Optimal Beam Search for Machine Translation

Alexander M. Rush **Yin-Wen Chang**
MIT CSAIL,
Cambridge, MA 02139, USA
{srush, yinwen}@csail.mit.edu

Michael Collins
Department of Computer Science,
Columbia University,
New York, NY 10027, USA
mcollins@cs.columbia.edu

Abstract

Beam search is a fast and empirically effective method for translation decoding, but it lacks formal guarantees about search error. We develop a new decoding algorithm that combines the speed of beam search with the optimal certificate property of Lagrangian relaxation, and apply it to phrase- and syntax-based translation decoding. The new method is efficient, utilizes standard MT algorithms, and returns an exact solution on the majority of translation examples in our test data. The algorithm is 3.5 times faster than an optimized incremental constraint-based decoder for phrase-based translation and 4 times faster for syntax-based translation.

1 Introduction

Beam search (Koehn et al., 2003) and cube pruning (Chiang, 2007) have become the *de facto* decoding algorithms for phrase- and syntax-based translation. The algorithms are central to large-scale machine translation systems due to their efficiency and tendency to produce high-quality translations (Koehn, 2004; Koehn et al., 2007; Dyer et al., 2010). However despite practical effectiveness, neither algorithm provides any bound on possible decoding error.

In this work we present a variant of beam search decoding for phrase- and syntax-based translation. The motivation is to exploit the effectiveness and efficiency of beam search, but still maintain formal guarantees. The algorithm has the following benefits:

- In theory, it can provide a certificate of optimality; in practice, we show that it produces optimal hypotheses, with certificates of optimality, on the vast majority of examples.
- It utilizes well-studied algorithms and extends off-the-shelf beam search decoders.
- Empirically it is very fast, results show that it is 3.5 times faster than an optimized incremental constraint-based solver.

While our focus is on fast decoding for machine translation, the algorithm we present can be applied to a variety of dynamic programming-based decoding problems. The method only relies on having a constrained beam search algorithm and a fast unconstrained search algorithm. Similar algorithms exist for many NLP tasks.

We begin in Section 2 by describing constrained hypergraph search and showing how it generalizes translation decoding. Section 3 introduces a variant of beam search that is, in theory, able to produce a certificate of optimality. Section 4 shows how to improve the effectiveness of beam search by using weights derived from Lagrangian relaxation. Section 5 puts everything together to derive a fast beam search algorithm that is often optimal in practice.

Experiments compare the new algorithm with several variants of beam search, cube pruning, A* search, and relaxation-based decoders on two translation tasks. The optimal beam search algorithm is able to find exact solutions with certificates of optimality on 99% of translation examples, significantly more than other baselines. Additionally the optimal

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 210–221,
Seattle, Washington, USA, 18-21 October 2013.

beam search algorithm is much faster than other exact methods.

2 Background

The focus of this work is decoding for statistical machine translation. Given a source sentence, the goal is to find the target sentence that maximizes a combination of translation model and language model scores. In order to analyze this decoding problem, we first abstract away from the specifics of translation into a general form, known as a hypergraph. In this section, we describe the hypergraph formalism and its relation to machine translation.

2.1 Notation

Throughout the paper, scalars and vectors are written in lowercase, matrices are written in uppercase, and sets are written in script-case, e.g. $\mathcal{X}$. All vectors are assumed to be column vectors. The function $\delta(j)$ yields an indicator vector with $\delta(j)_j = 1$ and $\delta(j)_i = 0$ for all $i \neq j$.

2.2 Hypergraphs and Search

A directed hypergraph is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1 \dots |\mathcal{V}|\}$ is a set of vertices, and $\mathcal{E}$ is a set of directed hyperedges. Each hyperedge $e \in \mathcal{E}$ is a tuple $\left\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \right\rangle$ where $v_i \in \mathcal{V}$ for $i \in \{1 \dots |v|\}$. The *head* of the hyperedge is $h(e) = v_1$. The *tail* of the hyperedge is the ordered sequence $t(e) = \langle v_2, \dots, v_{|v|} \rangle$. The size of the tail $|t(e)|$ may vary across different hyperedges, but $|t(e)| \geq 1$ for all edges and is bounded by a constant. A directed graph is a directed hypergraph with $|t(e)| = 1$ for all edges $e \in \mathcal{E}$.

Each vertex $v \in \mathcal{V}$ is either a *non-terminal* or a *terminal* in the hypergraph. The set of non-terminals is $\mathcal{N} = \{v \in \mathcal{V} : h(e) = v \text{ for some } e \in \mathcal{E}\}$. Conversely, the set of terminals is defined as $\mathcal{T} = \mathcal{V} \setminus \mathcal{N}$.

All directed hypergraphs used in this work are acyclic: informally this implies that no hyperpath (as defined below) contains the same vertex more than once (see Martin et al. (1990) for a full definition). Acyclicity implies a partial topological ordering of the vertices. We also assume there is a distinguished *root* vertex 1 where for all $e \in \mathcal{E}$, $1 \notin t(e)$.

Next we define a hyperpath as $x \in \{0, 1\}^{|\mathcal{E}|}$ where $x(e) = 1$ if hyperedge e is used in the hyperpath, $x(e) = 0$ otherwise. The set of valid hyperpaths is defined as

$$\mathcal{X} = \left\{ x : \begin{array}{l} \displaystyle\sum_{e \in \mathcal{E}: h(e) = 1} x(e) = 1, \\ \\ \displaystyle\sum_{e: h(e) = v} x(e) = \sum_{e: v \in t(e)} x(e) \quad \forall\, v \in \mathcal{N} \setminus \{1\} \end{array} \right\}$$

The first problem we consider is *unconstrained* hypergraph search. Let $\theta \in R^{|\mathcal{E}|}$ be the weight vector for the hypergraph and let $\tau \in \mathbb{R}$ be a weight offset.[1] The unconstrained search problem is to find

$$\max_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} \theta(e) x(e) + \tau = \max_{x \in \mathcal{X}} \theta^\top x + \tau$$

This maximization can be computed for any weights and directed acyclic hypergraph in time $O(|\mathcal{E}|)$ using dynamic programming. Figure 1 shows this algorithm which is simply a version of the CKY algorithm.

procedure BESTPATHSCORE(θ, τ)
  $\pi[v] \leftarrow 0$ for all $v \in \mathcal{T}$
  for $e \in \mathcal{E}$ in topological order **do**
    $\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \rangle \leftarrow e$
    $s \leftarrow \theta(e) + \sum_{i=2}^{|v|} \pi[v_i]$
    if $s > \pi[v_1]$ **then** $\pi[v_1] \leftarrow s$
  return $\pi[1] + \tau$

Figure 1: Dynamic programming algorithm for unconstrained hypergraph search. Note that this version only returns the highest score: $\max_{x \in \mathcal{X}} \theta^\top x + \tau$. The optimal hyperpath can be found by including back-pointers.

Next consider a variant of this problem: *constrained* hypergraph search. Constraints will be necessary for both phrase- and syntax-based decoding. In phrase-based models, the constraints will ensure that each source word is translated exactly once. In syntax-based models, the constraints will be used to intersect a translation forest with a language model.

In the constrained hypergraph problem, hyperpaths must fulfill additional linear hyperedge constraints. Define the set of constrained hyperpaths as

$$\mathcal{X}' = \{x \in \mathcal{X} : Ax = b\}$$

[1] The purpose of the offset will be clear in later sections. For this section, the value of τ can be taken as 0.

where we have a constraint matrix $A \in \mathbb{R}^{|b| \times |\mathcal{E}|}$ and vector $b \in R^{|b|}$ encoding $|b|$ constraints. The optimal constrained hyperpath is $x^* = \arg\max_{x \in \mathcal{X}'} \theta^\top x + \tau$.

Note that the constrained hypergraph search problem may be NP-Hard. Crucially this is true even when the corresponding unconstrained search problem is solvable in polynomial time. For instance, phrase-based decoding is known to be NP-Hard (Knight, 1999), but we will see that it can be expressed as a polynomial-sized hypergraph with constraints.

Example: Phrase-Based Machine Translation Consider translating a source sentence $w_1 \dots w_{|w|}$ to a target sentence in a language with vocabulary Σ. A simple phrase-based translation model consists of a tuple $(\mathcal{P}, \omega, \sigma)$ with

- $\mathcal{P}$; a set of pairs (q, r) where $q_1 \dots q_{|q|}$ is a sequence of source-language words and $r_1 \dots r_{|r|}$ is a sequence of target-language words drawn from the target vocabulary Σ.
- $\omega : \mathbb{R}^{|\mathcal{P}|}$; parameters for the translation model mapping each pair in $\mathcal{P}$ to a real-valued score.
- $\sigma : \mathbb{R}^{|\Sigma \times \Sigma|}$; parameters of the language model mapping a bigram of target-language words to a real-valued score.

The translation decoding problem is to find the best derivation for a given source sentence. A derivation consists of a sequence of *phrases* $p = p_1 \dots p_n$. Define a phrase as a tuple (q, r, j, k) consisting of a span in the source sentence $q = w_j \dots w_k$ and a sequence of target words $r_1 \dots r_{|r|}$, with $(q, r) \in \mathcal{P}$. We say the source words $w_j \dots w_k$ are *translated* to r.

The score of a derivation, $f(p)$, is the sum of the translation score of each phrase plus the language model score of the target sentence

$$f(p) = \sum_{i=1}^{n} \omega(q(p_i), r(p_i)) + \sum_{i=0}^{|u|+1} \sigma(u_{i-1}, u_i)$$

where u is the sequence of words in Σ formed by concatenating the phrases $r(p_1) \dots r(p_n)$, with boundary cases $u_0 = \texttt{<s>}$ and $u_{|u|+1} = \texttt{</s>}$.

Crucially for a derivation to be valid it must satisfy an additional condition: it must translate every source word *exactly* once. The decoding problem for phrase-based translation is to find the highest-scoring derivation satisfying this property.

We can represent this decoding problem as a constrained hypergraph using the construction of Chang and Collins (2011). The hypergraph weights encode the translation and language model scores, and its structure ensures that the count of source words translated is $|w|$, i.e. the length of the source sentence. Each vertex will remember the preceding target-language word and the count of source words translated so far.

The hypergraph, which for this problem is also a directed graph, takes the following form.

- Vertices $v \in \mathcal{V}$ are labeled (c, u) where $c \in \{1 \dots |w|\}$ is the count of source words translated and $u \in \Sigma$ is the last target-language word produced by a partial hypothesis at this vertex. Additionally there is an initial terminal vertex labeled $(0, \texttt{<s>})$.
- There is a hyperedge $e \in \mathcal{E}$ with head (c', u') and tail $\langle (c, u) \rangle$ if there is a valid corresponding phrase (q, r, j, k) such that $c' = c + |q|$ and $u' = r_{|r|}$, i.e. c' is the count of words translated and u' is the last word of target phrase r. We call this phrase $p(e)$.

  The weight of this hyperedge, $\theta(e)$, is the translation model score of the pair plus its language model score

$$\theta(e) = \omega(q, r) + \left(\sum_{i=2}^{|r|} \sigma(r_{i-1}, r_i) \right) + \sigma(u, r_1)$$

- To handle the end boundary, there are hyperedges with head 1 and tail $\langle (|w|, u) \rangle$ for all $u \in \Sigma$. The weight of these edges is the cost of the stop bigram following u, i.e. $\sigma(u, \texttt{</s>})$.

While any valid derivation corresponds to a hyperpath in this graph, a hyperpath may not correspond to a valid derivation. For instance, a hyperpath may translate some source words more than once or not at all.

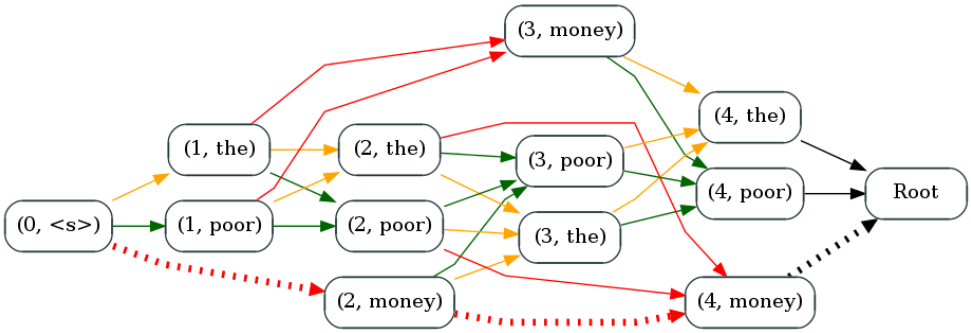

Figure 2: Hypergraph for translating the sentence $w = \texttt{les}_1\ \texttt{pauvres}_2\ \texttt{sont}_3\ \texttt{demunis}_4$ with set of pairs $\mathcal{P} = \{(\texttt{les}, \texttt{the}), (\texttt{pauvres}, \texttt{poor}), (\texttt{sont demunis}, \texttt{don't have any money})\}$. Hyperedges are color-coded by source words translated: orange for $\texttt{les}_1$, green for $\texttt{pauvres}_2$, and red for $\texttt{sont}_3\ \texttt{demunis}_4$. The dotted lines show an invalid hyperpath x that has signature $Ax = \langle 0, 0, 2, 2 \rangle \neq \langle 1, 1, 1, 1 \rangle$.

We handle this problem by adding additional constraints. For all source words $i \in \{1 \dots |w|\}$, define ρ as the set of hyperedges that translate w_i

$$\rho(i) = \{e \in \mathcal{E} : j(p(e)) \leq i \leq k(p(e))\}$$

Next define $|w|$ constraints enforcing that each word in the source sentence is translated exactly once

$$\sum_{e \in \rho(i)} x(e) = 1 \quad \forall\, i \in \{1 \dots |w|\}$$

These linear constraints can be represented with a matrix $A \in \{0, 1\}^{|w| \times |\mathcal{E}|}$ where the rows correspond to source indices and the columns correspond to edges. We call the product Ax the *signature*, where in this case $(Ax)_i$ is the number of times word i has been translated. The full set of constrained hyperpaths is $\mathcal{X}' = \{x \in \mathcal{X} : Ax = \mathbf{1}\}$, and the best derivation under this phrase-based translation model has score $\max_{x \in \mathcal{X}'} \theta^\top x + \tau$.

Figure 2.2 shows an example hypergraph with constraints for translating the sentence `les pauvres sont demunis` into English using a simple set of phrases. Even in this small example, many of the possible hyperpaths violate the constraints and correspond to invalid derivations.

Example: Syntax-Based Machine Translation Syntax-based machine translation with a language model can also be expressed as a constrained hypergraph problem. For the sake of space, we omit the definition. See Rush and Collins (2011) for an in-depth description of the constraint matrix used for syntax-based translation.

3 A Variant of Beam Search

This section describes a variant of the beam search algorithm for finding the highest-scoring constrained hyperpath. The algorithm uses three main techniques: (1) dynamic programming with additional signature information to satisfy the constraints, (2) beam pruning where some, possibly optimal, hypotheses are discarded, and (3) branch-and-bound-style application of upper and lower bounds to discard provably non-optimal hypotheses.

Any solution returned by the algorithm will be a valid constrained hyperpath and a member of $\mathcal{X}'$. Additionally the algorithm returns a certificate flag opt that, if true, indicates that no beam pruning was used, implying the solution returned is optimal. Generally it will be hard to produce a certificate even by reducing the amount of beam pruning; however in the next section we will introduce a method based on Lagrangian relaxation to tighten the upper bounds. These bounds will help eliminate most solutions before they trigger pruning.

3.1 Algorithm

Figure 3 shows the complete beam search algorithm. At its core it is a dynamic programming algorithm filling in the chart π. The beam search chart indexes hypotheses by vertex $v \in \mathcal{V}$ as well as a signature $sig \in \mathbb{R}^{|b|}$ where $|b|$ is the number of constraints. A new hypothesis is constructed from each hyperedge and all possible signatures of tail nodes. We define the function SIGS to take the tail of an edge and re-

turn the set of possible signature combinations

$$\textsc{Sigs}(v_2, \dots v_{|v|}) = \prod_{i=2}^{|v|} \{sig : \pi[v_i, sig] \neq -\infty\}$$

where the product is the Cartesian product over sets. Line 8 loops over this entire set.[2] For hypothesis x, the algorithm ensures that its signature sig is equal to Ax. This property is updated on line 9.

The signature provides proof that a hypothesis is still valid. Let the function $\textsc{Check}(sig)$ return true if the hypothesis can still fulfill the constraints. For example, in phrase-based decoding, we will define $\textsc{Check}(sig) = (sig \leq \mathbf{1})$; this ensures that each word has been translated 0 or 1 times. This check is applied on line 11.

Unfortunately maintaining all signatures is inefficient. For example we will see that in phrase-based decoding the signature is a bit-string recording which source words have been translated; the number of possible bit-strings is exponential in the length of the sentence. The algorithm includes two methods for removing hypotheses, bounding and pruning.

Bounding allows us to discard provably non-optimal solutions. The algorithm takes as arguments a lower bound on the optimal score $\mathrm{lb} \leq \theta^\top x^* + \tau$, and computes upper bounds on the outside score for all vertices v: $\mathrm{ubs}[v]$, i.e. an overestimate of the score for completing the hyperpath from v. If a hypothesis has score s, it can only be optimal if $s + \mathrm{ubs}[v] \geq \mathrm{lb}$. This bound check is performed on line 11.

Pruning removes weak partial solutions based on problem-specific checks. The algorithm invokes the black-box function, PRUNE, on line 13, passing it a pruning parameter β and a vertex-signature pair. The parameter β controls a threshold for pruning. For instance for phrase-based translation, it specifies a hard-limit on the number of hypotheses to retain. The function returns true if it prunes from the chart. Note that pruning may remove optimal hypotheses, so we set the certificate flag opt to false if the chart is modified.

[2]For simplicity we write this loop over the entire set. In practice it is important to use data structures to optimize lookup. See Tillmann (2006) and Huang and Chiang (2005).

1: **procedure** BEAMSEARCH($\theta, \tau, \mathrm{lb}, \beta$)
2: $\mathrm{ubs} \leftarrow \textsc{Outside}(\theta, \tau)$
3: $\mathrm{opt} \leftarrow \mathrm{true}$
4: $\pi[v, sig] \leftarrow -\infty$ for all $v \in \mathcal{V}, sig \in \mathbb{R}^{|b|}$
5: $\pi[v, 0] \leftarrow 0$ for all $v \in \mathcal{T}$
6: **for** $e \in \mathcal{E}$ in topological order **do**
7: $\langle \langle v_2, \dots, v_{|v|} \rangle, v_1 \rangle \leftarrow e$
8: **for** $sig^{(2)} \dots sig^{(|v|)} \in \textsc{Sigs}(v_2, \dots, v_{|v|})$ **do**
9: $sig \leftarrow A\delta(e) + \sum_{i=2}^{|v|} sig^{(i)}$
10: $s \leftarrow \theta(e) + \sum_{i=2}^{|v|} \pi[v_i, sig^{(i)}]$
11: **if** $\begin{pmatrix} s > \pi[v_1, sig] \wedge \\ \textsc{Check}(sig) \wedge \\ s + \mathrm{ubs}[v_1] \geq \mathrm{lb} \end{pmatrix}$ **then**
12: $\pi[v_1, sig] \leftarrow s$
13: **if** $\textsc{Prune}(\pi, v_1, sig, \beta)$ **then** $\mathrm{opt} \leftarrow \mathrm{false}$
14: $\mathrm{lb}' \leftarrow \pi[1, c] + \tau$
15: **return** $\mathrm{lb}', \mathrm{opt}$

Input:

$(\mathcal{V}, \mathcal{E}, \theta, \tau)$	hypergraph with weights
(A, b)	matrix and vector for constraints
$\mathrm{lb} \in \mathbb{R}$	lower bound
β	a pruning parameter

Output:

lb'	resulting lower bound score
opt	certificate of optimality

Figure 3: A variant of the beam search algorithm. Uses dynamic programming to produce a lower bound on the optimal constrained solution and, possibly, a certificate of optimality. Function OUTSIDE computes upper bounds on outside scores. Function SIGS enumerates all possible tail signatures. Function CHECK identifies signatures that do not violate constraints. Bounds lb and ubs are used to remove provably non-optimal solutions. Function PRUNE, taking parameter β, returns true if it prunes hypotheses from π that could be optimal.

This variant on beam search satisfies the following two properties (recall x^* is the optimal constrained solution)

Property 3.1 (Primal Feasibility)**.** *The returned score* lb' *lower bounds the optimal constrained score, that is* $\mathrm{lb}' \leq \theta^\top x^* + \tau$.

Property 3.2 (Dual Certificate)**.** *If beam search returns with* $\mathrm{opt} = \mathrm{true}$*, then the returned score is optimal, i.e.* $\mathrm{lb}' = \theta^\top x^* + \tau$.

An immediate consequence of Property 3.1 is that the output of beam search, lb', can be used as the input lb for future runs of the algorithm. Furthermore,

procedure PRUNE(π, v, sig, β)
  $\mathcal{C} \leftarrow \{(v', sig') : ||sig'||_1 = ||sig||_1, \pi[v', sig'] \neq -\infty\}$
  $\mathcal{D} \leftarrow \mathcal{C} \setminus \text{mBEST}(\beta, \mathcal{C}, \pi)$
  $\pi[v', sig'] \leftarrow -\infty$ for all $v', sig' \in \mathcal{D}$
  if $\mathcal{D} = \emptyset$ **then return** true
  else return false

Input: (v, sig) the last hypothesis added to the chart
$\beta \in \mathbb{Z}$ # of hypotheses to retain
Output: true, if π is modified

Figure 4: Pruning function for phrase-based translation. Set $\mathcal{C}$ contains all hypotheses with $||sig||_1$ source words translated. The function prunes all but the top-β scoring hypotheses in this set.

if we loosen the amount of beam pruning by adjusting the pruning parameter β we can produce tighter lower bounds and discard more hypotheses. We can then iteratively apply this idea with a sequence of parameters $\beta_1 \dots \beta_K$ producing lower bounds $\text{lb}^{(1)}$ through $\text{lb}^{(K)}$. We return to this idea in Section 5.

Example: Phrase-based Beam Search. Recall that the constraints for phrase-based translation consist of a binary matrix $A \in \{0,1\}^{|w| \times |\mathcal{E}|}$ and vector $b = \mathbf{1}$. The value sig_i is therefore the number of times source word i has been translated in the hypothesis. We define the predicate CHECK as CHECK$(sig) = (sig \leq \mathbf{1})$ in order to remove hypotheses that already translate a source word more than once, and are therefore invalid. For this reason, phrase-based signatures are called *bit-strings*.

A common beam pruning strategy is to group together items into a set $\mathcal{C}$ and retain a (possibly complete) subset. An example phrase-based beam pruner is given in Figure 4. It groups together hypotheses based on $||sig_i||_1$, i.e. the number of source words translated, and applies a hard pruning filter that retains only the β highest-scoring items $(v, sig) \in \mathcal{C}$ based on $\pi[v, sig]$.

3.2 Computing Upper Bounds

Define the set $\mathcal{O}(v, x)$ to contain all outside edges of vertex v in hyperpath x (informally, hyperedges that do not have v as an ancestor). For all $v \in \mathcal{V}$, we set the upper bounds, ubs, to be the best unconstrained outside score

$$\text{ubs}[v] = \max_{x \in \mathcal{X}: v \in x} \sum_{e \in \mathcal{O}(v,x)} \theta(e) + \tau$$

This upper bound can be efficiently computed for all vertices using the standard outside dynamic programming algorithm. We will refer to this algorithm as OUTSIDE(θ, τ).

Unfortunately, as we will see, these upper bounds are often quite loose. The issue is that unconstrained outside paths are able to violate the constraints without being penalized, and therefore greatly overestimate the score.

4 Finding Tighter Bounds with Lagrangian Relaxation

Beam search produces a certificate only if beam pruning is never used. In the case of phrase-based translation, the certificate is dependent on all groups $\mathcal{C}$ having β or less hypotheses. The only way to ensure this is to bound out enough hypotheses to avoid pruning. The effectiveness of the bounding inequality, $s + \text{ubs}[v] < \text{lb}$, in removing hypotheses is directly dependent on the tightness of the bounds.

In this section we propose using Lagrangian relaxation to improve these bounds. We first give a brief overview of the method and then apply it to computing bounds. Our experiments show that this approach is very effective at finding certificates.

4.1 Algorithm

In Lagrangian relaxation, instead of solving the constrained search problem, we relax the constraints and solve an unconstrained hypergraph problem with modified weights. Recall the constrained hypergraph problem: $\max_{x \in \mathcal{X}: Ax = b} \theta^\top x + \tau$. The Lagrangian dual of this optimization problem is

$$\begin{aligned} L(\lambda) &= \max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b) \\ &= \left(\max_{x \in \mathcal{X}} (\theta - A^\top \lambda)^\top x \right) + \tau + \lambda^\top b \\ &= \max_{x \in \mathcal{X}} \theta'^\top x + \tau' \end{aligned}$$

where $\lambda \in R^{|b|}$ is a vector of dual variables and define $\theta' = \theta - A^\top \lambda$ and $\tau' = \tau + \lambda^\top b$. This maximization is over $\mathcal{X}$, so for any value of λ, $L(\lambda)$ can be calculated as `BestPathScore`(θ', τ').

Note that for all valid constrained hyperpaths $x \in \mathcal{X}'$ the term $Ax - b$ equals 0, which implies that these hyperpaths have the same score under the modified weights as under the original weights, $\theta^\top x + \tau = \theta'^\top x + \tau'$. This leads to the following two properties,

procedure LRROUND(α_k, λ)
$x \leftarrow \arg\max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b)$
$\lambda' \leftarrow \lambda - \alpha_k (Ax - b)$
$\text{opt} \leftarrow Ax = b$
$\text{ub} \leftarrow \theta^\top x + \tau$
return $\lambda', \text{ub}, \text{opt}$
procedure LAGRANGIANRELAXATION(α)
$\lambda^{(0)} \leftarrow 0$
for k in $1 \dots K$ **do**
$\lambda^{(k)}, \text{ub}, \text{opt} \leftarrow \text{LRROUND}(\alpha_k, \lambda^{(k-1)})$
if opt **then return** $\lambda^{(k)}, \text{ub}, \text{opt}$
return $\lambda^{(K)}, \text{ub}, \text{opt}$

Input: $\alpha_1 \dots \alpha_K$ sequence of subgradient rates

Output:
- λ final dual vector
- ub upper bound on optimal constrained solution
- opt certificate of optimality

Figure 5: Lagrangian relaxation algorithm. The algorithm repeatedly calls LRROUND to compute the subgradient, update the dual vector, and check for a certificate.

where $x \in \mathcal{X}$ is the hyperpath computed within the max,

Property 4.1 (Dual Feasibility). *The value $L(\lambda)$ upper bounds the optimal solution, that is $L(\lambda) \geq \theta^\top x^* + \tau$*

Property 4.2 (Primal Certificate). *If the hyperpath x is a member of $\mathcal{X}'$, i.e. $Ax = b$, then $L(\lambda) = \theta^\top x^* + \tau$.*

Property 4.1 states that $L(\lambda)$ always produces some upper bound; however, to help beam search, we want as tight a bound as possible: $\min_\lambda L(\lambda)$.

The Lagrangian relaxation algorithm, shown in Figure 5, uses subgradient descent to find this minimum. The subgradient of $L(\lambda)$ is $Ax - b$ where x is the argmax of the modified objective $x = \arg\max_{x \in \mathcal{X}} \theta'^\top x + \tau'$. Subgradient descent iteratively solves unconstrained hypergraph search problems to compute these subgradients and updates λ. See Rush and Collins (2012) for an extensive discussion of this style of optimization in natural language processing.

Example: Phrase-based Relaxation. For phrase-based translation, we expand out the Lagrangian to

$$L(\lambda) = \max_{x \in \mathcal{X}} \theta^\top x + \tau - \lambda^\top (Ax - b) =$$
$$\max_{x \in \mathcal{X}} \sum_{e \in \mathcal{E}} \left(\theta(e) - \sum_{i=j(p(e))}^{k(p(e))} \lambda_i \right) x(e) + \tau + \sum_{i=1}^{|s|} \lambda_i$$

The weight of each edge $\theta(e)$ is modified by the dual variables λ_i for each source word translated by the edge, i.e. if $(q, r, j, k) = p(e)$, then the score is modified by $\sum_{i=j}^{k} \lambda_i$. A solution under these weights may use source words multiple times or not at all. However if the solution uses each source word exactly once ($Ax = \mathbf{1}$), then we have a certificate and the solution is optimal.

4.2 Utilizing Upper Bounds in Beam Search

For many problems, it may not be possible to satisfy Property 4.2 by running the subgradient algorithm alone. Yet even for these problems, applying subgradient descent will produce an improved estimate of the upper bound, $\min_\lambda L(\lambda)$.

To utilize these improved bounds, we simply replace the weights in beam search and the outside algorithm with the modified weights from Lagrangian relaxation, θ' and τ'. Since the result of beam search must be a valid constrained hyperpath $x \in \mathcal{X}'$, and for all $x \in \mathcal{X}'$, $\theta^\top x + \tau = \theta'^\top x + \tau'$, this substitution does not alter the necessary properties of the algorithm; i.e. if the algorithm returns with opt equal to true, then the solution is optimal.

Additionally the computation of upper bounds now becomes

$$\text{ubs}[v] = \max_{x \in \mathcal{X}: v \in x} \sum_{e \in \mathcal{O}(v,x)} \theta'(e) + \tau'$$

These outside paths may still violate constraints, but the modified weights now include penalty terms to discourage common violations.

5 Optimal Beam Search

The optimality of the beam search algorithm is dependent on the tightness of the upper and lower bounds. We can produce better lower bounds by varying the pruning parameter β; we can produce better upper bounds by running Lagrangian relaxation. In this section we combine these two ideas and present a complete optimal beam search algorithm.

Our general strategy will be to use Lagrangian relaxation to compute modified weights and to use beam search over these modified weights to attempt to find an optimal solution. One simple method for doing this, shown at the top of Figure 6, is to run

in stages. The algorithm first runs Lagrangian relaxation to compute the best λ vector. The algorithm then iteratively runs beam search using the parameter sequence β_k. These parameters allow the algorithm to loosen the amount of beam pruning. For example in phrase based pruning, we would raise the number of hypotheses stored per group until no beam pruning occurs.

A clear disadvantage of the staged approach is that it needs to wait until Lagrangian relaxation is completed before even running beam search. Often beam search will be able to quickly find an optimal solution even with good but non-optimal λ. In other cases, beam search may still improve the lower bound lb.

This motivates the alternating algorithm OPTBEAM shown Figure 6. In each round, the algorithm alternates between computing subgradients to tighten ubs and running beam search to maximize lb. In early rounds we set β for aggressive beam pruning, and as the upper bounds get tighter, we loosen pruning to try to get a certificate. If at any point either a primal or dual certificate is found, the algorithm returns the optimal solution.

procedure OPTBEAMSTAGED(α, β)
  $\lambda, \mathrm{ub}, \mathrm{opt} \leftarrow$LAGRANGIANRELAXATION($\alpha$)
  if opt **then return** ub
  $\theta' \leftarrow \theta - A^\top \lambda$
  $\tau' \leftarrow \tau + \lambda^\top b$
  $\mathrm{lb}^{(0)} \leftarrow -\infty$
  for k in $1 \dots K$ **do**
    $\mathrm{lb}^{(k)}, \mathrm{opt} \leftarrow$ BEAMSEARCH($\theta', \tau', \mathrm{lb}^{(k-1)}, \beta_k$)
    if opt **then return** $\mathrm{lb}^{(k)}$
  return $\max_{k \in \{1 \dots K\}} \mathrm{lb}^{(k)}$

procedure OPTBEAM(α, β)
  $\lambda^{(0)} \leftarrow 0$
  $\mathrm{lb}^{(0)} \leftarrow -\infty$
  for k in $1 \dots K$ **do**
    $\lambda^{(k)}, \mathrm{ub}^{(k)}, \mathrm{opt} \leftarrow$ LRROUND($\alpha_k, \lambda^{(k-1)}$)
    if opt **then return** $\mathrm{ub}^{(k)}$
    $\theta' \leftarrow \theta - A^\top \lambda^{(k)}$
    $\tau' \leftarrow \tau + \lambda^{(k)\top} b$
    $\mathrm{lb}^{(k)}, \mathrm{opt} \leftarrow$ BEAMSEARCH($\theta', \tau', \mathrm{lb}^{(k-1)}, \beta_k$)
    if opt **then return** $\mathrm{lb}^{(k)}$
  return $\max_{k \in \{1 \dots K\}} \mathrm{lb}^{(k)}$

Input: $\begin{bmatrix} \alpha_1 \dots \alpha_K & \text{sequence of subgradient rates} \\ \beta_1 \dots \beta_K & \text{sequence of pruning parameters} \end{bmatrix}$
Output: optimal constrained score or lower bound

Figure 6: Two versions of optimal beam search: staged and alternating. Staged runs Lagrangian relaxation to find the optimal λ, uses λ to compute upper bounds, and then repeatedly runs beam search with pruning sequence $\beta_1 \dots \beta_k$. Alternating switches between running a round of Lagrangian relaxation and a round of beam search with the updated λ. If either produces a certificate it returns the result.

6 Related Work

Approximate methods based on beam search and cube-pruning have been widely studied for phrase-based (Koehn et al., 2003; Tillmann and Ney, 2003; Tillmann, 2006) and syntax-based translation models (Chiang, 2007; Huang and Chiang, 2007; Watanabe et al., 2006; Huang and Mi, 2010).

There is a line of work proposing exact algorithms for machine translation decoding. Exact decoders are often slow in practice, but help quantify the errors made by other methods. Exact algorithms proposed for IBM model 4 include ILP (Germann et al., 2001), cutting plane (Riedel and Clarke, 2009), and multi-pass A* search (Och et al., 2001). Zaslavskiy et al. (2009) formulate phrase-based decoding as a traveling salesman problem (TSP) and use a TSP decoder. Exact decoding algorithms based on finite state transducers (FST) (Iglesias et al., 2009) have been studied on phrase-based models with limited reordering (Kumar and Byrne, 2005). Exact decoding based on FST is also feasible for certain hierarchical grammars (de Gispert et al., 2010). Chang and Collins (2011) and Rush and Collins (2011) develop Lagrangian relaxation-based approaches for exact machine translation.

Apart from translation decoding, this paper is closely related to work on column generation for NLP. Riedel et al. (2012) and Belanger et al. (2012) relate column generation to beam search and produce exact solutions for parsing and tagging problems. The latter work also gives conditions for when beam search-style decoding is optimal.

7 Results

To evaluate the effectiveness of optimal beam search for translation decoding, we implemented decoders for phrase- and syntax-based models. In this section we compare the speed and optimality of these

decoders to several baseline methods.

7.1 Setup and Implementation

For phrase-based translation we used a German-to-English data set taken from Europarl (Koehn, 2005). We tested on 1,824 sentences of length at most 50 words. For experiments the phrase-based systems uses a trigram language model and includes standard distortion penalties. Additionally the unconstrained hypergraph includes further derivation information similar to the graph described in Chang and Collins (2011).

For syntax-based translation we used a Chinese-to-English data set. The model and hypergraphs come from the work of Huang and Mi (2010). We tested on 691 sentences from the newswire portion of the 2008 NIST MT evaluation test set. For experiments, the syntax-based model uses a trigram language model. The translation model is tree-to-string syntax-based model with a standard context-free translation forest. The constraint matrix A is based on the constraints described by Rush and Collins (2011).

Our decoders use a two-pass architecture. The first pass sets up the hypergraph in memory, and the second pass runs search. When possible the baselines share optimized construction and search code.

The performance of optimal beam search is dependent on the sequences α and β. For the step-size α we used a variant of Polyak's rule (Polyak, 1987; Boyd and Mutapcic, 2007), substituting the unknown optimal score for the last computed lower bound: $\alpha_k \leftarrow \frac{\text{ub}^{(k)} - \text{lb}^{(k)}}{||Ax^{(k)} - b||_2^2}$. We adjust the order of the pruning parameter β based on a function μ of the current gap: $\beta_k \leftarrow 10^{\mu(\text{ub}^{(k)} - \text{lb}^{(k)})}$.

Previous work on these data sets has shown that exact algorithms do not result in a significant increase in translation accuracy. We focus on the efficiency and model score of the algorithms.

7.2 Baseline Methods

The experiments compare optimal beam search (OPTBEAM) to several different decoding methods. For both systems we compare to: BEAM, the beam search decoder from Figure 3 using the original weights θ and τ, and $\beta \in \{100, 1000\}$; LR-TIGHT, Lagrangian relaxation followed by incremental tightening constraints, which is a reimplementation of Chang and Collins (2011) and Rush and Collins (2011).

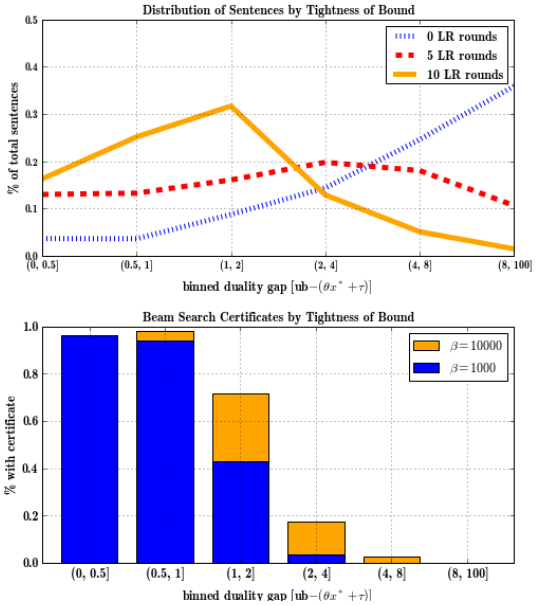

Figure 7: Two graphs from phrase-based decoding. Graph (a) shows the duality gap distribution for 1,824 sentences after 0, 5, and 10 rounds of LR. Graph (b) shows the % of certificates found for sentences with differing gap sizes and beam search parameters β. Duality gap is defined as, ub - $(\theta^\top x^* + \tau)$.

For phrase-based translation we compare with: MOSES-GC, the standard Moses beam search decoder with $\beta \in \{100, 1000\}$ (Koehn et al., 2007); MOSES, a version of Moses without gap constraints more similar to BEAM (see Chang and Collins (2011)); ASTAR, an implementation of A* search using original outside scores, i.e. OUTSIDE(θ, τ), and capped at 20,000,000 queue pops.

For syntax-based translation we compare with: ILP, a general-purpose integer linear programming solver (Gurobi Optimization, 2013) and CUBEPRUNING, an approximate decoding method similar to beam search (Chiang, 2007), tested with $\beta \in \{100, 1000\}$.

7.3 Experiments

Table 1 shows the main results. For phrase-based translation, OPTBEAM decodes the optimal translation with certificate in 99% of sentences with an average time of 17.27 seconds per sentence. This

	11-20 (558)			21-30 (566)			31-40 (347)			41-50 (168)			all (1824)		
Phrase-Based	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
BEAM (100)	2.33	19.5	38.0	8.37	1.6	7.2	24.12	0.3	1.4	71.35	0.0	0.0	14.50	15.3	23.2
BEAM (1000)	2.33	37.8	66.3	8.42	3.4	18.9	21.60	0.6	3.2	53.99	0.6	1.2	12.44	22.6	36.9
BEAM (100000)	3.34	83.9	96.2	18.53	22.4	60.4	46.65	2.0	18.1	83.53	1.2	6.5	23.39	43.2	62.4
MOSES (100)	0.18	0.0	81.0	0.36	0.0	45.6	0.53	0.0	14.1	0.74	0.0	6.0	0.34	0.0	52.3
MOSES (1000)	2.29	0.0	97.8	4.39	0.0	78.8	6.52	0.0	43.5	9.00	0.0	19.6	4.20	0.0	74.6
ASTAR (cap)	11.11	99.3	99.3	91.39	53.9	53.9	122.67	7.8	7.8	139.61	1.2	1.2	67.99	58.8	58.8
LR-TIGHT	4.20	100.0	100.0	23.25	100.0	100.0	88.16	99.7	99.7	377.9	97.0	97.0	60.11	99.7	99.7
OPTBEAM	2.85	100.0	100.0	10.33	100.0	100.0	28.29	100.0	100.0	84.34	97.0	97.0	17.27	99.7	99.7
ChangCollins	10.90	100.0	100.0	57.20	100.0	100.0	203.4	99.7	99.7	679.9	97.0	97.0	120.9	99.7	99.7
MOSES-GC (100)	0.14	0.0	89.4	0.27	0.0	84.1	0.41	0.0	75.8	0.58	0.0	78.6	0.26	0.0	84.9
MOSES-GC (1000)	1.33	0.0	89.4	2.62	0.0	84.3	4.15	0.0	75.8	6.19	0.0	79.2	2.61	0.0	85.0

	11-20 (192)			21-30 (159)			31-40 (136)			41-100 (123)			all (691)		
Syntax-Based	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact	time	cert	exact
BEAM (100)	0.40	4.7	75.9	0.40	0.0	66.0	0.75	0.0	43.4	1.66	0.0	25.8	0.68	5.72	58.7
BEAM (1000)	0.78	16.9	79.4	2.65	0.6	67.1	6.20	0.0	47.5	15.5	0.0	36.4	4.16	12.5	65.5
CUBE (100)	0.08	0.0	77.6	0.16	0.0	66.7	0.23	0.0	43.9	0.41	0.0	26.3	0.19	0.0	59.0
CUBE (1000)	1.76	0.0	91.7	4.06	0.0	95.0	5.71	0.0	82.9	10.69	0.0	60.9	4.66	0.0	85.0
LR-TIGHT	0.37	100.0	100.0	1.76	100.0	100.0	4.79	100.0	100.0	30.85	94.5	94.5	7.25	99.0	99.0
OPTBEAM	0.23	100.0	100.0	0.50	100.0	100.0	1.42	100.0	100.0	7.14	93.6	93.6	1.75	98.8	98.8
ILP	9.15	100.0	100.0	32.35	100.0	100.0	49.6	100.0	100.0	108.6	100.0	100.0	40.1	100.0	100.0

Table 1: Experimental results for translation experiments. Column *time* is the mean time per sentence in seconds, *cert* is the percentage of sentences solved with a certificate of optimality, *exact* is the percentage of sentences solved exactly, i.e. $\theta^\top x + \tau = \theta^\top x^* + \tau$. Results are grouped by sentence length (group 1-10 is omitted for space).

is seven times faster than the decoder of Chang and Collins (2011) and 3.5 times faster then our reimplementation, LR-TIGHT. ASTAR performs poorly, taking lots of time on difficult sentences. BEAM runs quickly, but rarely finds an exact solution. MOSES without gap constraints is also fast, but less exact than OPTBEAM and unable to produce certificates.

For syntax-based translation. OPTBEAM finds a certificate on 98.8% of solutions with an average time of 1.75 seconds per sentence, and is four times faster than LR-TIGHT. CUBE (100) is an order of magnitude faster, but is rarely exact on longer sentences. CUBE (1000) finds more exact solutions, but is comparable in speed to optimal beam search. BEAM performs better than in the phrase-based model, but is not much faster than OPTBEAM.

Figure 7.2 shows the relationship between beam search optimality and duality gap. Graph (a) shows how a handful of LR rounds can significantly tighten the upper bound score of many sentences. Graph (b) shows how beam search is more likely to find optimal solutions with tighter bounds. BEAM effectively uses 0 rounds of LR, which may explain why it finds so few optimal solutions compared to OPTBEAM.

Table 2 breaks down the time spent in each part of the algorithm. For both methods, beam search has the most time variance and uses more time on longer sentences. For phrase-based sentences, Lagrangian relaxation is fast, and hypergraph construction dominates. If not for this cost, OPTBEAM might be comparable in speed to MOSES (1000).

		≥ 30		all	
		mean	median	mean	median
PB	Hypergraph	56.6%	69.8%	59.6%	69.6%
	Lag. Relaxation	10.0%	5.5%	9.4%	7.6%
	Beam Search	33.4%	24.6%	30.9%	22.8%
SB	Hypergraph	0.5%	1.6%	0.8%	2.4%
	Lag. Relaxation	15.0%	35.2%	17.3%	41.4%
	Beam Search	84.4%	63.1%	81.9 %	56.1%

Table 2: Distribution of time within optimal beam search, including: hypergraph construction, Lagrangian relaxation, and beam search. *Mean* is the percentage of total time. *Median* is the distribution over the median values for each row.

8 Conclusion

In this work we develop an optimal variant of beam search and apply it to machine translation decoding. The algorithm uses beam search to produce constrained solutions and bounds from Lagrangian relaxation to eliminate non-optimal solutions. Results show that this method can efficiently find exact solutions for two important styles of machine translation.

Acknowledgments Alexander Rush, Yin-Wen Chang and Michael Collins were all supported by NSF grant IIS-1161814. Alexander Rush was partially supported by an NSF Graduate Research Fellowship.

References

David Belanger, Alexandre Passos, Sebastian Riedel, and Andrew McCallum. 2012. Map inference in chains using column generation. In *NIPS*, pages 1853–1861.

Stephen Boyd and Almir Mutapcic. 2007. Subgradient methods.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 26–37. Association for Computational Linguistics.

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Adria de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical Phrase-Based Translation with Weighted Finite-State Transducers and Shallow-n Grammars. In *Computational linguistics*, volume 36, pages 505–533.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathen Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vlad Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 228–235.

Inc. Gurobi Optimization. 2013. Gurobi optimizer reference manual.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 380–388, Athens, Greece, March. Association for Computational Linguistics.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL '03, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. *Machine translation: From real users to research*, pages 115–124.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

R. Kipp Martin, Rardin L. Rardin, and Brian A. Campbell. 1990. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of the workshop on Data-driven methods in machine translation - Volume 14*, DMMT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Boris Polyak. 1987. *Introduction to Optimization*. Optimization Software, Inc.

Sebastian Riedel and James Clarke. 2009. Revisiting optimal decoding for machine translation IBM model 4. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 5–8. Association for Computational Linguistics.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut: delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational*

Natural Language Learning, pages 732–743. Association for Computational Linguistics.

Alexander M Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 72–82.

Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Christoph Tillmann. 2006. Efficient dynamic programming search algorithms for phrase-based SMT. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, CHSLP '06, pages 9–16.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 777–784, Morristown, NJ, USA. Association for Computational Linguistics.

Mikhail Zaslavskiy, Marc Dymetman, and Nicola Cancedda. 2009. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 333–341, Stroudsburg, PA, USA. Association for Computational Linguistics.