On graph-based reentrancy-free semantic parsing

Alban Petit and **Caio Corro**
Universite Paris-Saclay, CNRS, LISN, 91400, Orsay, France
{alban.petit,caio.corro}@lisn.upsaclay.fr

Abstract

We propose a novel graph-based approach for semantic parsing that resolves two problems observed in the literature: (1) seq2seq models fail on compositional generalization tasks; (2) previous work using phrase structure parsers cannot cover all the semantic parses observed in treebanks. We prove that both MAP inference and latent tag anchoring (required for weakly-supervised learning) are NP-hard problems. We propose two optimization algorithms based on constraint smoothing and conditional gradient to approximately solve these inference problems. Experimentally, our approach delivers state-of-the-art results on GEOQUERY, SCAN and CLEVR, both for i.i.d. splits and for splits that test for compositional generalization.

1 Introduction

Semantic parsing aims to transform a natural language utterance into a structured representation that can be easily manipulated by a software (for example to query a database). As such, it is a central task in human-computer interfaces. Andreas et al. (2013) first proposed to rely on machine translation models for semantic parsing, where the target representation is linearized and treated as a foreign language. Due to recent advances in deep learning and especially in sequence-to-sequence (seq2seq) with attention architectures for machine translation (Bahdanau et al., 2015), it is appealing to use the same architectures for standard structured prediction problems (Vinyals et al., 2015). This approach is indeed common in semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016; Wang et al., 2020), *inter alia*. Unfortunately, there are well known limitations to seq2seq architectures for semantic parsing. First, at test time, the decoding algorithm is typically based on beam search as the model is autoregressive and does not make any independence assumption. In case of prediction failure, it is therefore unknown if this is due to errors in the weighting function or to the optimal solution failing out of the beam. Secondly, they are known to fail when compositional generalization is required (Lake and Baroni, 2018; Finegan-Dollak et al., 2018a; Keysers et al., 2020).

In order to bypass these problems, Herzig and Berant (2021) proposed to represent the semantic content associated with an utterance as a phrase structure, *i.e.* using the same representation usually associated with syntactic constituents. As such, their semantic parser is based on standard span-based decoding algorithms (Hall et al., 2014; Stern et al., 2017; Corro, 2020) with additional well-formedness constraints from the semantic formalism. Given a weighting function, MAP inference is a polynomial time problem that can be solved via a variant of the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke, 1970). Experimentally, Herzig and Berant (2021) show that their approach outperforms seq2seq models in terms of compositional generalization, therefore effectively bypassing the two major problems of these architectures.

The complexity of MAP inference for phrase structure parsing is directly impacted by the search space considered (Kallmeyer, 2010). Importantly, (ill-nested) discontinuous phrase structure parsing is known to be NP-hard, even with a bounded block-degree (Satta, 1992). Herzig and Berant (2021) explore two restricted inference algorithms, both of which have a cubic time complexity with respect to the input length. The first one only considers continuous phrase structures, *i.e.* derived trees that could have been generated by a context-free grammar, and the second one also considers a specific type of discontinuities, see Corro (2020, Section

This work has been accepted for publication in TACL. This version is a pre-MIT Press publication version.

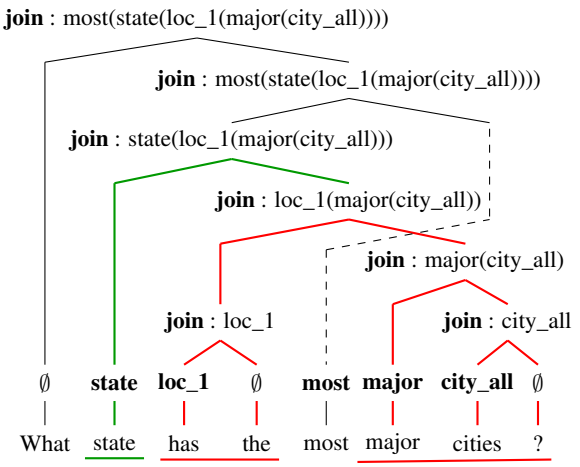

Figure 1: Example of a semantic phrase structure from GEOQUERY. This structure is outside of the search space of the parser of Herzig and Berant (2021) as the constituent in red is discontinuous and also has a discontinuous parent (in red+green).

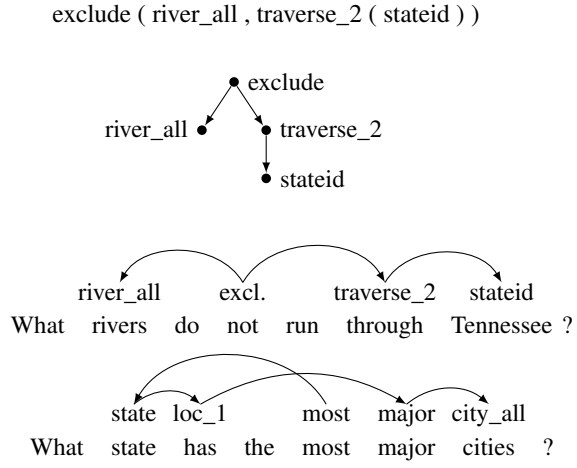

Figure 2: **(top)** The semantic program corresponding to the sentence "What rivers do not run through Tennessee?" in the GEOQUERY dataset. **(middle)** The associated AST. **(bottom)** Two examples illustrating the intuition of our model: we jointly assign predicates/entities and identify argument dependencies. As such, the resulting structure is strongly related to a syntactic dependency parse, but where the dependency structure do not cover all words.

3.6). Both algorithms fail to cover the full set of phrase structures observed in semantic treebanks, see Figure 1.

In this work, we propose to reduce semantic parsing without reentrancy (*i.e.* a given predicate or entity cannot be used as an argument for two different predicates) to a bi-lexical dependency parsing problem. As such, we tackle the same semantic content as aforementioned previous work but using a different mathematical representation (Rambow, 2010). We identify two main benefits to our approach: (1) as we allow crossing arcs, *i.e.* "non-projective graphs", all datasets are guaranteed to be fully covered and (2) it allows us to rely on optimization methods to tackle inference intractability of our novel graph-based formulation of the problem. More specifically, in our setting we need to jointly assign predicates/entities to words that convey a semantic content and to identify arguments of predicates via bi-lexical dependencies. We show that MAP inference in this setting is equivalent to the maximum generalized spanning arborescence problem (Myung et al., 1995) with supplementary constraints to ensure well-formedness with respect to the semantic formalism. Although this problem is NP-hard, we propose an optimization algorithm that solves a linear relaxation of the problem and can deliver an optimality certificate.

Our contributions can be summarized as follows:

- We propose a novel graph-based approach for semantic parsing without reentrancy;
- We prove the NP-hardness of MAP inference and latent anchoring inference;
- We propose a novel integer linear programming formulation for this problem together with an approximate solver based on conditional gradient and constraint smoothing;
- We tackle the training problem using variational approximations of objective functions, including the weakly-supervised scenario;
- We evaluate our approach on GEOQUERY, SCAN and CLEVR and observe that it outperforms baselines on both i.i.d. splits and splits that test for compositional generalization.

Code to reproduce the experiments is available online.[1]

2 Graph-based semantic parsing

We propose to reduce semantic parsing to parsing the abstract syntax tree (AST) associated to a semantic program. We focus on semantic programs

[1] https://github.com/alban-petit/semantic-dependency-parser

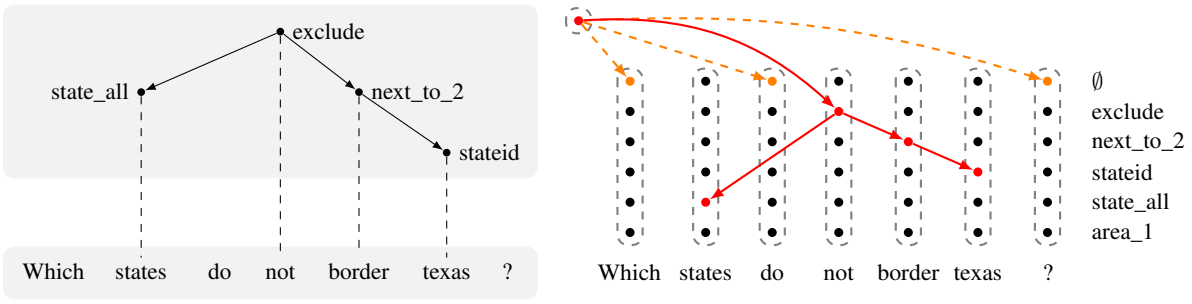

Figure 3: **(a)** Example of a sentence and its associated AST (solid arcs) from the GEOQUERY dataset. The dashed edges indicate predicates and entities anchors (note that this information is not available in the dataset). **(b)** The corresponding generalized valency-constrained not-necessarily-spanning arborescence (red arcs). The root is the isolated top left vertex. Adding $\emptyset$ tags and dotted orange arcs produces a generalized spanning arborescence.

whose ASTs do not have any reentrancy, *i.e.* a single predicate or entity cannot be the argument of two different predicates. Moreover, we assume that each predicate or entity is anchored on exactly one word of the sentence and each word can be the anchor of at most one predicate or entity. As such, the semantic parsing problem can be reduced to assigning predicates and entities to words and identifying arguments via dependency relations, see Figure 2. In order to formalize our approach to the semantic parsing problem, we will use concepts from graph theory. We therefore first introduce the vocabulary and notions that will be useful in the rest of this article. Notably, the notions of cluster and generalized arborescence will be used to formalize our prediction problem.

Notations and definitions. Let $G = \langle V, A \rangle$ be a directed graph with vertices V and arcs $A \subseteq V \times V$. An arc in A from a vertex $u \in V$ to a vertex $v \in V$ is denoted either $a \in A$ or $u \to v \in A$. For any subset of vertices $U \subseteq V$, we denote $\sigma^+_G(U)$ (resp. $\sigma^-_G(U)$) the set of arcs leaving one vertex of U and entering one vertex of $V \setminus U$ (resp. leaving one vertex of $V \setminus U$ and entering one vertex of U) in the graph G. Let $B \subseteq A$ be a subset of arcs. We denote $V[B]$ the cover set of B, *i.e.* the set of vertices that appear as an extremity of at least one arc in B. A graph $G = \langle V, A \rangle$ is an arborescence[2] rooted at $u \in V$ if and only if (iff) it contains $|V| - 1$ arcs and there is a directed path from u to each vertex in V. In the rest of this work, we will assume that the root is always vertex $0 \in V$. Let $B \subseteq A$ be a set of arcs such that $G' = \langle V[B], B \rangle$ is an arborescence. Then G' is a spanning arborescence of G iff $V[B] = V$.

Let $\pi = \{V_0, ..., V_n\}$ be a partition of V containing $n + 1$ clusters. G' is a generalized not-necessarily-spanning arborescence (resp. generalized spanning arborescence) on the partition π of G iff G' is an arborescence and $V[B]$ contains at most one vertex per cluster in π (resp. contains exactly one).

Let $W \subseteq V$ be a set of vertices. Contracting W consists in replacing in G the set W by a new vertex $w \notin V$, replacing all the arcs $u \to v \in \sigma^-(W)$ by an arc $u \to w$ and all the arcs $u \to v \in \sigma^+(W)$ by an arc $w \to v$. Given a graph with partition π, the contracted graph is the graph where each cluster in π has been contracted. While contracting a graph may introduce parallel arcs, it is not an issue in practice, even for weighted graphs.

2.1 Semantic grammar and AST.

The semantic programs we focus on take the form of a functional language, *i.e.* a representation where each predicate is a function that takes other predicates or entities as arguments. The semantic language is typed in the same sense than in "typed programming languages". For example, in GEOQUERY, the predicate `capital_2` expects an ar-

[2] In the NLP community, arborescences are often called (directed) trees. We stick with the term arborescence as it is more standard in the graph theory literature, see for example Schrijver (2003). Using the term tree introduces a confusion between two unrelated algorithms, Kruskal's maximum spanning tree algorithm (Kruskal, 1956) that operates on undirected graphs and Edmond's maximum spanning arborescence algorithm (Edmonds, 1967) that operates on directed graphs. Moreover, this prevents any confusion between the graph object called arborescence and the semantic structure called AST.

gument of type `city` and returns an object of type `state`. In the datasets we use, the typing system disambiguates the position of arguments in a function: for a given function, either all arguments are of the same type or the order of arguments is unimportant — an example of both is the predicate `intersection_river` in GEOQUERY that takes two arguments of type `river`, but the result of the execution is unchanged if the arguments are swapped.[3]

Formally, we define the set of valid semantic programs as the set of programs that can be produced with a semantic grammar $\mathcal{G} = \langle E, T, f_{\text{TYPE}}, f_{\text{ARGS}} \rangle$ where:

- E is the set of predicates and entities, which we will refer to as the set of tags — w.l.o.g. we assume that ROOT $\notin E$ where ROOT is a special tag used for parsing;
- T is the set of types;
- $f_{\text{TYPE}} : E \to T$ is a typing function that assigns a type to each tag;
- $f_{\text{ARGS}} : E \times T \to \mathbb{N}$ is a valency function that assigns the numbers of expected arguments of a given type to each tag.

A tag $e \in E$ is an entity iff $\forall t \in T : f_{\text{ARGS}}(e, t) = 0$. Otherwise, e is a predicate.

A semantic program in a functional language can be equivalently represented as an AST, a graph where instances of predicates and entities are represented as vertices and where arcs identify arguments of predicates. Formally, an AST is a labeled graph $G = \langle V, A, l \rangle$ where function $l : V \to E$ assigns a tag to each vertex and arcs identify the arguments of tags, see Figure 2. An AST G is well-formed with respect to the grammar $\mathcal{G}$ iff G is an arborescence and the valency and type constraints are satisfied, *i.e.* $\forall u \in V, t \in T$:

$$f_{\text{ARGS}}(l(u), t) = |\sigma^+_G(\{u\}, t)|$$

where:

$$\sigma^+_G(\{u\}, t) = \left\{ \begin{array}{l} u \to v \in \sigma^+_G(\{u\}) \\ \text{s.t. } f_{\text{TYPE}}(l(v)) = t \end{array} \right\}.$$

[3]There are a few corner cases like `exclude_river`, for which we simply assume arguments are in the same order as they appear in the input sentence.

2.2 Problem reduction and complexity

In our setting, semantic parsing is a joint sentence tagging and dependency parsing problem (Bohnet and Nivre, 2012; Li et al., 2011; Corro et al., 2017): each content word (*i.e.* words that convey a semantic meaning) must be tagged with a predicate or an entity, and dependencies between content words identify arguments of predicates, see Figure 2. However, our semantic parsing setting differs from standard syntactic analysis in two ways: (1) the resulting structure is not-necessarily-spanning, there are words (*e.g.* function words) that must not be tagged and that do not have any incident dependency — and those words are not known in advance, they must be identified jointly with the rest of the structure; (2) the dependency structure is highly constrained by the typing mechanism, that is the predicted structure must be a valid AST. Nevertheless, similarly to aforementioned works, our parser is graph-based, that is for a given input we build a (complete) directed graph and decoding is reduced to computing a constrained subgraph of maximum weight.

Given a sentence $\boldsymbol{w} = w_1 ... w_n$ with n words and a grammar $\mathcal{G}$, we construct a clustered labeled graph $G = \langle V, A, \pi, \bar{l} \rangle$ as follows. The partition $\pi = \{V_0, ..., V_n\}$ contains $n + 1$ clusters, where V_0 is a root cluster and each cluster V_i, $i \neq 0$, is associated to word w_i. The root cluster $V_0 = \{0\}$ contains a single vertex that will be used as the root and every other cluster contains $|E|$ vertices. The extended labeling function $\bar{l} : V \to E \cup \{\text{ROOT}\}$ assigns a tag in E to each vertex $v \in V \setminus \{0\}$ and ROOT to vertex 0. Distinct vertices in a cluster V_i cannot have the same label, *i.e.* $\forall u, v \in V_i : u \neq v \implies \bar{l}(u) \neq \bar{l}(v)$.

Let $B \subseteq A$ be a subset of arcs. The graph $G' = \langle V[B], B \rangle$ defines a 0-rooted generalized valency-constrained not-necessarily-spanning arborescence iff it is a generalized arborescence of G, there is exactly one arc leaving 0 and the subarborescence rooted at the destination of that arc is a valid AST with respect to the grammar $\mathcal{G}$. As such, there is a one-to-one correspondence between ASTs anchored on the sentence $\boldsymbol{w}$ and generalized valency-constrained not-necessarily-spanning arborescences in the graph G, see Figure 3b.

For any sentence $\boldsymbol{w}$, our aim is to find the AST that most likely corresponds to it. Thus, after building the graph G as explained above, the neural network described in Appendix B is used to pro-

duce a vector of weights $\boldsymbol{\mu} \in \mathbb{R}^{|V|}$ associated to the set of vertices V and a vector of weights $\boldsymbol{\phi} \in \mathbb{R}^{|A|}$ associated to the set of arcs A. Given these weights, graph-based semantic parsing is reduced to an optimization problem called the maximum generalized valency-constrained not-necessarily-spanning arborescence (MGVCNNSA) in the graph G.

Theorem 1. *The MGVCNNSA problem is NP-hard.*

The proof is in Appendix A.

2.3 Mathematical program

Our graph-based approach to semantic parsing has allowed us to prove the intrinsic hardness of the problem. We follow previous work on graph-based parsing (Martins et al., 2009; Koo et al., 2010), *inter alia*, by proposing an integer linear programming (ILP) formulation in order to compute (approximate) solutions.

Remember that in the joint tagging and dependency parsing interpretation of the semantic parsing problem, the resulting structure is not-necessarily-spanning, meaning that some words may not be tagged. In order to rely on well-known algorithms for computing spanning arborescences as a subroutine of our approximate solver, we first introduce the notion of extended graph. Given a graph $G = \langle V, A, \pi, \bar{l} \rangle$, we construct an extended graph $\overline{G} = \langle \overline{V}, \overline{A}, \overline{\pi}, \bar{l} \rangle$[4] containing n additional vertices $\{\overline{1}, ..., \overline{n}\}$ that are distributed along clusters, *i.e.* $\overline{\pi} = \{V_0, V_1 \cup \{\overline{1}\}, ..., V_n \cup \{\overline{n}\}\}$, and arcs from the root to these extra vertices, *i.e.* $\overline{A} = A \cup \{0 \rightarrow \bar{i} | 1 \leq i \leq n\}$. Let $B \subseteq A$ be a subset of arcs such that $\langle V[B], B \rangle$ is a generalized not-necessarily-spanning arborescence on G. Let $\overline{B} \subseteq \overline{A}$ be a subset of arcs defined as $\overline{B} = B \cup \{0 \rightarrow \bar{i} | \sigma^-_{\langle V[B], B \rangle}(V_i) = \emptyset\}$. Then, there is a one-to-one correspondence between generalized not-necessarily-spanning arborescences $\langle V[B], B \rangle$ and generalized spanning arborescences $\langle \overline{V}[\overline{B}], \overline{B} \rangle$, see Figure 3b.

Let $\boldsymbol{x} \in \{0,1\}^{|\overline{V}|}$ and $\boldsymbol{y} \in \{0,1\}^{|\overline{A}|}$ be variable vectors indexed by vertices and arcs such that a vertex $v \in V$ (resp. an arc $a \in A$) is selected iff $x_v = 1$ (resp. $y_a = 1$). The set of 0-rooted generalized valency-constrained spanning arborescences on $\overline{G}$ can be written as the set of variables $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ satisfying the following linear constraints. First, we restrict $\boldsymbol{y}$ to structures that are spanning arborescences over $\overline{G}$ where clusters have been contracted:

$$\sum_{a \in \sigma^-_{\overline{G}}(V_0)} y_a = 0 \tag{1}$$

$$\sum_{a \in \sigma^-_{\overline{G}}\left(\bigcup_{\overline{U} \in \overline{\pi'}} \overline{U}\right)} y_a \geq 1 \quad \forall \overline{\pi'} \subseteq \overline{\pi} \setminus \{V_0\} \tag{2}$$

$$\sum_{a \in \sigma^-_{\overline{G}}(\overline{V_i})} y_a = 1 \quad \forall \overline{V_i} \in \overline{\pi} \setminus \{V_0\} \tag{3}$$

Constraints (2) ensure that the contracted graph is weakly connected. Constraints (3) force each cluster to have exactly one incoming arc. The set of vectors $\boldsymbol{y}$ that satisfy these three constraints are exactly the set of 0-rooted spanning arborescences on the contracted graph, see Schrijver (2003, Section 52.4) for an in-depth analysis of this polytope. The root vertex is always selected and other vertices are selected iff they have one incoming selected arc:

$$x_0 = 1 \tag{4}$$

$$x_u = \sum_{a \in \sigma^-_{\overline{G}}(\{u\})} y_a \quad \forall u \in \overline{V} \setminus \{0\} \tag{5}$$

Note that constraints (1)–(3) do not force selected arcs to leave from a selected vertex as they operate at the cluster level. This property will be enforced via the valency constraints:

$$\sum_{u \in V \setminus \{0\}} y_{0 \rightarrow u} = 1 \tag{6}$$

$$\sum_{a \in \sigma^+_G(\{u\}, t)} y_a = x_u f_{\text{ARGS}}(l(u), t) \quad \forall t \in T, \; u \in V \setminus \{0\} \tag{7}$$

Constraint (6) forces the root to have exactly one outgoing arc into a vertex $u \in V \setminus \{0\}$ (*i.e.* a vertex that is not part of the extra vertices introduced in the extended graph) that will be the root of the AST. Constraints (7) force the selected vertices and arcs to produce a well-formed AST with respect to the grammar $\mathcal{G}$. Note that these constraints are only defined for vertices in $V \setminus \{0\}$, *i.e.* they are neither defined for the root vertex nor for the extra vertices introduced in the extended graph.

To simplify notations, we introduce the following sets:

$$\mathcal{C}^{(\text{sa})} = \left\{ \begin{array}{l} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0,1\}^{\overline{V}} \times \{0,1\}^{\overline{A}} \\ \text{s.t. } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ satisfy (1)–(5)} \end{array} \right\},$$

$$\mathcal{C}^{(\text{val})} = \left\{ \begin{array}{l} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \{0,1\}^{\overline{V}} \times \{0,1\}^{\overline{A}} \\ \text{s.t. } \boldsymbol{x} \text{ and } \boldsymbol{y} \text{ satisfy (6)–(7)} \end{array} \right\},$$

[4] The labeling function is unchanged as there is no need for types for vertices in $\overline{V} \setminus V$.

and $\mathcal{C} = \mathcal{C}^{(\text{sa})} \cap \mathcal{C}^{(\text{val})}$. Given vertex weights $\boldsymbol{\mu} \in \mathbb{R}^{|\overline{V}|}$ and arc weights $\boldsymbol{\phi} \in \mathbb{R}^{|\overline{A}|}$, computing the MGVCNNSA is equivalent to solving the following ILP:

$$\text{(ILP1)} \quad \max_{\boldsymbol{x},\boldsymbol{y}} \quad \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}$$
$$\text{s.t.} \quad \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{sa})} \text{ and } \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{val})}$$

Without constraint $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{(\text{val})}$, the problem would be easy to solve. The set $\mathcal{C}^{(\text{sa})}$ is the set of spanning arborescences over the contracted graph, hence to maximize over this set we can simply: (1) contract the graph and assign to each arc in the contracted graph the weight of its corresponding arc plus the weight of its destination vertex in the original graph; (2) run the the maximum spanning arborescence algorithm (MSA, Edmonds, 1967; Tarjan, 1977) on the contracted graph, which has a $\mathcal{O}(n^2)$ time-complexity. This process is illustrated on Figure 5 (top). Note that the contracted graph may have parallel arcs, which is not an issue in practice as only the one of maximum weight can appear in a solution of the MSA.

We have established that MAP inference in our semantic parsing framework is a NP-hard problem. We proposed an ILP formulation of the problem that would be easy to solve if some constraints were removed. This property suggests the use of an approximation algorithm that introduces the difficult constraints as penalties. As a similar setting arises from our weakly supervised loss function, the presentation of the approximation algorithm is deferred until Section 4.

3 Training objective functions

3.1 Supervised training objective

We define the likelihood of a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ via the Boltzmann distribution:

$$p_{\boldsymbol{\mu},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} - c(\boldsymbol{\mu}, \boldsymbol{\phi})),$$

where $c(\boldsymbol{\mu}, \boldsymbol{\phi})$ is the log-partition function:

$$c(\boldsymbol{\mu}, \boldsymbol{\phi}) = \log \sum_{\langle \boldsymbol{x}', \boldsymbol{y}' \rangle \in \mathcal{C}} \exp(\boldsymbol{\mu}^\top \boldsymbol{x}' + \boldsymbol{\phi}^\top \boldsymbol{y}').$$

During training, we aim to maximize the log-likelihood of the training dataset. The log-likelihood of an observation $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is defined as:

$$\begin{aligned} \ell(\boldsymbol{\mu}, \boldsymbol{\phi}; \boldsymbol{x}, \boldsymbol{y}) &= \log p_{\boldsymbol{\mu},\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) \\ &= \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} - c(\boldsymbol{\mu}, \boldsymbol{\phi}). \end{aligned}$$

Unfortunately, computing the log-partition function is intractable as it requires summing over all feasible solutions. Instead, we rely on a surrogate lower-bound as an objective function. To this end, we derive an upper bound (because it is negated in ℓ) to the second term: a sum of log-sum-exp functions that sums over each cluster of vertices independently and over incoming arcs in each cluster independently, which is tractable. This loss can be understood as a generalization of the head selection loss used in dependency parsing (Zhang et al., 2017). We now detail the derivation and prove that it is an upper bound to the log-partition function.

Let $\boldsymbol{U}$ be a matrix such that each row contains a pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ and $\Delta^{|\mathcal{C}|}$ be the simplex of dimension $|\mathcal{C}| - 1$, *i.e.* the set of all stochastic vectors of dimension $|\mathcal{C}|$. The log-partition function can then be rewritten using its variational formulation:

$$c(\boldsymbol{\mu}, \boldsymbol{\phi}) = \max_{\boldsymbol{p} \in \Delta^{|\mathcal{C}|}} \boldsymbol{p}^\top \left(\boldsymbol{U} \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} \right) + H[\boldsymbol{p}],$$

where $H[\boldsymbol{p}] = -\sum_i p_i \log p_i$ is the Shannon entropy. We refer the reader to Boyd and Vandenberghe (2004, Example 3.25), Wainwright and Jordan (2008, Section 3.6) and Beck (2017, Section 4.4.10). Note that this formulation remains impractical as $\boldsymbol{p}$ has an exponential size. Let $\mathcal{M} = \text{conv}(\mathcal{C})$ be the marginal polytope, *i.e.* the convex hull of the feasible integer solutions, we can rewrite the above variational formulation as:

$$= \max_{\boldsymbol{m} \in \mathcal{M}} \boldsymbol{m}^\top \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} + H_{\mathcal{M}}[\boldsymbol{m}]$$

where $H_{\mathcal{M}}$ is a joint entropy function defined such that the equality holds. The maximization in this reformulation acts on the marginal probabilities of parts (vertices and arcs) and has therefore a polynomial number of variables. We refer the reader to Wainwright and Jordan (2008, 5.2.1) and Blondel et al. (2020, Section 7) for more details. Unfortunately, this optimization problem is hard to solve as $\mathcal{M}$ cannot be caracterized in an explicit manner and $H_{\mathcal{M}}$ is defined indirectly and lacks a polynomial closed form (Wainwright and Jordan, 2008, Section 3.7). However, we can derive an upper bound to the log-partition function by decomposing the entropy term $H_{\mathcal{M}}$ (Cover, 1999, Property 4 on page 41, *i.e.* H is an upper bound) and by using an outer approximation to the marginal polytope $\mathcal{L} \supseteq \mathcal{M}$ (*i.e.* increasing the search space):

$$\leq \max_{\boldsymbol{m} \in \mathcal{L}} \boldsymbol{m}^\top \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\phi} \end{bmatrix} + H[\boldsymbol{m}]$$

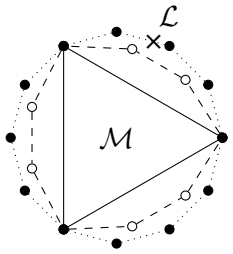

Figure 4: Polyhedrons illustration. The solid lines represent the convex hull of feasible solutions of ILP1, denoted $\mathcal{M}$, whose vertices are feasible integer solutions (black vertices). The dashed lines represent the convex hull of feasible solutions of the linear relaxation of ILP1, which has non integral vertices (in white). Finally, the dotted lines represent the polyhedron $\mathcal{L}$ that is used to approximate $c(\boldsymbol{\mu}, \boldsymbol{\phi})$. All its vertices are integral, but some of them are not feasible solutions of ILP1.

In particular, we observe that each pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}$ has exactly one vertex selected per cluster $\overline{V_i} \in \overline{\pi}$ and one incoming arc selected per cluster $\overline{V_i} \in \overline{\pi} \setminus \{V_0\}$. We denote $\mathcal{C}^{(\text{one})}$ the set of all the pairs $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ that satisfy these constraints. By using $\mathcal{L} = \text{conv}(\mathcal{C}^{(\text{one})})$ as an outer approximation to the marginal polytope (see Figure 4) the optimization problem can be rewritten as a sum of independent problems. As each of these problems is the variational formulation of a log-sum-exp term, the upper bound on $c(\boldsymbol{\mu}, \boldsymbol{\phi})$ can be expressed as a sum of log-sum-exp functions, one over vertices in each cluster $\overline{V_i} \in \overline{\pi} \setminus \{V_0\}$ and one over incoming arcs $\sigma_{\overline{G}}^{-}(\overline{V_i})$ for each cluster $\overline{V_i} \in \overline{\pi} \setminus \{V_0\}$. Although this type of approximation may not result in a Bayes consistent loss (Corro, 2023), it works well in practice.

3.2 Weakly-supervised training objective

Unfortunately, training data often does not include gold pairs $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ but instead only the AST, without word anchors (or word alignment). This is the case for the three datasets we use in our experiments. We thus consider our training signal to be the set of all structures that induce the annotated AST, which we denote $\mathcal{C}^*$.

The weakly-supervised loss is defined as:

$$\widetilde{\ell}(\boldsymbol{\mu}, \boldsymbol{\phi}; \mathcal{C}^*) = \log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} p_{\boldsymbol{\mu}, \boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}),$$

i.e. we marginalize over all the structures that induce the gold AST. We can rewrite this loss as:

$$= \left(\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}) \right) - c(\boldsymbol{\mu}, \boldsymbol{\phi}).$$

The two terms are intractable. We approximate the second term using the bound defined in Section 3.1.

We now derive a tractable lower bound to the first term. Let q be a proposal distribution such that $q(\boldsymbol{x}, \boldsymbol{y}) = 0$ if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \notin \mathcal{C}^*$. We derive the following lower bound via Jensen's inequality:

$$\begin{aligned} &\log \sum_{\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^*} \exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}) \\ &\quad = \log \mathbb{E}_q \left[\frac{\exp(\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y}))}{q(\boldsymbol{x}, \boldsymbol{y})} \right] \\ &\quad \geq \mathbb{E}_q \left[\boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} \right] + H[q]. \end{aligned}$$

This bound holds for any distribution q satisfying the aforementioned condition. We choose to maximize this lower bound using a distribution that gives a probability of one to a single structure, as in "hard" EM (Neal and Hinton, 1998, Section 6).

For a given sentence $\boldsymbol{w}$, let $G = \langle V, A, \pi, \bar{l} \rangle$ be a graph defined as in Section 2.2 and $G' = \langle V', A', l' \rangle$ be an AST defined as in Section 2.1. We aim to find the GVCNNSA in G of maximum weight whose induced AST is exactly G'. This is equivalent to aligning each vertex in V' with one vertex of $V \setminus \{0\}$ s.t. there is at most one vertex per cluster of π appearing in the alignment and where the weight of an alignment is defined as:

1. for each vertex $u' \in V'$, we add the weight of the vertex $u \in V$ it is aligned to — moreover, if u' is the root of the AST we also add the weight of the arc $0 \to u$;
2. for each arc $u' \to v' \in A'$, we add the weight of the arc $u \to v$ where $u \in V$ (resp. $v \in V$) is the vertex u' (resp. v') it is aligned with.

Note that this latent anchoring inference consists in computing a (partial) alignment between vertices of G and G', but the fact that we need to take into account arc weights forbids the use of the Kuhn–Munkres algorithm (Kuhn, 1955).

Theorem 2. *Computing the anchoring of maximum weight of an AST with a graph G is NP-hard.*

The proof is in Appendix A.

Therefore, we propose an optimization-based approach to compute the distribution q. Note that

Algorithm 1 Unconstrained alignment of maximum weight between a graph G and an AST G'

function DPALIGNMENT(G, G')
  for $u' \in V'$ in reverse topological order **do**
    for $u \in \{v \in V | \bar{l}(v) = l'(u')\}$ **do** ▷ We can only map u' to vertices u if they have the same tag.
      $\textsc{Chart}[u', u] \leftarrow \mu_u + \sum_{u' \to v' \in \sigma^+_{G'}(\{u'\})} \left(\max_{v \in V} \textsc{Chart}[v', v] + \phi_{u \to v} \right)$
  return $\max_{u \in V} \textsc{Chart}[r', u] + \phi_{0 \to u}$ ▷ Where $r' \in A'$ is the root of the AST.

Algorithm 2 Conditional gradient

function CONDITIONALGRADIENT(G, G')
  Let $\boldsymbol{z}^{(0)} \in \text{conv}(\mathcal{C}^{\text{(easy)}})$
  for $k \in \{0....K\}$ **do** ▷ Where K is the maximum number of iterations
    $\boldsymbol{d} \leftarrow \left(\text{lmo}_{\text{conv}(\mathcal{C}^{\text{(easy)}})}(\nabla g(\boldsymbol{z}^{(k)})) \right) - \boldsymbol{z}^{(k)}$ ▷ Compute the update direction
    if $\nabla g(\boldsymbol{z}^{(k)})^\top \boldsymbol{d} \leq \epsilon$ **then return** $\boldsymbol{z}^{(k)}$ ▷ If the dual gap is small, $\boldsymbol{z}^{(k)}$ is (almost) optimal
    $\gamma \in \arg\max_{\gamma \in [0,1]} g(\boldsymbol{z}^{(k)} + \gamma \boldsymbol{d})$ ▷ Compute or approximate the optimal stepsize
    $\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \gamma \boldsymbol{d}$ ▷ Update the current point
  return $\boldsymbol{z}^{(k)}$

the problem has a constraint requiring each cluster $V_i \in \pi$ to be aligned with at most one vertex $v' \in V'$, *i.e.* each word in the sentence can be aligned with at most one vertex in the AST. If we remove this constraint, then the problem becomes tractable via dynamic programming. Indeed, we can recursively construct a table $\textsc{Chart}[u', u]$, $u' \in V'$ and $u \in V$, containing the score of aligning vertex u' to vertex u plus the score of the best alignment of all the descendants of u'. To this end, we simply visit the vertices V' of the AST in reverse topological order, see Algorithm 1. The best alignment can be retrieved via back-pointers.

Computing q is therefore equivalent to solving the following ILP:

$$\begin{aligned} \text{(ILP2)} \quad \max_{\boldsymbol{x}, \boldsymbol{y}} \quad & \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{\phi}^\top \boldsymbol{y} \\ \text{s.t.} \quad & \langle \boldsymbol{x}, \boldsymbol{y} \rangle \in \mathcal{C}^{*\text{(relaxed)}}, \\ & \sum_{u \in V_i} x_u \leq 1 \qquad \forall V_i \in \pi. \end{aligned}$$

The set $\mathcal{C}^{*\text{(relaxed)}}$ is the set of feasible solutions of the dynamic program in Algorithm 1, whose convex hull can be described via linear constraints (Martin et al., 1990).

4 Efficient inference

In this section, we propose an efficient way to solve the linear relaxations of MAP inference (ILP1) and latent anchoring inference (ILP2) via constraint smoothing and the conditional gradient method.

We focus on problems of the following form:

$$\begin{aligned} \max_{\boldsymbol{z}} \quad & f(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{z} \in \text{conv}(\mathcal{C}^{\text{(easy)}}) \\ & \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \quad \text{or} \quad \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b} \end{aligned}$$

where the vector $\boldsymbol{z}$ is the concatenation of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ defined previously and conv denotes the convex hull of a set. We explained previously that if the set of constraints of form $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$ for (ILP1) or $\boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}$ for (ILP2) was absent, the problem would be easy to solve under a linear objective function. In fact, there exists an efficient linear maximization oracle (LMO), *i.e.* a function that returns the optimal integral solution, for the set $\text{conv}(\mathcal{C}^{\text{(easy)}})$. This setting covers both (ILP1) and (ILP2) where we have $\mathcal{C}^{\text{(easy)}} = \mathcal{C}^{\text{(sa)}}$ and $\mathcal{C}^{\text{(easy)}} = \mathcal{C}^{*\text{(relaxed)}}$, respectively.

An appealing approach in this setting is to introduce the problematic constraints as penalties in the objective:

$$\begin{aligned} \max_{\boldsymbol{z}} \quad & f(\boldsymbol{z}) - \delta_S(\boldsymbol{A}\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{z} \in \text{conv}(\mathcal{C}^{\text{(easy)}}) \end{aligned}$$

where δ_S is the indicator function of the set S:

$$\delta_S(\boldsymbol{A}\boldsymbol{z}) = \begin{cases} 0 & \text{if } \boldsymbol{A}\boldsymbol{z} \in S, \\ +\infty & \text{otherwise.} \end{cases}$$

In the equality case, we use $S = \{\boldsymbol{b}\}$ and in the inequality case, we use $S = \{\boldsymbol{u} | \boldsymbol{u} \leq \boldsymbol{b}\}$.

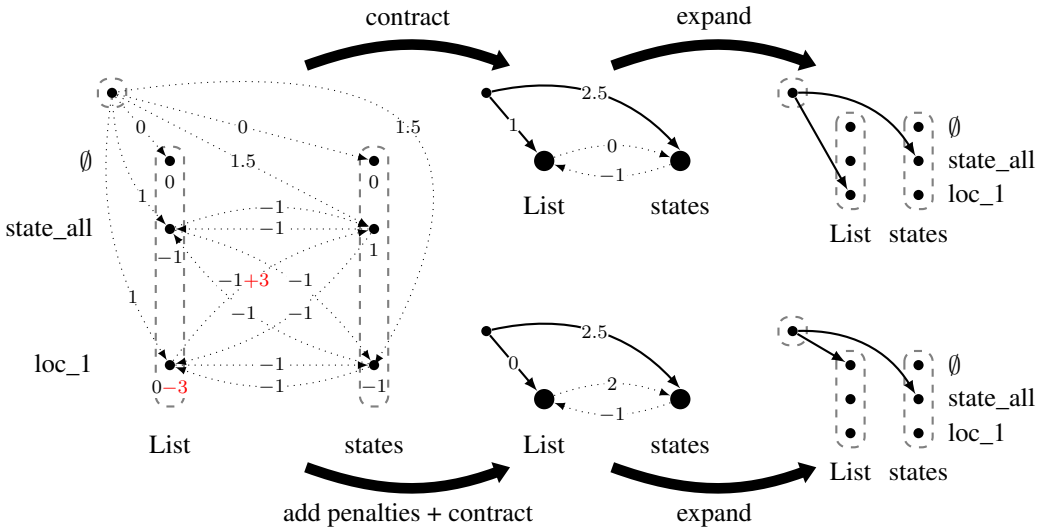

Figure 5: Illustration of the approximate inference algorithm on the two-word sentence "List states", where we assume the grammar has one entity `state_all` and one predicate `loc_1` that takes exactly one entity as argument. The left graph is the extended graph for the sentence, including vertices and arcs weights (in black). If we ignore constraints (6)–(7), inference is reduced to computing the MSA on the contracted graph (solid arcs in the middle column). This may lead to solutions that do not satisfy constraints (6)–(7) on the expanded graph (top example). However, the gradient of the smoothed constraint (7) will induce penalties (in red) to vertex and arc scores that will encourage the `loc_1` predicate to either be dropped from the solution or to have an outgoing arc to a `state_all` argument. Computing the MSA on the contracted graph with penalties results in a solution that satisfies constraints (6)–(7) (bottom example).

4.1 Conditional gradient method

Given a proper, smooth and differentiable function g and a nonempty, bounded, closed and convex set $\mathrm{conv}(\mathcal{C}^{(\text{easy})})$, the conditional gradient method (a.k.a. Frank-Wolfe, Frank and Wolfe, 1956; Levitin and Polyak, 1966; Lacoste-Julien and Jaggi, 2015) can be used to solve optimization problems of the following form:

$$\max_{\boldsymbol{z}} \quad g(\boldsymbol{z}) \quad \text{s.t.} \quad \boldsymbol{z} \in \mathrm{conv}(\mathcal{C}^{(\text{easy})})$$

Contrary to the projected gradient method, this approach does not require to compute projections onto the feasible set $\mathrm{conv}(\mathcal{C}^{(\text{easy})})$ which is, in most cases, computationally expensive. Instead, the conditional gradient method only relies on a LMO:

$$\mathrm{lmo}_{\mathcal{C}^{(\text{easy})}}(\boldsymbol{\psi}) \in \operatorname*{arg\,max}_{\boldsymbol{z} \in \mathrm{conv}(\mathcal{C}^{(\text{easy})})} \boldsymbol{\psi}^\top \boldsymbol{z}.$$

The algorithm constructs a solution to the original problem as a convex combination of elements returned by the LMO. The pseudo-code is given in Algorithm 2. An interesting property of this method is that its step size range is bounded. This allows for simple linesearch techniques.

4.2 Smoothing

Unfortunately, the function $g(\boldsymbol{z}) = f(\boldsymbol{z}) - \delta_S(\boldsymbol{A}\boldsymbol{z})$ is non-smooth due to the indicator function term, preventing the use of the conditional gradient method. We propose to rely on the framework proposed by Yurtsever et al. (2018) where the indicator function is replaced by a smooth approximation. The indicator function of the set S can be rewritten as:

$$\delta_S(\boldsymbol{A}\boldsymbol{z}) = \delta_S^{**}(\boldsymbol{A}\boldsymbol{z}) = \sup_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{z} - \sigma_S(\boldsymbol{u}),$$

where δ_S^{**} denotes the Fenchel biconjugate of the indicator function and $\sigma_S(\boldsymbol{u}) = \sup_{\boldsymbol{t} \in S} \boldsymbol{u}^\top \boldsymbol{t}$ is the support function of S. More details can be found in Beck (2017, Section 4.1 and 4.2). In order to smooth the indicator function, we add a β-parameterized convex regularizer $-\frac{\beta}{2}\|\cdot\|_2^2$ to its Fenchel biconjugate:

$$\delta_{S,\beta}^{**}(\boldsymbol{A}\boldsymbol{z}) = \max_{\boldsymbol{u}} \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{z} - \sigma_S(\boldsymbol{u}) - \frac{\beta}{2}\|\boldsymbol{u}\|_2^2$$

where $\beta > 0$ controls the quality and the smoothness of the approximation (Nesterov, 2005).

Equalities. In the case where $S = \{\boldsymbol{b}\}$, with a few computations that are detailed by Yurtsever

et al. (2018), we obtain:

$$\delta^{**}_{\{\boldsymbol{b}\},\beta}(\boldsymbol{Az}) = \frac{1}{2\beta}\|\boldsymbol{Az} - \boldsymbol{b}\|_2^2.$$

That is, we have a quadratic penalty term in the objective. Note that this term is similar to the term introduced in an augmented Lagrangian (Nocedal and Wright, 1999, Equation 17.36), and adds a penalty in the objective for vectors $\boldsymbol{z}$ s.t. $\boldsymbol{Az} \neq \boldsymbol{b}$.

Inequalities. In the case where $S = \{\boldsymbol{u}|\boldsymbol{u} \leq \boldsymbol{b}\}$, similar computations lead to:

$$\delta^{**}_{\leq \boldsymbol{b},\beta}(\boldsymbol{Az}) = \frac{1}{2\beta}\|[\boldsymbol{Az} - \boldsymbol{b}]_+\|_2^2$$

where $[\cdot]_+$ denotes the Euclidian projection into the non-negative orthant (*i.e.* clipping negative values). Similarly to the equality case, this term introduces a penalty in the objective for vectors $\boldsymbol{z}$ s.t. $\boldsymbol{Az} > \boldsymbol{b}$. This penalty function is also called the Courant-Beltrami penalty function.

Figure 5 (bottom) illustrates how the gradient of the penalty term can "force" the LMO to return solutions that satisfy the smoothed constraints.

4.3 Practical details

Smoothness. In practice, we need to choose the smoothness parameter β. We follow Yurtsever et al. (2018) and use $\beta^{(k)} = \frac{\beta^{(0)}}{\sqrt{k+1}}$ where k is the iteration number and $\beta^{(0)} = 1$.

Step size. Another important choice in the algorithm is the step size γ. We show that when the smoothed constraints are equalities, computing the optimal step size has a simple closed form solution if the function f is linear, which is the case for (ILP1), *i.e.* MAP decoding. The step size problem formulation at iteration k is defined as:

$$\arg\max_{\gamma \in [0,1]} f(\boldsymbol{z}^{(k)} + \gamma \boldsymbol{d}) - \frac{\|\boldsymbol{A}(\boldsymbol{z}^{(k)} + \gamma \boldsymbol{d}) - \boldsymbol{b}\|^2}{2\beta}$$

By assumption, f is linear and can be written as $f(\boldsymbol{z}) = \boldsymbol{\theta}^\top \boldsymbol{z}$. Ignoring the box constraints on γ, by first order optimality conditions, we have:

$$\gamma = \frac{-\beta\theta^\top \boldsymbol{d} + (\boldsymbol{Ad})^\top \boldsymbol{b} - (\boldsymbol{Ad})^\top (\boldsymbol{Az}^{(k)})}{\|\boldsymbol{Ad}\|^2}$$

We can then simply clip the result so that it satisfies the box constraints. Unfortunately, in the inequalities case, there is no simple closed form solution. We approximate the step size using 10 iterations of the bisection algorithm for root finding.

Non-integral solutions. As we solve the linear relaxation of original ILPs, the optimal solutions may not be integral. Therefore, we use simple heuristics to construct a feasible solution to the original ILP in these cases. For MAP inference, we simply solve the ILP[5] using CPLEX but introducing only variables that have a non-null value in the linear relaxation, leading to a very sparse problem which is fast to solve. For latent anchoring, we simply use the Kuhn–Munkres algorithm using the non-integral solution as assignment costs.

5 Experiments

We compare our method to baseline systems both on i.i.d. splits (IID) and splits that test for compositional generalization for three datasets. The neural network is described in Appendix B.

Datasets. SCAN (Lake and Baroni, 2018) contains natural language navigation commands. We use the variant of Herzig and Berant (2021) for semantic parsing. The IID split is the *simple* split (Lake and Baroni, 2018). The compositional splits are *primitive right* (RIGHT) and *primitive around right* (ARIGHT) (Loula et al., 2018).

GEOQUERY (Zelle and Mooney, 1996) uses the FunQL formalism (Kate et al., 2005) and contains questions about the US geography. The IID split is the standard split and compositional generalization is evaluated on two splits: LENGTH where the examples are split by program length and TEMPLATE (Finegan-Dollak et al., 2018a) where they are split such that all semantic programs having the same AST are in the same split.

CLEVR (Johnson et al., 2017) contains synthetic questions over object relations in images. CLOSURE (Bahdanau et al., 2019) introduces additional question templates that require compositional generalization. We use the original split as our IID split and the CLOSURE split as a compositional split where the model is evaluated on CLOSURE.

Baselines. We compare our approach against the architecture proposed by Herzig and Berant (2021) (SPANBASEDSP) as well as the seq2seq baselines they used. In SEQ2SEQ (Jia and Liang, 2016), the encoder is a bi-LSTM over pre-trained GloVe embeddings (Pennington et al., 2014) or ELMO (Peters et al., 2018) and the decoder is an attention-based LSTM (Bahdanau et al., 2015). BERT2SEQ replaces the encoder with `BERT-base`. GRAM-

[5] We use the multi-commodity flow formulation of Martins et al. (2009) instead of the cycle breaking constraints (2).

	SCAN			GEOQUERY			CLEVR	
	IID	RIGHT	ARIGHT	IID	TEMPLATE	LENGTH	IID	CLOSURE
Baselines (denotation accuracy only)								
SEQ2SEQ	99.9	11.6	0	78.5	46.0	24.3	**100**	59.5
+ ELMO	**100**	54.9	41.6	79.3	50.0	25.7	**100**	64.2
BERT2SEQ	**100**	77.7	95.3	81.1	49.6	26.1	**100**	56.4
GRAMMAR	**100**	0.0	4.2	72.1	54.0	24.6	**100**	51.3
BART	**100**	50.5	**100**	87.1	67.0	19.3	**100**	51.5
SPANBASEDSP	**100**	**100**	**100**	86.1	82.2	63.6	96.7	98.8
Our approach								
Denotation accuracy	**100**	**100**	**100**	**92.9**	**89.9**	**74.9**	**100**	**99.6**
↳ Corrected executor				91.8	88.7	74.5		
Exact match	**100**	**100**	**100**	90.7	86.2	69.3	**100**	**99.6**
↳ w/o CPLEX heuristic	**100**	**100**	**100**	90.0	83.0	67.5	**100**	98.0

Table 1: Denotation and exact match accuracy on the test sets. All the baseline results were taken from Herzig and Berant (2021). For our approach, we also report exact match accuracy, *i.e.* the percentage of sentences for which the prediction is identical to the gold program. The last line reports the exact match accuracy without the use of CPLEX to round non integral solutions (Section 4.3).

MAR is similar to SEQ2SEQ but the decoding is constrained by a grammar. BART (Lewis et al., 2020) is pre-trained as a denoising autoencoder.

Results. We report the denotation accuracies in Table 1. Our approach outperforms all other methods. In particular, the seq2seq baselines suffer from a significant drop in accuracy on splits that require compositional generalization. While SPANBASEDSP is able to generalize, our approach outperforms it. Note that we observed that the GEOQUERY execution script used to compute denotation accuracy in previous work contains several bugs that overestimate the true accuracy. Therefore, we also report denotation accuracy with a corrected executor[6] (see Appendix C) for fair comparison with future work.

We also report exact match accuracy, with and without the heuristic to construct integral solutions from fractional ones. The exact match accuracy is always lower or equal to the denotation accuracy. This shows that our approach can sometimes provide the correct denotation even though the prediction is different from the gold semantic program. Importantly, while our approach outperforms baselines, its accuracy is still significantly worse on the split that requires to generalize to longer programs.

[6] https://github.com/alban-petit/geoquery-funql-executor

6 Related work

Graph-based methods. Graph-based methods have been popularized by syntactic dependency parsing (McDonald et al., 2005) where MAP inference is realized via the maximum spanning arborescence algorithm (Chu and Liu, 1965; Edmonds, 1967). A benefit of this algorithm is that it has a $\mathcal{O}(n^2)$ time-complexity (Tarjan, 1977), *i.e.* it is more efficient than algorithms exploring more restricted search spaces (Eisner, 1997; Gómez-Rodríguez et al., 2011; Pitler et al., 2012, 2013).

In the case of semantic structures, Kuhlmann and Jonsson (2015) proposed a $\mathcal{O}(n^3)$ algorithm for the maximum non-necessarily-spanning acyclic graphs with a noncrossing arc constraint. Without the noncrossing constraint, the problem is known to be NP-hard (Grötschel et al., 1985). To bypass this computational complexity, Dozat and Manning (2018) proposed to handle each dependency as an independent binary classification problem, that is they do not enforce any constraint on the output structure. Note that, contrary to our work, these approaches allow for reentrancy but do not enforce well-formedness of the output with respect to the semantic grammar. Lyu and Titov (2018) use a similar approach for AMR parsing where tags are predicted first, followed by arc predictions and finally heuristics are used to ensure the output graph is valid. On the contrary, we do not use a pipeline and we focus on joint decoding where validity of

the output is directly encoded in the search space.

Previous work in the literature has also considered reduction to graph-based methods for other problems, *e.g.* for discontinuous constituency parsing (Fernández-González and Martins, 2015; Corro et al., 2017), lexical segmentation (Constant and Le Roux, 2015) and machine translation (Zaslavskiy et al., 2009), *inter alia*.

Compositional generalization. Several authors observed that compositional generalization insufficiency is an important source of error for semantic parsers, especially ones based on seq2seq architectures (Lake and Baroni, 2018; Finegan-Dollak et al., 2018b; Herzig and Berant, 2019; Keysers et al., 2020). Wang et al. (2021) proposed a latent re-ordering step to improve compositional generalization, whereas Zheng and Lapata (2021) relied on latent predicate tagging in the encoder. There has also been an interest in using data augmentation methods to improve generalization (Jia and Liang, 2016; Andreas, 2020; Akyürek et al., 2021; Qiu et al., 2022; Yang et al., 2022).

Recently, Herzig and Berant (2021) showed that span-based parsers do not exhibit such problematic behavior. Unfortunately, these parsers fail to cover the set of semantic structures observed in English treebanks, and we hypothesize that this would be even worse for free word order languages. Our graph-based approach does not exhibit this downside. Previous work by Jambor and Bahdanau (2022) also considered graph-based methods for compositional generalization, but their approach predicts each part independently without any well-formedness or acyclicity constraint.

7 Conclusion

In this work, we focused on graph-based semantic parsing for formalisms that do not allow reentrancy. We conducted a complexity study of two inference problems that appear in this setting. We proposed ILP formulations of these problems together with a solver for their linear relaxation based on the conditional gradient method. Experimentally, our approach outperforms comparable baselines.

One downside of our semantic parser is speed (we parse approximately 5 sentences per second for GeoQuery). However, we hope this work will give a better understanding of the semantic parsing problem together with baseline for faster methods.

Future research will investigate extensions for (1) ASTs that contain reentrancies and (2) prediction algorithms for the case where a single word can be the anchor of more than one predicate or entity. These two properties are crucial for semantic representations like Abstract Meaning Representation (Banarescu et al., 2013). Moreover, even if our graph-based semantic parser provides better results than previous work on length generalization, this setting is still difficult. A more general research direction on neural architectures that generalize better to longer sentences is important.

Acknowledgments

We thank François Yvon and the anonymous reviewers for their comments and suggestions. We thank Jonathan Herzig and Jonathan Berant for fruitful discussions. This work benefited from computations done on the Saclay-IA platform and on the HPC resources of IDRIS under the allocation 2022-AD011013727 made by GENCI.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. In *International Conference on Learning Representations*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron C. Courville. 2019.

CLOSURE: Assessing systematic generalization of CLEVR models. *CoRR*, abs/1912.05783.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Amir Beck. 2017. *First-Order Methods in Optimization*. SIAM.

Mathieu Blondel, André F.T. Martins, and Vlad Niculae. 2020. Learning with Fenchel-Young losses. *Journal of Machine Learning Research*, 21(35):1–69.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea. Association for Computational Linguistics.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*.

John Cocke. 1970. *Programming Languages and Their Compilers: Preliminary Notes*. New York University.

Matthieu Constant and Joseph Le Roux. 2015. Dependency representations for lexical segmentation. In *6th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2015), Bilbao, Spain, July*.

Caio Corro. 2020. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from O(nˆ6) down to O(nˆ3). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2753–2764, Online. Association for Computational Linguistics.

Caio Corro. 2023. On the inconsistency of separable losses for structured prediction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1644–1654, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas M. Cover. 1999. *Elements of Information Theory*. John Wiley & Sons.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Christophe Duhamel, Luis Gouveia, Pedro Moura, and Mauricio Souza. 2008. Models and heuristics for a minimum arborescence problem. *Networks*, 51(1):34–47.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards – B. Mathematics and Mathematical Physics*.

Jason Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, pages 54–65, Boston/Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018a. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018b. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Martin Grötschel, Michael Jünger, and Gerhard Reinelt. 1985. *Acyclic Subdigraphs and Linear Orderings: Polytopes, Facets, and a Cutting Plane Algorithm*, pages 217–264. Springer Netherlands, Dordrecht.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, Maryland. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2019. Don't paraphrase, detect! Rapid and effective data collection for semantic parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3810–3820, Hong Kong, China. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 908–921, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Dora Jambor and Dzmitry Bahdanau. 2022. LAGr: Label aligned graphs for better systematic generalization in semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3295–3308, Dublin, Ireland. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer Science & Business Media.

Tadao Kasami. 1965. *An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages*. Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1062–1068. AAAI Press.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

Joseph B Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, 3:559–570.

Harold W. Kuhn. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Simon Lacoste-Julien and Martin Jaggi. 2015. On the global linear convergence of Frank-Wolfe optimization variants. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Evgeny S. Levitin and Boris T. Polyak. 1966. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for Chinese POS tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191, Edinburgh, Scotland, UK. Association for Computational Linguistics.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment.

In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 397–407, Melbourne, Australia. Association for Computational Linguistics.

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

André Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. 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*, pages 342–350, Suntec, Singapore. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132, Prague, Czech Republic. Association for Computational Linguistics.

Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. 1995. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

Radford M. Neal and Geoffrey E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Yu Nesterov. 2005. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 478–488, Jeju Island, Korea. Association for Computational Linguistics.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association for Computational Linguistics*, 1:13–24.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, California. Association for Computational Linguistics.

V. Venkata Rao and R. Sridharan. 2002. Minimum-weight rooted not-necessarily-spanning arborescence problem. *Networks*, 39(2):77–87.

Giorgio Satta. 1992. Recognition of linear context-free rewriting systems. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 89–95, Newark, Delaware, USA. Association for Computational Linguistics.

Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Martin J. Wainwright and Michael Irwin Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Structured reordering for modeling latent alignments in sequence transduction. In *Advances in Neural Information Processing Systems*, volume 34, pages 13378–13391. Curran Associates, Inc.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. SUBS: Subtree substitution for compositional semantic parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 169–174, Seattle, United States. Association for Computational Linguistics.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

Alp Yurtsever, Olivier Fercoq, Francesco Locatello, and Volkan Cevher. 2018. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5727–5736. PMLR.

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*, pages 333–341, Suntec, Singapore. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2021. Compositional generalization via semantic tagging. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1022–1032, Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Proofs

Proof: Theorem 1. We prove Theorem 1 by reducing the maximum not-necessarily-spanning arborescence (MNNSA) problem, which is known to be NP-hard (Rao and Sridharan, 2002; Duhamel et al., 2008), to the MGVCNNSA.

Let $G = \langle V, A, \boldsymbol{\psi} \rangle$ be a weighted graph where $V = \{0, ..., n\}$ and $\boldsymbol{\psi} \in \mathbb{R}^{|A|}$ are arc weights. The MNNSA problem aims to compute the subset of arcs $B \subseteq A$ such that $\langle V[B], B \rangle$ is an arborescence of maximum weight, where its weight is defined as $\sum_{a \in B} \psi_a$.

Let $\mathcal{G} = \langle E, T, f_{\text{TYPE}}, f_{\text{ARGS}} \rangle$ be a grammar such that $E = \{0, ..., n-1\}$, $T = \{t\}$ and $\forall e \in E : f_{\text{TYPE}}(e) = t \wedge f_{\text{ARGS}}(e, t) = e$. Intuitively, a tag $e \in E$ will be associated to vertices that require exactly e outgoing arcs.

We construct a clustered labeled weighted graph $G' = \langle V', A', \pi, l, \boldsymbol{\psi}' \rangle$ as follows. $\pi = \{V'_0, ..., V'_n\}$ is a partition of V' such that each cluster V'_i contains $n-1$ vertices and represents the vertex $i \in V$. The labeling function l assigns a different tag to each vertex in a cluster, *i.e.* $\forall V'_i \in \pi, \forall u', v' \in V'_i : u' \neq v' \Rightarrow l(u') \neq l(v')$. The set of arcs is defined as $A' = \{u' \to v' | \exists i \to j \in A \text{ s.t. } u' \in V'_i \wedge v' \in V'_j\}$. The weight vector $\boldsymbol{\psi}' \in \mathbb{R}^{|A'|}$ is such that $\forall u' \to v' \in A' : u' \in V'_u \wedge v' \in V'_v \Rightarrow \psi'_{u' \to v'} = \psi_{u \to v}$.

As such, there is a one-to-one correspondence between solutions of the MNNSA on graph G and solutions of the MGVCNNSA on graph G'. □

Note that our proof considers that arcs leaving from the root cluster satisfy constraints defined by the grammar whereas we previously only required the root vertex to have a single outgoing arc. The latter constraint can be added directly in a grammar, but we omit presentation for brevity. The constrained arity case presented by McDonald and Satta (2007) focuses on spanning arborescences with an arity constraint by reducing the Hamiltonian path problem to their problem. While the arity constraint is similar in their problem and ours, our proof considers the not-necessarily-spanning case instead of the spanning one. Although the two problems seem related, they need to be studied separately, *e.g.* computing the maximum spanning arborescence is a polynomial time problem whereas computing the MNNSA is a NP-hard problem.

Proof: Theorem 2. We prove Theorem 2 by reducing the maximum directed Hamiltonian path problem, which is known to be NP-hard (Garey and Johnson, 1979, Appendix A1.3), to the latent anchoring.

Let $G = \langle V, A, \boldsymbol{\psi} \rangle$ be a weighted graph where $V = \{1, ..., n\}$ and $\boldsymbol{\psi} \in \mathbb{R}^{|A|}$ are arc weights. The maximum Hamiltonian path problem aims to compute the subset of arcs $B \subseteq A$ such that $V[B] = V$ and $\langle V[B], B \rangle$ is a path of maximum weight, where its weight is defined as $\sum_{a \in B} \psi_a$.

Let $\mathcal{G} = \langle E, T, f_{\text{TYPE}}, f_{\text{ARGS}} \rangle$ be a grammar such that $E = \{0, 1\}$, $T = \{t\}$ and $\forall e \in E : f_{\text{TYPE}}(e) = t \wedge f_{\text{ARGS}}(e, t) = e$.

We construct a clustered labeled weighted graph $G' = \langle V', A', \pi, l, \boldsymbol{\psi}' \rangle$ as follows. $\pi = \{V'_0, ..., V'_n\}$ is a partition of V' such that $V'_0 = \{0\}$, each cluster $V'_i \neq V'_0$ contains 2 vertices and represents the vertex $i \in V$. The labeling function l assigns a different tag to each vertex in a cluster except the root, *i.e.* $\forall V'_i \in \pi, i > 0, \forall u', v' \in V'_i : u' \neq v' \Rightarrow l(u') \neq l(v')$. The set of arcs is defined as $A' = \{0 \to u' | u' \in V' \setminus \{0\}\} \cup \{u' \to v' | i \to j \in A \wedge u' \in V'_i \wedge v' \in V'_j\}$. The weight vector $\boldsymbol{\psi}' \in \mathbb{R}^{|A'|}$ is such that $\forall u' \to v' \in A' : u' \in V'_u \wedge v' \in V'_v \Rightarrow \psi'_{u' \to v'} = \psi_{u \to v}$ and arcs leaving 0 have null weights.

We construct an AST $G'' = \langle V'', A'', l' \rangle$ such that $V'' = \{1, ..., n\}$, $A'' = \{i \to i+1 \mid 1 \leq i < n\}$ and the labeling function l' assigns the tag 0 to n and the tag 1 to every other vertex.

As such, there is a one-to-one correspondence between solutions of the maximum Hamiltonian path problem on graph G and solutions of the mapping of maximum weight of G'' with G'. □

B Experimental setup

The neural architecture used in our experiments to produce the weights $\boldsymbol{\mu}$ and $\boldsymbol{\phi}$ is composed of: (1) an embedding layer of dimension 100 for SCAN or `BERT-base` (Devlin et al., 2019) for the other datasets, followed by a bi-LSTM (Hochreiter and Schmidhuber, 1997) with a hidden size of 400; (2) a linear projection of dimension 500 over the output of the bi-LSTM followed by a TANH activation and another linear projection of dimension $|E|$ to obtain $\boldsymbol{\mu}$; (3) a linear projection of dimension 500 followed by a TANH activation and a bi-affine layer (Dozat and Manning, 2017) to obtain $\boldsymbol{\phi}$.

We apply dropout with a probability of 0.3 over the outputs of `BERT-base` and the bi-LSTM and after both TANH activations. The learning rate is 5×10^{-4} and each batch is composed of 30 exam-

ples. We keep the parameters that obtain the best accuracy on the development set after 25 epochs. Training the model takes between 40 minutes for GEOQUERY and 8 hours for CLEVR. However, note that the bottleneck is the conditional gradient method which is computed on the CPU.

C GEOQUERY denotation accuracy issue

The denotation accuracy is evaluated by checking whether the denotation returned by an executor is the same when given the gold semantic program and the prediction of the model. It can be higher than the exact match accuracy when different semantic programs yield the same denotation.

When we evaluated our approach using the same executor as the baselines of Herzig and Berant (2021), we observed two main issues regarding the behaviour of several predicates: (1) Several predicates have undefined behaviours (*e.g.* `population_1` and `traverse_2` in the case of an argument of type `country`), in the sense that they are not implemented; (2) The behaviour of some predicates are incorrect with respect to their expected semantic (*e.g.* `traverse_1` and `traverse_2`). These two sources of errors result in incorrect denotation for several semantic programs, leading to an overestimation of the denotation accuracy when both the gold and predicted programs return by accident an empty denotation (potentially for different reasons, due to aforementioned implementation issues).

We implemented a corrected executor addressing the issues that we found.