1 Sub-problems

Following Cao et al.’s algorithm, we also consider six sub-problems when we construct a maximum dependency graph on a given interval $[i, k] \in V$. Because $C$ sub-problem is too complex and rare in linguistic analysis, we ignore it in this algorithm. What’s more, we use a flag to indicate whether some edge exists or not and we still allow crossing sub-problem to degenerate to Int sub-problem. The sub-problems are explained as follows:

**Int**$[i, j]$ It represents an interval from $i$ to $j$ inclusively. And there is no edge $e_{i', j'}$ such that $i' \in [i, j]$ and $j' \notin [i, j]$. We further distinguish two types for Int. $Int_O[i, j]$ may or may not contain edge $e_{i, j}$, while $Int_C[i, j]$ contains $e_{i, j}$.

**LR**$[i, j, x]$ It represents an interval from $i$ to $j$ inclusively and an external vertex $x$. $\forall p \in (i, j), pt[x, p] = i$ or $j$. $LR[i, j, x]$ disallow $e_{i, j}$, $e_{x,i}$ or $e_{x,j}$. And $e_{i, j}$ will be captured in the procedure of generating $LR[i, j, x]$.

**N**$[i, j, x]$ It represents an interval from $i$ to $j$ inclusively and an external vertex $x$. $\forall p \in (i, j), pt[x, p] \notin [i, j]$. $N$ could contain $e_{i, j}$ but disallows $e_{x,i}$. If there exists $e_{i, j}$, this sub-problem should degenerate to Int sub-problem. We further distinguish two types for $N$. $N_O[i, j, x]$ may or may not contain $e_{x,i}$. While $N_C[i, j, x]$ disallows $e_{x,i}$ because it is captured in the procedure of generating $N_C[i, j, x]$.

**L**$[i, j, x]$ It represents an interval from $i$ to $j$ inclusively as well as an external vertex $x$. $\forall p \in (i, j), pt[x, p] = i$. $L$ could contain $e_{i, j}$ but disallows $e_{x,i}$. We further distinguish two types for $L$. $L_O[i, j, x]$ may or may not contain $e_{x,i}$. While $L_C[i, j, x]$ disallows $e_{x,i}$ because it is captured in the procedure of generating $L_C[i, j, x]$.

**R**$[i, j, x]$ It represents an interval from $i$ to $j$ inclusively as well as an external vertex $x$. $\forall p \in (i, j), pt[x, p] = j$. $R$ disallows $e_{x,j}$ and $e_{x,i}$. We further distinguish two types for $R$. $R_O[i, j, x]$ may or may not contain $e_{i, j}$. While $R_C[i, j, x]$ disallows $e_{i, j}$ because it is captured in the procedure of generating $R_C[i, j, x]$.

In this algorithm, we add all crossing edges during decomposition and add noncrossing edges in $Int_C$ for consideration of high-order.

2 Decomposing an Int Sub-problem

Consider $Int_O[i, j]$ and $Int_C[i, j]$ sub-problem. Because $Int_C[i, j]$ is very similar to $Int_O[i, j]$ and needs to expand in second-order, we just show the decomposition of $Int_C[i, j]$. Assume that $k \in [i, j] \cup \emptyset$ is the farthest vertex from $i$ that is linked with $i$, and $x = pt[i, k]$ ($x$ may be $\emptyset$). There are some cases as following:

**Case 1: No Arc From i** Vertex $k = \emptyset$ and $x = \emptyset$.
We can remove $i$ and consider interval $[i + 1, j]$. Because there exist no edge from $i$ to some node in $[i + 1, j]$, interval $[i + 1, j]$ is still an $Int_O$. The problem is decomposed to $Int_O[i + 1, j] + e_{i, j}$.

**Case 2: $e_{i, k}$ is noncrossing** Vertex $k \in (i, j)$ and $x = \emptyset$. Obviously, $[i, k]$ and $[k, j]$ are still Int since $e_{i, k}$ is noncrossing. The problem is decomposed to $Int_C[i, k] + Int_O[k, j] + e_{i, j}$.

**Case 3: $x \in (k, j]$** In this case, $e_{i, k}$ must be a crossing edge. Vertex $k$ and $x$ divide the
interval \([i,j]\) into three subparts: \([i,k]\), \([k,x]\), \([x,j]\). Because \(x\) may be \(j\), interval \([x,j]\) may only contain \(j\) and become an empty interval. We define \(x'\) as pencil point of all edges from \([i,k]\) to \(x\), and divide this case into two subproblems according to \(x'\) as Cao et al.’s algorithm.

First we assume there exist edges from \(k\) to \((x,j)\), so \(x'\) can only be \(k\) and pencil point of edges from \(k\) to \((x,j)\) is \(x\). Thus interval \([i,k]\) is an \(R\) with external vertex \(x\). What’s more, \([i,k]\) is an \(R_O\) because we have captured \(e_{(i,k)}\). Any edge from within \([k,x]\) to an external vertex violates 1-endpoint-crossing restriction, thus interval \([k,x]\) is an \(Int_O\). Since \(x\) is pencil point of edge from \(k\) to \((x,j)\), interval \([x,j]\) is an \(L_O\) with external vertex \(k\). In summary, we can decompose it into \(R_O[i,k,x] + Int_O[k,x] + L_O[x,j,k] + e_{(i,k)} + e_{(i,j)}\).

Second we assume there is no edge from \(k\) to \([x,j]\), so \(x'\) can be \(i\) or \(k\) and \([k,x]\), \([x,j]\) are \(Int_O\). And the result is \(LR[i,k,x] + Int_O[k,x] + Int_O[x,j] + e_{(i,k)} + e_{(i,j)}\).

**Case 4: \(x \in (i,k)\)** In this case, \(e_{(i,k)}\) must also be a crossing edge. Vertex \(k\) and \(x\) divide the interval \([i,j]\) into three subparts: \([i,x]\), \([x,k]\), \([k,j]\).

First we assume there exist edges from \(i\) to \((x,k)\), so pencil point of edges from \(x\) to \((k,j)\) is \(i\). Thus interval \([k,j]\) is an \(N_O\) with external vertex \(x\) because neither \(k\) nor \(j\) is pencil point. And interval \([i,x]\) should be \(Int_O\). Since \(x\) is pencil point of edges from \(i\) to \((x,k)\), interval \([x,k]\) is an \(L\) with external vertex \(i\). What’s more, \([x,k]\) is an \(L_C\) because we have captured \(e_{(i,k)}\). And the decomposition is \(Int_O[i,x] + L_C[x,k,i] + N_O[k,j,x] + e_{(i,k)} + e_{(i,j)}\).

Second we assume there is no edge from \(i\) to \([x,k]\), but edge from \(k\) to \([i,x]\). So pencil point of edges from \(x\) to \((k,j)\) is \(k\). Thus interval \([k,j]\) is an \(L_O\) with external vertex \(x\). And interval \([x,k]\) should be \(Int_O\). Since \(x\) is pencil point of edges from \([k,x]\), interval \([i,x]\) is an \(R_O\) with external vertex \(k\). And the decomposition is \(R_O[i,x,k] + Int_O[x,k] + L_O[k,j,x] + e_{(i,k)} + e_{(i,j)}\).

For \(Int_O[i,j]\), because there may be \(e_{(i,j)}\), we should add one more decomposition \(Int_O[i,j] = Int_C[i,j]\), and we don’t need to add \(e_{(i,j)}\) in all cases.

### 3 Decomposing an N Sub-problem

Consider \(N_O[i,j,x]\) and \(N_C[i,j,x]\) subproblem. And we show the decomposition of \(N_O[i,j,x]\).

#### Case 1: If there is no more edge from \(x\) to \((i,j)\), then it will degenerate to \(Int_O[i,j]\).

#### Case 2: If there exists \(e_{(x,j)}\) then it will reduced to \(N_C[i,j,x] + e_{(x,j)}\).
Case 3: If there is edge from $x$ to $(i, j)$, we define $e_{(x,k)} (k \in (i, j))$ as the farthest edge from $i$ and it divides $[i, j]$ into $[i, k]$ and $[k, j]$. Because neither $i$ nor $j$ is pencil point of $e_{(x,k)}$, $[i, k]$ and $[k, j]$ will be $N_C[i, k, x]$ and $Int_O[k, j]$ respectively. The decomposition is $N_C[i, k, x] + Int_O[k, j] + e_{(x,k)}$.

For $N_C[i, j, x]$, we just ignore Case 2 and follow the others.

![Figure 2: Decomposition for $N[i, j, x]$.](image)

### 4 Decomposing an L Sub-problem

Consider $L_O[i, j, x]$ and $L_C[i, j, x]$ subproblem. And we show the decomposition of $L_O[i, j, x]$.

**Case 1:** If there is no more edge from $x$ to $(i, j)$, then it will degenerate to $Int_O[i, j]$.

**Case 2:** If there exists $e_{(x,j)}$, then it will degenerate to $L_C[i, j, x] + e_{(x,j)}$.

**Case 3:** If there is edge from $x$ to $(i, j)$, we define $e_{(x,k)} (k \in [i, j])$ as the farthest edge from $i$ and it divides $[i, j]$ into $[i, k]$ and $[k, j]$. First, if there is edge from $x$ to $(i, j)$, $[i, k]$ and $[k, j]$ will be $N_O[i, k, j]$ and $R_O[k, j, x]$ respectively. However, $e_{(k,j)}$ will be calculated twice following this decomposition. So we define $N_O[i, k, j]$ as a special $N_C[i, k, j]$ to disallow it generating $e_{(k,j)}$. The decomposition is $N_C[i, k, j] + R_O[k, j, x] + e_{(x,k)}$.

Second, if there is no edge from $x$ to $(k, j)$, $[i, k]$ and $[k, j]$ will be $R_O[i, k, j]$ and $Int_O[k, j]$ respectively. The decomposition is $R_O[i, k, j] + Int_O[k, j] + e_{(x,k)}$.

For $R_C[i, j, x]$, we can still ignore Case 2. Specialy, we disallow $R_C$ to be $Int_C$. $R_C$ can only be produced by $R_O$’s Case 2 and Int’s Case 3. For $R_O$’s Case 2, $R_O$ can be $Int_O$ firstly and then be $Int_C$. For Int’s Case 3, we can use Int’s Case 2 directly to get $Int_C$ instead. So we don’t need to degenerate $R_C$.

### 6 Decomposing an LR Sub-problem

Because we don’t consider $C$ subproblem in Cao et al., there must be a vertex $k$ within $[i, j]$ which divides $[i, j]$ into $[i, k]$ and $[k, j]$. And $i$ is the pencil point of edges from $x$ to $(i, k)$ and $j$ is the pencil point of edges from $x$ to $(k, j)$. Obviously, $[i, k]$ is an $L_O$ and $[k, j]$ is an $R_O$ with external $x$. Thus the problem is decomposed as $L_O[i, k, x] + R_O[k, j, x]$.

Of course, either $i$ or $j$ may not be a pencil point. If the common pencil point of all edges from $x$ to $(i, j)$ is $i$, then the model is the same as $L_O[i, j, x]$. Similarly, if the common pencil point is $j$, then the model is the same as $R_C[i, j, x]$. And if neither $i$ nor $j$ is pencil point, it will be an Int problem.

However, we don’t need to consider this two special cases. If the common pencil point is only $i$, $i$ is the pencil point of edges from $x$ to $(i, k)$ but there must be no edge from $x$ to $(k, j)$ and $[k, j]$ is an Int. Thus we can still use above decomposition to express this case, just degenerate $R_O[k, j, x]$ to $Int_O[k, j]$. If the common pencil point is $j$, this case is equal to Int’s Case3.1. If neither $i$ or $j$ is pencil point, this case is equal to Int’s Case2.
7 Complexity and summary

We discuss each subproblem by enumerating different cases to get only one edge at once. In subproblem can decompose by discussing whether has a crossing arc and position of its pencil point. For subproblem, we simplify the decomposition and ignore subproblem. For other crossing problem, we consider whether it can degenerate and the number of arcs from to . Obviously, this algorithm has the same time and space complexity with Çağ et al.’s degenerated algorithm.

References