Minimalist Parsing of Heavy NP Shift

Lei Liu
Stony Brook University
lei.liu.1@stonybrook.edu

Abstract

This paper studies Heavy NP Shift (HNPS) from the perspective of parsing using Minimalist Grammar. Based on memory usage of the MG parsers, processing difficulties of HNPS as derived by rightward movement, PP movement and remnant movement are each compared with a non-movement structure. A set of complexity metrics show that shifted structures are indeed easier to parse than a non-movement structure when the NP is long.

1 Introduction

Heavy NP Shift (HNPS) refers to the tendency that long or phonologically "heavy" phrases are shifted to positions other than where they canonically occur. An English HNPS sentence is shown in (1a).

(1) a. Max put [$_{PP}$ in his car] [$_{NP}$ all the boxes of home furnishings]. (Larson 2014)
b. Max put [$_{NP}$ all the boxes of home furnishings] [$_{PP}$ in his car].
c. ??Max put [$_{PP}$ in his car] [$_{NP}$ boxes].

The canonically word-ordered or "unshifted" version of (1a) is the sentence in (1b). When the object NP is short, however, shifted word order is marked, as shown in (1c).

Popular analyses of HNPS include: *rightward movement* of NP (Ross 1986), where the heavy NP moves to the right edge of the constituent; the *PP movement* analysis (Kayne 1994), where the *PP* leftward moves; and the *remnant movement* analysis (Rochemont and Culicover 1997), where the heavy NP moves first, followed by movement of the "remnant" VP. The above analyses are schematized in (2-4) respectively.

(2) Max put t in his car <put> [all ... furnishings.]

(3) Max put __ all ... furnishings <put> [in his car]

(4) Max put __ __ [[all ... furnishings] <put> in his car]

These syntactic analyses, distinct as they are, are equally successful in deriving the English HNPS word order. However, since structural properties of a sentence predicts how hard it is for humans to process it, it is unclear what processing predictions these analyses make, nor is it clear whether these predictions are born out in observed human processing preferences.

Psycholinguistic studies on human sentence processing have shown that sentences with HNPS word order are preferred in production over the canonical word order when the NP is long (Stallings et al. 1998). Additionally, it has been observed that the likelihood of shifting heavy NPs relates not only to the length of NPs, but of PPs as well. As the length of a PP increases, i.e., as the length difference between the NP and the VP decreases, HNPS is less likely to happen (Stallings and MacDonald 2011). It is then interesting to explore whether and how well these psycholinguistic findings are predicted by a given structural analysis.

Minimalist Grammar (MG) parsing (Stabler 2013, Graf et al. 2017) provides a quantitative way to answer precisely these questions. As it will hopefully become clear, It is possible to infer and compare processing difficulties that associate with syntactic structures by observing the parser's behavior when conjecturing those structures. This enables us to see whether the reported human processing findings are expected when a certain syntactic structure is assumed.

In this paper, I investigate processing predictions the three aforementioned structural proposals make from the perspective of Minimalist parsing. I will show that the parser's behavior suggest that the *rightward movement* analysis correctly predict processing biases based on memory usage. *PP movement* and *remnant movement* analyses make correct predictions when unpronounced nodes are ignored by the memory usage calculation. Moreover, when contrasted with previous studies on complexity metrics (Graf et al. 2015, Zhang 2017, Graf et al. 2017), the same set of metrics that isd making correct processing predictions for relative clauses works for HNPS structures as well.

This paper is structured as follows. Section 2 introduces Minimalist Grammars, parsers as well as complexity metrics. Section 3 discusses how the comparisons are set up and what the results are. I conclude the paper with discussions in section 4.

2 Minimalist Grammars, Parsers and Complexity Metrics

The evaluation and comparisons of how difficult given syntactic structures are processed are based on complexity metrics measuring behaviors of MG parsers. In this section, I first discuss Minimalist Grammars and MG parsers. Complexity metrics will then be introduced with examples.

2.1 MG and Its Parser

Minimalist Grammars are grammar formalisms based on the Minimalist Program (Chomsky 2014). The formalism is mathematically defined in Stabler (1996), Graf (2012). Intuitively, MG rules are expressed in lexical items, which are essentially feature bundles containing information such as pronunciation, category, movement, etc. Similar to the standard Minimalist Program-styled derivation, these lexical items are built into sentences (trees) via `merge`, which combines lexical items and/or phrases; and `move`, which regulates movements. For a concrete example, lexical items for an English sentence *Max packed boxes* are listed in (5) while an MG derivation tree (modulo features on nodes) is shown in (6), next to a standard syntactic tree in (7).

(5) *Max* :: $\underset{\texttt{cat.}}{D^-}$ $\underset{\texttt{mvmt}}{nom^-}$

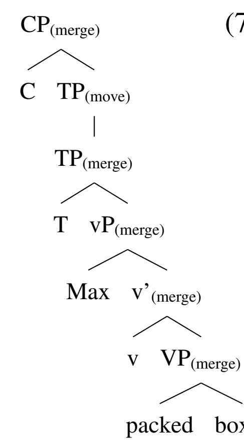
packed :: $\underset{\texttt{sel.}}{D^+}$ $\underset{\texttt{cat.}}{V^-}$

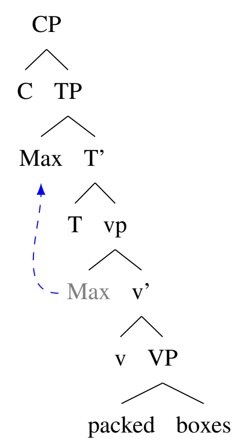
boxes :: $\underset{\texttt{cat.}}{D^-}$

C :: $\underset{\texttt{sel.}}{T^+}$ $\underset{\texttt{cat.}}{C^-}$

T :: $\underset{\texttt{sel.}}{v^+}$ $\underset{\texttt{mvmt}}{nom^+}$ $\underset{\texttt{cat.}}{T^-}$

v :: $\underset{\texttt{sel.}}{V^+}$ $\underset{\texttt{sel.}}{D^+}$ $\underset{\texttt{cat.}}{v^-}$

(6)


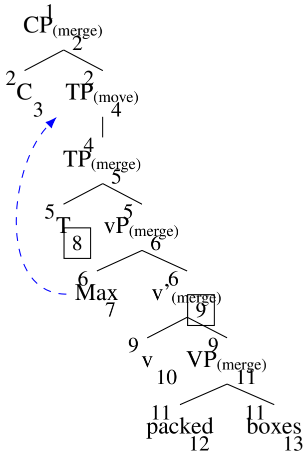
(7)


In MG, a derivation tree such as (6) is built from lexical items such as (5). Take the bottom-most VP in (6) for instance. The verb *packed* has a `sel`(ection) features marked for D^+. This means it must merge with a feature matching item, in this case, *boxes*, which is marked as D^-. The product of this merge is a VP, which is feature marked as V^-, the remaining feature on the head *packed* after D^+ and D^- "checks" each other out.

An MG parser essentially does `merge` and `move` in reverse. It takes a set of MG rules, a sentence, and conjectures derivation trees in a recursive-descend fashion. Again, take the sentence *Max packed boxes* for example. An annotated derivation tree outlining the parser's behavior is shown in (8).

32nd Pacific Asia Conference on Language, Information and Computation
Hong Kong, 1-3 December 2018

(8)



The numbers on the two corners of each nodes in (8) indicate steps at which the node is conjectured (superscripted numbers, or indices) and confirmed (subscripted numbers, or outdices) by the parser. In the example, the parser starts building the derivation tree from a CP (step 1-2); "un-"merges the CP into C and TP with an EPP movement landing site (step 2-5). Next, it "un-"merges TP into T and vP (step 5-8). The box on 8 indicates that the T node has been kept in the memory for non-trivial steps (steps > 2). According to the feature specification, T node could only be confirmed after all its features are checked. In this case, the movement licensor feature nom^+ is not checked until the the parser confirms the mover, *Max*, at step 7.

2.2 Complexity Metrics

Using the indices and outdices on a derivation tree, it is possible to infer memory usage of the parser when conjecturing that tree. Specifically, following Graf et al. (2017), one can measure how long a node is kept in memory, or **Tenure**; how many nodes are kept in memory, or **Payload** and how long movement dependencies stretch, or **Size**. Take the same annotated tree in (8) for example. The node T has a **Tenure** of 3 ($= 8 - 5$), which means it was kept in memory for 3 steps, as briefly mentioned before. The whole tree has a **Payload** of 2, which is the number of non-trivial nodes, namely, T and v', as indicated by the boxes on the outdices. The only movement in the tree has a **Size** of 2, which is calculated by subtract the step at which the landing site is confirmed (4 in this case) from the step at which a mover is conjectured (6 in this case).

From these notions, one can build a set of metrics quantify how how much more memory-usage intense a structure P is than Q (i.e., how much harder to parse it should be). Each metric predicts that a structure P is harder than Q if certain conditions are met. For example, **MaxT** predicts that P is harder to parse than Q when the maximum tenure among all tenured nodes in P is greater than that in Q; **SumS** predicts that P is harder to parse than Q when the sum of lengths of all the movements in P is greater that in Q; and **MaxS**^{R} makes the same prediction about P and Q when the furthest movement dependency in P is great than that in Q, while any ties between the two structures are ignored. The total number of base metrics used for this study is 20, among which the above three in the examples are arguably the most reliable ones, as we will see in the results.

3 Comparisons and Results

3.1 Setting Up Comparisons

Now that the metrics are defined, we can measure how difficult a proposed HNPS structure is compared to a canonical structure given the memory usage of the MG parser when conjecturing these structures. The comparisons are set up by specifying three parameters: a) target sentences, b) human processing biases of those sentences found in experiments, and c) syntactic structures of the target sentences.

The target sentences are constructed by controlling the length of DPs and PPs of the sentences (long and short, $2 \times 2 = 4$), and the word order (DP before PP and PP before DP). A total of four pairs of sentences are used in the comparisons. (9-12) show the DP before PP examples of the pairs.

(9) Max put [$_{\text{DP}}$ boxes] [$_{\text{PP}}$ in his car].
short DP, short PP (`sdp_spp`)

(10) Max put [$_{\text{DP}}$ boxes] [$_{\text{PP}}$ in his car made in Stuttgart]. `sdp_lpp`

(11) Max put [$_{\text{DP}}$ all the boxes of home furnishings] [$_{\text{PP}}$ in his car].
`ldp_spp` (heavy NP)

(12) Max put [$_{\text{DP}}$ all the boxes of home furnishings] [$_{\text{PP}}$ in his car made in Stuttgart].
`ldp_lpp`

Human processing difficulties of the sentences of the above four types have been reported in experimental and theoretical literature. The HNPS word order is preferred over the canonical word order with long NPs and short PPs (`ldp_spp`) (Stallings et al. 1998). As the length of PPs increases (`ldp_lpp`), the HNPS word order is no longer preferred (Stallings and MacDonald 2011). Additionally, for the sentences with short DPs, the shifted word order is ungrammatical (Ross 1986), or at least not preferred, as shown in (13).

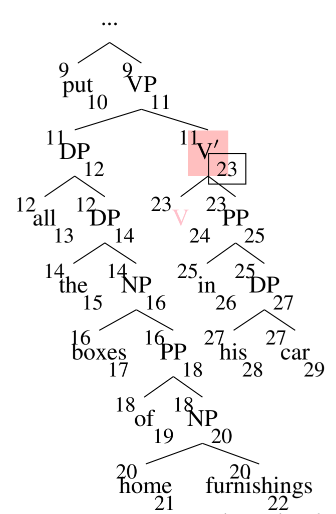
(13) a. ??Max put [$_{PP}$ in his car] [$_{DP}$ boxes].
b. *Max put [$_{PP}$ in his car made in Stuttgart] [$_{DP}$ boxes].

Syntactic proposals deriving each sentence with shifted word order were *rightward movement*, *PP movement* and *remnant movement*, as discussed earlier. They are compared in a pairwise fashion with *no movement*, which derives canonical word order. A total of 12 comparisons are conducted (4 phrase lengths × 3 syntactic proposals = 12). Each comparison asks the question whether the metrics can predict reported processing difficulties across sentence types given a certain analysis. For instance, if it is *rightward movement* that is currently in question, the comparison is setup such that, for the `ldp_spp` condition, i.e., the HNPS configuration, the rightward movement structure is easier to parse than the no movement structure. For the remaining three conditions, the no movement structure is easier to parse. And we test how successful the metrics are in predicting these processing biases. A collection of Python scripts are used for the comparisons, taking as input pairs of syntactic structures, processing bias of these pairs, and a set of complexity metrics; and outputs whether the set of metrics are successful, unsuccessful, or neutral in predicting those bias.

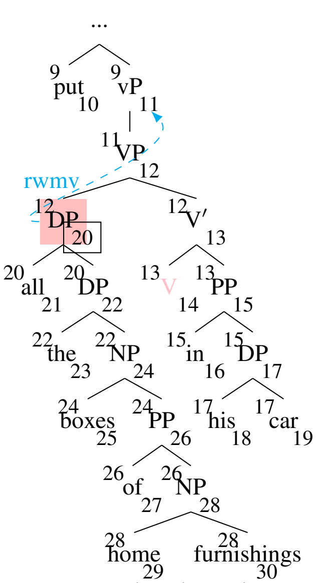
3.2 Results

Results of the comparisons show that, first, the MG parser's behavior predicts that HNPS sentence is less difficult to parse than its canonically ordered counterpart, as expected. Recall that among the four length conditions, a shifted structure is predicted to be easier in the pair only for the `ldp_spp` condition. 8 out of 10 tenure based metrics were able to predict this processing bias for *rightward movement* analysis. The performance of each of the 20 metrics in the twelve conditions can be found in Appendix A.

Relevant annotated derivation trees confirms these results. For the heavy NP condition (`ldp_spp`), if the heavy NP does not move, the parser would have to fully build the heavy NP part until it can go back to the earlier branch to continue work on the PP. This causes a greater tenure on the V' node as shown in (14). In contrast, rightward movement essentially delays the heavy lifting of building the NP. Since the size of PP, or of the right branch, is much smaller than its left branch, the tenure on the left branching node is smaller than that on the right branch of a canonical structure, as shown in (15).

(14)


Canonical word order

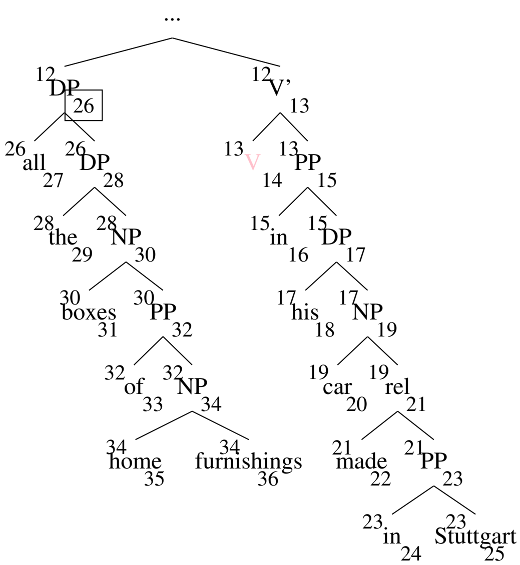
(15)


HNPS with rightward movement

32nd Pacific Asia Conference on Language, Information and Computation
Hong Kong, 1-3 December 2018

It is also not difficult to see from above that as the right sibling of the heavy NP, or in fact, the lower PP grows in length, the shifted order would no longer be preferred by the same complexity metrics. Comparison results from the `ldp_lpp` condition show exactly this, as demonstrated by a *rightward movement* case in (16).

(16)



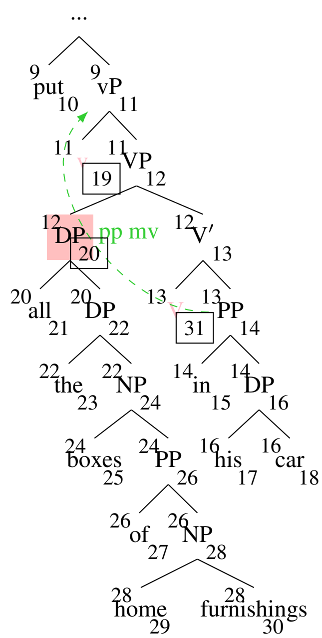
ldp_lpp with rightward movement

Recall that for a rightward movement structure, the parser builds the right branch before it returns to the left branch, the DP that has rightward moved. When the PP is also long, the tenure on the left branching DP node increases as a result. And in this particular case, the shifted structure (16), is no longer preferred in terms of memory usage when compared to the canonical structure in (14), because of the greater tenure on the DP node.

Second, the results suggest that the *PP movement* and *remnant movement* analyses also predict processing advantage of HNPS when unpronounced nodes are ignored. 7 out of 10 and 8 out of 10 tenure-based filtered metrics were successful in predicting processing biases for the PP movement and remnant movement analyses respectively. A performance summary of the metrics can be found in Appendix A.

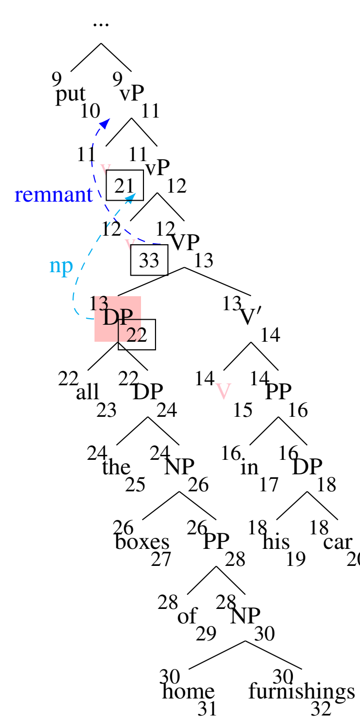
Graf et al. (2017) note that excluding unpronounced nodes from memory usage calculation can improve the performance of tenure-based metrics. In our case, as can be seen in (17) and (18), the nodes with large tenure are unpronounced nodes Vs and vs. Once these nodes are excluded for memory usage calculation, HNPS are predicted to be easier to process for the *PP movement* and *remnant movement* analyses, as indicated by the relative small tenures on those shaded tenured nodes.

(17)



HNPS with PP movement

(18)



HNPS with remnant movement

Moreover, ranked complexity metrics that are successful in predicting processing bias for other

32nd Pacific Asia Conference on Language, Information and Computation
Hong Kong, 1-3 December 2018

syntactic structures also make correct predictions for HNPS when a *rightward movement* structure is assumed. Ranked metrics are metrics of the form $< M1, M2 >$, which compares structures according to metric M1. When M1 is a tie, M2 is used. Graf et al. (2017), Zhang (2017) note that the ranked metrics $<$ **MaxT, SumS** $>$ and $<$ **MaxT, MaxS**^{R} $>$ were able to make correct processing predictions for relative clauses across several languages. These two metrics were also successful in predicting sentence processing bias across conditions in the current study, when assuming *rightward movement* analysis.

4 Discussion and conclusion

The results of the current study first provide evidence for a memory usage-based view of HNPS as discussed in incremental language production model (Stallings and MacDonald 2011). On the one hand, memory usage by the parser reliably predicts processing advantage of HNPS structures. On the other hand, the relation between DP-PP length conditions and their processing difficulties follow directly from the syntactic structures that the MG parser is building.

Furthermore, the MG parsing model provides a fresh perspective of viewing the three competing analyses. Given the processing predictions, complexity metrics favor *rightward movement* analysis over the rest. This is because, from the parser's perspective, assuming *rightward movement* delays building a large phrase, which decreases tenure on its sister node. Assuming the other two analyses does not have this effect.

To conclude, this paper studies HNPS from a MG parsing perspective. Memory usage-based metrics suggest that HNPS are easier to parse. For correct processing predictions, *rightward movement* is favored by the parser while *PP movement* and *remnant movement* analyses requires filtering unpronounced nodes.

References

Chomsky, N. (2014). *The minimalist program*. MIT press.

Graf, T. (2012). Locality and the complexity of minimalist derivation tree languages. In *Formal Grammar*, pages 208–227. Springer.

Graf, T., Fodor, B., Monette, J., Rachiele, G., Warren, A., and Zhang, C. (2015). A refined notion of memory usage for minimalist parsing. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*, pages 1–14.

Graf, T., Monette, J., and Zhang, C. (2017). Relative clauses as a benchmark for Minimalist parsing. *Journal of Language Modelling*, 5:57–106.

Kayne, R. S. (1994). *The antisymmetry of syntax*. Number 25. mit Press.

Larson, R. K. (2014). *On shell structure*. Routledge.

Rochemont, M. and Culicover, P. W. (1997). Deriving dependent right adjuncts in english. *Rightward movement*, pages 279–300.

Ross, J. R. (1986). *Infinite syntax*. Ablex Publishing Corporation.

Stabler, E. (1996). Derivational minimalism. In *International Conference on Logical Aspects of Computational Linguistics*, pages 68–95. Springer.

Stabler, E. P. (2013). Two models of minimalist, incremental syntactic analysis. *Topics in cognitive science*, 5(3):611–633.

Stallings, L. M. and MacDonald, M. C. (2011). It's not just the "heavy np": relative phrase length modulates the production of heavy-np shift. *Journal of psycholinguistic research*, 40(3):177–187.

Stallings, L. M., MacDonald, M. C., and O'Seaghdha, P. G. (1998). Phrasal ordering constraints in sentence production: Phrase length and verb disposition in heavy-np shift. *Journal of Memory and Language*, 39(3):392–417.

Zhang, C. (2017). *Stacked Relatives: Their Structure, Processing and Computation*. PhD thesis, State University of New York at Stony Brook.

Appendix A. Comparison Tables

Metric	Filters	sdp_spp	sdp_lpp	ldp_spp	ldp_lpp
AvgS		×	×	✓	×
AvgS’		×	×	✓	×
AvgT		✓	✓	✓	✓
AvgT’		✓	✓	✓	✓
BoxT		✓	✓	Tie	Tie
BoxT’		✓	✓	×	✓
MaxS		Tie	Tie	Tie	Tie
MaxS’		Tie	Tie	Tie	Tie
MaxSR		✓	✓	×	✓
MaxSR’		✓	✓	×	✓
MaxT		✓	✓	✓	✓
MaxT’		✓	✓	✓	✓
MaxTR		✓	✓	✓	✓
MaxTR’		✓	✓	✓	✓
Movers		✓	✓	×	✓
Movers’		✓	✓	×	✓
SumS		✓	✓	×	✓
SumS’		✓	✓	×	✓
SumT		✓	✓	✓	✓
SumT’		✓	✓	✓	✓

Table 1: no movement vs. rightward movement

Metric	Filters	sdp_spp	sdp_lpp	ldp_spp	ldp_lpp
AvgS		Tie	Tie	Tie	Tie
AvgS’		Tie	Tie	Tie	Tie
AvgT	U	✓	✓	✓	✓
AvgT’	U	✓	✓	✓	×
BoxT	U	✓	✓	Tie	Tie
BoxT’	U	✓	✓	×	✓
MaxS		Tie	Tie	Tie	Tie
MaxS’		Tie	Tie	Tie	Tie
MaxSR		✓	✓	×	✓
MaxSR’		✓	✓	×	✓
MaxT	U	✓	✓	✓	✓
MaxT’	U	✓	✓	✓	✓
MaxTR	U	✓	✓	✓	✓
MaxTR’	U	✓	✓	✓	✓
Movers		✓	✓	×	✓
Movers’		✓	✓	×	✓
SumS		✓	✓	×	✓
SumS’		✓	✓	×	✓
SumT	U	✓	✓	✓	✓
SumT’	U	✓	✓	✓	✓

Table 2: no movement vs. PP movement
(U = ignore unpronounced nodes)

Metric	Filters	sdp_spp	sdp_lpp	ldp_spp	ldp_lpp
AvgS		×	×	✓	×
AvgS’		×	×	✓	×
AvgT	U	✓	✓	✓	✓
AvgT’	U	✓	✓	✓	✓
BoxT	U	✓	✓	Tie	Tie
BoxT’	U	✓	✓	×	✓
MaxS		Tie	Tie	Tie	Tie
MaxS’		Tie	Tie	Tie	Tie
MaxSR		✓	✓	×	✓
MaxSR’		✓	✓	×	✓
MaxT	U	✓	✓	✓	✓
MaxT’	U	✓	✓	✓	✓
MaxTR	U	✓	✓	✓	✓
MaxTR’	U	✓	✓	✓	✓
Movers		✓	✓	×	✓
Movers’		✓	✓	×	✓
SumS		✓	✓	×	✓
SumS’		✓	✓	×	✓
SumT	U	✓	✓	✓	✓
SumT’	U	✓	✓	✓	✓

Table 3: no movement vs. remnant movement
(U = ignore unpronounced nodes)