2N-2

図表参照文を利用した文書レイアウト生成

冨田 恭平 † (東京大学)
原 忠義 ‡ (国立情報学研究所)
相澤 彰子 § (東京大学/国立情報学研究所)

1 はじめに

近年，電子端末上で文書が読まれる機会が増え，レイアウトを閲覧環境に合わせて変更するための手法が数多く提案されている [1]．ここで多くの既存手法では，見た目の綺麗さを重視しており，文書の読みやすさが考慮されていない．読みやすさも考慮した手法としては，Marriott ら [2] による多段組文書における図表配置位置を決定する手法が挙げられる．この手法では図表配置問題は最適化問題として扱われ，読みやすさ向上のために図表と参照文との距離が目的関数に組み込まれている．しかし，図表 1 つに対する複数の参照文の存在を仮定していない，参照文と図表の結びつきの強さを区別せず扱っているなどの問題点もある．

本稿では，文書中に存在する参照文を全て抽出する手法，および抽出された参照文に加えてその参照の強さも考慮して図表の配置位置を決定する方法を提案する．図表参照文を利用して図表の配置位置を決定するための問題を定式化した上で，その問題を解く手法を提案することが本稿の主な貢献である．

2 問題設定

2.1 用語定義と図表配置位置の指定方法

本稿における文書は，段落要素と図表要素の集合である．段落要素は 1 つ以上の文の集合からなる段落で章や節見出しを含む．図表要素は画像または表であり，キャプションを伴うこともある．段落要素は順序を持っているが，図表要素は文からの参照により対応付けられているため任意の順序で表示することができる．本稿では文書を HTML5 文書として扱うことを想定し，段落要素は HTML5 の p 要素と h1-h5 要素，図表要素は HTML5 の figure 要素を指すものとする．

次に図表要素の配置位置の指定方法を定義する．図表要素の配置位置は (垂直位置, 水平位置, 倍率) という 3 つ組により定義される．垂直位置は，図表要素が挿入される HTML ソースコード上での直前の段落を指定する．水平位置は，左寄せ，中央寄せ，右寄せのうちから一つ選ばれる．左・右寄せの場合のみ CSS の float 属性を指定して段落要素を左右に回り込ませる．倍率は実数が指定され，倍率によって図表が拡大・縮小した上で指定場所に図表要素が挿入される．

本稿では図表要素の複製を許すことにより，最小限のスクロール・視線移動で図表要素を目視できるレイアウトを目指す．また，通常 HTML 文書はブラウザで開いてすぐに全体のレイアウトが固定されるが，本稿ではスクロールにより画面に表示された部分から順にレイアウトを固定していく形で文書を表示することで図表配置の所要時間を減らす．

2.2 最適化問題の定式化

本節では図表の配置位置を決定する問題を最適化問題として定式化する．文書中の文を $\mathrm{s}_0, \cdots, \mathrm{s}_N$ とし，このうち既表示のものを部分集合 S とする．また図表要素を $\mathrm{f}_0, \cdots, \mathrm{f}_M$ で表す．図表要素の部分集合 F に対して第 2.1 節の 3 つ組を割り当てた図表配置を P とし，目的関数を次のように定義する．

$$g(P) = \sum_{\substack{\mathrm{s}_i \in S \\ \mathrm{f}_j \in F}} \mathrm{Dis}(\mathrm{s}_i, \mathrm{f}_j) + \sum_{\mathrm{f}_j \in F} \mathrm{Cls}(\mathrm{f}_j) + \sum_{\mathrm{f}_j \in F} \mathrm{Iso}(\mathrm{f}_j) + \alpha \cdot \sum_{\substack{\mathrm{s}_i \in S \\ \mathrm{f}_j \in F}} \mathrm{Rsc}(\mathrm{s}_i, \mathrm{f}_j) + \beta \cdot \sum_{\mathrm{f}_j \in F} \mathrm{Rsz}(\mathrm{f}_j)$$

第 1 項は図表要素とその参照文の距離に課されるコストで，$\mathrm{Dis}(\mathrm{s}_i, \mathrm{f}_j) = \deg(\mathrm{s}_i, \mathrm{f}_j) \cdot d(\mathrm{s}_i, \mathrm{f}_j)$ で定義される．ただし，$\deg(\mathrm{s}_i, \mathrm{f}_j) \in [0, 1]$ は文が図表要素をどの程度参照しているかを意味し，$d(\mathrm{e}_1, \mathrm{e}_2)$ は要素 e_1 と e_2 の画面上での距離 [px] である．

第 2 項は同じ図表要素が同一画面内に収まる場合に課されるコストで，既に画面内に図表要素 f_j が

Document Layout Formatting Using Sentences Referring to Figures

†Kyohei Tomita (The University of Tokyo)
‡Tadayoshi Hara (National Institute of Informatics)
§Akiko Aizawa (The University of Tokyo / National Institute of Informatics)

固定されているにもかかわらず P で新たに f_j を配置する場合に $\mathrm{Cls}(\mathrm{f}_j)$ は ∞，それ以外では 0 となる．

第 3 項は画面に図表要素を参照する文が存在していないのに図表要素が独立して配置されている場合に課されるコストで，図表要素 f_j が独立して存在している場合に $\mathrm{Iso}(\mathrm{f}_j)$ は ∞，それ以外では 0 となる．

第 4 項は画面に表示されている文が参照する図表要素を見るためにスクロールが必要となってしまう場合に課されるコストである．そのような場合に，$\mathrm{Rsc}(\mathrm{s}_i, \mathrm{f}_j) = 1$ となり，それ以外では 0 となる．

第 5 項は図表要素のサイズ変更に対するコストで，$\mathrm{Rsz}(\mathrm{f}_j) = (1 - m) \cdot \mathrm{chr}(\mathrm{f}_j)$ で定義される．ここで $m \in (0, 1]$ は図表要素の倍率，$\mathrm{chr}(\mathrm{f}_j)$ は図表 f_j 上の文字数を意味する．これは，文字数が多い図表要素は縮小すると読みづらくなるという考えに基づいている．

α, β はスクロールコストと図表要素の倍率変更に対する重みであり，本稿では α は閲覧環境のウインドウの高さ [px] に応じて，β はユーザの好みに応じて変更してもらうことを想定する．

スクロールされた際の新しく固定される図表配置 P_{new} は，目的関数を最小化することで $P_{\mathrm{new}} = \underset{P}{\mathrm{argmin}}\, g(P)$ として得られる．

3 提案手法

3.1 参照関係の抽出

文書中に含まれる文と図表要素の組ごとに，文 s_i が図表 f_j をどの程度参照しているか $(\deg(\mathrm{s}_i, \mathrm{f}_j) \in [0, 1])$ を出力する．

まず，文と図表要素のそれぞれを，各対象要素内での各単語の出現数が格納されたベクトルに変換する．図表要素に関しては，キャプションに加え画像上に出現する単語についても OCR を使用して検出する．また，文と図表要素のベクトルのコサイン類似度 $\mathrm{sim}(\mathrm{s}_i, \mathrm{f}_j)$ を計算しておく．

「図 1」のような参照語による直接的な図表要素の参照を正規表現で検出したあと，直接的な参照の近くにあるコサイン類似度が閾値以上の文も参照文として検出する．参照関係のある組に対しては $\deg(\mathrm{s}_i, \mathrm{f}_j) = \mathrm{sim}(\mathrm{s}_i, \mathrm{f}_j)$ とし，そうでない組に対しては $\deg(\mathrm{s}_i, \mathrm{f}_j) = 0$ とする．

3.2 最適化の実装

第 2.2 節で定義した最適化問題を実際に全探索で解くと，図表配置 P の組み合わせが図表要素数に対して指数的に増加することから最適化の実行時間が問題となる．そこで，(1) 目的関数が無限大となる図表配置 P は生成しない，(2) 図表要素の表示順序は，それを参照する文の範囲が重複していない限り参照文の出現順に固定する，という 2 つの制約を課すことで最適化の所要時間を実用的な範囲に収めるよう試みた．

図表要素数	参照文の重なり	実行時間 (秒)
2	あり	0.80
3	なし	0.19
4	なし	1.10
5	なし	3.47

表 1: 図表要素数と実行時間

4 評価

提案手法の参照関係抽出 (第 3.1 節) を Python で，図表配置の最適化 (第 3.2 節) をブラウザの拡張機能として Javascript を用いて実装した．また，図表配置最適化の部分の実行時間を実用的な長さにするため，本稿の実装では図表要素の倍率については等倍で固定することにした．

表 1 に実際の文書の一部に対して本実装を適用した際の実行時間を示す．図表要素が 4 枚以下の文書については約 1 秒以内で処理が終了し，図表要素の枚数が少ない場合には実用的な時間で実行可能であることが確認できる．ただし，図表要素が 5 枚を超えると処理時間が 3 秒を超えることから，単純な図表配置法に切り替えるなどの工夫が必要である．

5 おわりに

本稿では，図表参照文を利用してスクロールされるごとに図表要素の位置を順次固定していく方法を提案した．今後の課題としては，(1) 図表要素の倍率変更も含めた本手法をより多くの図表要素を含む文書へ適用可能にすること，(2) 生成された文書レイアウトの定量的な評価をすることが挙げられる．

参考文献

[1] N. Hurst, W. Li and K. Marriott, “Review of Automatic Document Formatting,” Proceedings of the 9th ACM Symposium on Document Engineering, 2009.

[2] K. Marriott, P. Moulder and N. Hurst, “Automatic float placement in multi-column documents,” Proceedings of the 7th ACM Symposium on Document Engineering, 2007.