シソーラスブラウザxthesにおけるDAG構造の描画アルゴリズムとその評価

市丸, 夏樹
九州大学大学院システム情報科学研究院知能システム学部門

https://doi.org/10.15017/1516213

出版情報：九州大学大学院システム情報科学紀要. 10 (2), pp.97-102, 2005-09-26. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第10巻　第2号　平成17年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.10, No.2, September 2005

シソーラスブラウザ xthes における DAG 構造の描画アルゴリズムとその評価

市丸夏樹*

Evaluation of an Algorithm to Draw DAG Structure Used in the Thesaurus Browser 'xthes'

Natsuki ICHIMARU

(Received June 10, 2005)

Abstract : In this paper, we present a method to plot a directed acyclic graph (DAG) which represents a hierarchical structure with multiple inheritances. This method is used in the thesaurus browser 'xthes' to display 'is-a' relationship between nodes in a thesaurus or ontology. Because it is noted that the problem of minimizing arc crossings in a DAG is NP-hard, we need a fast heuristic algorithm. 'xthes' draws a DAG as follows. At first, it splits a DAG into a tree and other non-tree arcs. Then, it rearranges the order of children nodes by clustering subtrees, and reverses the order of subtrees in the clusters, to make the length of inter-subtree links as short as possible. Finally, it minimizes the distance between subtrees in bottom-up manner, and decides the position of the nodes and arcs. By the experiment, it was confirmed that our node-exchanging method saves about 50% of the total length of the crossing arcs. It seems that our method can display a DAG in a comprehensible form, and that it is fast enough to update the diagram interactively.

Keywords : Directed acyclic graph, Tree, Thesaurus, Minimizing arc crossings

1. はじめに

シソーラスは自然言語の処理において単語間や概念間の類似度計算，単語の意味的な曖昧性の絞り込み，曖昧検索等に使用される．生物学分野でも古くから遺伝子情報などに基づくシソーラスやオントロジが用いられている．

一般にシソーラスは周回のない有向グラフ（DAG）であり，その構造は，多重継承枝を除けば，ほぼ木構造状である．そこで本研究では，シソーラスを木と多重継承枝に分離し，ノードを自動的に並べ替えることによって DAG を判別しやすい形で高速に描画することを目指す．

DAG 構造の美的描画問題に関して，ノードを最適な順序に並べ替える問題は NP-hard であることが知られており，$O(n^2)$ で解けるヒューリスティックによる近似解法がいくつか提案されている[1]．一方，木構造の描画については，$O(n)$ で動作するアルゴリズム[4]が知られている．本稿では，DAG の図のインタラクティブなアップデートを可能にするため，必ずしも最適な図となる保証はないが，DAG を高速に描画する手法を提案する．

平成17年6月10日受付
* 知能システム学部門

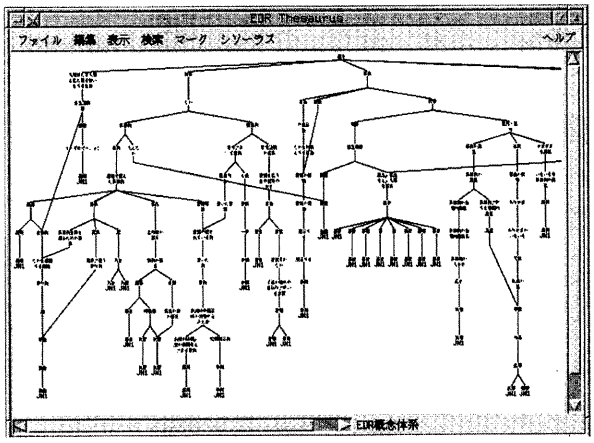


Fig. 1 A screenshot of the thesaurus browser 'xthes'.

2. DAG 構造の描画

著者らが開発したシソーラスブラウザ xthes[5] は交差する枝を短く，しかも高速に表示でき，自然言語処理研究のためのツールとして国内のいくつかの研究機関において使用されている（**Fig. 1**）．xthes は次のようにして DAG を描画する．

1．DAG 構造を木構造とそれ以外の枝に分離する（3. 節）．

2．交差する枝の長さができるだけ短くなるように部分

木の順序の並べ替えを行う（5.節）.

3．木構造のノード配置アルゴリズムによって，ノードの配置を決定する（4.節）.

4．木構造に多重継承枝をオーバーラップして描画する.

以下，この手法の詳細について述べる.

3. 多重継承枝の分離

ノードの集合をN，枝の集合をAとするとき，DAG G＝〈N, A〉を木T＝〈N, A'〉とA＝A'∪Xを満たす枝の集合Xに分離する.DAG中の複数の上位ノードを持つノードについて，その上位ノードのうち最も深い位置にあるもの1つを選択し，木構造上の親ノードとする．その木構造上の親ノード以外の上位ノードからの枝を多重継承枝と呼び，Xに格納する．シソーラスに複数の頂点ノードが存在する場合には，それらを子ノード列とする仮想的な唯一の頂点ノードを追加することによって木構造を作成する.

4. 木構造のノード配置アルゴリズム

木構造の描画は次のような描画規則に従って行うものとする.

- ルートノードを最上層とし，全ての枝の向きを下向きに揃え，ルートノードからの段数により各ノードを層状に配置する.
- 親ノードは子ノード列の中心に配置する.
- 各ノードは大きさ可変の長方形とする.
- 枝は直線とする.

ここでは，部分木接近戦略による木構造のノード配置アルゴリズムを示す[3]．このアルゴリズムは文献7）に示されている2分木に対するアルゴリズムを一般のn分木に拡張したものである.

4.1 ノードのy座標の決定

木構造中のノードをルートノードからの段数によって層状に分け，ルートノードを最上層とし，各層のy座標を一定に揃える．各層の間隔は与えられた最小間隔値にノードの高さの層別の最大値を加えたものとする.

4.2 部分木間の間隔の決定

部分木の概形を用いて複数の部分木をできるだけ緊密に配置する．葉ノードから始めてボトムアップに部分木を構成し，隣り合う部分木の間隔を，子孫同士が重ならない限り最小幅で配置する．ある親ノードに対する下位部分木列の概形が全て完成すれば，親ノードを子ノードの中心に据える.

4.2.1 左右端リスト

まず，部分木の概形を表すためのデータ構造を準備する.以下，順序を持ったm個の要素からなるリストLについて，各々の要素を順に$L=\{L_1, L_2, \ldots L_m\}$と書き，その要素数を$|L|=m$と書く．また，本稿では∪を順序を保ったリストの連結とする.

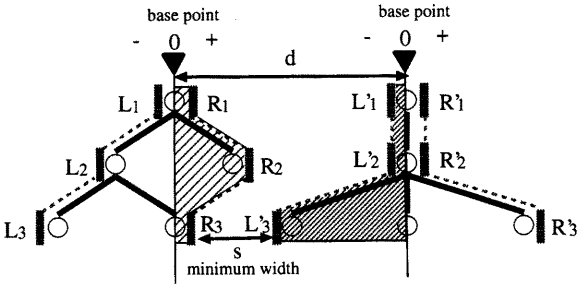


Fig. 2 Minimizing distance between subtrees using L-R lists.

ノードの左右端の相対x座標の最小値と最大値を層別に求め，このリストに格納する．部分木の頂点のノードの中心に基準点をとり，部分木内のノードの位置をその基準点からの相対座標で表す．木構造のルートを上とした場合，右方向を正とする．層別に求めた基準点からの相対座標の最小値のリストを左端リストL，最大値のリストを右端リストRと呼ぶ（**Fig. 2**）.

4.2.2 部分木列の概形

次に，2つの部分木間の間隔を，部分木の両端が重ならない限り最小にすることを考える．そのためには，左側の部分木の右端リストの突出幅と，右側の部分木の左端リストの突出幅を層別に加え，その和の最大値を求める．その値に与えられた最小間隔sを加えた値を部分木間の間隔dとする（**Fig. 2**）.

隣接しない遠方の部分木とノードが重ならないようにするためには，横方向に連なる部分木列を一単位とすると取り扱いが容易になる．そこで，部分木列を成す各部分木の頂点ノードの並びをα，α中の左端ノードと右端ノードの中心間の幅をw，左右端リストをL，Rとするとき，部分木列の概形を〈α，w，L，R〉という4項組で表す.

4.2.3 間隔決定アルゴリズム

左右端リストおよび部分木列の概形を帰納的に定義し，ボトムアップに配置する.

(1) 葉ノードの場合

ノードを表す矩形の中心を基準点とした時の，幅nの葉ノードNの左右端リストL, Rは，次のように与えられる.これを初期状態の部分木〈$\{N\}$，0，L，R〉と表す.

$$L=\{-n/2\},\quad R=\{n/2\}. \tag{1}$$

(2) 部分木の結合

共通の親ノードを持つ，隣合う部分木列〈α'，w'，L'，R'〉，〈α''，w''，L''，R''〉を左右方向に結合し，新しい部分木列〈$\alpha' \cup \alpha''$，w, L, R〉を構成する（**Fig. 3**）.

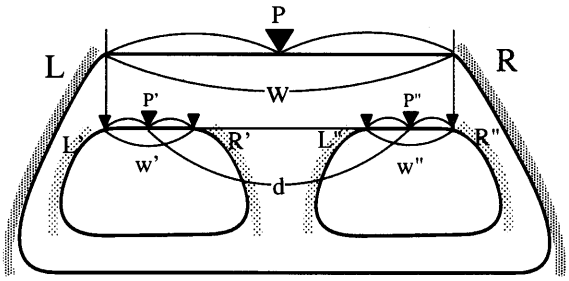


Fig. 3 Concatenating subtrees.

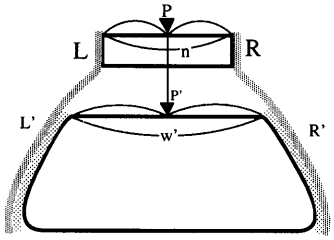


Fig. 4 Settling a parent node on the center of children.

ノード間の最小間隔を表す定数を s とすると，各部分木の基準点間 P'，P'' 間の距離 d，α' の左端と α'' の右端のノードの中心間の間隔 w，新しい部分木列の左端リスト $L=\{L_i \mid 1\leq i\leq \max\{|L'|,|L''|\}\}$，右端リスト $R=\{R_i \mid 1\leq i\leq \max\{|R'|,|R''|\}\}$ は次のように求められる．

$$d=s+\max\{R'_i-L''_i \mid 1\leq i\leq \min\{|R'|,|L''|\}\}. \quad (2)$$

$$w=d+(w'+w'')/2. \quad (3)$$

$$L_i=\begin{cases} L'_i+(w'-w)/2 & if \quad 1\leq i\leq |L'|, \\ L''_i+(w-w'')/2 & if \quad |L'|<i\leq |L''|. \end{cases} \quad (4)$$

$$R_i=\begin{cases} R''_i+(w-w'')/2 & if \quad 1\leq i\leq |R''|, \\ R'_i+(w'-w)/2 & if \quad |R''|<i\leq |R'|. \end{cases} \quad (5)$$

(3) 親ノードの配置

部分木列 $\langle \alpha', w', L', R' \rangle$ の親ノード N は，子ノード列の中心に配置され，N を頂点とする木 $\langle \{N\},0,L,R \rangle$ を成す（**Fig. 4**）．このときノード N の幅を n とすると，左右端リスト L，R は次のように与えられる．

$$L=\{-n/2\}\cup L', \quad R=\{n/2\}\cup R'. \quad (6)$$

4.3 ノードの x 座標の決定

ルートノード N_0 まで辿り着いたら，今度はトップダウンに各部分木の頂点ノードの絶対座標を計算する．

(1) ルートノードの x 座標

木全体 $\langle \{N_0\},0,L,R \rangle$ のルートノード N_0 の絶対 x 座標 P は，次のように求められる．

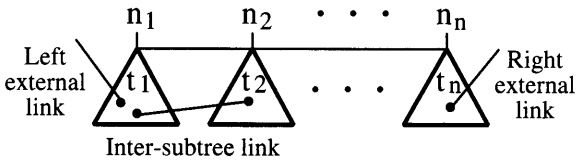


Fig. 5 Inter-subtree links and two kinds of external links.

$$P=-\min L. \quad (7)$$

(2) 部分木の x 座標

基準点の座標が P である部分木列 $\langle \alpha, w, L, R \rangle$ が，$\alpha=\alpha'\cup\alpha''$ を満たす 2 つの部分木列 $\langle \alpha', w', L', R' \rangle$，$\langle \alpha'', w'', L'', R'' \rangle$ に分けられるとき，各部分木列の基準点の x 座標 P'，P'' は次のように求められる（**Fig. 3**）．

$$P'=P+(w'-w)/2, \quad P''=P+(w-w'')/2. \quad (8)$$

部分木列の要素数が 1 のとき，基準点の x 座標と，部分木の頂点ノードの x 座標が一致する．子部分木列の基準点は親ノードの x 座標を継承する．以上により，木構造内の全てのノードの座標が確定する．

5. ノードの並べ替え

前節のアルゴリズムで描画された木構造の上に多重継承枝をオーバーラップさせて描画すると，枝が錯綜して図が読み取り難くなってしまう場合がある．そこで，ノードを並べ替えて部分木を結合させる順序を制御し，必要に応じて部分木を左右反転させることによって，部分木間の多重継承枝ができるだけ短くなるように配置することを考える．

ここでは簡単のため，各部分木はほぼ等しいサイズであるものと仮定し，ノードの並べ替えは木構造を描画する前にトップダウンに行うものとする．

5.1 部分木間の相互リンクと外部へのリンク

ある木構造中のノード n'_0 の子ノード列 $n_1, \ldots, n_n$ を頂点とする部分木列 $t_1, \ldots, t_n$ について考える．頂点ノード n_i，n_j の木構造上の子孫ノード n'_i，n'_j 間が多重継承枝で結ばれている時，この多重継承枝を t_i, t_j 間の部分木間リンクと呼ぶ．また，部分木 t_i 中のノードと部分木 $t_i, \ldots, t_n$ 外のノードを結ぶリンクを外部リンクと呼ぶ．親ノード n'_0 の全祖先ノードについて，子ノード列の並べ替えが完了しているものとすると，ルートノード n_0 から n'_0 までを結ぶパス上から見て左側のノードとの外部リンクを左外部リンク，右側のノードとの外部リンクを右外部リンクと呼ぶ（**Fig. 5**）．

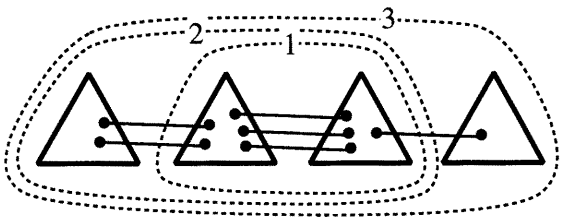


Fig. 6 An example clustering result.

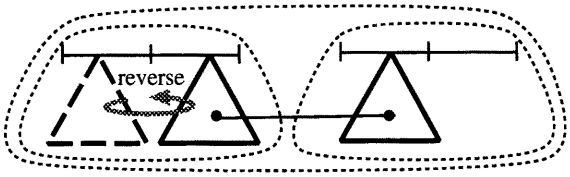


Fig. 7 Adjoining reversed clusters.

5.2 部分木のクラスタリング

部分木 t_1, ..., t_n を部分木間のリンク数が多い順に結合させ，クラスタリングを行う（**Fig. 6**）．2つのクラスタに含まれる全ての部分木間リンクのうち始点と終点が共に同一クラスタ中に含まれるものを除いたものをクラスタ間リンクと呼ぶ．クラスタリングは次のように行われる．

[部分木のクラスタリング手続き]

1．1部分木 t_i を1クラスタ $C_i=\{t_i\}$ とするクラスタ列 $C=\{C_1,...,C_n\}$ を用意する．

2．もし全てのクラスタ間リンク数が0ならば，クラスタリング手続きを終了する．

3．クラスタのペアの全組み合わせのうち，（クラスタ間リンク数−外部リンク数）が最大の組 C_i, C_j を求める．

4．C_i, C_j を連結した新しいクラスタ C'_i を作成して C に加え，元の C_i, C_j を C から取り除く．

5．2へ戻る．

ここで，2つのクラスタを結合する際には，必要があればいずれかのクラスタを左右反転し，クラスタ間リンクの長さが短くなるようにする．その際，リンクの実際の長さは未確定であるため，クラスタ内の部分木の頂点ノードの順位において，リンクの端点を含む部分木が前半であるか後半であるかによって左右の片寄りを判定する（**Fig. 7**）．

以上により，リンクの多い部分木同士が近くに配置され，クラスタ列 C 中のクラスタ間には相互結合がない状態になる．

5.3 クラスタの配置

各クラスタの内部から外部への枝が交差する部分木の数が少なくなるように，各クラスタ内部を左右反転する．部分木 t_i に含まれるノードへの左外部リンク数を $X_L(t_i)$，右外部リンク数を $X_R(t_i)$ とおくと，クラスタ内の部分木の並び $C'=\{t_1, ..., t_n\}$ と，それを左右反転した $C''=\{t_n, ..., t_1\}$ について，次の評価関数 $F(C')$, $F(C'')$ が小さくなる方をクラスタ内の最終的な部分木の並び $C=\arg\min_{x\in\{C',C''\}}F(x)$ とする．関数 $F(C)$ は，左右の外部リンクが交差するクラスタ C 中の部分木の数の推定値を表す．

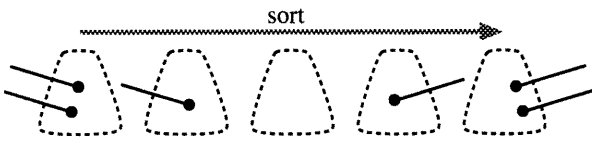


Fig. 8 Sorting clusters.

$$F(\{t_1,...,t_n\})=\sum_i\{(i-1)X_R(t_i)+(n-i)X_L(t_i)\} \tag{9}$$

外部へのリンクが多いクラスタを外側に配置するため，各クラスタ C_i への右外部リンク数と左外部リンク数との差の合計 $K(C_i)$ をキーとしてクラスタの並び C_1, ..., C_m をソートする（**Fig. 8**）．

$$K(C_i)=\sum_{t\in C_i}\{X_R(t)-X_L(t)\} \tag{10}$$

最後に，ソート済みのクラスタ列に含まれる部分木の頂点ノードを連結したものを，ノード n'_0 の最終的な子ノードの並びとする．

6. 計　算　量

ノード数を n，多重継承枝数を l とし，ノードの深さ，子ノード数，親ノード数の最大値をそれぞれ d, k, p とすると，木構造描画アルゴリズムに $O(ndk)$，部分木間リンクの抽出に $O(np+ld)$，外部リンクの抽出に $O(nk^2+ld)$，ノードの並べ替えに $O(nk^3)$ の計算量が必要となる．表示するシソーラスを1つに決め，そのうち n ノードからなる部分を表示する問題を考えると，d, k, p はシソーラスによって決まる定数であり，l は n に比例すると考えられる

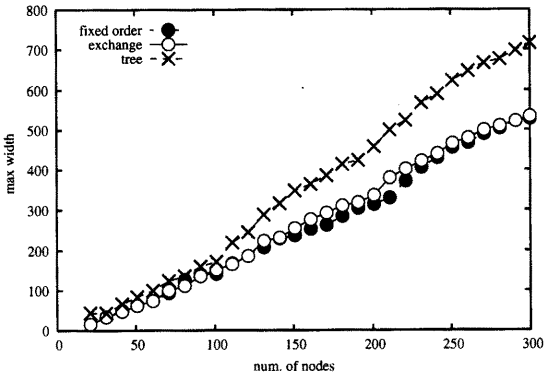
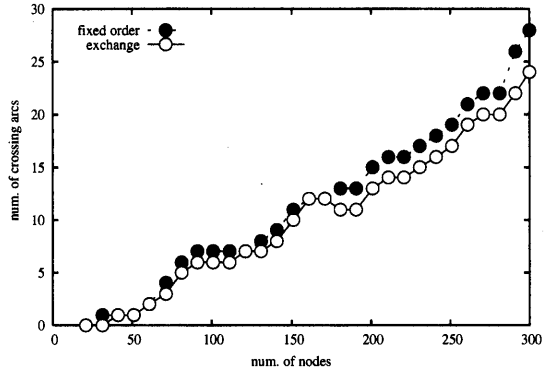
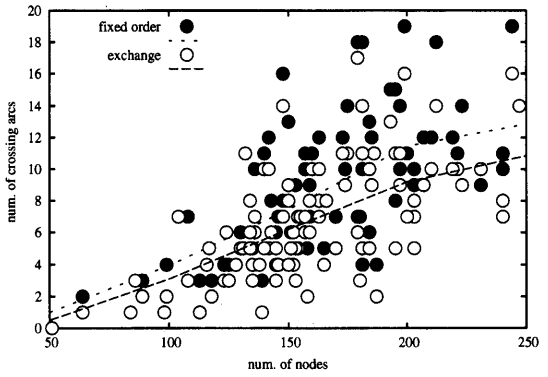


Fig. 9 Maximum width of a DAG which consists of top n nodes.

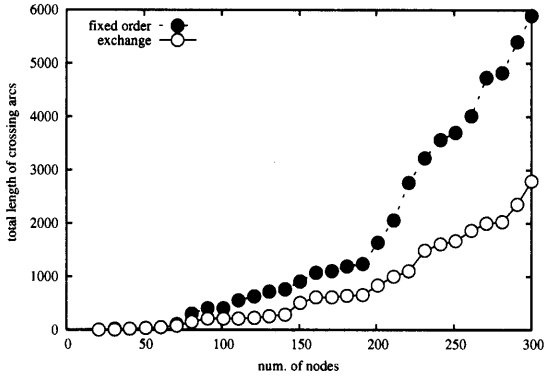


(A) Top n dodes.

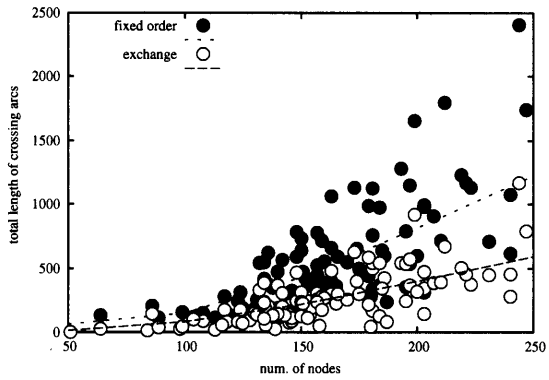


(B) Search results.

Fig. 10 Num. of crossing arcs.



(A) Top n dodes.



(B) Search results.

Fig. 11 Total length of crossing arcs.

ため，提案手法の時間計算量は，木構造描画アルゴリズムと同じく，高々 $O(n)$ となる．

7. 実験による評価

実際のシソーラスを用いて，描画幅，多重継承枝の交差数と長さの合計，描画時間等を測定する実験を行った．

7.1 実験条件

試験用のシソーラスデータとしては，EDR 概念体系に含まれる上位下位関係と EDR 日本語単語辞書[2] を使用した．実験では，次の 2 種類の方法で選択したノードを，全ての先祖ノードと共に DAG 構造として描画した．

（A）シソーラスの上位階層の全体像を把握するため，頂点から子孫数の降順に n ノードを抽出したもの．

（B）同音異議語数の降順に名詞 100 語を選び，その仮名表記を検索語として与えた場合の検索結果．検索された語とその全祖先ノードを 1 つの DAG 構造の形で表示する．

いずれの場合も多重継承枝数はノード数の10%程度であった．

7.2 図全体の幅の評価

まず，データ（A）について，シソーラスを提案手法によって DAG として表示した場合と，多重継承ノードを何重にも複製して木構造に展開して表示した場合の図全体の幅を **Fig. 9** に示す．その結果，DAG 表示の面積は木として表示した場合の 40～80%程度となった．

7.3 枝の交差の評価

枝の交差に対するノードの並べ替えの効果を調べるため，5. 節の方法でノードを並べ替えた場合と，ノードの並びを子孫数の降順に固定した場合について，データ（A），（B）の DAG 表示から次の値を求めた．

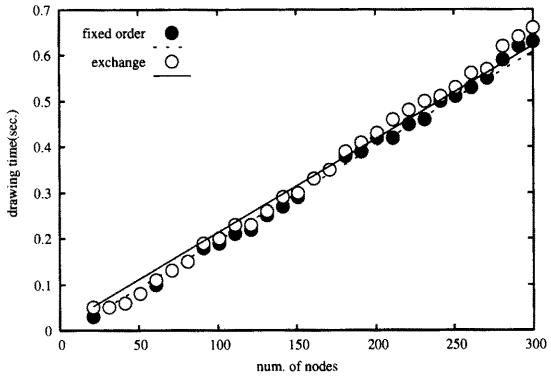
(1) 多重継承枝の交差数（**Fig. 10**）．

(2) 交差する多重継承枝の図面上の長さの合計（**Fig. 11**）．

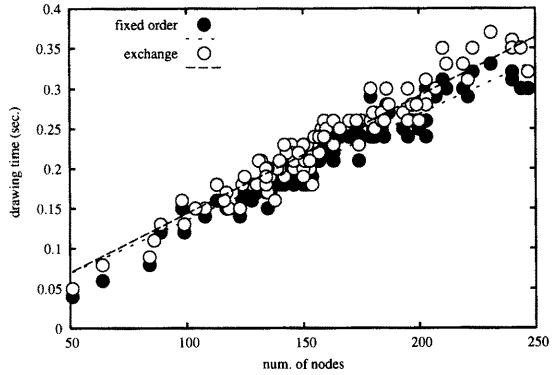
枝の長さの計測には 1 文字の幅を 1 とする座標系を使用した．

この結果によると，ノードを並べ替えることにより交差の数は 0～15%程度減少し，交差する枝の長さの合計は平均的には約 50%減少した．提案手法を用いると，交差数自体はあまり変わらないが，交差する枝の長さを大幅に短くできるため，ノード間の関係の視認性向上に有効であると考えられる．

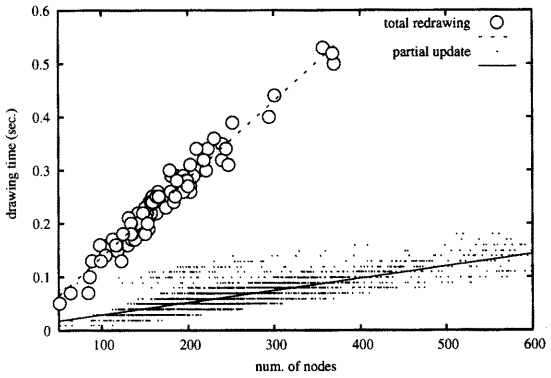
なお，**Fig. 11**(A) の並べ替えを行わない場合のグラフの勾配が，$n=200$ 以降で急に大きくなっているのは，この



(A) Top n dodes.



(B) Search results.



(B') Partial update.

Fig. 12 Drawing time and partial update time (sec.).

あたりから，例えば8本の多重継承枝を持つ「人」という概念など，多くの多重継承枝を持つ概念が出現し，その枝の間の部分木の幅が拡大するに伴って枝の長さの合計が急速に増大するためであると思われる．また，(B)にばらつきが生じているのは，検索語によって選ばれるノードにより多重継承枝数が大きく異なるためである．

7.4 描画および部分的な更新にかかる時間の評価

前節と同じDAGの描画時間を**Fig. 12**に示す．この値は，2.5GHzのIntel Pentium 4 CPUを搭載し，Ultra 160 SCSI HDD 2台でRAID-0を構成したDell Precision 340を使用した場合のCPU時間である．xthesはperl 5とTkライブラリを用いて記述されており，X11を有するUNIX互換環境上，またはWindows用のActive Perl上で動作するが，ここではLinux(Fedora core 2)上で行った実験結果を示す．

このうち(A)，(B)は白紙の状態からDAG構造を描画し終わるまでの時間を表す．描画時間はノード数nに比例し，100ノードで0.15〜0.2 sec.程度であった．ノードの並べ替えによる速度低下は僅かであった．

一方**Fig. 12**（B′）には，ノードの並べ替えが有効である時に，(B)の図の一部を修正する際，差分のみ座標を再計算して図をupdateする場合と，修正後と同等の図を白紙の状態から描画した場合の描画時間を示す．図の修正はDAG中のノードのうち1〜10個の子ノードを持つものを選び，その子ノード列を図に追加していくことによって行った．差分のみ修正した場合の描画時間は全体を再描画した場合の約23%であり，DAG全体のノード数が300ノード場合の再描画時間がほぼ0.1秒以内であった．よって，ノード数nが十分小さければ瞬間的な修正が可能であると言える．

8. おわりに

シソーラスブラウザxthesで用いられている高速なDAG描画アルゴリズムについて述べ，実験によってその有効性を示した．その結果，提案手法を用いることによって交差する多重継承枝の長さを50%程度まで短縮でき，シソーラスを効率的かつ明解に描画できることを示した．

参考文献

1) P. Eades and K. Sugiyama. How to draw a directed graph. *Journal on Information Processing*, Vol.13, No.4, pp.424 − 437, 1990.
2) EDR. EDR電子化辞書．日本電子化辞書研究所，第2版，1999.
3) 市丸夏樹．名詞のシソーラス構造の表示．夏のプログラミング・シンポジウム「可視化」報告集，pp.55−64．情報処理学会，1993.
4) John Q. Walker II. A node-positioning algorithm for general trees. *Software-Practice and Experience*, Vol.20, No.7, pp.685 − 705, 1990.
5) 右田和寛．検索語の絞り込み機能を持つシソーラスブラウザの作成．平成8年度九州大学情報工学科卒業論文，1997.
6) Edward M. Reingold and John S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, No.2, pp.223−228, 1981.
7) 田中実．木の図示．情報処理，Vol.44, No.10, pp.1082 − 1086, 2003.
8) Charles Wetherell and Alfred Shannon. Tidy drawings of trees. *IEEE Transactions on Software Engineering*, No.5, pp.514 −520, 1979.