On Amortizing Inference Cost for Structured Prediction

Vivek Srikumar* and **Gourab Kundu*** and **Dan Roth**
University of Illinois, Urbana-Champaign
Urbana, IL. 61801
{vsrikum2, kundu2, danr}@illinois.edu

Abstract

This paper deals with the problem of predicting structures in the context of NLP. Typically, in structured prediction, an inference procedure is applied to each example independently of the others. In this paper, we seek to optimize the time complexity of inference over entire datasets, rather than individual examples. By considering the general inference representation provided by integer linear programs, we propose three exact inference theorems which allow us to re-use earlier solutions for certain instances, thereby completely avoiding possibly expensive calls to the inference procedure. We also identify several approximation schemes which can provide further speedup. We instantiate these ideas to the structured prediction task of semantic role labeling and show that we can achieve a speedup of over 2.5 using our approach while retaining the guarantees of exactness and a further speedup of over 3 using approximations that do not degrade performance.

1 Introduction

Typically, in structured prediction applications, every example is treated independently and an inference algorithm is applied to each one of them. For example, consider a dependency parser that uses the maximum spanning tree algorithm (McDonald et al., 2005) or its integer linear program variants (Riedel and Clarke, 2006; Martins et al., 2009) to make predictions. Given a trained model, the parser addresses each sentence separately and runs the inference algorithm to predict the parse tree. Thus, the time complexity of inference over the test set is linear in the size of the corpus.

In this paper, we ask the following question: *For a given task, since the inference procedure predicts structures from the same family of structures (dependency trees, semantic role structures, etc.), can the fact that we are running inference for a large number of examples help us improve the time complexity of inference?* In the dependency parsing example, this question translates to asking whether, having parsed many sentences, we can decrease the parsing time for the next sentence.

Since any combinatorial optimization problem can be phrased as an integer linear program (ILP), we frame inference problems as ILPs for the purpose of analysis. By analyzing the objective functions of integer linear programs, we identify conditions when two ILPs have the same solution. This allows us to reuse solutions of previously solved problems and theoretically guarantee the optimality of the solution. Furthermore, in some cases, even when the conditions are not satisfied, we can reuse previous solutions with high probability of being correct.

Given the extensive use of integer linear programs for structured prediction in Natural Language Processing over the last few years, these ideas can be applied broadly to NLP problems. We instantiate our improved inference approaches in the structured prediction task of semantic role labeling, where we use an existing implementation and a previous trained model that is based on the approach of (Punyakanok et al., 2008). We merely modify the inference pro-

* These authors contributed equally to this work.

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1114–1124, Jeju Island, Korea, 12–14 July 2012.

cess to show that we can realize the theoretical gains by making fewer calls to the underlying ILP solver.

Algorithm	Speedup
Theorem 1	2.44
Theorem 2	2.18
Theorem 3	2.50

Table 1: The speedup for semantic role labeling corresponding to the three theorems described in this paper. These theorems guarantee the optimality of the solution, thus ensuring that the speedup is not accompanied by any loss in performance.

Table 1 presents a preview of our results, which are discussed in Section 4. All three approaches in this table improve running time, while guaranteeing optimum solutions. Allowing small violations to the conditions of the theorems provide an even higher improvement in speedup (over 3), without loss of performance.

The primary contributions of this paper are:

1. We pose the problem of optimizing inference costs over entire datasets rather than individual examples. Our approach is agnostic to the underlying models and allows us to use pretrained scoring functions.

2. We identify equivalence classes of ILP problems and use this notion to prove exact conditions under which no inference is required. These conditions lead to algorithms that can speed up inference problem without losing the exactness guarantees. We also use these conditions to develop approximate inference algorithms that can provide a further speedup.

3. We apply our approach to the structured prediction task of semantic role labeling. By not having to perform inference on some of the instances, those that are equivalent to previously seen instances, we show significant speed up in terms of the number of times inference needs to be performed. These gains are also realized in terms of wall-clock times.

The rest of this paper is organized as follows: In section 2, we formulate the problem of amortized inference and provide motivation for why amortized gains can be possible. This leads to the theoretical discussion in section 3, where we present the meta-algorithm for amortized inference along with several exact and approximate inference schemes. We instantiate these schemes for the task of semantic role labeling (Section 4). Section 5 discusses related work and future research directions.

2 Motivation

Many NLP tasks can be phrased as structured prediction problems, where the goal is to jointly assign values to many inference variables while accounting for possible dependencies among them. This decision task is a combinatorial optimization problem and can be solved using a dynamic programming approach if the structure permits. In general, the inference problem can be formulated and solved as integer linear programs (ILPs).

Following (Roth and Yih, 2004) Integer linear programs have been used broadly in NLP. For example, (Riedel and Clarke, 2006) and (Martins et al., 2009) addressed the problem of dependency parsing and (Punyakanok et al., 2005; Punyakanok et al., 2008) dealt with semantic role labeling with this technique.

In this section, we will use the ILP formulation of dependency parsing to introduce notation. The standard approach to framing dependency parsing as an integer linear program was introduced by (Riedel and Clarke, 2006), who converted the MST parser of (McDonald et al., 2005) to use ILP for inference. The key idea is to build a complete graph consisting of tokens of the sentence where each edge is weighted by a learned scoring function. The goal of inference is to select the maximum spanning tree of this weighted graph.

2.1 Problem Formulation

In this work, we consider the general inference problem of solving a 0-1 integer linear program. To perform inference, we assume that we have a model that assigns scores to the ILP decision variables. Thus, our work is applicable not only in cases where inference is done after a separate learning phase, as in (Roth and Yih, 2004; Clarke and Lapata, 2006; Roth and Yih, 2007) and others, but also when inference is done during the training phase, for algorithms like

the structured perceptron of (Collins, 2002), structured SVM (Tsochantaridis et al., 2005) or the constraints driven learning approach of (Chang et al., 2007).

Since structured prediction assigns values to a collection of inter-related binary decisions, we denote the i^{th} binary decision by $y_i \in \{0, 1\}$ and the entire structure as $\mathbf{y}$, the vector composed of all the binary decisions. In our running example, each edge in the weighted graph generates a single decision variable (for unlabeled dependency parsing). For each y_i, let $c_i \in \Re$ denote the weight associated with it. We denote the entire collection of weights by the vector $\mathbf{c}$, forming the *objective* for this ILP.

Not all assignments to these variables are valid. Without loss of generality, these constraints can be expressed using linear inequalities over the inference variables, which we write as $\mathbf{M}^T\mathbf{y} \leq \mathbf{b}$ for a real valued matrix $\mathbf{M}$ and a vector $\mathbf{b}$. In dependency parsing, for example, these constraints ensure that the final output is a spanning tree.

Now, the overall goal of inference is to find the highest scoring structure. Thus, we can frame inference as an optimization problem $\mathbf{p}$ with n inference variables as follows:

$$\arg\max_{\mathbf{y}\in\{0,1\}^n} \quad \mathbf{c}^T\mathbf{y} \tag{1}$$

$$\text{subject to} \quad \mathbf{M}^T\mathbf{y} \leq \mathbf{b}. \tag{2}$$

For brevity, we denote the space of feasible solutions that satisfy the constraints for the ILP problem $\mathbf{p}$ as $K_\mathbf{p} = \{\mathbf{y} \in \{0, 1\}^n | \mathbf{M}^T\mathbf{y} \leq \mathbf{b}\}$. Thus, the goal of inference is to find

$$\arg\max_{\mathbf{y}\in K_\mathbf{p}} \mathbf{c}^T\mathbf{y}.$$

We refer to $K_\mathbf{p}$ as the feasible set for the inference problem $\mathbf{p}$ and $\mathbf{y}_\mathbf{p}$ as its solution.

In the worst case, integer linear programs are known to be NP-hard. Hence, solving large problems, (that is, problems with a large number of constraints and/or variables) can be infeasible.

For structured prediction problems seen in NLP, we typically solve many instances of inference problems. In this paper, we investigate whether an inference algorithm can use previous predictions to speed up inference time, thus giving us an *amortized gain* in inference time over the lifetime of the program. We refer to inference algorithms that have this capability as **amortized inference algorithms**.

In our running example, each sentence corresponds to a separate ILP. Over the lifetime of the dependency parser, we create one inference instance (that is, one ILP) per sentence and solve it. An amortized inference algorithm becomes *faster* at parsing as it parses more and more sentences.

2.2 Why can inference costs be amortized over datasets?

In the rest of this section, we will argue that the time cost of inference can be amortized because of the nature of inference in NLP tasks. Our argument is based on two observations, which are summarized in Figure (1): (1) Though the space of possible structures may be large, only a very small fraction of these occur. (2) The distribution of observed structures is heavily skewed towards a small number of them.

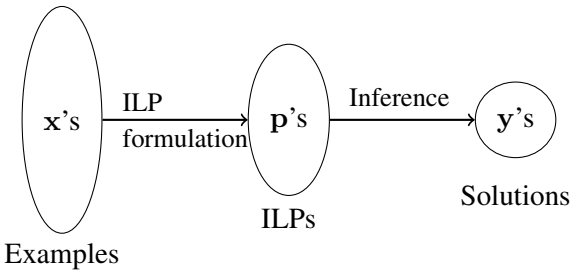
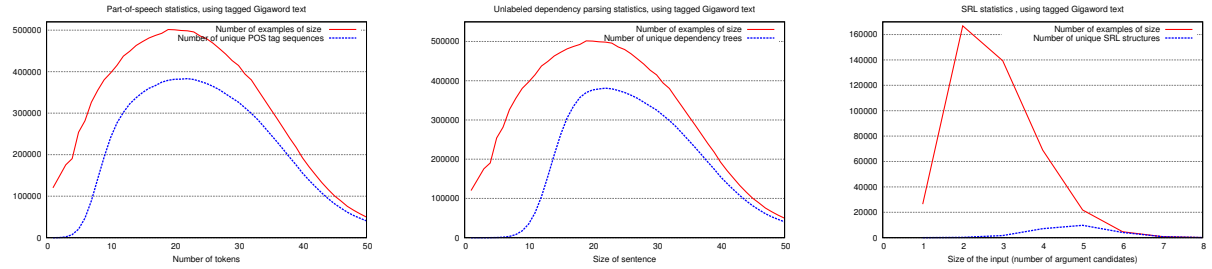


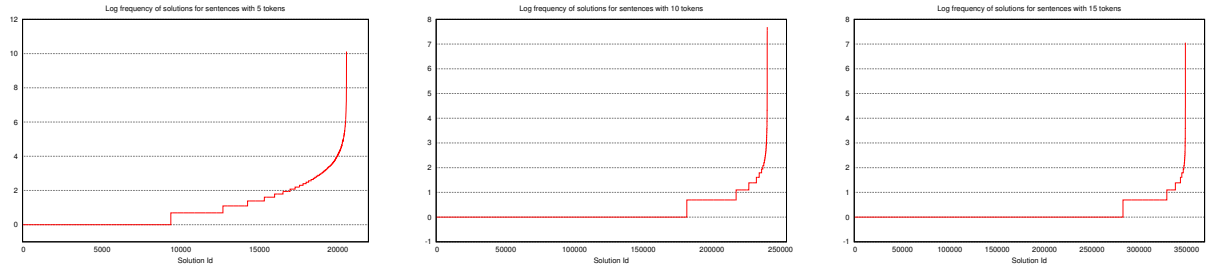
Figure 1: For a structured prediction task, the inference problem $\mathbf{p}$ for an example $\mathbf{x}$ needs to be formulated before solving it to get the structure $\mathbf{y}$. In structured prediction problems seen in NLP, while an exponential number of structures is possible for a given instance, in practice, only a small fraction of these ever occur. This figure illustrates the empirical observation that there are fewer inference problems $\mathbf{p}$'s than the number of examples and the number of observed structures $\mathbf{y}$'s is even lesser.

As an illustration, consider the problem of part-of-speech tagging. With the standard Penn Treebank tag set, each token can be assigned one of 45 labels. Thus, for a sentence of size n, we could have 45^n structures out of which the inference process needs to choose one. However, a majority of these structures never occur. For example, we cannot have a



(a) Part-of-speech tagging (b) Unlabeled dependency parsing (c) Semantic role labeling

Figure 2: Number of inference instances for different input sizes (red solid lines) and the number of unique structures for each size (blue dotted lines). The x-axis indicates the size of the input (number of tokens for part of speech and dependency, and number of argument candidates for SRL.) Note that the number of instances is not the number of unique examples of a given length, but the number of times an inference procedure is called for an input of a given size.



(a) Sentence length = 5 (b) Sentence length = 10 (c) Sentence length = 15

Figure 3: These plots show the log-frequencies of occurrences of part-of-speech sequences for sentences with five, ten and fifteen tokens. The x-axes list different unique part-of-speech tag sequences for the entire sentence. These plots show that for sentences of a given length, most structures (solutions) that are possible never occur, or occur very infrequently; only a few of the possible structures (solutions) actually occur frequently.

sentence where all the tokens are determiners.

Furthermore, many sentences of the same size share the same part-of-speech tag sequence. To quantify the redundancy of structures, we part-of-speech tagged the English Gigaword corpus (Graff and Cieri, 2003). Figure (2a) shows the number of sentences in the corpus for different sentence lengths. In addition, it also shows the number of unique part-of-speech tag sequences (over the entire sentence) for each size. We see that the number of structures is much fewer than the number of instances for any sentence size. Note that 45^n quickly outgrows the number of sentences as n increases. The figures (2b) and (2c) show similar statistics for unlabeled dependency parsing and semantic role labeling. In the former case, the size of the instance is the number of tokens in a sentence, while in the latter, the size is the number of argument candidates that need to be labeled for a given predicate. In both cases, we see that the number of empirically observed structures is far fewer than the number of instances to be labeled.

Thus, for any given input size, the number of instances of that size (over the lifetime of the program) far exceeds the number of observed structures for that size. Moreover, the number of observed structures is significantly smaller than the number of theoretically possible structures. Thus, we have a small number of structures that form optimum structures for many inference instances of the same size.

Our second observation deals with the distribution of structures for a given input size. Figure (3)

shows the log frequencies of part-of-speech tagging sequences for sentences of lengths five, ten and fifteen. In all cases, we see that a few structures are most frequent. We observed similar distributions of structures for all input sizes for dependency parsing and semantic role labeling as well.

Since the number of structures for a given example size is small, many examples $\mathbf{x}$'s, and hence many inference problems $\mathbf{p}$'s, are associated with the same structure $\mathbf{y}$. These observations suggest the possibility of getting an amortized gain in inference time by characterizing the set of inference problems that produce the same structure. Then, for a new inference problem, if we can identify that it belongs to a known set, that is, will yield a solution that we have already seen, we do not have to run inference at all. The second observation also suggests that this characterization of sets of problems that have the same solution can be done in a data-driven way because characterizing a small number of structures can give us high coverage.

3 Amortizing inference costs

In this section, we present different schemes for amortized inference leading up to an inference meta-algorithm. The meta-algorithm is both agnostic to the underlying inference algorithm that is used by the problem and maintains the exactness properties of the underlying inference scheme. That is, if we have an exact/approximate inference algorithm with a certain guarantees, the meta-algorithm will have the same guarantees, but with a speedup.

3.1 Notation

For an integer linear program $\mathbf{p}$ with $n_{\mathbf{p}}$ variables, we denote its objective coefficients by $\mathbf{c_p}$ and its feasible set by $K_{\mathbf{p}}$. We denote its solution as as $\mathbf{y_p}$. We represent vectors by boldfaced symbols and their i^{th} component using subscripts.

We consider many instantiations of the inference problem and use superscripts to denote each individual instance. Thus, we have a large collection of inference instances $P = \{\mathbf{p}^1, \mathbf{p}^2, \cdots\}$ along with their respective solutions $\{\mathbf{y}_{\mathbf{p}}^1, \mathbf{y}_{\mathbf{p}}^2, \cdots\}$.

Definition 1 (Equivalence classes of ILPs)**.** *Two integer linear programs are said to be in the same equivalence class if they have the same number of inference variables and the same feasible set.*

We square brackets to denote equivalence classes. If $[P]$ is an equivalence class of ILPs, we use the notation $K_{[P]}$ to denote its feasible set and $n_{[P]}$ to denote the number of variables. Also, for a program $\mathbf{p}$, we use the notation $\mathbf{p} \sim [P]$ to indicate that it belongs to the equivalence class $[P]$.

3.2 Exact theorems

Our goal is to characterize the set of objective functions which will have the same solution for a given equivalence class of problems.

Suppose we have solved an ILP $\mathbf{p}$ to get a solution $\mathbf{y_p}$. For every inference variable that is active in the solution (i.e., whose value is 1), increasing the corresponding objective value will not change the optimal assignment to the variables. Similarly, for all other variables (whose value in the solution is 0), decreasing the objective value will not change the optimal solution. This intuition gives us our first theorem for checking whether two ILPs have the same solution by looking at the difference between their objective coefficients.

Theorem 1. *Let $\mathbf{p}$ denote an inference problem posed as an integer linear program belonging to an equivalence class $[P]$. Let $\mathbf{q} \sim [P]$ be another inference instance in the same equivalence class. Define $\delta\mathbf{c} = \mathbf{c_q} - \mathbf{c_p}$ to be the difference of the objective coefficients of the ILPs. Then, $\mathbf{y_p}$ is the solution of the problem $\mathbf{q}$ if for each $i \in \{1, \cdots, n_{\mathbf{p}}\}$, we have*

$$(2\mathbf{y}_{\mathbf{p},i} - 1)\delta\mathbf{c}_i \geq 0 \tag{3}$$

The condition in the theorem, that is, inequality (3), requires that the objective coefficients corresponding to values $\mathbf{y}_{\mathbf{p},i}$ that are set to 1 in $\mathbf{p}$ increase, and those that correspond to values of $\mathbf{y}_{\mathbf{p},i}$ set to 0, decrease. Under these conditions, if $\mathbf{y_p}$ is the maximizer of the original objective, then it maximizes the new objective too.

Theorem 1 identifies perturbations of an ILP's objective coefficients that will not change the optimal assignment. Next, we will characterize the sets of objective values that will have the same solution using a criterion that is independent of the actual solution. Suppose we have two ILPs $\mathbf{p}$ and $\mathbf{q}$ in an equivalence class $[P]$ whose objective values are $\mathbf{c_p}$ and $\mathbf{c_q}$ respectively. Suppose $\mathbf{y}^*$ is the solution to

both these programs. That is, for every $\mathbf{y} \in K_{[P]}$, we have $\mathbf{c_p}^T\mathbf{y} \leq \mathbf{c_p}^T\mathbf{y}^*$ and $\mathbf{c_q}^T\mathbf{y} \leq \mathbf{c_q}^T\mathbf{y}^*$. Multiplying these inequalities by any two positive real numbers x_1 and x_2 and adding them shows us that $\mathbf{y}^*$ is also the solution for the ILP in $[P]$ which has the objective coefficients $x_1\mathbf{c_p} + x_2\mathbf{c_q}$. Extending this to an arbitrary number of inference problems gives us our next theorem.

Theorem 2. *Let P denote a collection $\{\mathbf{p}^1, \mathbf{p}^2, \cdots, \mathbf{p}^m\}$ of m inference problems in the same equivalence class $[P]$ and suppose that all the problems have the same solution, $\mathbf{y_p}$. Let $\mathbf{q} \sim [P]$ be a new inference program whose optimal solution is $\mathbf{y}$. Then $\mathbf{y} = \mathbf{y_p}$ if there is some $\mathbf{x} \in \Re^m$ such that $\mathbf{x} \geq \mathbf{0}$ and*

$$\mathbf{c_q} = \sum_j \mathbf{x}_j \mathbf{c_p}^j. \tag{4}$$

From the geometric perspective, the pre-condition of this theorem implies that if the new coefficients lie in the cone formed by the coefficients of the programs that have the same solution, then the new program shares the solution.

Theorems 1 and 2 suggest two different approaches for identifying whether a new ILP can use the solution of previously solved inference instances. These theorems can be combined to get a single criterion that uses the objective coefficients of previously solved inference problems *and* their common solution to determine whether a new inference problem will have the same solution. Given a collection of solved ILPs that have the same solution, from theorem 2, we know that an ILP with the objective coefficients $\mathbf{c} = \sum_j \mathbf{x}_j \mathbf{c_p}^j$ will share the solution. Considering an ILP whose objective vector is $\mathbf{c}$ and applying theorem 1 to it gives us the next theorem.

Theorem 3. *Let P denote a collection $\{\mathbf{p}^1, \mathbf{p}^2, \cdots, \mathbf{p}^m\}$ of m inference problems belonging to the same equivalence class $[P]$. Furthermore, suppose all the programs have the same solution $\mathbf{y_p}$. Let $\mathbf{q} \sim [P]$ be a new inference program in the equivalence class. For any $\mathbf{x} \in \Re^m$, define $\Delta\mathbf{c}(\mathbf{x}) = \mathbf{c_q} - \sum_j \mathbf{x}_j \mathbf{c_p}^j$. The assignment $\mathbf{y_p}$ is the optimal solution of the problem $\mathbf{q}$ if there is some $\mathbf{x} \in \Re^m$ such that $\mathbf{x} \geq \mathbf{0}$ and for each $i \in \{1, n_\mathbf{p}\}$, we have*

$$(2\mathbf{y}_{\mathbf{p},i} - 1)\Delta\mathbf{c}_i \geq 0 \tag{5}$$

Theorem	Condition
Theorem 1	$\forall i \in \{1, \cdots, n_\mathbf{p}\}$, $(2\mathbf{y}_{\mathbf{p},i} - 1)\delta\mathbf{c}_i \geq 0;\ \forall i.$
Theorem 2	$\exists\, \mathbf{x} \in \Re^m$, such that $\mathbf{x} \geq 0$ and $\mathbf{c_q} = \sum_j \mathbf{x}_j \mathbf{c_p}^j$
Theorem 3	$\exists\, \mathbf{x} \in \Re^m$, such that $\mathbf{x} \geq \mathbf{0}$ and $(2\mathbf{y}_{\mathbf{p},i} - 1)\Delta\mathbf{c}_i \geq 0;\ \forall i.$

Table 2: Conditions for checking whether $\mathbf{y_p}$ is the solution for an inference problem $\mathbf{q} \sim [P]$ according to theorems 1, 2 and 3. Please refer to the statements of the theorems for details about the notation.

3.3 Implementation

Theorems 1, 2 and 3 each specify a condition that checks whether a pre-existing solution is the optimal assignment for a new inference problem. These conditions are summarized in Table 2. In all cases, if the condition matches, the theorems guarantee that the two solutions will be the same. That is, applying the theorems will not change the performance of the underlying inference procedure. Only the number of inference calls will be decreased.

In our implementation of the conditions, we used a database[1] to cache ILPs and implemented the retrieval of equivalence classes and solutions as queries to the database. To implement theorem 1, we iterate over all ILPs in the equivalence class and check if the condition is satisfied for one of them. The conditions of theorems 2 and 3 check whether a collection of linear (in)equalities has a feasible solution using a linear program solver.

We optimize the wall-clock time of theorems 2 and 3 by making two observations. First, we do not need to solve linear programs for all possible observed structures. Given an objective vector, we only need consider the highest scoring structures within an equivalence class. (All other structures cannot be the solution to the ILP.) Second, since theorem 2 checks whether an ILP lies within a cone, we can optimize the cache for theorem 2 by only storing the ILPs that form on the boundary of the cone. A similar optimization can be performed for theorem 3 as well. Our implementation uses the following weaker version of this optimization: while caching ILPs, we

[1] We used the H2 database engine, which can be downloaded from `http://h2database.com`, for all caching.

do not add an instance to the cache if it already satisfies the theorem. This optimization reduces the size of the linear programs used to check feasibility.

3.4 Approximation schemes

So far, in the above three theorems, we retain the guarantees (in terms of exactness and performance) of the underlying inference procedure. Now, we will look at schemes for approximate inference. Unlike the three theorems listed above, with the following amortized inference schemes, we are not guaranteed an optimal solution.

3.4.1 Most frequent solution

The first scheme for approximation uses the observation that the most frequent solution occurs an overwhelmingly large number of times, compared to the others. (See the discussion in section 2.2 and figures 3a, 3b and 3c for part-of-speech tagging.) Under this approximation scheme, given an ILP problem, we simply pick the most frequent solution for that equivalence class as the solution, provided this solution has been seen a sufficient number of times. If the support available in the cache is insufficient, we call the underlying inference procedure.

3.4.2 Top-K approximation

The previous scheme for approximate amortized inference is agnostic to the objective coefficients of integer linear program to be solved and uses only its equivalence class to find a candidate structure. The top-K approach extends this by scoring the K most frequent solutions using the objective coefficients and selecting the highest scoring one as the solution to the ILP problem. As with the previous scheme, we only consider solutions that have sufficient support.

3.4.3 Approximations to theorems 1 and 3

The next approximate inference schemes relaxes the conditions in theorems 1 and 3 by allowing the inequalities to be violated by ϵ. That is, the inequality (3) from Theorem 1 now becomes

$$(2\mathbf{y}_{\mathbf{p},i} - 1)\delta\mathbf{c}_i + \epsilon \geq 0. \quad (6)$$

The inequality (5) from Theorem 3 is similarly relaxed as follows:

$$(2\mathbf{y}_{\mathbf{p},i} - 1)\Delta\mathbf{c}_i + \epsilon \geq 0 \quad (7)$$

3.5 Amortized inference algorithm

Each exact and approximate inference approach described above specifies a condition to check whether an inference procedure should be called for a new problem. This gives us the following meta-algorithm for amortized inference, parameterized by the actual scheme used: If the given input instance $\mathbf{p}$ satisfies the condition specified by the scheme, then use the cached solution. Otherwise, call the inference procedure and cache the solution for future use.

4 Experiments

In this section, we apply the theory from Section 3 to the structure prediction problem of semantic role labeling. Since the inference schemes presented above are independent of the learning aspects, we use an off-the-shelf implementation and merely modify the inference as discussed in Section 3.5.

The goal of the experiments is to show that using an amortized inference algorithm, we can make fewer calls to the underlying inference procedure. For the exact inference algorithms, doing so will not change the performance as compared to the underlying system. For the approximations, we can make a trade-off between the inference time and performance.

4.1 Experimental setup

Our goal is to simulate a long-running NLP process that can use a cache of already solved problems to improve inference time. Given a new input problem, our theorems require us to find all elements in the equivalence class of that problem along with their solutions. Intuitively, we expect a higher probability of finding members of an arbitrary equivalence class if the size of the cache is large. Hence, we processed sentences from the Gigaword corpus and cached the inference problems for our task.

The wall-clock time is strongly dependent on such specific implementation of the components, which are independent of the main contributions of this work. Also, in most interesting applications, the computation time for each step will be typically dominated by the number of inference steps, especially with efficient implementations of caching and retrieval. Hence, the number of calls to the underlying procedure is the appropriate complexity param-

eter. Let N_{Base} be the number of times we would need to call the underlying inference procedure had we not used an amortized algorithm. (This is the same as the number of inference problems.) Let N_A be the number of times the underlying inference procedure is actually called using an amortized algorithm A. We define the *speedup* of A as

$$Speedup(A) = \frac{N_{Base}}{N_A}. \quad (8)$$

We also report the *clock speedup* of our implementation for all algorithms, which is the ratio of the wall-clock time taken by the baseline algorithm to that of the amortized algorithm. For measuring time, we only measure the time for inference as the other aspects (feature extraction, scoring, etc.) are not changed.

4.2 Semantic Role Labeling

The goal of Semantic Role Labeling (SRL) (Palmer et al., 2010) is to identify and assign semantic roles to arguments of verb predicates in a sentence. For example, consider the the sentence *John gave the ball to Mary.* The verb *give* takes three arguments, *John*, *the ball* and *to Mary*, which are labeled A0, A1 and A2 respectively.

We used the system of (Punyakanok et al., 2008) as our base SRL system. It consists of two classifiers trained on the Propbank corpus. The first one, called the argument identifier, filters argument candidates which are generated using a syntactic parse-based heuristic. The second model scores each candidate that has not been filtered for all possible argument labels. The scores for all candidates of a predicate are combined via inference. As in the system of (Punyakanok et al., 2008), the softmax function is applied to the raw classifier scores to ensure that they are in the same numeric range.

Inference mandates that certain structural and linguistic constraints hold over the full predicate-argument structure for a verb. (Punyakanok et al., 2008) modeled inference via an integer linear program instance, where each assignment of labels to candidates corresponds to one decision variable. Given a set of argument candidates, the feasible set of decisions is dependent of the number of argument candidates and the verb predicate. Thus, in terms of the notation used in this paper, the equivalence classes are defined by the pair *(predicate, number of argument candidates)*.

We ran the semantic role labeler on 225,000 verb predicates from the Gigaword corpus and cached the equivalence classes, objective coefficients and solutions generated by the SRL system. We report speedup for the various amortized inference schemes on the standard Penn Treebank test set. On this data, the unaltered baseline system, processes 5127 integer linear programs and achieves an F1 of 75.85%.

Table 3 shows the speedup and performance for the various inference schemes. The most frequent and top-K systems are both naive solutions that take advantage of the cache of stored problems. In spite of their simplicity, they attain F1 scores of 62% and 70.06% because few structures occur most frequently, as described in section 2.2. We see that all the exact theorems attain a speedup higher than two without losing performance. (The variation in F1 between them is because of the existence of different equivalent solutions in terms of the objective value.) This shows us that we can achieve an amortized gain in inference. Note that a speedup of 2.5 indicates that the solver is called only for 40% of the examples. The approximate versions of theorems 1 and 3 (with $\epsilon = 0.3$ in both cases, which was not tuned) attain an even higher gain in speedup over the baseline than the base versions of the theorems. Interestingly, the SRL performance in both cases does not decline much even though the conditions of the theorems may be violated.

5 Related work and Future directions

In recent years, we have seen several approaches to speeding up inference using ideas like using the cutting plane approach (Riedel, 2009), dual decomposition and Lagrangian relaxation (Rush et al., 2010; Chang and Collins, 2011). The key difference between these and the work in this paper is that all these approaches solve one instance at a time. Since we can use any inference procedure as a underlying system, the speedup reported in this paper is applicable to all these algorithms.

Decomposed amortized inference In this paper, we have taken advantage of redundancy of structures that can lead to the re-use of solutions. In the

Type	Algorithm	# instances	# solver calls	Speedup	Clock speedup	F1
Exact	Baseline	5127	5217	1.0	1.0	75.85
Exact	Theorem 1	5127	2134	2.44	1.54	75.90
Exact	Theorem 2	5127	2390	2.18	1.14	75.79
Exact	Theorem 3	5127	2089	2.50	1.36	75.77
Approx.	Most frequent (Support = 50)	5127	2812	1.86	1.57	62.00
Approx.	Top-10 solutions (Support = 50)	5127	2812	1.86	1.58	70.06
Approx.	Theorem 1 (approx, $\epsilon = 0.3$)	5127	1634	3.19	1.81	75.76
Approx.	Theorem 3 (approx, $\epsilon = 0.3$)	5127	1607	3.25	1.50	75.46

Table 3: Speedup and performance for various inference methods for the task of Semantic Role Labeling. All the exact inference algorithms get a speedup higher than two. The speedup of the approximate version of the theorems is even higher without loss of performance. The clock speedup is defined as the ratio of the inference times of the baseline and the given algorithm. All numbers are averaged over ten trials.

part of speech example, we showed redundancy of structures at the sentence level (Figure 2a). However, for part-of-speech tagging, the decisions are rarely, if at all, dependent on a very large context. One direction of future work is to take advantage of the fact that the inference problem can be split into smaller sub-problems. To support this hypothesis, we counted the number of occurrences of ngrams of tokens (including overlapping and repeated mentions) for $n <= 10$ and compared this to the number of unique part-of-speech ngrams of this length. Figure 4 shows these two counts. Following the argument in Section 2.2, this promises a large amortized gain in inference time. We believe that such decomposition can also be applied to other, more complex structured prediction tasks.

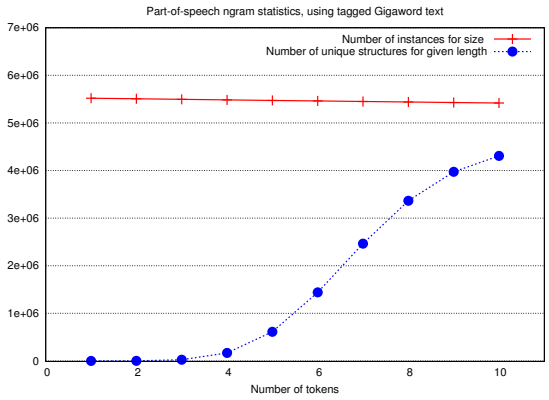


Figure 4: The red line shows the number of ngrams of tokens (including overlapping and repeated occurrences) in the Gigaword corpus and the blue line shows the number of unique POS tag sequences.

The value of approximate inference From the experiments, we see that the first two approximate inference schemes (most frequent solution and the top-K scheme) can speed up inference with the only computational cost being the check for preconditions of the exact theorems. Effectively, these algorithms have parameters (i.e., the support parameter) that allow us to choose between the inference time and performance. Figure 5 shows the performance of the most frequent and top-K baselines for different values of the support parameter, which indicates how often a structure must occur for it to be considered. We see that for lower values of support, we can get a very high speedup but pay with poorer performance.

However, the prediction of the approximate algorithms can be used to warm-start any solver that can accept an external initialization. Warm-starting a solver can give a way to get the exact solution and yet take advantage of the frequency of structures that have been observed.

Lifted inference The idea of amortizing inference time over the dataset is conceptually related to the idea of lifted inference (de Salvo Braz et al., 2005). We abstract many instances into equivalence classes and deal with the inference problem with respect to the equivalence classes in the same way as done in lifted inference algorithms.

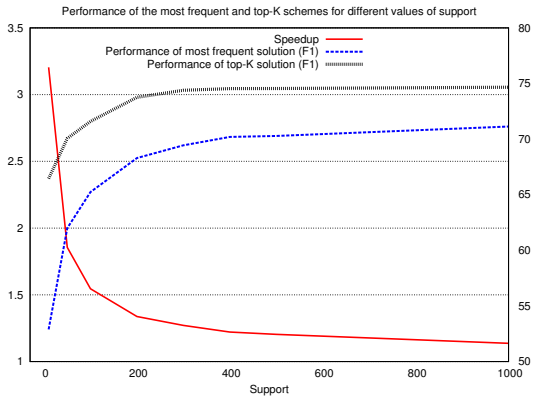


Figure 5: Most frequent solutions and top-K: Speedup and SRL performance (F1) for different values of the support parameter, using the most-frequent solutions (dashed blue line) and the top-K scheme (thick gray line). Support indicates how many times a structure should be seen for it to be considered. Note that the speedup values for both schemes are identical (red line).

6 Conclusion

In this paper, we addressed structured prediction in the context of NLP and proposed an approach to improve inference costs over an entire dataset, rather than individual instances. By treating inference problems as instances of integer linear programs, we proposed three exact theorems which identify examples for which the inference procedure need not be called at all and previous solutions can be re-used with the guarantee of optimality. In addition, we also proposed several approximate algorithms. We applied our algorithms, which are agnostic to the actual tasks, to the problem semantic role labeling, showing significant decrease in the number of inference calls without any loss in performance. While the approach suggested in this paper is evaluated in semantic role labeling, it is generally applicable to any NLP task that deals with structured prediction.

Acknowledgements

The authors wish to thank Sariel Har-Peled and the members of the Cognitive Computation Group at the University of Illinois for insightful discussions and the anonymous reviewers for their valuable feedback. This research is sponsored by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053. The authors also gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. This work is also supported by the Intelligence Advanced Research Projects Activity (IARPA) Foresight and Understanding from Scientific Exposition (FUSE) Program via Department of Interior National Business Center contract number D11PC2015. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of ARL, DARPA, AFRL, IARPA, or the US government.

References

Y-W. Chang and M. Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. EMNLP.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*.

J. Clarke and M. Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *ACL*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

R. de Salvo Braz, E. Amir, and D. Roth. 2005. Lifted first-order probabilistic inference. In *IJCAI*.

D Graff and C. Cieri. 2003. English gigaword.

A. Martins, N. A. Smith, and E. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *EMNLP*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

M. Palmer, D. Gildea, and N. Xue. 2010. *Semantic Role Labeling*, volume 3. Morgan & Claypool Publishers.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*.

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

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *EMNLP*.

S. Riedel. 2009. Cutting Plane MAP Inference for Markov Logic. *Machine Learning*.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *CoNLL*.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*.

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

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*.