidefinite relaxations (Anstreicher, 2009) suggests they may provide tighter bounds at the expense of greater computation time.

Perhaps even more important than tightening the bounds at each node are search heuristics (e.g., surface cues) and priors (e.g., universal grammar) that guide our global search by deciding *which* node to expand next (Chomsky and Lasnik, 1993).

References

Tobias Achterberg. 2007. *Constraint integer programming*. Ph.D. thesis, TU Berlin.

Warren P. Adams and Hanif D. Sherali. 1986. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, October. ArticleType: research-article / Full publication date: Oct., 1986 / Copyright 1986 INFORMS.

Kurt Anstreicher. 2009. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2):471–484.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, DeNero, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL*, June.

Samuel Burer and Dieter Vandenbussche. 2009. Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, 43(2):181–195.

Olivier Chapelle, Vikas Sindhwani, and S. Sathiya Keerthi. 2007. Branch and bound for semi-supervised support vector machines. In *Proc. of NIPS 19*, pages 217–224. MIT Press.

E. Charniak. 1993. *Statistical language learning*. MIT press.

Yunmei Chen and Xiaojing Ye. 2011. Projection onto a simplex. *arXiv:1101.6081*, January.

Noam Chomsky and Howard Lasnik. 1993. Principles and parameters theory. In *Syntax: An International Handbook of Contemporary Research*. Berlin: de Gruyter.

Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of HLT-NAACL*, pages 74–82, June.

Shay Cohen and Noah A. Smith. 2010. Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *Proc. of ACL*, pages 1502–1511, July.

S. B. Cohen, K. Gimpel, and N. A. Smith. 2009. Logistic normal priors for unsupervised probabilistic grammar induction. In *Proceedings of NIPS*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proc. of ACL (Volume 1: Long Papers)*, pages 223–231. Association for Computational Linguistics, July.

George B. Dantzig and Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, January.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*, pages 457–464, June.

Jennifer Gillenwater, Kuzman Ganchev, Joo Graa, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199. Association for Computational Linguistics, July.

K. Gimpel and N. A. Smith. 2012. Concavity and initialization for unsupervised dependency parsing. In *Proc. of NAACL*.

M. Held and R. M. Karp. 1970. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162.

D. Hsu, S M Kakade, and T. Zhang. 2009. A spectral algorithm for learning hidden markov models. In *COLT 2009 - The 22nd Conference on Learning Theory*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*, pages 478–485, July.

D. E. Knuth. 1975. Estimating the efficiency of backtrack programs. *Mathematics of computation*, 29(129):121–136.

L. Lobjois and M. Lemaître. 1998. Branch and bound algorithm selection by performance prediction. In *Proc. of the National Conference on Artificial Intelligence*, pages 353–358.

Franco M. Luque, Ariadna Quattoni, Borja Balle, and Xavier Carreras. 2012. Spectral learning for non-deterministic dependency parsing. In *Proc. of EACL*, pages 409–419, April.

Thomas L. Magnanti and Laurence A. Wolsey. 1994. *Optimal Trees*. Center for Operations Research and Econometrics.

Alexander Martin. 2000. Integer programs with block structure. Technical Report SC-99-03, ZIB.

André Martins, Noah A. Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*, pages 342–350, August.

Garth P. McCormick. 1976. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10(1):147–175.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP*, pages 1234–1244, October.

P. M. Pardalos. 1991. Global optimization algorithms for linearly constrained indefinite quadratic problems. *Computers & Mathematics with Applications*, 21(6):87–97.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

Xian Qian and Yang Liu. 2013. Branch and bound algorithm for dependency parsing with non-local features. *TACL*, 1:37—48.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of EMNLP*, pages 129–137, July.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut—Delayed column and row generation for graph based parsers. In *Proc. of EMNLP-CoNLL*, pages 732–743, July.

Hanif D. Sherali and Warren P. Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, August.

H. Sherali and L. Liberti. 2008. Reformulation-linearization technique for global optimization. *Encyclopedia of Optimization*, 2:3263–3268.

Hanif D. Sherali and Cihan H. Tuncbilek. 1995. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 7(1):1–31.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proc. of COLING-ACL*, pages 569–576, July.

N.A. Smith. 2006. *Novel estimation methods for unsupervised discovery of latent structure in natural language text*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010a. From baby steps to leapfrog: How Less is more in unsupervised dependency parsing. In *Proc. of HLT-NAACL*, pages 751–759. Association for Computational Linguistics, June.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010b. Viterbi training improves unsupervised dependency parsing. In *Proc. of CoNLL*, pages 9–17. Association for Computational Linguistics, July.

S. A. Vavasis. 1991. *Nonlinear optimization: complexity issues*. Oxford University Press, Inc.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proc of ACL-HLT*, pages 532–540. Association for Computational Linguistics, June.