An Exact Dual Decomposition Algorithm for Shallow Semantic Parsing with Constraints

Dipanjan Das* **André F. T. Martins*†** **Noah A. Smith***
*Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
†Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{dipanjan,afm,nasmith}@cs.cmu.edu

Abstract

We present a novel technique for jointly predicting semantic arguments for lexical predicates. The task is to find the best matching between semantic roles and sentential spans, subject to structural constraints that come from expert linguistic knowledge (e.g., in the FrameNet lexicon). We formulate this task as an integer linear program (ILP); instead of using an off-the-shelf tool to solve the ILP, we employ a dual decomposition algorithm, which we adapt for exact decoding via a branch-and-bound technique. Compared to a baseline that makes local predictions, we achieve better argument identification scores and avoid all structural violations. Runtime is nine times faster than a proprietary ILP solver.

1 Introduction

Semantic knowledge is often represented declaratively in resources created by linguistic experts. In this work, we strive to exploit such knowledge in a principled, unified, and intuitive way. An example resource where a wide variety of knowledge has been encoded over a long period of time is the FrameNet lexicon (Fillmore et al., 2003),[1] which suggests an analysis based on frame semantics (Fillmore, 1982). This resource defines hundreds of semantic **frames**. Each frame represents a gestalt event or scenario, and is associated with several semantic **roles**, which serve as participants in the event that the frame signifies (see Figure 1 for an example). Along with storing the above data, FrameNet also provides a hierarchy of relationships between frames, and semantic relationships between pairs of roles. In prior NLP research using FrameNet, these interactions have been largely ignored, though they have the potential to improve the quality and consistency of semantic analysis.

In this paper, we present an algorithm that finds the full collection of arguments of a predicate given its semantic frame. Although we work within the conventions of FrameNet, our approach is generalizable to other semantic role labeling (SRL) frameworks. We model this **argument identification** task as constrained optimization, where the constraints come from expert knowledge encoded in a lexicon. Following prior work on PropBank-style SRL (Kingsbury and Palmer, 2002) that dealt with similar constrained problems (Punyakanok et al., 2004; Punyakanok et al., 2008, *inter alia*), we incorporate this declarative knowledge in an integer linear program (ILP).

Because general-purpose ILP solvers are proprietary and do not fully exploit the structure of the problem, we turn to a class of optimization techniques called **dual decomposition** (Komodakis et al., 2007; Rush et al., 2010; Martins et al., 2011a). We derive a modular, extensible, parallelizable approach in which semantic constraints map not just to declarative components of the algorithm, but also to procedural ones, in the form of "workers." While dual decomposition algorithms only solve a relaxation of the original problem, we make a novel contribution by wrapping the algorithm in a branch-and-bound search procedure, resulting in *exact* solutions.

We experimentally find that our algorithm achieves accuracy comparable to a state-of-the-art system, while respecting all imposed linguistic constraints. In comparison to inexact beam search that violates many of these constraints, our exact decoder has less than twice the runtime; furthermore, it decodes nine times faster than CPLEX, a state-of-the-art, proprietary, general-purpose exact ILP solver.

[1] http://framenet.icsi.berkeley.edu

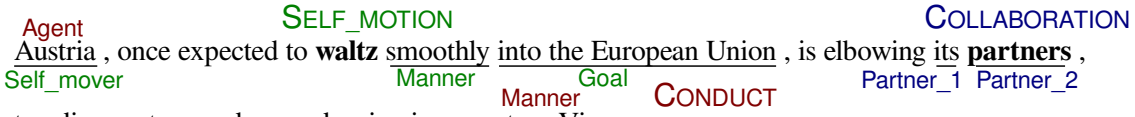

Figure 1: An example sentence from the annotations released as part of FrameNet 1.5 with three predicates marked in bold. Each predicate has its evoked semantic frame marked above it, in a distinct color. For each frame, its semantic roles are shown in the same color, and the spans fulfilling the roles are underlined. For example, **manner** evokes the CONDUCT frame, and has the Agent and Manner roles fulfilled by Austria and most un-Viennese respectively.

2 Collective Argument Identification

Here, we take a declarative approach to modeling argument identification using an ILP and relate our formulation to prior work in shallow semantic parsing. We show how knowledge specified in a linguistic resource can be used to derive the constraints used in our ILP. Finally, we draw connections of our specification to graphical models, a popular formalism in AI, and describe how the constraints can be treated as factors in a factor graph.

2.1 Declarative Specification

Let us denote a predicate by t and the semantic frame it evokes within a sentence $\mathbf{x}$ by f. In this work, we assume that the semantic frame f is given, which is traditionally the case in controlled experiments used to evaluate SRL systems (Màrquez et al., 2008). Given the semantic frame of a predicate, the semantic roles that might be filled are assumed to be given by the lexicon (as in PropBank and FrameNet). Let the set of roles associated with the frame f be $\mathcal{R}_f$. In sentence $\mathbf{x}$, the set of candidate spans of words that might fill each role is enumerated, usually following an overgenerating heuristic;[2] let this set of spans be $\mathcal{S}_t$. We include the null span $\emptyset$ in $\mathcal{S}_t$; connecting it to a role $r \in \mathcal{R}_f$ denotes that the role is not overt. Our approach assumes a scoring function that gives a strength of association between roles and candidate spans. For each role $r \in \mathcal{R}_f$ and span $s \in \mathcal{S}_t$, this score is parameterized as:

$$c(r, s) = \boldsymbol{\psi} \cdot \mathbf{h}(t, f, \mathbf{x}, r, s), \tag{1}$$

where $\boldsymbol{\psi}$ are model weights and $\mathbf{h}$ is a feature function that looks at the predicate t, the evoked frame f, sentence $\mathbf{x}$, and its syntactic analysis, along with r and s. The SRL literature provides many feature functions of this form and many ways to use machine learning to acquire $\boldsymbol{\psi}$. Our presented method does not make any assumptions about the score except that it has the form in Eq. 1.

We define a vector $\mathbf{z}$ of binary variables $z_{r,s} \in \{0, 1\}$ for every role and span pair. We have that: $\mathbf{z} \in \{0, 1\}^d$, where $d = |\mathcal{R}_f| \times |\mathcal{S}_t|$. $z_{r,s} = 1$ means that role r is filled by span s. Given the binary $\mathbf{z}$ vector, it is straightforward to recover the collection of arguments by checking which components $z_{r,s}$ have an assignment of 1; we use this strategy to find arguments, as described in §4.2 (strategies 4 and 6). The joint argument identification task can be represented as a constrained optimization problem:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{S}_t} c(r, s) \times z_{r,s} \\ \text{with respect to} \quad & \mathbf{z} \in \{0, 1\}^d \\ \text{such that} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b}. \end{aligned} \tag{2}$$

The last line imposes constraints on the mapping between roles and spans; these are motivated on linguistic grounds and are described next.[3]

Uniqueness: Each role r is filled by at most one span in $\mathcal{S}_t$. This constraint can be expressed by:

$$\forall r \in \mathcal{R}_f, \textstyle\sum_{s \in \mathcal{S}_t} z_{r,s} = 1. \tag{3}$$

There are $O(|\mathcal{R}_f|)$ such constraints. Note that since $\mathcal{S}_t$ contains the null span $\emptyset$, non-overt roles are also captured using the above constraints. Such a constraint is used extensively in prior literature (Punyakanok et al., 2008, §3.4.1).

Overlap: SRL systems commonly constrain roles to be filled by non-overlapping spans. For example, Toutanova et al. (2005) used dynamic programming over a phrase structure tree to prevent overlaps between arguments, and Punyakanok et al. (2008) used

[2] Here, as in most SRL literature, role fillers are assumed to be expressed as *contiguous* spans, though such an assumption is easy to relax in our framework.

[3] Note that equality constraints $\mathbf{a} \cdot \mathbf{z} = b$ can be transformed into double-side inequalities $\mathbf{a} \cdot \mathbf{z} \leq b$ and $-\mathbf{a} \cdot \mathbf{z} \leq -b$.

constraints in an ILP to respect this requirement. Inspired by the latter, we require that each input sentence position of $\mathbf{x}$ be covered by at most one argument. For each role $r \in \mathcal{R}_f$, we define:

$$\mathcal{G}_r(i) = \{s \mid s \in \mathcal{S}_t, s \text{ covers position } i \text{ in } \mathbf{x}\}. \quad (4)$$

We can define our overlap constraints in terms of $\mathcal{G}_r$ as follows, for every sentence position i:

$$\forall i \in \{1, \ldots, |\mathbf{x}|\}, \quad \sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{G}_r(i)} z_{r,s} \leq 1, \quad (5)$$

This gives us $O(|\mathbf{x}|)$ constraints.

Pairwise "Exclusions": For many predicate classes, there are pairs of roles forbidden to appear together in the analysis of a single predicate token. Consider the following two sentences:

A blackberry (Entity_1) **resembles** a loganberry (Entity_2). (6)

Most berries (Entities) **resemble** each other. (7)

Consider the uninflected predicate **resemble** in both sentences, evoking the same meaning. In example 6, two roles, which we call Entity_1 and Entity_2 describe two entities that are similar to each other. In the second sentence, a phrase fulfills a third role, called Entities, that collectively denotes some objects that are similar. It is clear that the roles Entity_1 and Entities cannot be overt for the same predicate at once, because the latter already captures the function of the former; a similar argument holds for the Entity_2 and Entities roles. We call this phenomenon the "excludes" relationship. Let us define a set of pairs from $\mathcal{R}_f$ that have this relationship:

$$Excl_f = \{(r_i, r_j) \mid r_i \text{ and } r_j \textit{ exclude} \text{ each other}\}$$

Using the above set, we define the constraint:

$$\forall (r_i, r_j) \in Excl_f,\ z_{r_i,\emptyset} + z_{r_j,\emptyset} \geq 1 \quad (8)$$

In English: if both roles are overt in a parse, this constraint will be violated, and we will not respect the "excludes" relationship between the pair. If neither or only one of the roles is overt, the constraint is satisfied. The total number of such constraints is $O(|Excl_f|)$, which is the number of pairwise "excludes" relationships of a given frame.

Pairwise "Requirements": The sentence in example 6 illustrates another kind of constraint. The predicate **resemble** cannot have only one of Entity_1 and Entity_2 as roles in text. For example,

* A blackberry **resembles**. (9)
(Entity_1)

Enforcing the overtness of two roles sharing this "requires" relationship is straightforward. We define the following set for a frame f:

$$Req_f = \{(r_i, r_j) \mid r_i \text{ and } r_j \textit{ require} \text{ each other}\}$$

This leads to constraints of the form

$$\forall (r_i, r_j) \in Req_f, z_{r_i,\emptyset} - z_{r_j,\emptyset} = 0 \quad (10)$$

If one role is overt (or absent), so must the other be. A related constraint has been used previously in the SRL literature, enforcing joint overtness relationships between core arguments and referential arguments (Punyakanok et al., 2008, §3.4.1), which are formally similar to the example above.[4]

Integer Linear Program and Relaxation: Plugging the constraints in Eqs. 3, 5, 8 and 10 into the last line of Eq. 2, we have the argument identification problem expressed as an ILP, since the indicator variables $\mathbf{z}$ are binary. In this paper, apart from the ILP formulation, we will consider the following *relaxation* of Eq. 2, which replaces the binary constraint $\mathbf{z} \in \{0,1\}^d$ by a unit interval constraint $\mathbf{z} \in [0,1]^d$, yielding a *linear* program:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{r \in \mathcal{R}_f} \sum_{s \in \mathcal{S}_t} c(r,s) \times z_{r,s} \\ \text{with respect to} \quad & \mathbf{z} \in [0,1]^d \\ \text{such that} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b}. \end{aligned} \quad (11)$$

There are several LP and ILP solvers available, and a great deal of effort has been spent by the optimization community to devise efficient generic solvers. An example is CPLEX, a state-of-the-art solver for mixed integer programming that we employ as a baseline to solve the ILP in Eq. 2 as well as its LP relaxation in Eq. 11. Like many of the best implementations, CPLEX is proprietary.

[4] We noticed in the annotated data, in some cases, the "requires" constraint is violated by the FrameNet annotators. This happens mostly when one of the required roles is absent in the sentence containing the predicate, but is rather instantiated in an earlier sentence; see Gerber and Chai (2010). We apply the hard constraint in Eq. 10, though extending our algorithm to seek arguments outside the sentence is straightforward (Chen et al., 2010).

2.2 Linguistic Constraints from FrameNet

Although enforcing the four different sets of constraints above is intuitive from a general linguistic perspective, we ground their use in definitive linguistic information present in the FrameNet lexicon (Fillmore et al., 2003). FrameNet, along with lists of semantic frames, associated semantic roles, and predicates that could evoke the frames, gives us a small number of annotated sentences with frame-semantic analysis. From the annotated data, we gathered that only 3.6% of the time is a role instantiated multiple times by different spans in a sentence. This justifies the uniqueness constraint enforced by Eq. 3. Use of such a constraint is also consistent with prior work in frame-semantic parsing (Johansson and Nugues, 2007; Das et al., 2010a). Similarly, we found that in the annotations, no arguments overlapped with each other for a given predicate. Hence, the overlap constraints in Eq. 5 are also justified.

Our third and fourth sets of constraints, presented in Eqs. 8 and 10, come from FrameNet, too; moreover, they are explicitly mentioned in the lexicon. Examples 6–7 are instances where the predicate **resemble** evokes the SIMILARITY frame, which is defined in FrameNet as: "Two or more distinct entities, which may be concrete or abstract objects or types, are characterized as being similar to each other. Depending on figure/ground relations, the entities may be expressed in two distinct frame elements and constituents, Entity_1 and Entity_2, or jointly as a single frame element and constituent, Entities."

For this frame, the lexicon lists several roles other than the three roles we have already observed, such as Dimension (the dimension along which the entities are similar), Differentiating_fact (a fact that reveals how the concerned entities are similar or different), and so forth. Along with the roles, FrameNet also declares the "excludes" and "requires" relationships noted in our discussion in Section 2.1. The case of the SIMILARITY frame is not unique; in Fig. 1, the frame COLLABORATION, evoked by the predicate **partners**, also has two roles Partner_1 and Partner_2 that share the "requires" relationship. In fact, out of 877 frames in FrameNet 1.5, the lexicon's latest edition, 204 frames have at least a pair of roles that share the "excludes" relationship, and 54 list at least a pair of roles that share the "requires" relationship.

2.3 Constraints as Factors in a Graphical Model

The LP in Eq. 11 can be represented as a maximum *a posteriori* (MAP) inference problem in an undirected graphical model. In the factor graph, each component of $\mathbf{z}$ corresponds to a binary variable, and each instantiation of a constraint in Eqs. 3, 5, 8 and 10 corresponds to a factor. Smith and Eisner (2008) and Martins et al. (2010) used such a representation to impose constraints in a dependency parsing problem; the latter discussed the equivalence of linear programs and factor graphs for representing discrete optimization problems. Each of our constraints take standard factor forms we can describe using the terminology of Smith and Eisner (2008) and Martins et al. (2010). The uniqueness constraint in Eq. 3 corresponds to an XOR factor, while the overlap constraint in Eq. 5 corresponds to an ATMOSTONE factor. The constraints in Eq. 8 enforcing the "excludes" relationship can be represented with an OR factor. Finally, each "requires" constraints in Eq. 10 is equivalent to an XORWITHOUTPUT factor.

In the following section, we describe how we arrive at solutions for the LP in Eq. 11 using dual decomposition, and how we adapt it to efficiently recover the *exact* solution of the ILP (Eq. 2), without the need of an off-the-shelf ILP solver.

3 "Augmented" Dual Decomposition

Dual decomposition methods address complex optimization problems in the dual, by dividing them into simple worker problems, which are repeatedly solved until a consensus is reached. The most simple technique relies on the subgradient algorithm (Komodakis et al., 2007; Rush et al., 2010); as an alternative, an augmented Lagrangian technique was proposed by Martins et al. (2011a, 2011b), which is more suitable when there are many small components—commonly the case in declarative constrained problems, such as the one at hand. Here, we present a brief overview of the latter, which is called *Dual Decomposition with the Alternating Direction Method of Multipliers* (AD^3).

Let us start by establishing some notation. Let $m \in \{1, \ldots, M\}$ index a factor, and denote by $\mathbf{i}(m)$

the vector of indices of variables linked to that factor. (Recall that each factor represents the instantiation of a constraint.) We introduce a new set of variables, $\mathbf{u} \in \mathbb{R}^d$, called the "witness" vector. We split the vector $\mathbf{z}$ into M overlapping pieces $\mathbf{z}_1, \ldots, \mathbf{z}_M$, where each $\mathbf{z}_m \in [0,1]^{|\mathbf{i}(m)|}$, and add M constraints $\mathbf{z}_m = \mathbf{u}_{\mathbf{i}(m)}$ to impose that all the pieces must agree with the witness (and therefore with each other). Each of the M constraints described in §2 can be encoded with its own matrix $\mathbf{A}_m$ and vector $\mathbf{b}_m$ (which jointly define $\mathbf{A}$ and $\mathbf{b}$ in Eq. 11). For convenience, we denote by $\mathbf{c} \in \mathbb{R}^d$ the score vector, whose components are $c(r,s)$, for each $r \in \mathcal{R}_f$ and $s \in \mathcal{S}_t$ (Eq. 1), and define the following scores for the mth subproblem:

$$c_m(r,s) = \delta(r,s)^{-1} c(r,s), \ \forall (r,s) \in \mathbf{i}(m), \quad (12)$$

where $\delta(r,s)$ is the number of constraints that involve role r and span s. Note that according to this definition, $\mathbf{c} \cdot \mathbf{z} = \sum_{m=1}^{M} \mathbf{c}_m \cdot \mathbf{z}_m$. We can rewrite the LP in Eq. 11 in the following equivalent form:

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{M} \mathbf{c}_m \cdot \mathbf{z}_m \\ \text{with respect to} \quad & \mathbf{u} \in \mathbb{R}^d, \ \mathbf{z}_m \in [0,1]^{\mathbf{i}(m)}, \quad \forall m \\ \text{such that} \quad & \mathbf{A}_m \mathbf{z}_m \leq \mathbf{b}_m, \quad \forall m \\ & \mathbf{z}_m = \mathbf{u}_{\mathbf{i}(m)}, \quad \forall m. \end{aligned} \quad (13)$$

We next augment the objective with a quadratic penalty term $\frac{\rho}{2} \sum_{m=1}^{M} \|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}\|^2$ (for some $\rho > 0$). This does not affect the solution of the problem, since the equality constraints in the last line force this penalty to vanish. However, as we will see, this penalty will influence the workers and will lead to faster consensus. Next, we introduce Lagrange multipliers $\boldsymbol{\lambda}_m$ for those equality constraints, so that the augmented Lagrangian function becomes:

$$\begin{aligned} L_\rho(\mathbf{z}, \mathbf{u}, \boldsymbol{\lambda}) \quad = \quad & \sum_{m=1}^{M} (\mathbf{c}_m + \boldsymbol{\lambda}_m) \cdot \mathbf{z}_m - \boldsymbol{\lambda}_m \cdot \mathbf{u}_{\mathbf{i}(m)} \\ & - \frac{\rho}{2} \|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}\|^2. \end{aligned} \quad (14)$$

The AD^3 algorithm seeks a saddle point of L_ρ by performing alternating maximization with respect to $\mathbf{z}$ and $\mathbf{u}$, followed by a gradient update of $\boldsymbol{\lambda}$. The result is shown as Algorithm 1. Like dual decomposition approaches, it repeatedly performs a *broadcast* operation (the $\mathbf{z}_m$-updates, which can be done in parallel, one constraint per "worker") and a *gather* operation (the $\mathbf{u}$- and $\boldsymbol{\lambda}$-updates). Each $\mathbf{u}$-operation can be seen as an averaged voting which takes into consideration each worker's results.

Algorithm 1 AD^3 for Argument Identification

input:
- role-span matching scores $\mathbf{c} := \langle c(r,s) \rangle_{r,s}$,
- structural constraints $\langle \mathbf{A}_m, \mathbf{b}_m \rangle_{m=1}^{M}$,
- penalty $\rho > 0$

initialize $\mathbf{u}$ uniformly (*i.e.*, $u(r,s) = 0.5, \ \forall r,s$)
initialize each $\boldsymbol{\lambda}_m = \mathbf{0}, \ \forall m \in \{1, \ldots, M\}$
initialize $t \leftarrow 1$
repeat
  for each $m = 1, \ldots, M$ **do**
    make a $\mathbf{z}_m$-update by finding the best scoring analysis for the mth constraint, with penalties for deviating from the consensus $\mathbf{u}$:

$$\mathbf{z}_m^{t+1} \leftarrow \operatorname*{arg\,max}_{\mathbf{A}_m \mathbf{z}_m \leq \mathbf{b}_m} (\mathbf{c}_m + \boldsymbol{\lambda}_m) \cdot \mathbf{z}_m - \frac{\rho}{2} \|\mathbf{z}_m - \mathbf{u}_{\mathbf{i}(m)}\|^2$$

  end for
  make a $\mathbf{u}$-update by updating the consensus solution, averaging $\mathbf{z}_1, \ldots, \mathbf{z}_m$:

$$u^{t+1}(r,s) \leftarrow \frac{1}{\delta(r,s)} \sum_{m:(r,s) \in \mathbf{i}(m)} z_m^{t+1}(r,s)$$

  make a $\boldsymbol{\lambda}$-update:

$$\boldsymbol{\lambda}_m^{t+1} \leftarrow \boldsymbol{\lambda}_m^t - \rho(\mathbf{z}_m^{(t+1)} - \mathbf{u}_{\mathbf{i}(m)}^{(t+1)}), \quad \forall m$$

  $t \leftarrow t + 1$
until convergence.
output: relaxed primal solution $\mathbf{u}^*$ and dual solution $\boldsymbol{\lambda}^*$. If $\mathbf{u}^*$ is integer, it will encode an assignment of spans to roles. Otherwise, it will provide an upper bound of the true optimum.

Like in the subgradient method, the $\boldsymbol{\lambda}$-updates can be regarded as price adjustments, which will affect the next round of $\mathbf{z}_m$-updates. The only difference with respect to the subgradient method (Rush et al., 2010) is that each subproblem involved in a $\mathbf{z}_m$-update also has a quadratic penalty that penalizes deviations from the previous average voting; it is this term that accelerates consensus and therefore convergence. Martins et al. (2011b) also provide stopping criteria for the iterative updates using primal and dual residuals that measure convergence; we refer the reader to that paper for details.

A key attraction of this algorithm is all the components of the declarative specification remain intact

in the procedural form. Each worker corresponds exactly to one constraint in the ILP, which corresponds to one linguistic constraint. There is no need to work out *when*, during the procedure, each constraint might have an effect, as in beam search.

Solving the subproblems. In a different application, Martins et al. (2011b, §4) showed how to solve each $\mathbf{z}_m$-subproblem associated with the XOR, XORWITHOUTPUT and OR factors in runtime $O(|\mathbf{i}(m)| \log |\mathbf{i}(m)|)$. The only subproblem that remains is that of the ATMOSTONE factor, to which we now turn. The problem can be transformed into that of projecting a point $(a_1, \ldots, a_k)$ onto the set

$$\mathcal{S}_m = \left\{ \mathbf{z}_m \in [0,1]^{|\mathbf{i}(m)|} \;\middle|\; \sum_{j=1}^{|\mathbf{i}(m)|} z_{m,j} \leq 1 \right\}.$$

This projection can be computed as follows:

1. Clip each a_j into the interval $[0, 1]$ (*i.e.*, set $a'_j = \min\{\max\{a_j, 0\}, 1\}$). If the result satisfies $\sum_{j=1}^{k} a'_j \leq 1$, then return $(a'_1, \ldots, a'_k)$.
2. Otherwise project $(a_1, \ldots, a_k)$ onto the probability simplex:

$$\left\{ \mathbf{z}_m \in [0,1]^{|\mathbf{i}(m)|} \;\middle|\; \sum_{j=1}^{|\mathbf{i}(m)|} z_{m,j} = 1 \right\}.$$

   This is precisely the XOR subproblem and can be solved in time $O(|\mathbf{i}(m)| \log |\mathbf{i}(m)|)$.

Caching. As mentioned by Martins et al. (2011b), as the algorithm comes close to convergence, many subproblems become unchanged and their solutions can be cached. By caching the subproblems, we managed to reduce runtime by about 60%.

Exact decoding. Finally, it is worth recalling that AD^3, like other dual decomposition algorithms, solves a *relaxation* of the actual problem. Although we have observed that the relaxation is often tight—cf. §4—this is not always the case. Specifically, a fractional solution may be obtained, which is not interpretable as an argument, and therefore it is desirable to have a strategy to recover the exact solution. Two observations are noteworthy. First, the optimal value of the relaxed problem (Eq. 11) provides an upper bound to the original problem (Eq. 2). This is because Eq. 2 has the additional integer constraint on the variables. In particular, any feasible dual point provides an upper bound to the original problem's optimal value. Second, during execution of the AD^3 algorithm, we always keep track of a sequence of feasible dual points. Therefore, each iteration constructs tighter and tighter upper bounds. With this machinery, we have all that is necessary for implementing a branch-and-bound search that finds the exact solution of the ILP. The procedure works recursively as follows:

1. Initialize $L = -\infty$ (our best value so far).
2. Run Algorithm 1. If the solution $\mathbf{u}^*$ is integer, return $\mathbf{u}^*$ and set L to the objective value. If along the execution we obtain an upper bound less than L, then Algorithm 1 can be safely stopped and return "infeasible"—this is the *bound* part. Otherwise (if $\mathbf{u}^*$ is fractional) go to step 3.
3. Find the "most fractional" component of $\mathbf{u}^*$ (call it u^*_j) and *branch*: constrain $u_j = 0$ and go to step 2, eventually obtaining an integer solution $\mathbf{u}^*_0$ or infeasibility; and then constrain $u_j = 1$ and do the same, obtaining $\mathbf{u}^*_1$. Return the $\mathbf{u}^* \in \{\mathbf{u}^*_0, \mathbf{u}^*_1\}$ that yields the largest objective value.

Although this procedure may have worst-case exponential runtime, we found it empirically to rapidly obtain the exact solution in all test cases.

4 Experiments and Results

4.1 Dataset, Preprocessing, and Learning

In our experiments, we use FrameNet 1.5, which contains a lexicon of 877 frames and 1,068 role labels, and 78 documents with multiple predicate-argument annotations (a superset of the SemEval shared task dataset; Baker et al., 2007). We used the same split as Das and Smith (2011), with 55 documents for training (containing 19,582 frame annotations) and 23 for testing (with 4,458 annotations). We randomly selected 4,462 predicates in the training set as development data. The raw sentences in all the training and test documents were preprocessed using MXPOST (Ratnaparkhi, 1996) and the MST dependency parser (McDonald et al., 2005).

The state-of-the-art system for this task is SEMAFOR, an open source tool (Das et al., 2010a)[5] that provides a baseline benchmark for our new algorithm. We use the components of SEMAFOR as-is to define the features $\mathbf{h}$ and train the weights $\boldsymbol{\psi}$ used in the scoring function c. We also use its

[5] http://www.ark.cs.cmu.edu/SEMAFOR

heuristic mechanism to find potential spans $\mathcal{S}_t$ for a given predicate t. SEMAFOR learns weights using ℓ_2-penalized log-likelihood; we augmented its dev set-tuning procedure to tune both the regularization strength and the AD^3 penalty strength ρ. We initialize $\rho = 0.1$ and follow Martins et al. (2011b) in dynamically adjusting it. Note that we do not use SEMAFOR's automatic frame identification component in our presented experiments, as we assume that we have gold frames on each predicate. This lets us compare the different argument identification methods in a controlled fashion.

4.2 Decoding Strategies

We compare the following algorithms:

1. **Local**: this is a naïve argument identification strategy that selects the best span for each role r, according to the score function $c(r, s)$. It ignores all constraints except "uniqueness."
2. **SEMAFOR**: this strategy employs greedy beam search to eliminate overlaps between predicted arguments (Das et al., 2010b, Algorithm 1). Note that it does not try to respect the "excludes" and "requires" constraints between pairs of roles. The default size of the beam in SEMAFOR was a safe 10,000; this resulted in extremely slow decoding times. We also tried beam sizes of 100 and 2 (the latter being the smallest size that achieves the same F_1 score on the dev set as beam width 100.)
3. **CPLEX**, *LP*: this uses CPLEX to solve the relaxed LP in Eq. 11. To handle fractional $\mathbf{z}$, for each role r, we choose the best span s^*, such that $s^* = \arg\max_{s \in \mathcal{S}_r} z_{r,s}$, solving ties arbitrarily.
4. **CPLEX**, *exact*: this tackles the actual ILP (Eq. 2) with CPLEX.
5. **AD^3**, *LP*: this is the counterpart of the LP version of CPLEX, where the relaxed problem is solved using AD^3. We choose the spans for each role in the same way as in strategy 3.
6. **AD^3**, *exact*: this couples AD^3 with branch-and-bound search to get the exact integer solution.

4.3 Results

Table 1 shows performance of the different decoding strategies on the test set. We report precision, recall, and F_1 scores.[6] Since these scores do not penalize structural violations, we also report the number of overlap, "excludes," and "requires" constraints that were violated in the test set. Finally, we tabulate each setting's decoding time in seconds on the whole test set averaged over 5 runs.[7] The Local model is very fast but suffers degradation in precision and violates one constraint roughly per nine predicates. SEMAFOR used a default beam size of 10,000, which is extremely slow; a faster version of beam size 100 results in the same precision and recall values, but is 15 times faster. Beam size 2 results in slightly worse precision and recall values, but is even faster. All of these, however, result in many constraint violations. Strategies involving CPLEX and AD^3 perform similarly to each other and SEMAFOR on precision and recall, but eliminate most or all of the constraint violations. SEMAFOR with beam size 2 is 11-16 times faster than the CPLEX strategies, but is only twice as fast than AD^3, and results in significantly more structural violations. The exact algorithms are slower than the LP versions, but compared to CPLEX, AD^3 is significantly faster and has a narrower gap between its exact and LP versions. We found that relaxation was tight 99.8% of the time on the test examples.

The example in Fig. 1 is taken from our test set, and shows an instance where two roles, Partner_1 and Partner_2 share the "requires" relationship; for this example, the beam search decoder misses the Partner_2 role, which is a violation, while our AD^3 decoder identifies both arguments correctly. Note that beam search makes plenty of linguistic violations, but has precision and recall values that are marginally better than AD^3. We found that beam search, when violating many "requires" constraints, often finds one role in the pair, which increases its recall. AD^3 is sometimes more conservative in such cases, predicting neither role. A second issue, as noted in footnote 4, is that the annotations sometimes violate these constraints. Overall, we found it interesting that imposing the constraints did not have much effect on standard measures of accuracy.

[6] We use the evaluation script from SemEval 2007 shared task, modified to evaluate only the argument identification output.

[7] We used a 64-bit machine with 2 2.6GHz dual-core CPUs (*i.e.*, 4 processors in all) with a total of 8GB of RAM. The workers in AD^3 were not parallelized, while CPLEX automatically parallelized execution.

Method	P	R	F_1	Violations: Overlap	Violations: Requires	Violations: Excludes	Time in Secs.	
Local	67.69	59.76	63.48	441	45	15	1.26	$\pm$ 0.01
SEMAFOR (beam = 2)	70.18	59.54	64.42	0	49	0	2.74	$\pm$ 0.10
SEMAFOR (beam = 100)	70.43	59.64	64.59	0	50	1	29.00	$\pm$ 0.25
SEMAFOR (beam = 10000)	70.43	59.64	64.59	0	50	1	440.67	$\pm$ 5.53
CPLEX, *LP*	70.34	59.43	64.43	0	1	0	32.67	$\pm$ 1.29
CPLEX, *exact*	70.31	59.45	64.43	0	0	0	43.12	$\pm$ 1.26
AD^3, *LP*	70.30	59.45	64.42	2	2	0	4.17	$\pm$ 0.01
AD^3, *exact*	70.31	59.45	64.43	0	0	0	4.78	$\pm$ 0.04

Table 1: Comparison of decoding strategies in §4.2. We evaluate in terms of precision, recall and F_1 score on a test set containing 4,458 predicates. We also compute the number of structural violations each model makes: number of overlapping arguments and violations of the "requires" and "excludes" constraints of §2. Finally decoding time (without feature computation steps) on the *whole* test set is shown in the last column averaged over 5 runs.

5 Related Work

Semantic role labeling: Most SRL systems use conventions from PropBank (Kingsbury and Palmer, 2002) and NomBank (Meyers et al., 2004), which store information about verbal and nominal predicates and corresponding symbolic and meaning-specific semantic roles. A separate line of work, including this paper, investigates SRL systems that use FrameNet conventions; while less popular, these systems, pioneered by Gildea and Jurafsky (2002), consider predicates of a wider variety of syntactic categories, use semantic frame abstractions, and employ explicit role labels. A common trait in prior work has been the use of a two-stage model that identifies arguments first, then labels them. They are treated jointly here, unlike what has typically been done in PropBank-style SRL (Màrquez et al., 2008).

Dual decomposition: Rush et al. (2010) proposed subgradient-based dual decomposition as a way of combining models which are tractable individually, but not jointly, by solving a relaxation of the original problem. This was followed by work adopting this method for syntax and translation (Koo et al., 2010; Auli and Lopez, 2011; DeNero and Macherey, 2011; Rush and Collins, 2011; Chang and Collins, 2011). Recently, Martins et al. (2011b) showed that the success of subgradient-based dual decomposition strongly relies on breaking down the original problem into a "good" decomposition, *i.e.*, one with few overlapping components. This leaves out many declarative constrained problems, for which such a good decomposition is not readily available. For those, Martins et al. (2011b) proposed the AD^3 algorithm, which retains the modularity of previous methods, but can handle thousands of small overlapping components.

Exact decoding: This paper contributes an exact branch-and-bound technique wrapped around AD^3. A related line of research is that of Rush and Collins (2011), who proposed a tightening procedure for dual decomposition, which can be seen as a cutting plane method (another popular approach in combinatorial optimization).

6 Conclusion

We presented a novel algorithm for incorporating declarative linguistic knowledge as constraints in shallow semantic parsing. It outperforms a naïve baseline that is oblivious to the constraints. Furthermore, it is significantly faster than a decoder employing a state-of-the-art proprietary solver, and less than twice as slow as beam search, which is inexact and does not respect all linguistic constraints. Our method is easily amenable to the inclusion of more constraints, which would require minimal programming effort. Our implementation of AD^3 within SEMAFOR will be publicly released at `http://www.ark.cs.cmu.edu/SEMAFOR`.

Acknowledgments

We thank the three anonymous reviewers for their valuable feedback. This material is based upon work supported by NSF grant IIS-1054319, Google's support of the Wordly Knowledge Project, a FCT/ICTI grant through the CMU-Portugal Program, and by Priberam, through the Discooperio project, contract 2011/18501 of the EU/FEDER program.

References

M. Auli and A. Lopez. 2011. A comparison of loopy belief propagation and dual decomposition for integrated ccg supertagging and parsing. In *Proc. of ACL*.

C. Baker, M. Ellsworth, and K. Erk. 2007. SemEval-2007 Task 19: Frame semantic structure extraction. In *Proc. of SemEval*.

Y.-W. Chang and Michael Collins. 2011. Exact decoding of Phrase-Based translation models through lagrangian relaxation. In *Proc. of EMNLP*. Association for Computational Linguistics.

D. Chen, N. Schneider, D. Das, and N. A. Smith. 2010. SEMAFOR: Frame argument resolution with log-linear models. In *Proc. of SemEval*.

D. Das and N. A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proc. of ACL*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010a. Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010b. SEMAFOR 1.0: a probabilistic frame-semantic parser. Technical report, CMU-LTI-10-001.

J. DeNero and K. Macherey. 2011. Model-based aligner combination using dual decomposition. In *Proc. of ACL*.

C. J. Fillmore, C. R. Johnson, and M. R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3).

C. J. Fillmore. 1982. Frame Semantics. In *Linguistics in the Morning Calm*. Hanshin.

M. Gerber and J. Y. Chai. 2010. Beyond nombank: A study of implicit arguments for nominal predicates. In *ACL*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

R. Johansson and P. Nugues. 2007. LTH: semantic structure extraction using nonprojective dependency trees. In *Proc. of SemEval*.

P. Kingsbury and M. Palmer. 2002. From TreeBank to PropBank. In *Proc. of LREC*.

N. Komodakis, N. Paragios, and G. Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34(2).

A. F. T. Martins, N. A. Smith, E. P. Xing, M. A. T. Figueiredo, and P. M. Q. Aguiar. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *EMNLP*.

A. F. T. Martins, M. A. T. Figueiredo, P. M. Q. Aguiar, N. A. Smith, and E. P. Xing. 2011a. An augmented Lagrangian approach to constrained MAP inference. In *Proc. of ICML*.

A F. T. Martins, N. A. Smith, P. M. Q. Aguiar, and M. A. T. Figueiredo. 2011b. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proc. of NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

V. Punyakanok, D. Roth, W.-T. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of COLING*.

V. Punyakanok, D. Roth, and W Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34:257–287.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

A. M. Rush and M. Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proc. of ACL*.

A. M. Rush, D. Sontag, M. Collins, and T. Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of EMNLP*.

D. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *EMNLP*.

K. Toutanova, A. Haghighi, and C. Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of ACL*.