# Time Mapping with Hypergraphs

**Jan W. Amtrup**
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003,USA
email: jamtrup@crl.nmsu.edu

**Volker Weber**
University of Hamburg,
Computer Science Department,
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany
email: weber@informatik.uni-hamburg.de

## Abstract

Word graphs are able to represent a large number of different utterance hypotheses in a very compact manner. However, usually they contain a huge amount of redundancy in terms of word hypotheses that cover almost identical intervals in time. We address this problem by introducing hypergraphs for speech processing. Hypergraphs can be classified as an extension to word graphs and charts, their edges possibly having several start and end vertices. By converting ordinary word graphs to hypergraphs one can reduce the number of edges considerably. We define hypergraphs formally, present an algorithm to convert word graphs into hypergraphs and state consistency properties for edges and their combination. Finally, we present some empirical results concerning graph size and parsing efficiency.

## 1 Introduction

The interface between a word recognizer and language processing modules is a crucial issue with modern speech processing systems. Given a sufficiently high word recognition rate, it suffices to transmit the most probable word sequence from the recognizer to a subsequent module (e.g. a parser). A slight extension over this *best chain* mode would be to deliver *n-best chains* to improve language processing results.

However, it is usually not enough to deliver just the best 10 or 20 utterances, at least not for reasonable sized applications given todays speech recognition technology. To overcome this problem, in most current systems word graphs are used as speech-language interface. Word graphs offer a simple and efficient means to represent a very high number of utterance hypotheses in a extremely compact way (Oerder and Ney, 1993; Aubert and Ney, 1995).

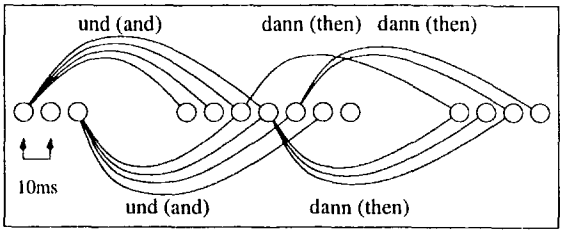


Figure 1: Two families of edges in a word graph

Although they are compact, the use of word graphs leads to problems by itself. One of them is the current lack of a reasonable measure for word graph size and evaluation of their contents (Amtrup et al., 1997). The problem we want to address in this paper is the presence of a large number of *almost identical* word hypotheses. By almost identical we mean that the start and end vertices of edges differ only slightly.

Consider figure 1 as an example section of a word graph. There are several word hypotheses representing the words und (and) and dann (then). The start and end points of them differ by small numbers of *frames*, each of them 10ms long. The reasons for the existence of these *families of edges* are at least twofold:

- Standard HMM-based word recognizers try to start (and finish) word models at each individual frame. Since the resolution is quite high (10ms, in many cases shorter than the word onset), a word model may have boundaries at several points in time.
- Natural speech (and in particular spontaneously produced speech) tends to blur word boundaries. This effect is in part responsible for the dramatic decrease in word recognition rate, given fluent speech as input in contrast to isolated words as input. Figure 1 demonstrates the inaccuracy

of word boundaries by containing several meeting points between und and dann, emphasized by the fact that both words end resp. start with the same consonant.

Thus, for most words, there is a whole set of word hypotheses in a word graph which results in several meets between two sets of hypotheses. Both facts are disadvantageous for speech processing: Many word edges result in a high number of lexical lookups and basic operations (e.g. bottom-up proposals of syntactic categories); many meeting points between edges result in a high number of possibly complex operations (like unifications in a parser).

The most obvious way to reduce the number of neighboring, identically labeled edges is to reduce the time resolution provided by a word recognizer (Weber, 1992). If a word edge is to be processed, the start and end vertices are mapped to the more coarse grained points in time used by linguistic modules and a redundancy check is carried out in order to prevent multiple copies of edges. This can be easily done, but one has to face the drawback on introducing many more paths through the graph due to artificially constructed overlaps. Furthermore, it is not simple to choose a correct resolution, as the intervals effectively appearing with word onsets and offsets change considerably with words spoken. Also, the introduction of cycles has to be avoided.

A more sophisticated schema would use interval graphs to encode word graphs. Edges of interval graphs do not have individual start and end vertices, but instead use intervals to denote the range of applicability of an edge. The major problem with interval graphs lies with the complexity of edge access methods. However, many formal statements shown below will use interval arithmetics, as the argument will be easier to follow.

The approach we take in this paper is to use hypergraphs as representation medium for word graphs. What one wants is to carry out operations only once and record the fact that there are several start and end points of words. Hypergraphs (Gondran and Minoux, 1984, p. 30) are generalizations of ordinary graphs that allow multiple start and end vertices of edges.

We extend the approach of H. Weber (Weber, 1995) for time mapping. Weber considered sets of edges with identical start vertices but slightly different end vertices, for which the notion *family* was introduced. We use full hypergraphs as representation and thus additionally allow several start vertices, which results in a further decrease of 6% in terms of resulting chart edges while parsing (cf. section 3). Figure 2 shows the example section using hyperedges for the two families of edges. We adopt the way of dealing with different acoustical scores of word hypotheses from Weber.

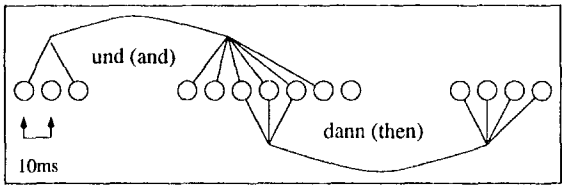


Figure 2: Two hyperedges representing families of edges

## 2 Word Graphs and Hypergraphs

As described in the introduction, word graphs consist of edges representing word hypotheses generated by a word recognizer. The start and end point of edges usually denote points in time. Formally, a word graph is a directed, acyclic, weighted, labeled graph with distinct root and end vertices. It is a quadruple $G = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{L})$ with the following components:

- A nonempty set of graph vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$. To associate vertices with points in time, we use a function $t : \mathcal{V} \longrightarrow N$ that returns the frame number for a given vertex.
- A nonempty set of weighted, labeled, directed edges $\mathcal{E} = \{e_1, \ldots, e_m\} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{W} \times \mathcal{L}$. To access the components of an edge $e = (v, v', w, l)$, we use functions $\alpha$, $\beta$, $w$ and $l$, which return the start vertex ($\alpha(e) = v$), the end vertex ($\beta(e) = v'$), the weight ($w(e) = w$) and the label ($l(e) = l$) of an edge, respectively.
- A nonempty set of edge weights $\mathcal{W} = \{w_1, \ldots, w_p\}$. Edge weights normally represent a the acoustic score assigned to the word hypothesis by a HMM based word recognizer.

- A nonempty set of Labels $\mathcal{L} = \{l_1, \ldots, l_o\}$, which represents information attached to an edge, usually words.

We define the relation of *reachability* for vertices ($\rightarrow$) as $\forall v, w \in \mathcal{V} : v \rightarrow w \Leftrightarrow \exists e \in \mathcal{E} : \alpha(e) = v \wedge \beta(e) = w$

The transitive hull of the reachability relation $\rightarrow$ is denoted by $\overset{*}{\rightarrow}$.

We already stated that a word graph is acyclic and distinctly rooted and ended.

### 2.1 Hypergraphs

Hypergraphs differ from graphs by allowing several start and end vertices for a single edge. In order to apply this property to word graphs, the definition of edges has to be changed. The set of edges $\mathcal{E}$ becomes a nonempty set of weighted, labeled, directed hyperedges $\mathcal{E} = \{e_1, \ldots, e_m\} \subseteq \mathcal{V}^* \backslash \emptyset \times \mathcal{V}^* \backslash \emptyset \times \mathcal{W} \times \mathcal{L}$.

Several notions and functions defined for ordinary word graphs have to be adapted to reflect edges having sets of start and end vertices.

- The accessor functions for start and end vertices have to be adapted to return sets of vertices. Consider an edge $e = (V, V', w, l)$, then we redefine

$$\alpha : \mathcal{E} \longrightarrow \mathcal{V}^*, \alpha(e) := V \quad (1)$$
$$\beta : \mathcal{E} \longrightarrow \mathcal{V}^*, \beta(e) := V' \quad (2)$$

- Two hyperedges $e, e'$ are adjacent, if they share a common vertex:

$$\beta(e) \cap \alpha(e') \neq \emptyset \quad (3)$$

- The reachability relation is now $\forall v, w \in \mathcal{V} : v \rightarrow w \Leftrightarrow \exists e \in \mathcal{E} : v \in \alpha(e) \wedge w \in \beta(e)$

Additionally, we define accessor functions for the first and last start and end vertex of an edge. We recur to the association of vertices with frame numbers, which is a slight simplification (in general, there is no need for a total ordering on the vertices in a word graph)[1]. Furthermore, the intervals covered by start and end vertices are defined.

$$\alpha_<(e) := \text{argmin}\{t(v) | v \in V\} \quad (4)$$
$$\alpha_>(e) := \text{argmax}\{t(v) | v \in V\} \quad (5)$$
$$\beta_<(e) := \text{argmin}\{t(v) | v \in V'\} \quad (6)$$
$$\beta_>(e) := \text{argmax}\{t(v) | v \in V'\} \quad (7)$$
$$\alpha_{[]}(e) := [t(\alpha_<(e)), t(\alpha_>(e))] \quad (8)$$
$$\beta_{[]}(e) := [t(\beta_<(e)), t(\beta_>(e))] \quad (9)$$

In contrast to interval graphs, we do not require the sets of start and end vertices to be contiguous, i.e. there may be vertices that fall in the range of the start or end vertices of an edge which are not members of that set. If we are not interested in the individual members of $\alpha(e)$ or $\beta(e)$, we merely talk about interval graphs.

### 2.2 Edge Consistency

Just like word graphs, we demand that hypergraphs are acyclic, i.e. $\forall v \overset{*}{\rightarrow} w : v \neq w$. In terms of edges, this corresponds to $\forall e : t(\alpha_>(e)) < t(\beta_<(e))$.

### 2.3 Adding Edges to Hypergraphs

Adding a simple word edge to a hypergraph is a simplification of merging two hyperedges bearing the same label into a new hyperedge. Therefore we are going to explain the more general case for hyperedge merging first. We analyze which edges of a hypergraph may be merged to form a new hyperedge without loss of linguistic information. This process has to follow three main principles:

- Edge labels have to be identical
- Edge weights (scores) have to be combined to a single value
- Edges have to be compatible in their start and end vertices and must not introduce cycles to the resulting graph

#### Simple Rule Set for Edge Merging

Let $e_1, e_2 \in \mathcal{E}$ be two hyperedges to be checked for merging, where $e_1 = (V_1, V_1', w_1, l_1)$ and $e_2 = (V_2, V_2', w_2, l_2)$. Then $e_1$ and $e_2$ could be merged into a new hyperedge $e_3 = (V_3, V_3', w_3, l_3)$ iff

$$l(e_1) = l(e_2) \quad (10)$$
$$\min(t(\beta_<(e_1)), t(\beta_<(e_2))) > \max(t(\alpha_>(e_1)), t(\alpha_>(e_2))) \quad (11)$$

where $e_3$ is:

$$l_3 = l_1 \quad (= l_2) \quad (12)$$
$$w_3 = \text{scorejoin}(w_1, w_2)^2 \quad (13)$$

[1] The total ordering on vertices is naturally given through the linearity of speech.

$$V_3 = V_1 \cup V_2 \quad (14)$$
$$V_3' = V_1' \cup V_2' \quad (15)$$

$e_1$ and $e_2$ have to be removed from the hypergraph while $e_3$ has to be inserted.

**Sufficiency of the Rule-Set**

Why is this set of two conditions sufficient for hyperedge merging? First of all it is clear that we can merge only hyperedges with the same label (this is prescribed by condition 10). Condition 11 gives advice which hyperedges could be combined and prohibits cycles to be introduced in the hypergraph. An analysis of the occuring cases shows that this condition is reasonable. Without loss of generality, we assume that $t(\beta_>(e_1)) \leq t(\beta_>(e_2))$.

1. $\alpha_{[]}(e_1) \cap \beta_{[]}(e_2) \neq \emptyset \vee \alpha_{[]}(e_2) \cap \beta_{[]}(e_1) \neq \emptyset$:
   This is the case where either the start vertices of $e_1$ and the end vertices of $e_2$ or the start vertices of $e_2$ and end vertices of $e_1$ overlap each other. The merge of two such hyperedges of this case would result in a hyperedge $e_3$ where $t(\alpha_>(e_3)) \geq t(\beta_<(e_3))$. This could introduce cycles to the hypergraph. So this case is excluded by condition 11.
2. $\alpha_{[]}(e_1) \cap \beta_{[]}(e_2) = \emptyset \wedge \alpha_{[]}(e_2) \cap \beta_{[]}(e_1) = \emptyset$:
   This is the complementary case to 1.
   (a) $t(\alpha_<(e_2)) \geq t(\beta_>(e_1))$
   This is the case where all vertices of hyperedge $e_1$ occur before all vertices of hyperedge $e_2$ or in other words the case where two individual independent word hypotheses with same label occur in the word graph. This case must also not result in an edge merge since $\beta_{[]}(e_1) \subseteq [t(\alpha_<(e_1)), t(\alpha_>(e_2))]$ in the merged edge. This merge is prohibited by condition 11 since all vertices of $\beta(e_1)$ have to be smaller than all vertices of $\alpha(e_2)$.
   (b) $t(\alpha_<(e_2)) < t(\beta_>(e_1))$
   This is the complementary case to (a).
      i. $t(\alpha_<(e_1)) \geq t(\beta_>(e_2))$
      This is only a theoretical case because $t(\alpha_<(e_1)) \leq t(\alpha_>(e_1)) < t(\beta_<(e_1)) \leq t(\beta_<(e_2)) \leq t(\beta_>(e_2))$ is required ($e_2$ contains the last end vertex).
      ii. $t(\alpha_<(e_1)) < t(\beta_>(e_2))$
      This is the complementary case to i. As a result of the empty intersections and the cases (b) and ii we get $t(\alpha_>(e_1)) < t(\beta_<(e_2))$ and $t(\alpha_>(e_2)) < t(\beta_<(e_1))$. That is in other words $\forall t_\alpha \in \alpha_{[]}(e_1) \cup \alpha_{[]}(e_2), t_\beta \in \beta_{[]}(e_1) \cup \beta_{[]}(e_2) : t_\alpha < t_\beta$ and just the case demanded by condition 2.

After analyzing all cases of merging of intersections between start and end vertices of two hyperedges we turn to insertion of word hypotheses to a hypergraph. Of course, a word hypothesis could be seen as interval edge with trivial intervals or as a hyperedge with only one start and one end vertex. Since this case of adding an edge to a hypergraph is rather easy to depict and is heavily used while parsing word graphs incrementally we discuss it in more detail.

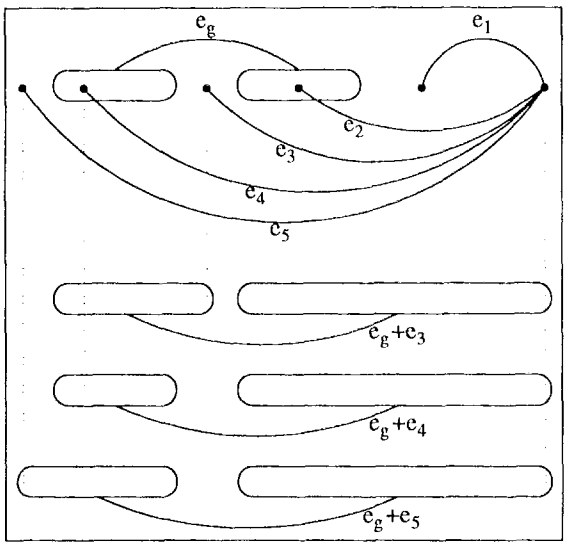


Figure 3: Cases for adding an edge to the graph

The speech decoder we use delivers word hypotheses incrementally and ordered by the time stamps of their end vertices. For practical reasons we further sort the start vertices with equal end vertex of a hypergraph by time. Under this precondition we get the cases shown in figure 3. The situation is such that $e_g$ is a hyperedge already constructed and $e_1 - e_5$ are candidates for insertion.

[2]Examples for the scorejoin operation are given later in the paragraph about score normalization.

```
function AddEdge(G:Hypergraph,e_n:Wordhypothesis)
        → HyperGraph
begin
[1]   if  ∃e_k ∈ E(G) with
          l(e_k) = l(e_n) ∧ t(β<(e_k)) > t(α(e_n)) then
[2]       Modify edge e_k
          e'_k := (α(e_k) ∪ {α(e_n)},
                   β(e_k) ∪ {β(e_n)},
                   max(w(e_k),
                       normalize(w(e_n), α(e_n), β(e_n))),
                   l(e_k))
          return
              G' := (V ∪ {α(e_n), β(e_n)},
                     E\e_k ∪ {e'_k},
                     W\w(e_k) ∪ {w(e'_k)}, L)
[3]   else
[4]       Add edge e'_n
          e'_n := ({α(e_n)}, {β(e_n)},
                   normalize(w(e_n), α(e_n), β(e_n)),
                   l(e_n))
          return
              G' := (V ∪ {α(e_n), β(e_n)}, E ∪ {e'_n},
                     W ∪ {w(e'_n)}, L ∪ {l(e'_n)})
end
```

Figure 4: An algorithm for adding edges to hypergraphs

It is not possible to add $e_1$ and $e_2$ to the hyperedge $e_g$ since they would introduce an overlap between the sets of start and end vertices of the potential new hyperedge. The resulting hyperedges of adding $e_3 - e_5$ are depicted below.

### Score Normalization

Score normalization is a necessary means if one wants to compare hypotheses of different lengths. Thus, edges in word graphs are assigned normalized scores that account for words of different extensions in time. The usual measure is the *score per frame*, which is computed by taking $\frac{\text{Score per word}}{\text{Length of word in frames}}$.

When combining several word edges as we do by constructing hyperedges, the combination should be assigned a single value that reflects a certain useful aspect of the originating edges. In order not to exclude certain hypotheses from consideration in score-driven language processing modules, the score of the hyperedge is inherited from the best-rated word hypothesis (cf. (Weber, 1995)). We use the minimum of the source acoustic scores, which corresponds to a highest recognition probability.

### Introducing a Maximal Time Gap

The algorithm depicted in figure 4 can be speeded up for practical reasons. Each vertex between the graph root and the start vertex of the new edge could be one of the start vertices of a matching hyperdege. In practice this is not needed and we could check a smaller amount of vertices. We do this by introducing a *maximal time gap* which gives us advice how far (in measures of time) we look backwards from the start vertex of a new edge to be inserted into the hypergraph to determine a compatible hyperedge of the hypergraph.

### Additional Paths

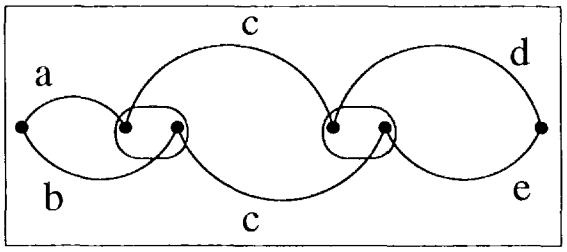


Figure 5: Additional paths by time mapping

It is possible to introduce additional paths into a graph by performing time mapping. Consider fig. 5 as an example. Taken as a normal word graph, it contains two label sequences, namely `a-c-d` and `b-c-e`. However, if time mapping is performed for the edges labelled `c`, two additional sequences are introduced: `a-c-e` and `b-c-d`. Thus, time mapping by hypergraphs is not information preserving in a strong sense. For practical applications this does not present any problems. The situations in which additional label sequences are introduced are quite rare, we did not observe any linguistic difference in our experiments.

## 2.4 Edge Combination

Besides merging the combination of hyperedges, to construct edges with new content is an important task within any speech processing module, e.g. for parsing. The assumption we will adopt here is that two hyperedges $e_1, e_2 \in \mathcal{E}$ may be combined if they are adjacent, i.e. they share a common vertex: $\beta(e_1) \cap \alpha(e_2) \neq \emptyset$. The label of the new edge $e_n$ (which may be used to represent linguistic content) is determined by the component building it, whereas the start and end vertices are determined by

$$\alpha(e_n) := \alpha(e_1) \quad (16)$$

$$\beta(e_n) := \beta(e_2) \quad (17)$$

This approach is quite analogous to edge combination methods for normal graphs, e.g. in chart parsing, where two edges are equally required to have a meeting point. However, score computation for hyperedge combination is more difficult. The goal is to determine a score per frame by selecting the smallest possible score under all possible meeting vertices. It is derived by examining all possible connecting vertices (all elements of $I := \beta(e_1) \cap \alpha(e_2)$) and computing the resulting score of the new edge: If $w(e_1) \leq w(e_2)$, we use
$w(e_n) := \frac{w(e_1)\cdot(t_< - t(\alpha_>(e_1))) + w(e_2)\cdot(t(\beta_>(e_2)) - t_<)}{t(\beta_>(e_2)) - t(\alpha_>(e_1))}$,
where $t_< = \min\{t(v)|v \in I\}$. If, on the other hand, $w(e_1) > w(e_2)$, we use
$w(e_n) := \frac{w(e_1)\cdot(t_> - t(\alpha_<(e_1))) + w(e_2)\cdot(t(\beta_<(e_2)) - t_>)}{t(\beta_<(e_2)) - t(\alpha_<(e_1))}$,
where $t_> = \max\{t(v)|v \in I\}$.

## 3 Experiments with Hypergraphs

The method of converting word graphs to hypergraphs has been used in two experiments so far. One of them is devoted to the study of connectionist unification in speech applications (Weber, forthcoming). The other one, from which the performance figures in this section are drawn, is an experimental speech translation system focusing on incremental operation and uniform representation (Amtrup, 1997).

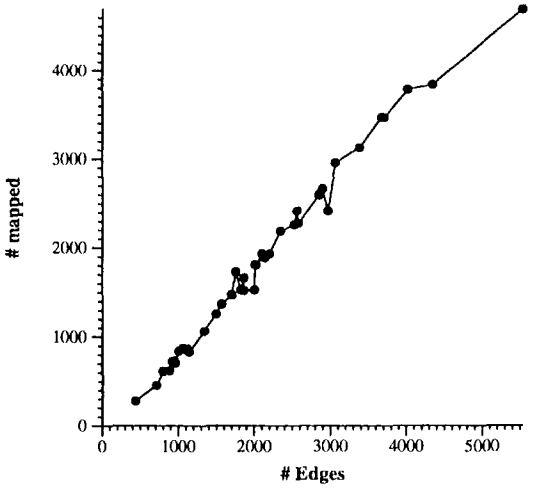


Figure 6: Word edge reduction

We want to show the effect of hypergraphs regarding edge reduction and parsing effort. In order to provide real-world figures, we used word graphs produced by the Hamburg speech recognition system (Huebener et al., 1996). The test data consisted of one dialogue within the Verbmobil domain. There were 41 turns with an average length of 4.65s speaking time per turn. The word graphs contained 1828 edges on the average. Figure 6 shows the amount of reduction in the number of edges by converting the graphs into hypergraphs. On the average, 1671 edges were removed (mapped), leaving 157 edges in hypergraphs, approximately 91% less than the original word graphs.

Next, we used both sets of graphs (the original word graphs and hypergraphs) as input to the speech parser used in (Amtrup, 1997). This parser is an incremental active chart parser which uses a typed feature formalism to describe linguistic entities. The grammar is focussed on partial parsing and contains rules mainly for noun phrases, prepositional phrases and such. The integration of complete utterances is neglected. Figure 7 shows the reduction in terms of chart edges at completion time.

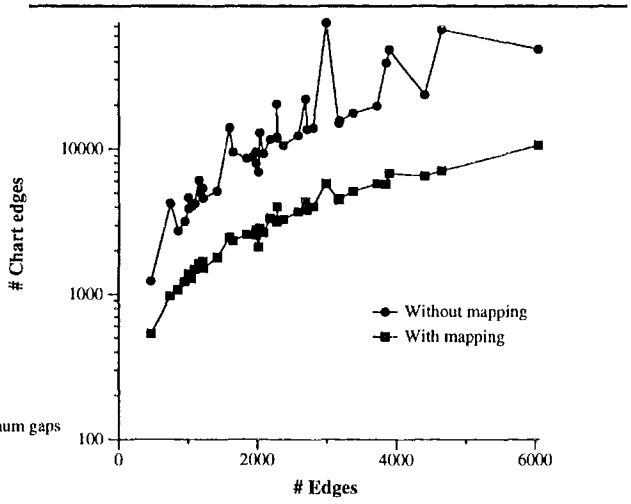


Figure 7: Chart edge reduction

The amount of reduction concerning parsing effort is much less impressive than pure edge reduction. On the average, parsing of complete graphs resulted in 15547 chart edges, while parsing of hypergraphs produced 3316 chart edges, a reduction of about 79%. Due to edge combinations, one could have expected a much higher value. The reason for this fact lies mainly with the redundancy test used in the parser. There

are many instances of edges which are not inserted into the chart at all, because identical hypotheses are already present.

Consequently, the amount of reduction in parse time is within the same bounds. Parsing ordinary graphs took 87.7s, parsing of hypergraphs 6.4s, a reduction of 93%. There are some extreme cases of word graphs, where hypergraph parsing was 94 times faster than word graph parsing. One of the turns had to be excluded from the test set, because it could not be fully parsed as word graph.

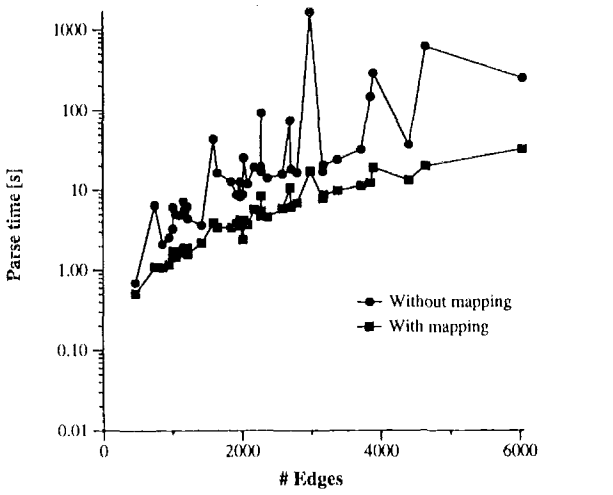


Figure 8: Parsing time reduction

## 4 Conclusion

In this paper, we have proposed the application of hypergraph techniques to word graph parsing. Motivated by linguistic properties of spontaneously spoken speech, we argued that bundles of edges in word graphs should be treated in an integrated manner. We introduced interval graphs and directed hypergraphs as representation devices. Directed hypergraphs extend the notion of a family of edges in that they are able to represent edges having several start and end vertices.

We gave a formal definition of word graphs and the necessary extensions to cover hypergraphs. The conditions that have to be fulfilled in order to merge two hyperedges and to combine two adjacent hyperedges were stated in a formal way; an algorithm to integrate a word hypothesis into a hypergraph was presented.

We proved the applicability of our mechanisms by parsing one dialogue of real-world spoken utterances. Using hypergraphs resulted in a 91% reduction of initial edges in the graph and a 79% reduction in the total number of chart edges. Parsing hypergraphs instead of ordinary word graphs reduced the parsing time by 93%.

## References


Jan W. Amtrup, Henrik Heine, and Uwe Jost. 1997. What's in a Word Graph — Evaluation and Enhancement of Word Lattices. In *Proc. of Eurospeech 1997*, Rhodes, Greece, September.

Jan W. Amtrup. 1997. Layered Charts for Speech Translation. In *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '97*, Santa Fe, NM, July.

Xavier Aubert and Hermann Ney. 1995. Large Vocabulary Continuous Speech Recognition Using Word Graphs. In *ICASSP 95.*

Michel Gondran and Michel Minoux. 1984. *Graphs and algorithms.* Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons, Chichester.

Kai Huebener, Uwe Jost, and Henrik Heine. 1996. Speech Recognition for Spontaneously Spoken German Dialogs. In *ICSLP96*, Philadelphia.

Martin Oerder and Hermann Ney. 1993. Word Graphs: An Efficient Interface Between Continuous-Speech Recognition and Language Understanding. In *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech & Signal Processing, ICASSP*, pages II/119–II/122, Minneapolis, MN.

Hans Weber. 1992. Chartparsing in ASL-Nord: Berichte zu den Arbeitspaketen P1 bis P9. Technical Report ASL-TR-28-92/UER, Universität Erlangen-Nürnberg, Erlangen, December.

Hans Weber. 1995. *LR-inkrementelles, Probabilistisches Chartparsing von Worthypothesengraphen mit Unifikationsgrammatiken: Eine Enge Kopplung von Suche und Analyse.* Ph.D. thesis, Universität Hamburg.

Volker Weber. forthcoming. *Funktionales Konnektionistisches Unifikationsbasiertes Parsing.* Ph.D. thesis, Univ. Hamburg.