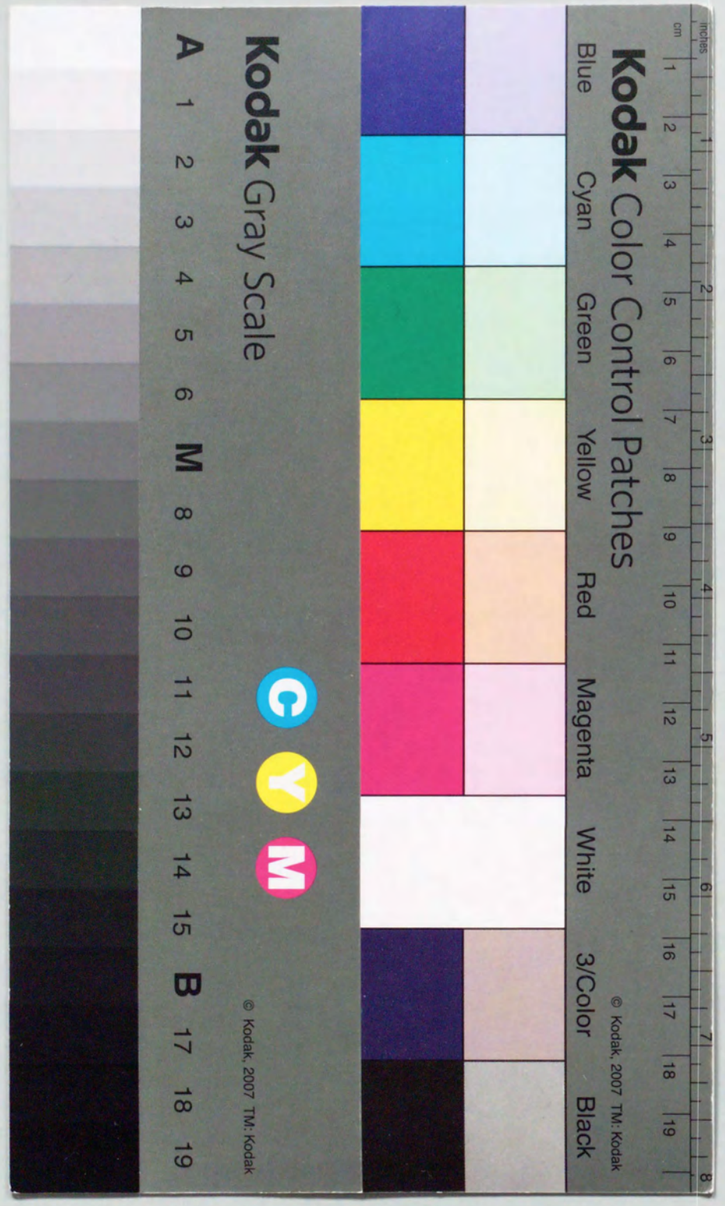


ユーザインタフェースにおける
アクセスリンク構造の意味的分類を保った
最適化問題

高田喜朗

大阪大学大学院基礎工学研究科

1997年1月

ユーザインタフェースにおける
アクセスリンク構造の意味的分類を保った
最適化問題

高田喜朗

大阪大学大学院基礎工学研究科

1997年1月

内容梗概

本論文は，著者が大阪大学大学院基礎工学研究科（物理系専攻）の学生として，都倉研究室において行った研究のうち，ユーザインタフェースにおけるアクセスリンク構造の最適化問題に関する研究をまとめたものである．

計算機アプリケーションの豊富な機能や提供する多量の情報にユーザが効率よくアクセスできるユーザインタフェースとして，メニューシステムやハイパーテキストがよく使われている．メニューシステムやハイパーテキストの構成要素が作る構造は，有向グラフとして表される．すなわち各メニューや情報ページが頂点に対応し，ユーザの選択操作によってアクセスされる頂点とその元の頂点の関係が有向辺に対応する．このようなグラフをアクセスリンク構造とここでは呼ぶ．本研究では，ユーザにとって使いやすいメニューシステムやハイパーテキストの構築を支援する著者支援のひとつとして，ユーザインタフェース設計者から与えられた大量の項目を自動的にアクセスリンク構造に配置する方法について考察する．

アクセスリンク構造を構成する際，1個の頂点内の項目数が多すぎても，逆にそれが少なくて階層構造が深くても，どちらもユーザのアクセスコストが大きくなる．そのため，アクセスコストを最小化するアクセスリンク構造は自明ではない．また，目標の項目に対する操作手順，すなわち途中で選択すべき上位階層の項目をユーザが知るためには，上位階層がその子孫の項目の意味的な分類となっていなければならない．つまり，上位階層の項目は分類名であり，目標の項目がどの分類に属しているかで選択すべき項目がわかる必要がある．

本研究で考える問題は，設計者が与えた項目の意味的分類を保ち，ユーザのアクセスコストである平均アクセス時間が最小であるようなアクセスリンク構造を構成する問題である．これを解く効率のよいアルゴリズムが構成できれば，設計者の負担を軽減できると期待できる．

以下，第1章では，アクセスリンク構造の最適化問題の意義，および本研究の結果について概説を述べている．

第2章では，最適メニュー階層構造問題について，問題を定式化し，問題を解くアルゴリズムを示している．このアルゴリズムは，与えられたカテゴリの数を n としたとき，最悪時に $\Theta(n^2)$ 時間，平均 $O(n \log n)$ 時間で解を求める．

第3章では，アクセスコストが最小なハイパーテキストの構成法について，最適リンク構造問題とそれを解くアルゴリズムを示している．このアルゴリ

ズムの時間計算量は，与えられたカテゴリの数を n としたとき，$O(n^3)$ である．さらに，ハイパーテキスト設計者から与えられた入力に頂点の追加が行われたときの更新問題について考察している．また，より拡張されたモデルに対しても考察を行っている．

最後に第4章では，本研究で得られた主な結果をまとめ，今後に残された問題について述べている．

目次

発表論文リスト

1. 学術雑誌論文

(1) 高田喜朗, 辻野嘉宏, 都倉信樹: 最適メニュー階層構造を求めるアルゴリズムについて, 情報処理学会論文誌, Vol. 36, No. 2, pp. 415–422 (1995-02).

(2) 高田喜朗, 梅木良太郎, 辻野嘉宏, 都倉信樹: 大きい目標の選択操作に対するFittsの法則の適合性, 電子情報通信学会論文誌（レター）, Vol. J79-D-I, No. 11, pp. 1001–1003 (1996-11).

(3) 高田喜朗, 辻野嘉宏, 都倉信樹: ユーザのアクセスコストを最小化するハイパーテキストの構成法, 情報処理学会論文誌, Vol. 38, No. 2 (1997-02 掲載予定).

2. 研究会

(4) 高田喜朗, 辻野嘉宏, 都倉信樹: 最適なメニュー階層構造を求めるアルゴリズム, 情報処理学会アルゴリズム研究会研究報告, 93-AL-35, pp. 11–18 (1993-10).

(5) 高田喜朗, 梅木良太郎, 辻野嘉宏, 都倉信樹: 大きい目標の選択操作に対するFittsの法則の適合性の評価, 情報処理学会ヒューマンインタフェース研究会研究報告, 95-HI-61, pp. 9–16 (1995-07).

(6) 高田喜朗, 辻野嘉宏, 都倉信樹: アクセスコストを最小化するハイパーテキストの構成法, 情報処理学会ソフトウェア工学研究会研究報告, 95-SE-105, pp. 17–24 (1995-09).

3. 口頭発表

(7) 高田喜朗, 篠田真由美, 魚井宏高, 辻野嘉宏, 都倉信樹: Fittsの法則に基づくマウスの操作方向の効率への影響, 情報処理学会第45回全国大会, 3T-3, 分冊5, pp. 217–218, (1992-10).

(8) 高田喜朗, 辻野嘉宏, 都倉信樹: 最適メニューシステムの設計法について, 情報処理学会第48回全国大会, 3K-04, 分冊5, pp. 351–352 (1994-03).

第1章

まえがき

計算機アプリケーションは高度化・大規模化する傾向にあり，機能が豊富で複雑になるにつれ，それらの機能にユーザが効率よくアクセスできるようなユーザインタフェースを備えることが重要となっている．そのようなユーザインタフェースとしてメニュー階層構造がよく使われており，特に近年，ウィンドウシステムの普及に伴って，基本的ユーザインタフェースとしてメニュー階層構造を用いたアプリケーションが一般的となっている．

メニューとは，ユーザに選択してもらう複数の項目を並べて表示したものである．メニューを使ったシステムでは，ユーザは，マウスやキーボードを使って，メニュー内の項目から必要なものを選択する．メニュー階層構造とは，項目を階層的に分類し，上位階層から順に下位へ選択していくようにした構造である．

メニューを画面にどのような形式で表示するか，あるいはユーザにどのような操作で選択してもらうかといったメニュー方式については，どのような方式が適しているかを知るためにすでにいろいろな研究が行われている [1, 15, 16, 17]．例えば，Walker ら [15] は，画面上部など画面上に固定された領域を選択操作するとメニューを表示するプルダウンメニュー方式と，マウスボタンを押したときのマウスカーソルの位置にメニューを表示するポップアップメニュー方式について，選択効率の比較実験を行った．それにより，プルダウンメニュー方式のほうがアクセスに必要な時間が小さいという結果が得られている．また，山本ら [16, 17] は，新しいメニュー方式である横型メニュー方式やリモートプルダウンメニュー方式を提案し，従来の方式との選択効率の比較実験を行っている．この他にも，項目を表示・選択する領域がパイを切ったような形のパイメニューの提案 [1] など，メニュー方式に関する研究は多く行われている．しかし，メニュー方式とは別に，アプリケーションに必要な各メニュー項目をどのようにメニュー内および階層構造に配置するかも，ユーザの使いやすさに大きく影響する．例えば，常に項目のメニュー内での位置が一定である方式と，メニューが表示されるたびに項目の配置を変え，よく使われる項目をメニュー内の上部に置く方式との比較実験が Mitchell らによって行われ [12]，項目の位置が一定であるほうが選択に必要な時間が小さいという結果が得ら

れている．また，メニュー内で項目をグループ化し，メニュー内にグループの区切りを表示することで，選択時間を短縮できることが知られている [11]．これらの研究はメニュー内での項目の配置に関する研究だが，多数の項目をユーザの使いやすさを考慮しながら各メニューに分けて階層構造に配置する方法も，よいユーザインタフェースを実現するために重要である．

また，計算機アプリケーションのいくつかの分野で，ハイパーテキストを利用したシステムがよく見られるようになってきている．アプリケーションのヘルプシステムやCD-ROMなどで配布される大容量の情報提示システム，分散データベースの一種であるWWW (World Wide Web) などがその例である．たとえば，WWWブラウザを使うと，表示される情報ページの文中に，地の文とは違う色やアンダーラインなどでハイライトされた語句（アンカー）がある．この部分をキーボードやマウスを使って選択すると，対応する情報ページに即座にアクセスできる．このような機構を持つものをハイパーテキストと呼んでいる．ハイパーテキストを用いることで，ユーザは，大量の情報の中を目的に応じて眺め歩くように検索でき，より快適に情報検索できると考えられている．

より検索しやすいハイパーテキストを構築するために，モデル，読者支援，著者支援，有効性評価，標準化などについてさまざまな研究がなされている [8]．ここでは著者支援に当たる，ユーザに提供する情報のハイパーテキストの構成法について考える．ここで考えるのは，大量の情報（文書）が個別に提供され，それらをユーザが効率よくアクセスできるように，メニューとその間のリンクを構築する問題である．WWWのように，ネットワーク中の各地の提供者からさまざまな情報が個別に提供されている状況で，それらを検索しやすいように索引を作っていくことなどを想定すればよい．ここでいうメニューとは，検索のために設けられたアンカーだけからなる頂点のことである．

典型的なハイパーテキストでは，各ページ上で他のページやメニューの題となるような語が，そのページやメニューに対応するアンカーにされる．これによりユーザは，アンカーとなっている語に興味を感じてそれを選択操作すれば，対応するページや対応するメニューから辿れるページ，すなわちその語に関連する情報を得られる．このアンカーにされる語を，関連する情報を検索するための語と言う意味で，ここではキーワードと呼ぶ．ハイパーテキストの設計者は，各ページ上で，ユーザが興味を感じ新しく検索を始めるのに適すると思われる語をキーワードに選びアンカーにする．これは一般のデータベース検索等で使われるキーワードと同質のものと考えられる．

特にWWWのように，さまざまな情報が個別に提供される状況では，設計者が選んだキーワードにぴたりと当てはまるページは与えられないことが多いと考えられる．よってここでは，キーワードに関連するページの集合（カテゴリ）を与えると，それらの中からユーザが適切なページを選べるように，自動的にメニューを配置することを考える．

以上より，本研究では，ユーザにとって使いやすいメニューシステムやハイパーテキストの構築を支援する著者支援のひとつとして，ユーザインタフェース設計者から与えられた大量の項目を自動的に階層構造に配置する方法について考察する．

ユーザが効率よくアクセスするためには，操作の手間が平均的に小さいことが必要になる．その尺度として，平均アクセス時間を考える．これは，適切な選択時間モデル上での，ユーザの目標となる各項目へのアクセス１回に必要な平均時間である．メニュー内の項目の数が多すぎても，逆に少なくて階層構造が深くても，どちらも平均アクセス時間が大きくなる．すなわち，前者の場合はメニューの中から必要な項目を探すのに，後者の場合は項目の選択を多数行わなければならないために，時間がかかる．このようなトレードオフがあり，最適なものを求めることが問題となる．

また，目標の項目に対する操作手順，すなわち途中で選択すべき上位階層の項目をユーザが知るためには，上位階層がその子孫の項目の意味的な分類となっていなければならない．つまり，上位階層の項目は分類名であり，目標の項目がどの分類に属しているかで選択すべき項目がわかる必要がある．

階層構造の設計の際，設計者はタスク分析や機能分析などを行い，各項目の使用頻度と意味による項目の分類を調査し与えることができると考えられる．そこで，今回考える問題は，設計者が与えた項目の意味的分類を保ち，ユーザのアクセスコストである平均アクセス時間を最小化する問題と定義できる．これを解く効率のよいアルゴリズムが構成できれば，設計者の負担を軽減できると期待できる．

ここで扱う平均アクセス時間は，対象とするメニュー階層構造やハイパーテキストのグラフ構造と，対象ユーザやシステムのパフォーマンスによって決まる定数とで与えられる式である．Cardらのキーストロークレベルモデル [2] では，ユーザの操作に必要な時間を，キー押下・目標へのマウス移動・手をキーボードなどのホームポジションに戻す動作・マウスの直線的な移動・動作の前の心理的準備・システムの反応時間の和とモデル化し，これらの項の値を実験的に求めている．本研究での平均アクセス時間もこのモデルに準じたものであり，キーストロークレベルモデルの研究の結果を応用して定数を与えられると考えられる．また，設計者が実際の対象システムや重視したい事柄に合わせて定数の値や意味づけを変更し，適用することもできる．

メニュー階層構造の構成法について，Fisherらは，メニュー設計者から与えられた項目の意味的分類がなされた階層構造から，その分類を崩さないようなある制限の下で平均アクセス時間を最小にする階層構造を求める方法を示した [5]．しかしFisherらの方法では次のような点で問題がある．

- Fisherらの方法では，制限が強すぎて，メニュー設計者から与えられた階層構造から得られる意味的分類を保った階層構造すべてを調べ尽くしてはいない．

- 現実のメニューでは多く見られるような，中間項目と終端項目が混在するメニューを許さない．

本論文では，与えられた項目の意味的分類を保つ階層構造すべての中からメニューの平均アクセス時間が最小な階層構造を求める問題について扱う．

以下の章では，メニュー階層構造とハイパーテキスト構造について，アクセスコストの最小化問題を考える．本研究で扱うこれらの問題は，それぞれ木およびDAG（有向無閉路グラフ）に対するアクセスコスト最小化問題に当たる．

第2章では，最適メニュー階層構造問題とそれを解くアルゴリズムを示している．このアルゴリズムは，与えられたカテゴリの数を n としたとき，最悪時に $\Theta(n^2)$ 時間，平均 $O(n \log n)$ 時間で解を求める．

第3章では，アクセスコストが最小なハイパーテキストの構成法について，最適リンク構造問題とそれを解くアルゴリズムを示している．このアルゴリズムの時間計算量は，与えられたカテゴリの数を n としたとき，$O(n^3)$ である．

第2章

最適メニュー階層構造

2.1 背 景

従来から，計算機アプリケーションのユーザインタフェースとしてメニューがよく用いられてきた．メニューとは，ユーザに選択してもらう項目を並べて表示したものである．メニューを使ったシステムでは，ユーザは，マウスやキーボードを使って，メニュー内の項目から必要なものを選択する．表示されているものから選択する方式のため，操作のために覚えるべき事柄が少なく，ユーザの認知的負荷を軽減する方式である．また，そのアプリケーションの操作に慣れていないユーザでも容易に操作できるようにする方式と言える．特に近年，ウィンドウシステムの普及に伴い，基本的ユーザインタフェースとしてメニューを用いたアプリケーションが一般的となっている．

計算機アプリケーションは年々高度化・大規模化する傾向にあり，機能が豊富で複雑になるにつれ，より多数の項目を持つメニューが見られるようになっている．項目が多い場合，1個のメニューにすべての項目を持たせるのでなく，一般にメニュー階層構造を使うことが多い．メニュー階層構造とは，項目を階層的に分類し，上位階層から順に下位へ選択していくようにした構造である．

よりよいユーザインタフェースを実現するためには，良好なメニュー方式とメニュー階層構造を設計する必要がある．ここでメニュー方式とは，メニューを画面に表示する形式およびユーザが選択を行う方法のことである．メニュー方式についてはすでにいろいろな研究が行われている [15, 16, 17]．しかし，メニュー方式とは別に，アプリケーションに必要な各メニュー項目をどのように階層構造に配置するかも，ユーザの使いやすさに大きく影響する [12]．そこで本論文では，メニュー階層構造の問題を扱う．

メニュー階層構造がユーザにとって使いやすいためには，操作に必要な運動の手間が平均的に小さいことと，目標の項目に対する操作手順が容易に理解できることの両方が求められる．

操作の手間の尺度として平均アクセス時間が考えられる．これは，適当な選択時間モデル上での，1回の選択操作に必要な平均時間である．メニュー項

目のうち，アプリケーションの機能が割り当てられ，ユーザの1回の選択操作の目標となる項目を，終端項目と呼ぶ．それ以外の項目，すなわち子メニューを選択するために置かれた項目を，中間項目と呼ぶ．ここで，終端項目の集合が与えられたとして，それをメニュー階層構造に配置することを考える．極端な解として，全終端項目を1個のメニューに置いた，高さが0の階層構造が考えられる．しかし，その1個のメニューでの選択時間が大きくなるため，平均アクセス時間は小さくない．あるいは各メニューの項目数を2とした，高さの大きい階層構造も考えられる．しかしこの場合も，各メニューごとの選択時間は小さいが，選択操作を階層の深さだけ繰り返すために，平均アクセス時間は小さくない．

また，目標の項目に対する操作手順，すなわち途中で選択すべき上位階層の項目をユーザが知るためには，上位階層がその子孫の項目の意味的な分類となっていなければならない．つまり，上位階層の項目は分類名であり，目標の項目がどの分類に属しているかで選択すべき項目がわかる必要がある．

メニューシステムの設計の際，設計者はタスク分析や機能分析などを行い，各項目の使用頻度と意味による項目の分類を調査し与えることができると考えられる．そこで，これらの情報から意味的分類を保ち平均アクセス時間が小さい階層構造を求める効率のよいアルゴリズムが構成できれば，設計者の負担を軽減できると期待できる．

Fisher らは，メニュー設計者から与えられた項目の意味的分類がなされた階層構造から，その分類を崩さないようなある制限の下で平均アクセス時間を最小にする階層構造を求める方法を示した [5]．しかし Fisher らの方法では次のような点で問題がある．

- Fisher らの方法では，制限が強すぎて，メニュー設計者から与えられた階層構造から得られる意味的分類を保った階層構造すべてを調べ尽くしてはいない．
- 現実のメニューでは多く見られるような，中間項目と終端項目が混在するメニューを許さない．

本章では，与えられた項目の意味的分類を保つ階層構造すべての中からメニューの平均アクセス時間が最小な階層構造を求める，最悪時 $O(n^2)$, 平均 $O(n \log n)$ 時間のアルゴリズムを示す．ただし，n はメニュー設計者から与えられる意味的分類の数である．

以下，2.2節では，本章で扱うメニュー階層構造と平均アクセス時間のモデルについて諸定義を行う．2.3節では，2.2節で定義したモデル上で最適なメニュー階層構造を求める効率のよいアルゴリズムを示し，評価を行う．最適なメニュー階層構造とは，与えられた項目の意味的分類を保つ階層構造すべてのうち平均アクセス時間が最小の階層構造である．2.4節では，より拡張されたモデルを採用した場合について議論する．

2.2 モデル

2.2.1 メニュー階層構造

定義 1 メニュー階層構造は，メニューを節点とする有向木である．メニューは，*1* 個以上の項目の集合である．

メニュー階層構造の一つの有向辺に対して，その有向辺の終点となるメニューを表す項目が，始点となるメニューにちょうど一つ存在する．その項目を中間項目と呼ぶ．それ以外の項目を終端項目と呼ぶ *(図 2.1)*. □

ユーザは，メニュー内の中間項目を選択操作することで，その中間項目から出る辺が入るメニューにアクセスできる．ユーザの1回の選択操作は，根メニューへのアクセスから始まり，中間項目の選択と子メニューへのアクセスを0回以上繰り返し，終端項目を選択して終了する．

中間項目の選択を行うとき，どれを選択すれば目的の項目のあるメニューにアクセス出来るかをユーザが理解できなければいけない．すなわち，各メニューから辿れる終端項目の集合が，各項目を選択することでアクセスできる各部分集合に，その項目名から理解できる分類によって分割されている必要がある．その分割は，一般的には，項目の意味的分類によって行われるのが自然である．意味によってグループ化された項目の集合をカテゴリと呼ぶ (図 2.2)．上述の要件を満たすため，ここでは次のような性質を満たすメニュー階層構造を対象とする．

定義 2 メニュー階層構造 G の任意のメニュー M から辿れる終端項目の集合と，M と同じ名前を持つカテゴリが一致するとき，G は *semantically well formed* (以下 *SWF* と略す) であるという． □

2.2.2 平均アクセス時間

ここでは，メニュー階層構造が与えられたときに，そのメニュー階層構造の終端項目にアクセスを行うときに必要な平均的な操作時間を考える．

定義 3 1個のメニュー M をアクセスするのに必要な平均時間を次のように定義する．

$$t(M) = c(|M|+1)/2+s+r,$$

c: 項目 *1* 個あたりの選択判定時間,
s: 選択操作 *(クリック，打鍵)* 時間,
r: 計算機の反応時間.

ただし，$|M|$ は M 内の項目の数，c, s, r はそれぞれ定数である． □

これは，メニュー内の各項目を順に1個ずつ目的のものかどうか判定し目的の項目に到達するまでの平均時間が $c(|M|+1)/2$，その後ユーザがマウスや

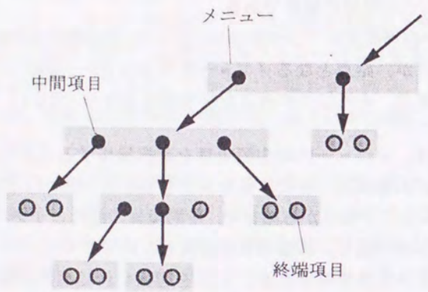


図 2.1: メニュー階層構造

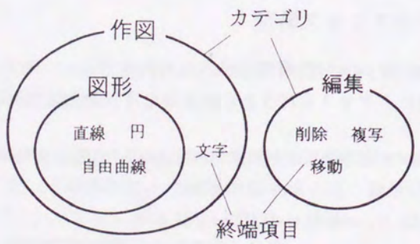


図 2.2: カテゴリ

キーボードを操作してその項目を選択するのに s，計算機がその操作を受け取り次のメニューの表示などを行うのに r だけ時間がかかるというモデルである．

与えられたメニュー階層構造について経験の浅いユーザの場合，各メニュー内にどのような項目があるかがわからないため，目的の項目かどうかを順に 1 個ずつ調べて検索を行うと考えられる．このときユーザは，目的の項目が見つかったところで判定を終えるとする．簡単のため，メニュー内での項目の配置順を考えないとし，ある項目が何番目に配置されるかは 1 番目から $|M|$ 番目まで等しい確率で起こるとすると，その項目が目的の項目であった場合の判定回数は平均 $(|M|+1)/2$ となる．このことから，上記の $t(M)$ は，このようなユーザの選択時間の第 1 次近似モデルとして妥当であると考えられる [5].

メニュー階層構造 G とそのメニュー M について，M を根とする G の部分階層構造を $G_{<M>}$ と表す.

定義 4 $G_{<M>}$ の平均アクセス時間 $ET[G_{<M>}]$ を次のように再帰的に定義する.

$$ET[G_{<M>}] = t(M) + \sum_{M' \in child[M]} ET[G_{<M'>}]\, P(M'|M).$$

$child[M]$ は M の子であるメニューの集合を表す．$P(M'|M)$ はメニュー M がアクセスされたときにその子 M' がアクセスされる条件付き確率である．□

アクセス確率については後述する.

定義 4は，根から必要な項目を選ぶのに $t(M)$ 時間かかり，その後全部分構造の平均アクセス時間の平均（アクセス確率で重みづけした和）の時間がかかるというモデルである．これは終端項目にアクセスするまでにかかる時間の期待値となる.

G の根を r とすると，G の平均アクセス時間 $ET[G]$ は，

$$ET[G] = ET[G_{<r>}]$$

となる.

アクセス確率

終端項目 o のアクセス確率を p_o とし，これは設計者が与えるとする．また，メニュー M のアクセス確率を $P(M)$，中間・終端項目 o_i のアクセス確率を $P(o_i)$ とする.

メニューのアクセス確率は，メニュー内の項目のアクセス確率の総和に等しい．また中間項目のアクセス確率は，それが指す子メニューのアクセス確率に等しい．よって，

$$P(M) = \sum_{o_i \in M} P(o_i),$$

$$P(o_i) = \begin{cases} p_{o_i}, & o_i \text{が終端項目} \\ P(M'), & o_i \text{が中間項目} \\ & (M' \text{は } o_i \text{が表す子メニュー}) \end{cases}$$

が成り立つ．メニュー Mがアクセスされたときにその子メニュー M'がアクセスされる条件付き確率 $P(M'|M)$ は，

$$P(M'|M) = P(M')/P(M)$$

となる．

2.2.3 縮　約

ここで考える問題は，SWF であるメニュー階層構造のうち平均アクセス時間が最小のものを求めることである．しかし，どのようなメニュー階層構造が平均アクセス時間を最小にするかは自明ではない．たとえば，図 2.2のようなカテゴリの集合が与えられたとき，図 2.3, 2.4の 3 つの構造はいずれも SWF であるが，どれが平均アクセス時間を最小にするかは，カテゴリの要素数，アクセス確率，定数 c, s, rによって異なり一般的にいえない．

ここでは，メニュー設計者が予め考えられるカテゴリをすべて与えるとし，それを基に各カテゴリが含む終端項目を変えないような階層構造の中から適するものを選ぶことを考える．

定義 5 与えられたカテゴリの集合に対し，次のように構成されたメニュー階層構造を *seed* 階層構造と呼ぶ．

- 各カテゴリに対し，同じ名前を持つメニューをちょうど *1* 個置く．
- 任意のカテゴリ A について，A の要素 qのうち，$q \in B$かつ $B \subseteq A$ であるような Bが存在しないものすべてを，メニュー A 内の終端項目とする．
- 任意のカテゴリ Aに対し，Aの部分集合 $B \subseteq A$のうち，$B \subseteq C \subseteq A$ であるような Cが存在しないものすべてについて，メニュー Bをメニュー A の子とする． □

図 2.3は，図 2.2のカテゴリの集合に対する seed 階層構造である．

このように構成された seed 階層構造は，SWF でありかつ各メニュー内の項目が最も少ない（すなわち階層が深い）階層構造になっている．seed 階層構造は，与えられたカテゴリの集合と一対一に対応する．

定義 6 次の操作を縮約という．

> メニュー M内の中間項目を，それから出る辺が入る子メニュー M' の項目全てに置き換える操作．

このとき，Mに M'を縮約するという． □

図 2.4の構造はどちらも図 2.3の構造に 1 回の縮約操作を行ったものである．すなわち，(i) は図形，(ii) は作図というメニューを縮約した．

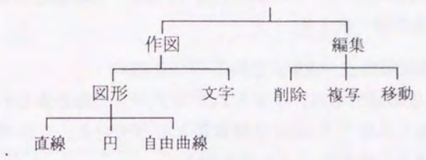


図 2.3: seed 階層構造

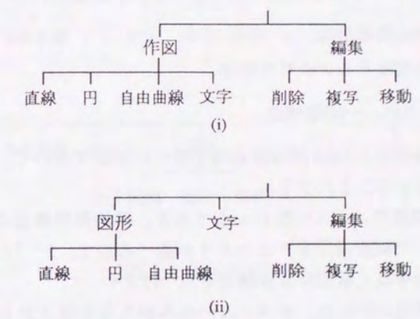


図 2.4: 縮約操作

定義 7 *seed* 階層構造に縮約操作を *0* 回以上繰り返して得られる階層構造を縮約階層構造と呼ぶ. □

図 2.5の構造は，図 2.3の構造を seed 階層構造としたときの縮約階層構造である．この場合，縮約階層構造はこの 8 通りが存在する．

補題 1 与えられたカテゴリの集合に対し，*SWF* である階層構造の集合と縮約階層構造の集合は一致する．

(証明)　縮約階層構造が SWF であるのは自明．

SWF である階層構造は，与えられたカテゴリの集合からいくつかのカテゴリを除いたものに対する seed 階層構造となっている．これは与えられた seed 階層構造に対する縮約によって得られる． □

以上より，ここで考える問題は，縮約階層構造のうち平均アクセス時間が最小のものを求めることになる．

定義 8 与えられた *seed* 階層構造に対する縮約階層構造のうち，平均アクセス時間が最小のものを最適なメニュー階層構造という． □

ここでは議論を簡単にするため，seed 階層構造が木であると仮定する．一般の場合（seed 階層構造が DAG の場合）については 2.4.1節で議論する．

2.3 最適メニュー階層構造問題

2.3.1 問題の定義

定義 9 最適メニュー階層構造問題は次のように定義される．

入力
- *seed* 階層構造，
- 終端項目のアクセス確率．

出力 最適なメニュー階層構造． □

カテゴリの集合と seed 階層構造は一対一に対応するので，seed 階層構造の形で入力を与えることにする．

seed 階層構造のメニュー数を n とすると，縮約階層構造は，根以外の各メニューについて縮約操作をするかどうか選べるので，2^{n-1}とおりある．これらの中から効率良く最適な階層構造を見つけたい．

縮約階層構造の中には，根メニューのみからなる高さが 0 の階層構造のような極端なものも含まれる．しかし，項目 1 個あたりの選択判定時間 c などのパラメータが自然な値に定義され，与えられた項目数が十分大きいなら，このような極端な階層構造は，その唯一のメニューの平均選択時間 $t(M)$ が大きな値となるため最適な階層構造とならない．また逆に，極端に深い階層構造も，メニュー選択回数が大きくなるため最適な階層構造にならない．よって，最適な階層構造を求めるアルゴリズムは，バランスのとれた階層構造を求めることが期待できる．

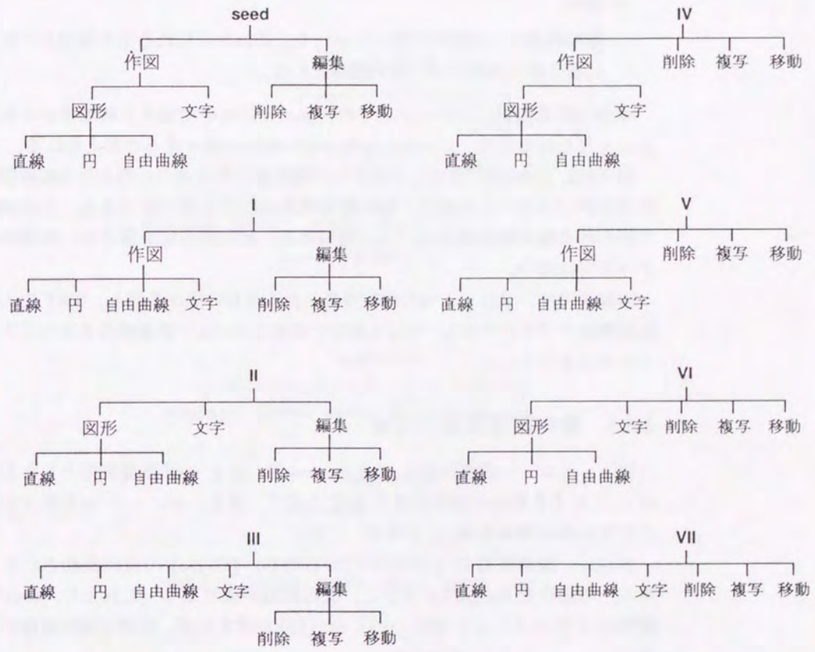


図 2.5: 縮約階層構造

既知の結果

Fisher らは，扱う縮約階層構造に制約を置いてその中で最適な階層構造を $O(n)$ 時間で得る方法を示した [5].

Fisher らの方法では，前節の定義に対し次のような違いがある．

- メニュー:

  メニューは，終端項目だけからなる集合（メニュー階層構造の葉）または中間項目だけからなる集合のいずれかである．

- 縮約:

  階層構造中の任意の中間メニューを，それから辿れる全終端項目の集合に置き換える操作を，縮約操作という．

現実の計算機アプリケーションのメニューには中間項目と終端項目の混在がよく見られるため，この混在を許さない制約は強すぎると考えられる．

図 2.6は，seed 階層構造と Fisher らの縮約操作の定義から得られる縮約階層構造の例である．この場合，縮約階層構造はこの 3 個だけである．この定義で得られる縮約階層構造は，seed 階層構造の意味的分類を崩さない階層構造すべてではない．

本論文では，メニュー内の中間項目と終端項目の混在を許し，SWF である階層構造すべての中から，$O(n^2)$ 時間で最適なメニュー階層構造を求めるアルゴリズムを示す．

2.3.2 縮約階層構造の性質

以下，メニュー階層構造 G の根を $root[G]$，G を seed 階層構造としたときの G に対する最適な階層構造を $《G》$ と表す．また，メニュー Mを根とする G の部分階層構造を $G_{<M>}$ と表す．

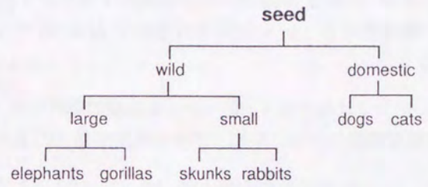
メニュー階層構造 G とメニュー Mに対し，$《G_{<M>}》$ の根に縮約されるメニューの集合を $Rdc[M]$ とすると，次の補題が成り立つ．これより，seed 階層構造 G の全メニュー Mについて $Rdc[M]$ が求まれば，最適な階層構造が求まる．

補題 2 *seed* 階層構造を G とする．最適な階層構造 $《G》$ の任意の部分階層構造 $《G》_{<M>}$は，$G_{<M>}$を *seed* 階層構造としたときの最適な階層構造である．

(証明)　背理法により示す．

$G_{opt} = 《G》$ とし，$H = (G_{opt})_{<M>}$が，$G_{<M>}$を seed 階層構造としたときの最適な階層構造でないと仮定する．すなわち，$G_{<M>}$を seed 階層構造としたときの最適な階層構造を H_{opt}とすると，$ET[H] > ET[H_{opt}]$ である．G_{opt}に対し，$(G_{opt})_{<M>}$を H_{opt}に換えた階層構造は，定義 4より，G_{opt}より平均アクセス時間が小さい．□

また，次の補題が成り立つ．



(seed 階層構造)

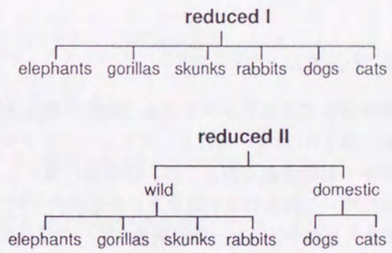


図 2.6: Fisher らの縮約操作

補題 3 メニュー階層構造 G の任意のメニュー M_1にメニュー M_2を縮約したとき，G の平均アクセス時間の変化は，M_1に M_2以外のメニューを縮約するかどうかによらない．

(証明)　M_1に M_2を縮約したとき，$G_{<M_1>}$を含まない部分階層構造の平均アクセス時間は変化しない．$G_{<M_1>}$を含む部分階層構造（$G_{<M,r>}$とする）の平均アクセス時間は，$ET[G_{<M_1>}]$ の変化を $P(M_1|M_r)$ 倍したものになる．

$ET[G_{<M_1>}]$ の変化は定義から計算すると

$$\frac{c}{2}(|M_2|-1)(1-P(M_2|M_1))-(c+s+r)P(M_2|M_1)$$

となる．これは M_1から M_2以外への辺を縮約するかどうかによらない．　□

メニュー階層構造 G と，$root[G]$ の任意の子 Mについて，$\langle G, M\rangle$ を次のように決める：

> $G_{<M>}$に含まれるメニューについてのみ縮約を行った，G に対する縮約階層構造のうち，平均アクセス時間が最小のもの．

補題 3より，$root[G]$ の子の集合を $\{M_1, M_2, \ldots, M_k\}$ としたとき，

$$《G》 = \langle \cdots \langle \langle G, M_1\rangle, M_2\rangle, \cdots, M_k\rangle$$

が成り立つ．

2.3.3　アルゴリズム

最適な階層構造を求めるアルゴリズム OMH の動作を説明する．アルゴリズムの詳細を，図 2.7に示す．ただし，メニューのアクセス確率は終端項目のアクセス確率から計算されるが，これらは項目の数に比例する時間で全部求まる．簡単のため，これらは予め計算され与えられているとする．

入力として与えられるメニュー階層構造のデータ構造は次のようにする．メニューは，中間項目のリストと項目数の組とする．中間項目は，それから出る辺が入る子メニューへのポインタとする．このとき 1 個のメニューの縮約は，中間項目のリストの連結となり，定数時間で実行できる．メニュー階層構造はその根へのポインタとし，縮約による変形や部分階層構造を扱う場合に階層構造の複写を行なわない．

OMH は次の手続きから構成されている．

- OMT　最適な階層構造を求める．
- OPT　部分 seed 階層構造 G を入力とし，$《G》$ を求める．実際は，$《G》$ の各メニューに縮約されるメニューの集合を求める．
- FINDR　部分 seed 階層構造 G について，$root[《G》]$ に縮約されるメニューの集合 Rを求める．


```
/* グローバル変数 */
Rdc: array[ menu ] of set of menu;
g: array[ menu ] of real;

OMH(inout G:menuHierarchy)
/* (入力) G: seed リンク構造,
   (出力) G: 最適リンク構造. */
 1 OPT(G)
 2 RESTR(G)

OPT(in G:menuHierarchy)
 1 r ← root[G]
 2 Rdc[r] ← ∅
 3 g[r] ← 0
 4 if G の高さが 0
 5    then return
 6 for each M ∈ child[r]
 7    do OPT(G_<M>)
 8       FINDR(G, M, R, d)
 9       Rdc[r] ← Rdc[r] ∪ R
10       g[r] ← g[r] + d

FINDR(in G:menuHierarchy; in M:menu; out R:set of menu; out d:real)
/* R: G_<M>のうち《G》の根に縮約されるメニューの集合,
   d: ET[⟨G, M⟩] − ET[G]. */
 1 R_1 ← {M}
 2 d_1 ← c(|M| − 1)/2 − t(M)P(M|root[G])
   /* Mを根に縮約したときの ET[G] の変化量 */
 3 for each M' ∈ child[M]
 4    do FINDR(G, M', R', d')
 5       R_1 ← R_1 ∪ R'
 6       d_1 ← d_1 + d'
   /* Mを縮約すると仮定したとき，《G》の根に
      縮約されるメニューと，G と ⟨G, M⟩ の ET の差 */
 7 if d_1 < g[M]P(M|root[G])
      /* G_<M>を最適化した場合と比較 */
 8    then R ← R_1
 9         d ← d_1
10    else R ← ∅
11         d ← g[M]P(M|root[G])

RESTR(inout G:menuHierarchy)
 /*   (入力)   G:   seed(部分) 階層構造,
      (出力)   G:   最適な階層構造. */
 1 for each M ∈ Rdc[root[G]]
 2    do root[G] に Mを縮約
 3 for each M ∈ child[root[G]]
 4    do RESTR(G_<M>)
```

図 2.7: アルゴリズム

- RESTR　OPT で求めた結果を基に，縮約操作を行って最適な階層構造を構成する．

OMH の入力である seed リンク構造を G とする．OMH はまず OPT を実行し，その後 RESTR によって，OPT の結果を基に最適な階層構造を構成する．

OPT は，G の各メニュー M について $Rdc[M]$ を求め，グローバル変数に代入する．具体的には，再帰呼出しにより根 r 以外の全メニューについて Rdc を求めた後，FINDR の計算結果を用いて $Rdc[r]$ を求める．同時に，G の全メニュー M について，$g[M] = ET[\langle\langle G_{<M>}\rangle\rangle] - ET[G_{<M>}]$ を計算する．これはグローバル変数に保持され，FINDR が参照する．

FINDR は，部分 seed 階層構造 G と，$root[G]$ の任意の子 M について，$root[\langle G, M\rangle]$ に縮約されるメニューの集合 R と，$d = ET[\langle G, M\rangle] - ET[G]$ を求める．具体的には，$root[G]$ の任意の子メニュー M について，

1) M を $root[G]$ に縮約し，そのリンク構造を G と置いて再帰的に $\langle G, M\rangle$ を求める．

2) M を縮約せず，$G_{<M>}$ を $\langle\langle G_{<M>}\rangle\rangle$ に置き換える．

の 2 通りを比較し，1) のほうが平均アクセス時間が小さい場合に M を R に含める．

2.3.4 実行例

図 2.8の階層構造を seed 階層構造としたときのアルゴリズムの出力を図 2.9および図 2.10に示す．ただし，これらの出力は，平均アクセス時間の定義に現れる定数すなわち定義 3中の定数 c, s, rを，それぞれ，$c = s = r = 1.0$ とした場合と，$c = 2.0, s = r = 1.0$ とした場合に対応する．

各図は，左側が上位層，右側が下位層になるように階層構造を描いている．●印の付いた名前が中間項目，それ以外の名前が終端項目を表している．例えば図 2.8で，メニュー「補正」は中間項目「階調補正」,「色調補正」を含み，メニュー「階調補正」は終端項目「階調の反転」,「平均化 (イコライズ)」と中間項目「階調を減らす」の 3 個の項目を含む．

各図の上部の ET(G) の行の数値は，その階層構造の平均アクセス時間である．図 2.9と図 2.10ではさらに，ET(seed) として，seed 階層構造の平均アクセス時間を示している．同じ階層構造でも定数 c, s, rの値によって平均アクセス時間が変わるので，図 2.9と図 2.10の seed 階層構造の平均アクセス時間の値は一致していない．

ここで用いた seed 階層構造は，Adobe Photoshop 3.0J というアプリケーションで実際に用いられているメニュー階層構造の一部（図 2.11）を基に作成した．実際のアプリケーションのメニュー階層構造は，ユーザの使いやすさを考慮して予め階層構造が深すぎないものになっているので，seed 階層構造と


```
c,s,r = 1.000000, 1.000000, 1.000000
ET(G)=15.593750
イメージ●+-補正●+-階調補正●+-階調の反転
          |       |          +-平均化（イコライズ）
          |       |          +-階調を減らす●+-2 階調化…
          |       |                          +-ポスタリゼーション…
          |       +-色調補正●+-色調●+-レベル●+-レベル補正…
          |                   |     |        +-トーンカーブ…
          |                   |     |        +-明るさ・コントラスト…
          |                   |     +-カラーバランス…
          |                   |     +-色相・彩度…
          |                   +-色の置き換え・選択●+-色の置き換え…
          |                   |                    +-特定色域の選択…
          |                   +-自動レベル調整
          |                   +-彩度を下げる
          |                   +-バリエーション…
          +-複製・重ね合わせ●+-複製…
          |                   +-重ね合わせ●+-画像操作…
          |                                 +-演算…
          +-画像変形●+-鏡像●+-水平方向
          |          |       +-垂直方向
          |          +-回転●+-固定角度●+-180 度
          |          |       |          +-90 度（時計回り）
          |          |       |          +-90 度（反時計回り）
          |          |       +-自由角度●+-角度入力…
          |          |                  +-マウスで自由に回転
          |          +-変形●+-拡大・縮小
          |                  +-平行四辺形
          |                  +-遠近法
          |                  +-自由な形に
          +-解像度・サイズ●+-画像解像度…
          |                 +-画像サイズ…
          +-ヒストグラム
          +-トラッピング…
```

図 2.8: seed 階層構造

```
c,s,r = 1.000000, 1.000000, 1.000000
ET(G)=13.062500  (ET(seed)=15.593750)

イメージ●+-階調補正●+-階調の反転
        |           +-平均化（イコライズ）
        |           +-2 階調化…
        |           +-ポスタリゼーション…
        +-色調補正●+-レベル補正…
        |           +-トーンカーブ…
        |           +-明るさ・コントラスト…
        |           +-カラーバランス…
        |           +-色相・彩度…
        |           +-色の置き換え…
        |           +-特定色域の選択…
        |           +-自動レベル調整
        |           +-彩度を下げる
        |           +-バリエーション…
        +-複製・重ね合わせ●+-複製…
        |                   +-画像操作…
        |                   +-演算…
        +-鏡像●+-水平方向
        |      +-垂直方向
        +-回転●+-180 度
        |      +-90 度（時計回り）
        |      +-90 度（反時計回り）
        |      +-角度入力…
        |      +-マウスで自由に回転
        +-変形●+-拡大・縮小
        |      +-平行四辺形
        |      +-遠近法
        |      +-自由な形に
        +-解像度・サイズ●+-画像解像度…
        |                 +-画像サイズ…
        +-ヒストグラム
        +-トラッピング…
```

図 2.9: 最適な階層構造 (c=s=r=1.0)


```
c,s,r = 2.000000, 1.000000, 1.000000
ET(G)=21.187500  (ET(seed)=23.562500)

イメージ●+-階調補正●+-階調の反転
        |           +-平均化（イコライズ）
        |           +-2 階調化…
        |           +-ポスタリゼーション…
        +-色調補正●+-レベル●+-レベル補正…
        |           |         +-トーンカーブ…
        |           |         +-明るさ・コントラスト…
        |           +-カラーバランス…
        |           +-色相・彩度…
        |           +-色の置き換え・選択●+-色の置き換え…
        |                                 +-特定色域の選択…
        |           +-自動レベル調整
        |           +-彩度を下げる
        |           +-バリエーション…
        +-複製・重ね合わせ●+-複製…
        |                   +-画像操作…
        |                   +-演算…
        +-鏡像●+-水平方向
        |      +-垂直方向
        +-回転●+-180 度
        |      +-90 度（時計回り）
        |      +-90 度（反時計回り）
        |      +-角度入力…
        |      +-マウスで自由に回転
        +-変形●+-拡大・縮小
        |      +-平行四辺形
        |      +-遠近法
        |      +-自由な形に
        +-解像度・サイズ●+-画像解像度…
        |                 +-画像サイズ…
        +-ヒストグラム
        +-トラッピング…
```

図 2.10: 最適な階層構造 (c=2.0, s=r=1.0)

```
c,s,r = 1.000000, 1.000000, 1.000000
ET(G)=13.968750

イメージ●+-階調補正●+-階調の反転
         |           +-平均化 (イコライズ)
         |           +-2 階調化…
         |           +-ポスタリゼーション…
         +-色調補正●+-レベル補正…
         |           +-トーンカーブ…
         |           +-明るさ・コントラスト…
         |           +-カラーバランス…
         |           +-色相・彩度…
         |           +-色の置き換え…
         |           +-特定色域の選択…
         |           +-自動レベル調整
         |           +-彩度を下げる
         |           +-バリエーション…
         +-複製…
         +-画像操作…
         +-演算…
         +-鏡像●+-水平方向
         |       +-垂直方向
         +-回転●+-180 度
         |       +-90 度 (時計回り)
         |       +-90 度 (反時計回り)
         |       +-角度入力…
         |       +-マウスで自由に回転
         +-変形●+-拡大・縮小
         |       +-平行四辺形
         |       +-遠近法
         |       +-自由な形に
         +-画像解像度…
         +-画像サイズ…
         +-ヒストグラム
         +-トラッピング…
```

図 2.11: Adobe Photoshop 3.0J のメニュー階層構造の一部

して用いるのには向かない．そこで，図 2.11の階層構造に対し新しくカテゴリを追加することで，seed 階層構造（図 2.8）を作成した．

また，終端項目のアクセス確率は，全終端項目について等しいと仮定した．

アルゴリズムの適用により，seed 階層構造からメニューが縮約され，平均アクセス時間の小さい階層構造が得られている（図 2.9, 2.10）．設計者の対象が，メニュー内で各項目を必要な項目かどうか判定するのに時間がかかるようなユーザやアプリケーションである場合は，図 2.10のように定数 c の値を他の定数に対して大きく与えることで対応できる．図 2.10の階層構造は図 2.9に較べて深いものになっており，各メニュー内の項目数が少ない出力が得られている．

図 2.9の階層構造は，単純な定数値やアクセス確率の設定にも関わらず実際のアプリケーションの階層構造に近いものが得られており，今回のアルゴリズムの出力が現実から大きく外れたものではないことがわかる．

アクセス確率にばらつきのある例

次に，同じく図 2.8の seed 階層構造について，終端項目のアクセス確率にばらつきのある値を与えた場合を考える．

英語で書かれた文書中の各単語の出現頻度について，Zipf の法則が成り立つことが知られている [9]．Zipf の法則とは，最も出現頻度の高い単語の出現確率を $P(1)$，2 番目に出現頻度の高い単語の出現確率を $P(2)$，同様に，r番目に出現頻度の高い単語の出現確率を $P(r)$ としたとき，$P(r) = C/r^{\alpha}$が成り立つというものである．ただし，Cは約 0.1，αは約 1 である．

終端項目のアクセス確率の分布が Zipf の法則に従うと仮定したときの最適階層構造を図 2.12，図 2.13に示す．ただし，これらの出力は，定義 3中の定数 c, s, rを，それぞれ，$c = s = r = 1.0$ とした場合と，$c = 2.0, s = r = 1.0$ とした場合に対応する．

アクセス確率の与え方は次のようにした．まず，図 2.8で，より上に書かれた終端項目ほどアクセス確率が高いとする．最もアクセス確率の高い「階調の反転」のアクセス確率を p としたとき，r番目の終端項目のアクセス確率を p/rとする．そして，全終端項目のアクセス確率の和が 1 になるよう p の値を決める．図 2.11は，各メニュー内の項目が実際のアプリケーションでのメニュー内の項目の配置と等しくなるように描かれている．図 2.8も，終端項目間の上下の関係は図 2.11と同じである．実際のアプリケーションでは，アクセス頻度が高いと思われる項目をメニュー内のより上に配置されていると推測されるため，上記のようにアクセス確率を与えた．

図 2.12，図 2.13とも，それぞれ図 2.9，図 2.10に較べて，図の上部ほど階層構造が浅くなっているのがわかる．実際のアプリケーション（図 2.11）では，図の全体が同じ程度の階層構造の深さになっている．このことから，実際のアプリケーションでは，終端項目のアクセスについて，Zipf の法則のように値のばらつきの大きい分布は仮定されていないことが推測できる．

```
c,s,r = 1.000000, 1.000000, 1.000000
ET(G)=11.851245  (ET(seed)=16.179459)

イメージ●+-階調の反転
         +-平均化（イコライズ）
         +-2 階調化…
         +-ポスタリゼーション…
         +-色調補正●+-レベル補正…
         |          +-トーンカーブ…
         |          +-明るさ・コントラスト…
         |          +-カラーバランス…
         |          +-色相・彩度…
         |          +-色の置き換え…
         |          +-特定色域の選択…
         |          +-自動レベル調整
         |          +-彩度を下げる
         |          +-バリエーション…
         +-複製・重ね合わせ●+-複製…
         |                  +-画像操作…
         |                  +-演算…
         +-画像変形●+-水平方向
         |          +-垂直方向
         |          +-180 度
         |          +-90 度（時計回り）
         |          +-90 度（反時計回り）
         |          +-角度入力…
         |          +-マウスで自由に回転
         |          +-変形●+-拡大・縮小
         |                  +-平行四辺形
         |                  +-遠近法
         |                  +-自由な形に
         +-解像度・サイズ●+-画像解像度…
         |                +-画像サイズ…
         +-ヒストグラム
         +-トラッピング…
```

図 2.12: アクセス確率を Zipf の法則で与えた例 (c=s=r=1.0)


```
c,s,r = 2.000000, 1.000000, 1.000000
ET(G)=19.307901  (ET(seed)=24.375243)

イメージ●+-階調の反転
         +-平均化（イコライズ）
         +-階調を減らす●+-2 階調化…
         |               +-ポスタリゼーション…
         +-色調補正●+-レベル補正…
         |          +-トーンカーブ…
         |          +-明るさ・コントラスト…
         |          +-カラーバランス…
         |          +-色相・彩度…
         |          +-色の置き換え・選択●+-色の置き換え…
         |          |                     +-特定色域の選択…
         |          +-自動レベル調整
         |          +-彩度を下げる
         |          +-バリエーション…
         +-複製・重ね合わせ●+-複製…
         |                  +-画像操作…
         |                  +-演算…
         +-画像変形●+-水平方向
         |          +-垂直方向
         |          +-固定角度●+-180 度
         |          |          +-90 度（時計回り）
         |          |          +-90 度（反時計回り）
         |          +-自由角度●+-角度入力…
         |          |          +-マウスで自由に回転
         |          +-変形●+-拡大・縮小
         |                  +-平行四辺形
         |                  +-遠近法
         |                  +-自由な形に
         +-解像度・サイズ●+-画像解像度…
         |                +-画像サイズ…
         +-ヒストグラム
         +-トラッピング…
```

図 2.13: アクセス確率を Zipf の法則で与えた例 (c=2.0, s=r=1.0)

2.3.5 アルゴリズムの正当性

補題 4 （*FINDR* の正当性） *seed* 階層構造 G の根以外のすべての頂点 qについて，$g[q] = ET[\langle\!\langle G_{<q>} \rangle\!\rangle] - ET[G_{<q>}]$ が正しく求まっているとする．

FINDR に，G と，$root[G]$ の任意の子 Mを入力したとする．このとき，*FINDR* が出力する Rは，$\langle G, M\rangle$ の根に縮約されるメニューの集合に等しい．また，*FINDR* が出力する d は，$d = ET[\langle G, M\rangle] - ET[G]$ を満たす．

(証明) まず，Mを $root[G]$ に縮約しない場合を考える．ここで，$G_{<M>}$の M 以外のメニューを縮約操作して得られる G の縮約階層構造のうち，平均アクセス時間が最小なものを G'とする．このとき，補題 2の証明と同様に，G'は，$G'_{<M>}$が，$G_{<M>}$を seed 階層構造としたときの最適な階層構造 $\langle\!\langle G_{<M>} \rangle\!\rangle$ と等しいような縮約階層構造である．

このことと，補題 3より，以下のことが言える:

- Mを $root[G]$ に縮約しないとき：$\langle G, M\rangle$ は，G 中の $G_{<M>}$を $\langle\!\langle G_{<M>} \rangle\!\rangle$ に置き換えたものに等しい．
- Mを $root[G]$ に縮約したとき：Mを $root[G]$ に縮約した G の縮約階層構造を G_1とする．また，Mの子の集合を $\{M_1, M_2, \ldots, M_k\}$ とする．このとき $\langle G, M\rangle = \langle \cdots \langle\langle G_1, M_1\rangle, M_2\rangle, \cdots, M_k\rangle$ となる．

以上より，$G_{<M>}$の高さについての帰納法によって，題意が言える． □

補題 5 （*OPT* の正当性） *OPT* に *seed* 階層構造 G を入力したとする．このとき，G の任意のメニュー Mについて，*OPT* が求めた $Rdc[M]$ は，$\langle\!\langle G_{<M>} \rangle\!\rangle$ の根に縮約されるメニューの集合に等しい．

(証明) 補題 4より，G の高さについての帰納法によって，題意が言える． □

定理 6 アルゴリズム *OMH* は，与えられた *seed* 階層構造 G から得られる縮約階層構造のうち平均アクセス時間が最小のものを返す．

(証明) 補題 5, 2より題意が言える． □

2.3.6 アルゴリズムの評価

seed 階層構造 H中のメニューの数を n とする．

最悪時間計算量

補題 7 *OMH* の最悪時間計算量は $\Theta(n^2)$ 時間である．

(証明) 部分階層構造 $G_{<M>}$について OPT が呼ばれたとき，

- OPT は再帰的に $G_{<M>}$のメニュー数だけ呼ばれる．これは $O(n)$ 個ある．

- OPT から FINDR が呼び出されたとき，FINDR は再帰的に $G_{<M'>}$（M'は Mの子）のメニュー数だけ呼ばれる．各 M'について $G_{<M'>}$のメニューは重複していないので，OPT が 1 回呼ばれたとき FINDR は $O(n)$ 回呼ばれる．
- RESTR の計算量は $O(n)$ である．

以上より，OMH の最悪時間計算量は $O(n^2)$ 時間である．

また，図 2.14のような階層構造を seed 階層構造 G とした場合を考える．OPT から FINDR(G, M, R, d) が呼ばれたとき，FINDR が再帰呼び出しされる回数は，$G_{<M>}$のメニュー数すなわち $G_{<M>}$の高さに 1 加えた値に等しい．このとき，OPT の時間計算量は $1 + 2 + \cdots + (n-1) = \Theta(n^2)$ である．よって OPT の最悪時間計算量は $\Omega(n^2)$ である．

以上より OMH の最悪時間計算量は $\Theta(n^2)$ 時間である． □

よって次の定理が成り立つ．

定理 8 最適なメニュー階層構造は $O(n^2)$ 時間で求められる． □

平均時間計算量

ここでの平均は，同じメニュー数の seed 階層構造の形がすべて等しい確からしさで現れるとして，その平均を考える．アクセス確率については最悪時で評価する．

まず，各メニューに含まれる中間項目の数を高々2 に制限した seed 階層構造について考える．

補題 9 *seed* 階層構造の各メニューに含まれる中間項目の数が高々2 であるとき，最適なメニュー階層構造は平均 $O(n \log n)$ 時間で求められる．

(証明) メニュー数 n のときの OMH の平均時間計算量を $T(n)$ とする．階層構造の左部分木のメニュー数が $k = 0, 1, \ldots, n-1$ である場合それぞれについて平均を取ることから

$$T(n) \leq \frac{1}{n} \sum_{k=0}^{n-1} \{T(k) + T(n-1-k)\} + O(n)$$

が成り立つ（図 2.15）．

$T(0), T(1)$ がともに $O(1)$ であるから $T(n) = O(n \log n)$ となる [4]． □

補題より，次の定理が成り立つ．

定理 10 最適なメニュー階層構造は平均 $O(n \log n)$ 時間で求められる．

(証明) 3 個以上の中間項目のあるメニューに対し，子孫メニューを追加して，各メニューに含まれる中間項目数が高々2 になるように変形した階層構造を考える．たとえば図 2.16(i) のメニューに対して (ii) のようなメニューを考える．

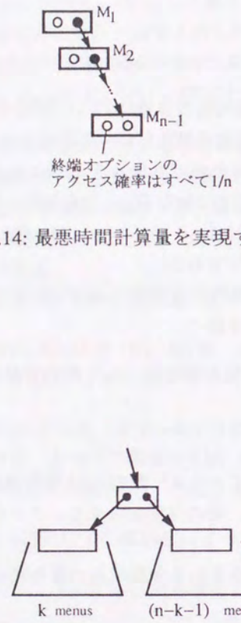


図 2.14: 最悪時間計算量を実現する場合

図 2.15: 中間項目が高々2 の seed 階層構造

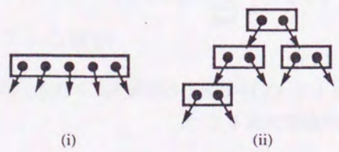


図 2.16: 階層構造の変形の例

ただし，このとき追加した子孫メニューが作る部分階層構造の葉の深さがなるべく均等になるようにする．

追加したメニュー数は，元のメニューの中間項目数すなわち子メニュー数を k として $k-2$ であるから，変形した階層構造のメニュー数は高々$2n$ である．したがって補題9より，変形した階層構造に対する OMH の平均時間計算量は $O(n \log n)$ 時間である．元の階層構造に対する時間計算量は，メニュー数が少ないので明らかにこれより大きくない．

よって，一般の seed 階層構造に対して，OMH の平均時間計算量は $O(n \log n)$ 時間である． □

2.4 議　論

2.4.1 DAG

以上では，任意の 2 個のカテゴリが，互いに素または一方が他方の部分集合であるような階層構造について考えた．しかし，ある終端項目が，互いに包含関係のない複数のカテゴリに属するような場合も考えられる．このとき，seed 階層構造は，木ではなく DAG（有向無閉路グラフ）となる（図 2.17）．

seed 階層構造が DAG であるとし，複数の親を持つメニューを根とする任意の部分階層構造を H とする．このとき H の扱い方に 2 通りが考えられる．

- H の根を親メニューに縮約しない．すなわち，H と同じ根を持つ縮約階層構造の部分階層構造が，その根のどの親メニューから見ても同じであるようにする（図 2.18）．

  H に対して最適な階層構造を求めた後その部分を固定することで前述のアルゴリズムが適用できる．このとき時間計算量は DAG のメニュー数に対し元のアルゴリズムと同じである．

- H を各親メニューごとに複写する（図 2.19）．

  複写により DAG を木に変形できるが変形後のメニュー数は元より多くなる．

2.4.2 メニュー内での項目の順序

以上では，メニュー内の項目の順序を考慮しない選択時間モデルについて考えた．しかし，各メニュー内で項目を順に 1 個ずつ目的のものかどうか判定しているとすれば，項目の選択時間はメニュー内での順序の関数とするモデルがより妥当である．そこで，各メニュー内で，順序が k 番目である項目の選択時間を

$$t'(k) = ck + s + r,$$

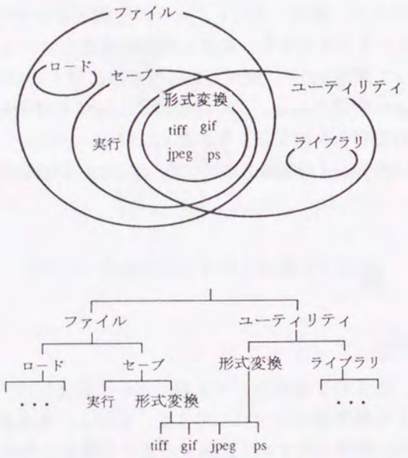


図 2.17: カテゴリの交差と seed 階層構造

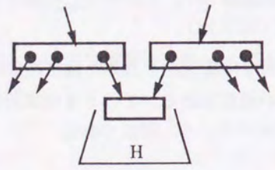


図 2.18: 部分階層構造を固定する場合

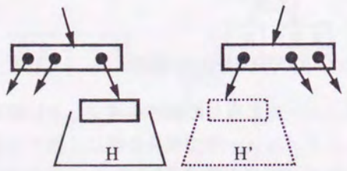


図 2.19: 木に変形する場合

階層構造 G とメニュー Mについて，部分階層構造 $G_{<M>}$の平均アクセス時間を

$$ET[G_{<M>}] = \sum_{o_i \in M} t'(index[o_i])\, P(o_i|M) + \sum_{M' \in child[M]} ET[G_{<M'>}]\, P(M'|M)$$

と仮定し，このモデルについて考える．ただし，c, s, r は 2.2.2節で定義した定数，$index[o_i]$ はメニュー内での o_iの位置である．

この場合，子メニューの各項目が親メニュー内で何番目になるかは，親メニューに縮約されている他のメニューに依存する．すなわち縮約時の平均アクセス時間の変化が親メニューの他の子メニューの縮約状況に依存する．このため，補題 3のような性質が成り立たず，前述のアルゴリズムは適用できない．

ところで，項目のグループ化によって選択時間を短縮できることが知られている [11, 16]．そこで，seed 階層構造で同じメニューに属していた項目が必ずメニュー内で連続しているという縮約階層構造のモデルも考えられる．しかしこの場合も同様に，縮約時の平均アクセス時間の変化が親メニューの他の子メニューの縮約状況に依存する．

しかし，次の補題が成り立つ．

補題 11 平均アクセス時間が最小であるような階層構造の，各メニュー内の項目の順序は，項目のアクセス確率の降順である．

(証明) 各メニュー内の項目の順序が項目のアクセス確率の降順でない，すなわち，あるメニュー内の項目 o_1, o_2について，$index[o_1] < index[o_2]$ かつ $P(o_1) < P(o_2)$ であるとする．

このとき，o_1と o_2を入れ換えた階層構造のほうがより平均アクセス時間が小さい． □

補題より，前述のアルゴリズムによって得られた階層構造の各メニューについて，項目をアクセス確率の降順にソートすることで，近似的な解が得られる．

2.5 第 2章のまとめ

本章では，与えられたメニュー階層構造について，項目の意味的分類を保つメニュー階層構造すべての中から，平均アクセス時間が最小な階層構造を求める効率のよいアルゴリズムを示した．このアルゴリズムは，与えられたメニュー階層構造のメニュー数を n としたとき，最悪時に $\Theta(n^2)$ 時間，平均 $O(n \log n)$ 時間で解を求める．

なお，2.3節で示したアルゴリズムでは，seed 階層構造 Hについて，根に縮約するメニューを Hの全メニューから探していた．しかし実際は，根の子メニュー Mと《$H_{<M>}$》の根に縮約されたメニューだけ調べれば良いことがわかっている [14]．これによって，オーダ的には変わらないが計算量を小さくできる．

また，seed 階層構造を DAG に拡張した場合や，メニュー内での項目の順序を考慮した選択時間モデルを採用した場合について考察した．

今後の課題として，これらの問題の計算量の下界，特に項目の順序を考慮した選択時間モデルに対する問題の計算量の下界を求めることがある．

第 3 章

アクセスコストが最小なハイパーテキストの構成法

3.1 背　景

計算機アプリケーションのいくつかの分野でハイパーテキストを利用したシステムがよく見られるようになってきている．アプリケーションのヘルプシステムや CD-ROM などで配布される大容量の情報提示システム，分散データベースの一種である WWW (World Wide Web) などがその例である．

たとえば，WWW ブラウザを使うと，表示される情報ページの文中に，地の文とは違う色やアンダーラインなどでハイライトされた語句（アンカー）がある．この部分をキーボードやマウスを使って選択すると，対応する情報ページに即座にアクセスできる．このような機構を持つものをハイパーテキストと呼んでいる．これをモデル化すると，ハイパーテキストとは，各頂点が文書や情報であるような有向グラフである．典型的なハイパーテキストシステムでは，頂点の中に，その頂点から出るリンクに対応するアンカーが置かれる．ユーザは，アンカーを選択操作することで，対応するリンクが入る頂点にアクセスできる [13]．ハイパーテキストを用いることで，ユーザは，大量の情報の中を目的に応じて眺め歩くように検索でき，より快適に情報検索できると考えられている．

より検索しやすいハイパーテキストを構築するために，モデル，読者支援，著者支援，有効性評価，標準化などについてさまざまな研究がなされている [8]．ここでは著者支援に当たる，ユーザに提供する情報のハイパーテキストの構成法について考える．ここで考えるのは，大量の情報（文書）が個別に提供され，それらをユーザが効率よくアクセスできるように，メニューとその間のリンクを構築する問題である．WWW のように，ネットワーク中の各地の提供者からさまざまな情報が個別に提供されている状況で，それらを検索しやすいように索引を作っていくことなどを想定すればよい．ここでいうメニューとは，検索のために設けられたアンカーだけからなる頂点のことである．

典型的なハイパーテキストでは，各ページ上で他のページやメニューの題となるような語が，そのページやメニューに対応するアンカーにされる．これによりユーザは，アンカーとなっている語に興味を感じてそれを選択操作すれば，対応するページや対応するメニューから辿れるページ，すなわちその語に関連する情報を得られる．このアンカーにされる語を，関連する情報を検索するための語と言う意味で，ここではキーワードと呼ぶ．ハイパーテキストの設計者は，各ページ上で，ユーザが興味を感じ新しく検索を始めるのに適すると思われる語をキーワードに選びアンカーにする．これは一般のデータベース検索等で使われるキーワードと同質のものと考えられる．

特にWWWのように，さまざまな情報が個別に提供される状況では，設計者が選んだキーワードにぴたりと当てはまるページは与えられないことが多いと考えられる．よってここでは，キーワードに関連するページの集合（カテゴリ）を与えると，それらの中からユーザが適切なページを選べるように，自動的にメニューを配置することを考える．カテゴリはその部分集合として別のカテゴリを含むことがあるので，メニューの項目は別のメニューを指すことがある（図3.1）．図3.1で，「art」というメニューから「CG」というメニューが指されている．また，図3.1の「CG」のように複数のカテゴリに含まれるような部分カテゴリも存在し得ることから，メニューが作るハイパーテキストの部分構造は一般にDAG（有向無閉路グラフ）となる．

図3.1で，実際の文書のページは長方形で，メニューは楕円形で表している．左から２つ目のページを開いているユーザがその中の「art」というアンカーをクリックすると，artというメニューが開き，関連するページや別のメニューへのアクセスができる．この例では「CG」という項目を選ぶと別のメニューが現れる．もちろん，メニュー「art」の中にメニュー「CG」の全項目を含めればメニュー「CG」はなくてもいいかも知れないが，その場合メニューが大きくなり過ぎて，そこでの検索がしにくいこともあり得る．

ユーザが効率よくアクセスするためには，操作の手間が平均的に小さいことが必要になる．その尺度として，平均アクセス時間を考える．これは，適切な選択時間モデル上での，ページへのアクセス１回に必要な平均時間である．メニュー内のアンカーの数が多すぎても，逆に各メニューのアンカーが少なくて階層構造が深くても，どちらも平均アクセス時間が大きくなる．すなわち，前者の場合はメニューの中から必要なアンカーを探すのに，後者の場合はアンカーの選択を多数行わなければならないために，時間がかかる．このようなトレードオフがあり，最適なものを求めることが問題となる．

また，ユーザが目的のページを検索する際に途中のメニューで迷わないように，キーワードからアクセスできるページの集合は，そのキーワードに意味的に関係のあるページの集合（カテゴリ）と対応していなければならない．

ハイパーテキスト構造の構築の際，設計者は実験や分析により，各ページの使用頻度とキーワードの集合および各キーワードに対するカテゴリを与えることができると考えられる．そこでこれらの情報から，キーワードからアクセスできるページの集合とカテゴリとの対応を保ち，平均アクセス時間が小さいハイパーテキスト構造を求める効率のよいアルゴリズムが構成できれば，設計者の負担を軽減できると期待できる．

前章「最適メニュー階層構造」では，グラフィカルユーザインタフェース(GUI)としてよく用いられるメニューシステムについて，与えられた項目の意味的分類を保ちながら平均アクセス時間が最小なメニュー階層構造を求める問題を扱った．今回扱うハイパーテキスト中のメニューが作るリンク構造の平均アクセス時間も，メニュー階層構造と同様に考えることができる．しかし，メニュー階層構造は１個の根を持つ木であるのに対し，ハイパーテキストのリンク構造は，多数の入り口（ページから出る辺が入るメニュー，すなわちキーワードを名前とするメニュー）を持つDAGである．つまり，ユーザが興味や目的に沿ってそれぞれの入り口にアクセスし，そこからメニューを辿ってページにアクセスするまでを１回の検索操作として，全入り口に対して平均的な検索時間を考慮しなければならない．また，2.4.1節ではDAGであるメニュー階層構造に対する考察も行っているが，これはアルゴリズムがDAGを木と同様に扱えるようにする方法を考察しただけで，DAGであるメニュー階層構造の最適性をどのように定義するかには触れていなかった．

本章では，まず，多数の入り口を持つDAGであるリンク構造と，その平均アクセス時間について，適切な定義づけを試みる．そして，与えられたキーワードとカテゴリの関係を保つリンク構造すべての中から，平均アクセス時間が最小なリンク構造を求める，多項式時間のアルゴリズムを示す．

以下，3.2節では，本章で扱うハイパーテキストの構造と平均アクセス時間のモデルについて諸定義を行う．3.3節では最適リンク構造問題を定義し，それを解く効率のよいアルゴリズムを示す．またアルゴリズムの正当性を示し，評価を行う．3.4節では，ハイパーテキストに頂点を追加した場合の更新問題や，より拡張されたモデルを採用した場合について議論する．

3.2 モデル

3.2.1 リンク構造

ここで扱うハイパーテキストの構成要素を定義する．

定義10 ハイパーテキストは有向グラフである．ハイパーテキストの頂点は，ページまたはメニューである． □

ページは，ユーザに提供される情報を収めた最小単位である．メニューは，ユーザが目的のページを検索するために置かれた頂点である．ハイパーテキストでのメニューや辺の役割を以下に説明する．

ページおよびメニュー内には，その頂点から出る辺に一対一に対応するアンカーが置かれる．ユーザがハイパーテキストを読む場合，頂点内のアンカーを選択操作することで，そのアンカーに対応する辺が入る頂点にアクセスで

きる．アンカーは，対応する辺が入る頂点を示す語や記号で表示される．ここでは，アンカーは，対応する辺が入る頂点の名前で表示されるとする．ユーザは，頂点内のアンカーから適するものを選択操作することを繰り返して目的のページにアクセスする．また，ページから，その中のアンカーを選択操作することで，別のページへアクセスするための操作を開始できる．

ここで考えるのは，与えられたページの集合から，メニューと辺を適切に設けてハイパーテキストを構成することである．各ページ内には，他のページへのアクセスの始点となるようなキーワードが0個以上含まれている．このキーワードをアンカーとし，ユーザがそこから辺を辿ることでそのキーワードに関連するページにアクセスできるようなハイパーテキストを構成するのが目的となる．キーワードが1個のページを指すのでなければ，上記のことを実現するために，そのキーワードを名前とするメニューを設ける必要がある．このとき，各メニューから，その名前であるキーワードに関連するページにアクセスできるよう辺を設ける．

ページから，その中のアンカーであるキーワードを選択操作し，辺を辿りながらメニューをアクセスした後，次にページをアクセスするまでを，ユーザの1回の検索操作と考える．ユーザは，この検索操作を，目的や興味に沿って必要なだけ繰り返す．そこで，1回の検索操作に必要な，メニューから出る辺だけからなるグラフを考える．

定義 11 ハイパーテキストから，ページから出る辺を除いたものをリンク構造と呼ぶ． □

ただし，ページから出る辺はページ内のアンカーであるキーワードによって表せるので，ハイパーテキストとリンク構造は，グラフとして異なるだけで，表す情報は等しい．ユーザの1回の検索操作は，リンク構造中のある頂点からアクセスを始め，リンク構造の辺を辿って，ページをアクセスしたところで終了する．検索操作の開始点となる頂点は，ハイパーテキスト中で，ページから出る辺が入る頂点である．これを入り口と呼ぶ．これはすなわち，ページ中のキーワードと同じ名前を持つ頂点である．

定義 12 ハイパーテキスト中で，ページから出る辺が入る頂点を入り口と呼ぶ． □

複数のキーワードに関連するページは複数のメニューから辿れる必要があるので，リンク構造は一般にDAG（有向無閉路グラフ）となる．

仮定 1 リンク構造は *DAG* である． □

キーワードに意味的に関連するページの集合をどのように決めるかという問題はここでは扱わず，これはハイパーテキストの設計者が与えるものとする．キーワードと関連するページについて，次のように定義する．

定義 13 意味的にグループ化されたページの集合をカテゴリと呼ぶ． □

設計者がキーワードに対して，それに関連するページとして与えるページの集合が，そのキーワードを名前とするカテゴリである．すなわち，各キーワードに対し，それを名前とするメニューとカテゴリがちょうど1個ずつ存在する（ただし，キーワードが直接1個のページを指す場合は，そのキーワードにはそれを名前とするメニューもカテゴリも存在しない）．各メニューから辿れるページがユーザにわかるようなリンク構造であるために，ここでは次のような性質を満たすリンク構造を対象とする．この定義は，メニュー階層構造に対する定義2と同様である．

定義 14 リンク構造 G の任意のメニュー M から辿れるページの集合と，M と同じ名前を持つカテゴリが一致するとき，G は *semantically well formed*（以下 *SWF* と略す）であるという． □

3.2.2 平均アクセス時間

ここでは，リンク構造が与えられたとき，そのリンク構造のページにアクセスを行うときに必要な平均的な操作時間を考える．

定義 15 1個の頂点 M をアクセスするのに必要な平均時間を次のように定義する．

$$t(M) = \begin{cases} c(|M|+1)/2+s+r, & M\text{がメニュー}, \\ r, & M\text{がページ}, \end{cases}$$

c: 出次辺 *1* 個あたりの選択判定時間，
s: 選択操作 *(*クリック，打鍵*)* 時間，
r: 計算機の反応時間．

ただし，$|M|$ は M から出る辺の数，c, s, r はそれぞれ定数である． □

これは，メニュー階層構造でのメニューをアクセスするのに必要な平均時間（定義3）と同様のモデルである．ただし，頂点がメニューとページの2種類となっているので，それに合わせて拡張されている．このモデルの妥当性については2.2.2節で触れた．WWWでは，目的のページに初めてアクセスするような場合が多いと考えられ，このモデルが比較的あてはまると推測される．

定義 16 リンク構造 G とその頂点 M について，M と M から到達可能な頂点を頂点集合とする G の誘導部分グラフ *(induced subgraph)*[1] を $G_{<M>}$ と表し，M を根とする G の部分リンク構造という． □

定義 17 $G_{<M>}$ の平均アクセス時間 $ET[G_{<M>}]$ を次のように再帰的に定義する．

$$ET[G_{<M>}] = t(M) + \sum_{M' \in child[M]} ET[G_{<M'>}]\, P(M'|M).$$

[1] グラフ $G = (V, E)$ と頂点集合 $V' \subseteq V$ について，V' を頂点集合とする G の極大な部分グラフ $G' = (V', E')$ を，V' を頂点集合とする G の誘導部分グラフという [7]．すなわち $E' = \{(u, v) \in E | u, v \in V'\}$．

child[M] は Mの子である頂点の集合を表す．$P(M'|M)$ はメニュー Mがアクセスされたときにその子 M'がアクセスされる割合を表す重みである． □

アクセスの重みについては次節で述べる．

定義 17は，メニュー階層構造の平均アクセス時間（定義 4）と同様である．

定義 18 リンク構造 G の平均アクセス時間 $ET[G]$ を次のように定義する．

$$ET[G] = \sum_{q \in \mathrm{PAGE}[G]} \frac{1}{|key[q]|} \sum_{M \in key[q]} ET[G_{<M>}]\, P(q).$$

ただし，PAGE[G] は G 中のページの集合，$P(q)$ はページ qのアクセス確率である．*key*[q] は，ハイパーテキスト中で，qから出る辺が入る頂点の集合である． □

3.2.1節より，ページ qから出る辺が入る頂点は，qに含まれるキーワードで表される．アクセス確率については次節で述べる．

定義 18は，リンク構造の平均アクセス時間を，各入り口を根とする部分リンク構造の平均アクセス時間を重みづけして足し合わせたものとしている．ただし，部分リンク構造の重みはキーワードがアクセスされる重みとし，それはキーワードを含む各ページのアクセス確率をそのページ中のキーワードの数で均等分して足し合わせたものとしている．よくアクセスされるページに少数だけあるキーワードはより多くアクセスされると考えられることから，定義 18はリンク構造の平均アクセス時間の定義として妥当と考えられる．

3.2.3 アクセス確率

リンク構造の平均的なアクセス時間を決めるために，各ページやメニューがアクセスされる割合を表す重みを決める必要がある．各ページがどのような頻度でアクセスされるかというアクセス確率は，メニュー階層構造の場合と同様，システム設計者が実験や分析，ログデータの収集によって与えることができると考えられる．そこで，与えられたページのアクセス確率を基にメニューのアクセスの重みを定義する．

ページ qのアクセス確率を $P(q)$ とする．これはシステム設計者が与える．確率の性質として，$P(q)$ は非負で，$\sum_{q \in \mathrm{PAGE}[G]} P(q) = 1$ を満たすように与えられるとする．

親メニュー Mがアクセスされた場合に子孫の頂点 M'がアクセスされる重み $P(M'|M)$ については次の２通りの定義が考えられる．

(1) 子のアクセス確率を複数の親に分配

子メニューまたはページ M'がアクセスされた場合に親メニュー Mがアクセスされている事後確率 $P(M|M')$ を考え，これを

$$P(M|M') = 1/|parent[M']|$$

とする．ただし，*parent*[M'] は M'の親の集合である．つまり，各親から均等な確率でアクセスされるとする．

均等分ではなく入力として与えることも可能である．この場合，事後確率は複数の親カテゴリに含まれる子カテゴリの，各親への関連度と言える．

上記の事後確率を用いて，メニューのアクセス確率 $P(M)$ を

$$P(M) = \sum_{M' \in child[M]} P(M|M')\, P(M')$$

とする．親メニュー Mがアクセスされた場合に子メニュー M'がアクセスされる確率 $P(M'|M)$ を次のようにする．

$$P(M'|M) = P(M|M')\, P(M')/P(M).$$

このとき $\sum_{M' \in child[M]} P(M'|M) = 1$ となる．

(2) アクセス確率を分配せず同じ値を親に渡す

メニューのアクセス重み $P(M)$ をそのメニューからたどれるページのアクセス確率の和と定義する．このとき

$$0 \le P(M) \le 1,$$

Mが M'の祖先の場合 $P(M) \ge P(M')$

が成り立つ．これはカテゴリのアクセス重みといえる．

親メニュー Mがアクセスされた場合に子孫 M'がアクセスされる重み $P(M'|M)$ を次のようにする．

$$P(M'|M) = P(M')/P(M).$$

このとき $\sum_{M' \in child[M]} P(M'|M) \ge 1$ となる．

(1) の定義では $\sum_{M' \in child[M]} P(M'|M) = 1$ が成り立ち，メニューのアクセス重みがそのアクセス確率に近いような定義になっている．(2) の定義は，あるページにアクセスできるパスが多数ある場合でも，各パス上のメニューにそのページのアクセス確率と等しい重みを加えるという定義になっている．(1) のほうがアクセス確率に近いが，リンク構造が一般の DAG の場合，アルゴリズムがうまく取り扱えないような性質がリンク構造に生じる．これについては 3.4.2節で議論する．

3.2.4 縮 約

ここで考える問題は，SWF であるリンク構造のうち平均アクセス時間が最小のリンク構造を求めることである．しかし，メニュー階層構造の場合と同様に，どのようなリンク構造が平均アクセス時間を最小にするかは自明ではない．たとえば，$B \subseteq A$ であるような２カテゴリ A, Bが与えられたとき，

- メニュー A, Bから別々にそれぞれのカテゴリの要素であるページへの辺を置いたリンク構造（図 3.2(a)），
- カテゴリ Bの要素に対しては A からの辺を置かず代わりに A から Bへの辺を置いたリンク構造（図 3.2(b)），

ともにSWF であるが，どちらが平均アクセス時間が小さいかはカテゴリの要素数，アクセス重み，定数 c, s, rによって異なり一般的にいえない．

定義 19 与えられたカテゴリの集合に対し，次のように構成されたリンク構造を *seed* リンク構造と呼ぶ．

- 任意のカテゴリ A とその要素 qについて，$q \in B$かつ $B \subseteq A$ であるような Bが存在しないとき，メニュー A から qへの辺を置く．
- $B \subseteq A$ であるような任意のカテゴリ A, Bについて，$B \subseteq C \subseteq A$ であるような Cが存在しないとき，メニュー A から Bへの辺を置く．
- 上記以外の辺を置かない． □

このように構成された seed リンク構造は，SWF でありかつ各メニュー内のアンカーが最も少ない（すなわち階層が深い）リンク構造になっている．seed リンク構造は，与えられたカテゴリの集合と一対一に対応する．

ここで，seed リンク構造に次のような仮定を置く．

仮定 2 *seed* リンク構造は「枝分かれ合流」のない *DAG* とする．「枝分かれ合流」とは，任意のメニュー Mとその子孫 M'の間に複数のパスがあることをいう． □

まずこの仮定を置いた場合について述べ，その緩和については 3.4.2節で議論する．

定義 20 次の操作を縮約という．

> メニューから出る 1 個の辺を，その辺の先のメニューから出る辺全てに置き換える操作．

ただし，辺の先がページである場合は縮約操作はできない． □

これは，メニュー階層構造におけるメニューの縮約と同様の操作である．

図 3.3は縮約の例である．図 3.3(a) のリンク構造でメニュー「computer」から「CG」への辺を置き換えると（「computer」に「CG」を縮約すると）図 3.3(b) になる．メニュー Mとその子 M'の間で縮約操作を行ったとすると，メニュー階層構造の場合は，M'に入る辺がなくなり，M'はユーザに使われることのないメニューとなる．しかしリンク構造の場合，一般に M'には M以外の親からの辺が入り，またハイパーテキスト中でページ内のアンカーから辺が入るため，縮約操作を行っても M'はユーザに使われ得る．

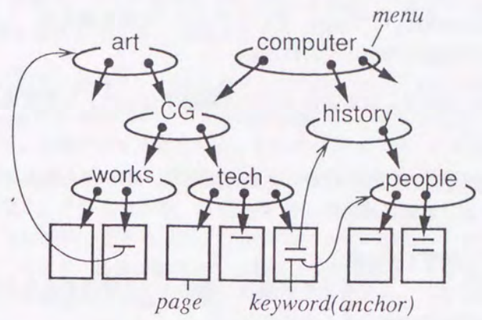


図 3.1: ハイパーテキスト構造

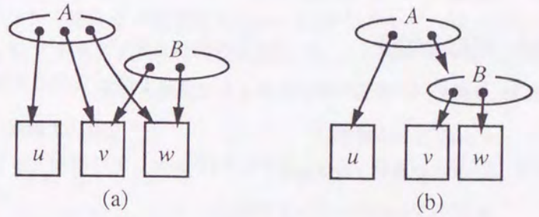


図 3.2: SWF である 2 リンク構造

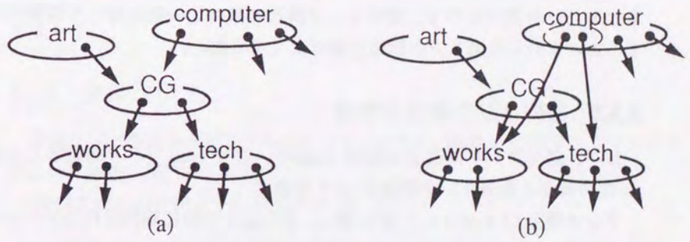


図 3.3: 縮約

縮約で辺を置き換えるとき，加えられるアンカーはその名前に縮約されるメニューの名前を付加するようにする．例えば上の例で「CG」というアンカーを「CG.works」,「CG.tech」というアンカーに置き換える．このようにすればカテゴリ分類の情報は失われない．

定義 21 *seed* リンク構造に *0* 回以上縮約を行ったリンク構造を縮約リンク構造という． □

補題 12 与えられたカテゴリの集合に対し，*SWF* であるリンク構造の集合と縮約リンク構造の集合は一致する．

(証明)　補題 1と同様． □

以上より，ここで考える問題は，縮約リンク構造のうち平均アクセス時間が最小のものを求めることになる．

定義 22 与えられた *seed* リンク構造に対する縮約リンク構造のうち，平均アクセス時間が最小のものを最適リンク構造という． □

3.3 最適リンク構造問題

3.3.1 問題の定義

定義 23 最適リンク構造問題は次のように定義される．

入力
- *seed* リンク構造，
- ページのアクセス確率，
- ページ中のキーワードの集合．

出力 最適リンク構造． □

カテゴリの集合と seed リンク構造は一対一に対応するので，seed リンク構造の形で入力を与えることにする．

seed リンク構造中のメニューからメニューへの辺それぞれについて縮約するかしないか選べるので，縮約リンク構造の数は辺の数に対して指数的になる．この中から最適リンク構造を効率よく求めたい．

3.3.2 縮約リンク構造の性質

以下，部分リンク構造 G の根を $root[G]$，G を seed リンク構造としたときの G に対する最適リンク構造を $《G》$ と表す．

リンク構造 G とメニュー M に対し，$《G_{<M>}》$ の根に縮約されるメニューの集合を $Rdc[M]$ とすると，以下の定理が成立する．

定理 13 *seed* リンク構造 G に対して次のように構成したリンク構造 G_{opt} は，G に対する最適リンク構造である：

- 頂点集合が G と等しい．
- G_{opt} の任意のメニュー M から出る辺は，G の同じメニュー M に $Rdc[M]$ の要素を縮約したときの M から出る辺と同じ．

(証明)　G の任意の部分リンク構造 H とその中のメニュー M について考える．ここで，$H_{<M>}$ 中の M 以外のメニューを縮約操作して得られる H の縮約リンク構造のうち，平均アクセス時間が最小なものを H' とする．このとき，H' は，$H'_{<M>}$ が，$H_{<M>}$ を seed リンク構造としたときの最適リンク構造 $《H_{<M>}》$ に等しいようなリンク構造である．なぜなら，もしそれらが等しくない場合，等しくなるように縮約し直すことで，H' の平均アクセス時間がより小さくなるからである．つまり，最適な部分リンク構造の任意の部分リンク構造は，その部分についての最適リンク構造である．

また，メニュー A から B への辺を縮約した場合，B から出る辺や B に入る他の辺には影響しない．すなわち A を含まないような部分リンク構造（$G_{<B>}$ を含む）の構造は変化しない．

よって，上のように構成した G_{opt} は，任意のメニュー M について $(G_{opt})_{<M>}$ が $《G_{<M>}》$ に等しい．以上より，G_{opt} は G に対する最適リンク構造となる． □

補題 14 リンク構造 G の任意のメニュー M_1 からメニュー M_2 への辺を縮約したとき，G の平均アクセス時間の変化は，M_1 から M_2 への辺以外の辺を縮約するかどうかによらない．

(証明)　補題 3と同様． □

部分リンク構造 G と，$root[G]$ の任意の子 M について，$\langle G, M\rangle$ を次のように決める：

> $G_{<M>}$ に含まれるメニューについてのみ縮約を行った，G に対する縮約リンク構造のうち，平均アクセス時間が最小のもの．

補題 14より，$root[G]$ の子の集合を $\{M_1, M_2, \ldots, M_k\}$ としたとき，

$$《G》 = \langle \cdots \langle \langle G, M_1\rangle, M_2\rangle, \cdots, M_k\rangle$$

が成り立つ．

3.3.3 アルゴリズム

最適リンク構造を求めるアルゴリズム OPTLS の動作を説明する．アルゴリズムの詳細を，図 3.4に示す．

OPTLS は次の手続きから構成されている．

- OPTLS　最適リンク構造を求める．
- OPTSUB　部分リンク構造 G を入力とし，$《G》$ を求める．実際は，$《G》$ の各メニューに縮約されるメニューの集合を求める．

```
/* グローバル変数 */
Rdc: array[ menu ] of set of vertex;
g: array[ menu ] of real;

OPTLS(in G: linkStructure; out G_opt: linkStructure)
/* (入力) G:    seed リンク構造,
   (出力) G_opt: 最適リンク構造. */

1  for each M ∈ キーワードの集合
2     do OPTSUB(G_<M>)
3  RESTR(G, G_opt)

OPTSUB(in G: linkStructure)

1  r ← root[G]
2  if optimized[r] /* 最適化が済んでいる */
3     then return
4  optimized[r] ← TRUE
5  Rdc[r] ← ∅
6  g[r] ← 0
7  if G の高さが 0
8     then return
9  for each M ∈ child[r]
10    do OPTSUB(G_<M>)
11       if M が葉でない
12          then FINDR(G, M, R, d)
13               Rdc[r] ← Rdc[r] ∪ R
14               g[r] ← g[r] + d

FINDR(in G: linkStructure; in M: menu; out R: set of menu; out d: real)
/* R: G_<M> のうち ⟪G⟫ の根に縮約されるメニューの集合,
   d: ET[⟨G, M⟩] − ET[G]. */

1  R_1 ← {M}
2  d_1 ← c(|M| − 1)/2 − t(M)P(M|root[G])
   /* M を根に縮約したときの ET[G] の変化量 */
3  for each M' ∈ child[M]
4     do if M' が葉でない
5           then FINDR(G, M', R', d')
6                R_1 ← R_1 ∪ R'
7                d_1 ← d_1 + d'
   /* M を縮約すると仮定したとき，⟪G⟫ の根に
   縮約されるメニューと，G と ⟨G, M⟩ の ET の差*/
8  if d_1 < g[M]P(M|root[G])
        /* G_<M> を最適化した場合と比較 */
9     then R ← R_1
10       d ← d_1
11    else R ← ∅
12       d ← g[M]P(M|root[G])

RESTR(in G: linkStructure; out G_opt: linkStructure)
/* (入力) G:    seed リンク構造,
   (出力) G_opt: 最適リンク構造. */

1  G_opt ← G    /* 複写 */
2  for each M ∈ G のメニュー集合
3     do for each M' ∈ Rdc[M]
4           do G_opt の辺 (M, M') を削除
5              for each M'' ∈ child[M']
6                 do G_opt に辺 (M, M'') を追加
```

図 3.4: アルゴリズム

- FINDR　部分リンク構造 G について，$root[\langle\langle G \rangle\rangle]$ に縮約されるメニューの集合 Rを求める．

- RESTR　OPTSUB で求めた結果を基に，縮約操作を行って最適リンク構造を構成する．

OPTLS の入力である seed リンク構造を G とする．OPTLS はまず，各入り口 Mについて，$G_{<M>}$に対し OPTSUB を実行する．その後，RESTR によって，OPTSUB の結果を基に最適リンク構造を構成する．

OPTSUB は，$G_{<M>}$の各メニュー M'について $Rdc[M']$ を求め，グローバル変数に代入する．これは，2.3節の OPT と同様のサブルーチンである．ただし，リンク構造では，メニュー Mが複数の親を持つとき，$G_{<M>}$に対して複数回 OPTSUB が呼ばれる．そこで計算量を減らすため，*optimized*という変数を使って，複数の親を持つ Mに対しても $Rdc[M]$ を一度しか計算しないようにしている．また，ページ（葉である頂点）Mに対して FINDR(G, M, R, d) を呼ばないようになっている．定義から Mに入る辺は縮約できないが，Mに対して FINDR が呼ばれないため，$Rdc[root[G]]$ に Mが含まれないことが保証されている．

FINDR は，2.3節の FINDR と同様のサブルーチンである．OPTSUB と同様，ページ Mに対して FINDR(G, M, R, d) を再帰的に呼ばないようになっている．

3.3.4 アルゴリズムの正当性

補題 15 （*FINDR* の正当性）　部分リンク構造 G の根以外のすべての頂点 q について，$g[q] = ET[\langle\langle G_{<q>} \rangle\rangle] - ET[G_{<q>}]$ が正しく求まっているとする．

FINDR に，G と，$root[G]$ の任意の子 Mを入力したとする．このとき，*FINDR* が出力する Rは，$\langle G, M \rangle$ の根に縮約されるメニューの集合に等しい．また，*FINDR* が出力する d は，$d = ET[\langle G, M \rangle] - ET[G]$ を満たす．

(証明)　補題 4と同様．　□

補題 16 （*OPTSUB* の正当性）　*OPTSUB* に部分リンク構造 G を入力したとする．このとき，G の任意のメニュー Mについて，*OPTSUB* が求めた $Rdc[M]$ は，$\langle\langle G_{<M>} \rangle\rangle$ の根に縮約されるメニューの集合に等しい．

(証明)　補題 5と同様．　□

定理 17 アルゴリズム *OPTLS* は，与えられた *seed* リンク構造 G から得られる縮約リンク構造のうち平均アクセス時間が最小のリンク構造を返す．

(証明)　補題 16と定理 13より題意が言える．　□

3.3.5 アルゴリズムの評価

seed リンク構造のメニューの数を n とする．

補題 18 *OPTLS* の最悪時間計算量は $O(n^3)$ である．

(証明)　部分リンク構造 $G_{<M>}$ について OPTSUB が呼ばれたとき，

- OPTSUB は再帰的に $G_{<M>}$ のメニュー数だけ呼ばれる．これは $O(n)$ 個ある．
- OPTSUB から FINDR が呼び出されたとき，FINDR は再帰的に $G_{<M'>}$（M' は M の子）のメニュー数だけ呼ばれる．各 M' について $G_{<M'>}$ のメニューは重複していないので，OPTSUB が 1 回呼ばれたとき FINDR は $O(n)$ 回呼ばれる．

また，RESTR の計算量は $O(n^3)$ である．
よって，OPTLS の最悪時間計算量は $O(n^3)$ となる．　□

定理 19 最適リンク構造問題は $O(n^3)$ で計算できる．　□

3.4 議　論

3.4.1 更新問題

この節では，最適リンク構造を求めた後に新しくメニューやページを追加した際に，再び seed リンク構造にアルゴリズム OPTLS を適用するより少ない手間で新しい最適リンク構造を求める方法を考察する．

カテゴリの追加

ハイパーテキストに，キーワードとカテゴリの対を新しく追加することを考える（図 3.5）．図 3.5では，$B \subseteq M \subseteq A$ であるようなカテゴリ M を追加した．すなわち，seed 階層構造にメニュー M を追加し，メニュー A から B および M の要素であるページへの辺を M からの辺に替え，A から M への辺を追加した．

このとき，アクセス重み $P(M)$ は，M の子メニューのアクセス重みから決まる．M 以外のメニューのアクセス重みは変化しない．キーワードが追加されることにより，各ページ中のキーワードの数すなわち定義 18の $|key[q]|$ の値が変化する．しかし，定理 13より，$|key[q]|$ の値が変化しても最適リンク構造は変化しない．ただし，平均アクセス時間の値は変化する．

このときの最適リンク構造の再計算について考える．追加したメニューを M とし，新しい seed リンク構造を G とする．まず，M を含まない部分リンク構造に対する最適部分リンク構造は変化しない．すなわち，図 3.5の $G_{<B>}$ や $G_{<C>}$ に対する最適部分リンク構造は，M を追加する前と同じである．

次に，M の任意の祖先 A について，$Rdc[A]$ を求めることを考える．これは，OPTSUB($G_{<A>}$) によって求まる．このとき，M の任意の子孫 B について，OPTSUB($G_{<B>}$) の結果は M を追加する前と変わらないので，これを実行する必要はない．同様に，M の子孫でも先祖でもないメニュー C についても，OPTSUB($G_{<C>}$) および FINDR($G_{<A>}, C, R, d$) を実行する必要はない．また，FINDR($G_{<A>}, B, R, d$) について，これも M を追加する前と結果が変わらない．なぜなら，B を A に縮約するためには M を A に縮約しなければならないが，G に対し M を A に縮約したリンク構造は，M を追加する前の seed リンク構造に等しいからである．

以上をまとめると，新しい最適リンク構造の計算は次の手順で出来る．すなわち，M を含まないすべての部分リンク構造 $G_{<B>}$ について $Rdc[B]$ と $g[B]$ が求まっている状態から始めて，M の祖先である入り口 A_i について OPTSUB($G_{<A_i>}$) を実行する．ただし，FINDR($G_{<A>}, M', R, d$) は，A の子孫かつ M の先祖または M 自身であるメニュー M' についてのみ実行すればよい．

この再計算の手間は，M の祖先の数が $O(n)$ であることから，補題 18の証明と同様に，$O(n^2)$ 時間となる．

ページの追加

seed リンク構造に新しいページ q を追加することを考える（図 3.6）．図 3.6では，カテゴリ M に属するページ q を追加した．すなわち，seed 階層構造に，ページ q とメニュー M から q への辺を追加した．アクセス重み $P(q)$ は設計者が与える．

このとき，q の祖先であるメニューのアクセス重みは，祖先でないメニューより相対的に重みが増す．よって，q の祖先であるメニュー A について，その子孫同士のアクセス重みの比が変わることから，$Rdc[A]$ が変化し得る．また，それ以外のメニューについて Rdc は変化しない．

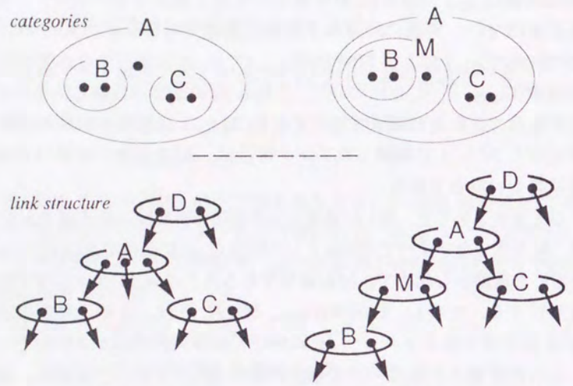
以上より，seed リンク構造 G にページ q を追加したとき，q の先祖であるメニュー M すべてについて OPTSUB($G_{<M>}$) を実行すれば，変化し得る Rdc がすべて求まる．この再計算の手間は，q の祖先の数が $O(n)$ であることから，補題 18の証明と同様に，$O(n^2)$ 時間となる．

3.4.2 枝分かれ合流のあるリンク構造

以上では seed リンク構造として枝分かれ合流のないものに制限した．もしこの制限がないと次のような問題が生じる．

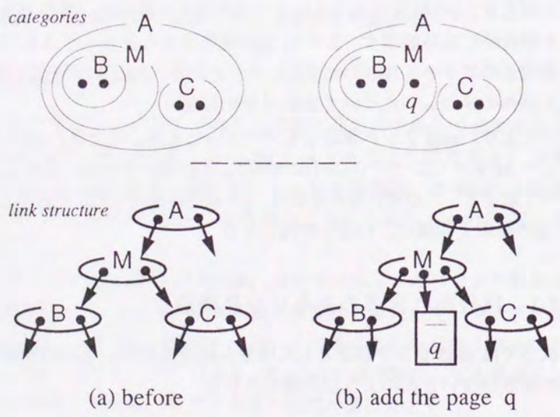
(1) 条件付き重み $P(D|A)$ の問題

A がアクセスされた場合に孫メニュー D がアクセスされる確率が，途中 B を通るか C を通るかで変わる（図 3.7(a)）．



(a) before (b) add the menu M

図 3.5: カテゴリの追加



(a) before (b) add the page q

図 3.6: ページの追加

縮約したときのコスト変化がパスによって変わる．よって縮約するかどうか，パスの数だけ調べないといけない．このときパスの数は一般に指数オーダとなる．

この問題は，アクセス確率の定義 (2) を使えば解消できる．

補題 20 アクセス確率の定義 *(2)* を用いた場合，メニュー M を根に縮約したときのコストの変化は，根から M までのパス上のどのメニューが根に縮約されているかによらない．

ただし，M が根に縮約可能であるためには，根から M までのパスのうちパス上のメニューがすべて根に縮約されているようなものが 1 個以上必要である．

(証明)　平均アクセス時間の定義より M を縮約した際のコストの変化は $|M|$ と $P(M|root)$ にのみ依存．□

(2) エイリアスの問題

縮約の結果発生した，同じ子メニューへの複数の辺をエイリアスという（図 3.7(b)）．

エイリアスを除去する場合，子メニューを縮約したときのコスト変化が，親メニューの縮約状況に依存する．よって補題 14が成り立たなくなりアルゴリズム OHTLS は正しくなくなる．逆にエイリアスを除去しない場合，最悪指数個の辺が発生し，出力の大きさが指数的になる．

枝分かれ合流の禁止という制限を緩めるため，リンク構造中の枝分かれ合流の数を定数個まで許すという方法が考えられる．定数個以下にした場合にはアルゴリズムの計算量は多項式時間に収まる．

3.4.3　通信コスト

WWW のように，各ページやメニューがネットワーク上に分散しておりそれぞれに対する通信コストにばらつきがある場合がある．この場合，メニューに対するアクセスコストのうち計算機の反応時間 r をメニューによって決まる関数 $r(M)$ とすることが考えられる．すなわち

$$t(M) = c(|M|+1)/2 + s + r(M)$$

と定義する．

この場合でも補題 14の証明の $ET[G_{<M_1>}]$ の変化は

$$\frac{c}{2}(|M_2|-1)(1-P(M_2|M_1)) - (c+s+r(M_2))P(M_2|M_1)$$

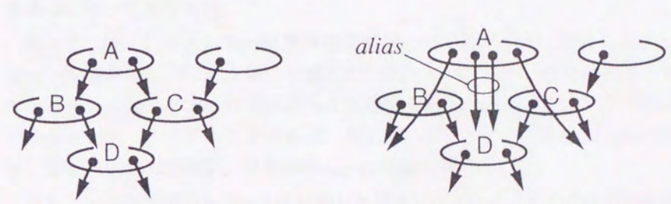
となって M_1 と M_2 によってのみ決まるので補題 14は成り立つ．よってこれまでの議論がそのまま有効となる．

3.5 第3章のまとめ

本章では，与えられたリンク構造について，キーワードからアクセスできるページとキーワードに対するカテゴリの対応を保つようなリンク構造すべての中から平均アクセス時間が最小なリンク構造を求める効率のよいアルゴリズムを示した．

なお，本章の最適リンク構造の定義は，要素が1個しかないカテゴリに対しても必ずメニュー頂点をアクセスしてからページにアクセスするような構造を構成する．このようなカテゴリについては，本章のアルゴリズムで求めた最適リンク構造に対しアンカーからの辺を置き換えてそのようなメニューを除去することで，より操作しやすいハイパーテキスト構造が構築できる．この操作を行ってもそのアンカーに対するアクセス以外に影響はなく，リンク構造の最適性は失われない．

今後の課題として，ハイパーテキストの別のモデルやハイパーメディア，より一般的なモデルに対する構成法を考えることがある．



(a) (b) Reduce B and C.

図 3.7: 枝分かれ合流の問題

第4章

あとがき

本論文では，計算機アプリケーションのユーザインタフェースとしてよく用いられているメニュー階層構造やハイパーテキストについて，設計者が与えた項目の意味的分類を保ちながら，アクセスコストが最小な構造を構成する方法について考察した．

第2章では，最適メニュー階層構造問題について考察した．与えられたメニュー階層構造について，項目の意味的分類を保つメニュー階層構造すべての中から，平均アクセス時間が最小な階層構造を求める効率のよいアルゴリズムを示した．このアルゴリズムは，与えられたカテゴリの数を n としたとき，最悪時に $\Theta(n^2)$ 時間，平均 $O(n \log n)$ 時間で解を求める．

また，seed 階層構造を DAG に拡張した場合や，メニュー内での項目の順序を考慮した選択時間モデルを採用した場合について考察した．

第3章では，アクセスコストが最小なハイパーテキストの構成法について考察した．まず，多数の入り口を持つ DAG であるリンク構造と，その平均アクセス時間について，適切な定義づけを試みた．そして，与えられたリンク構造について，キーワードからアクセスできるページとキーワードに対するカテゴリの対応を保つようなリンク構造すべての中から平均アクセス時間が最小なリンク構造を求める，多項式時間のアルゴリズムを示した．このアルゴリズムは，与えられたカテゴリの数を n としたとき，$O(n^3)$ 時間で解を求める．

また，ハイパーテキストに頂点を追加した場合の更新問題，および，より拡張されたモデルを採用した場合について考察した．

今後の課題として，より現実に近い選択時間モデルに対する効率のよいアルゴリズムを求めることがある．本論文で扱ったメニュー項目の選択時間モデルは，選択に必要な時間は項目数に線形で，また項目の表示される順序は考慮していなかった．マウスなどのポイント装置による目標選択操作について，選択操作時間は Fitts の法則に従うことが経験的に示されている [3, 6, 10]．これにより，既にその階層構造に慣れたユーザの場合，選択に必要な時間は項目数の対数に比例することが推測される．本論文のアルゴリズムは，選択に必要な時間が項目数に線形でないモデルに拡張することが容易ではない．ま

た，2.4.2節で述べたように，項目の選択時間がメニュー内での順序の関数であるモデルに拡張することも容易ではない．

他に，第3章で扱ったものとは異なるハイパーテキストのモデルなど，さまざまなモデルに対する最適な構造の構成法を考えることも，今後の課題として挙げられる．

謝　辞

本研究の全過程を通じて懇切なる御助言と御指導を賜わりました都倉信樹教授に深く感謝の意を表します．

本研究の内容に関し適切なる御意見を戴きました首藤勝教授，橋本昭洋教授，西川清史教授に心より感謝致します．

平素より御討論，御助言を戴き，本研究の全過程を通じて御指導を戴きました辻野嘉宏助教授に深謝致します．

種々の面で御指導，御援助を戴きました齊藤明紀講師，安留誠吾助手，西田知博助手に感謝致します．

著者の在学中，御討論，ならびに研究の上で様々な御支援を戴きました亀田益代技官をはじめ都倉研究室の皆様に感謝致します．

文　献

[1] Callahan, J., Hopkins, D., Weiser M. and Shneiderman B.: An Empirical Comparison of Pie vs. Linear Menus, *Proc. CHI '88 Conference on Human Factors in Computing Systems*, pp. 95–100 (April 1998).

[2] Card, S. K., Moran, T. P., Newell, A:: The Keystroke-Level Model for User Performance Time with Interactive Systems, *Communications of the ACM*, Vol. 23, No. 7, pp. 396–410 (July 1980).

[3] Card, S. K., Moran, T. P. and Newell, A.: *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, London, pp. 23–45 (1983).

[4] Cormen, T. H., Leiserson, C. E. and Rivest, R. L.: *Introduction to Algorithms*, McGraw-Hill, New York, pp. 165–167 (1990).

[5] Fisher, D. L., Yungkurth, E. J. and Moss, S. M.: Optimal Menu Hierarchy Design: Syntax and Semantics, *Human Factors*, Vol. 32, No. 6, pp. 665–683 (1990).

[6] Fitts, P. M.: The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement, *J. of the Experimental Psychology*, Vol. 47, No. 6, pp. 381–391 (June 1954).

[7] ハラリイ, F.: グラフ理論, 池田貞雄訳, 共立出版, p. 15 (1971).

[8] 金子朝男: ハイパーメディアの研究動向, 情報処理学会誌, Vol. 34, No. 1, pp. 60–71 (1993-01).

[9] Li, W.: Random texts exhibit Zipf's-law-like word frequency distribution, *IEEE Trans. Information Theory*, Vol. 38, No. 6, pp. 1842–1845 (November 1992).

[10] MacKenzie, I. S. and Buxton, W. : Extending Fitts' law to two-dimensional tasks, *Proc. CHI'92 Conference on Human Factors in Computing Systems*, pp. 219–226 (May 1992).

[11] McDonald, J. E., Stone, J. D. and Liebelt, L. S.: Searching for Items in Menus: The Effects of Organization and Type of Target, *Proc. of the Human Factors Society 27th Annual Meeting 1983*, pp. 834–837 (1983).

[12] Mitchell, J. and Shneiderman, B.: Dynamic versus Static Menus: an Exploratory Comparison, SIGCHI Bulletin, Vol. 20, No. 4, pp. 33–37 (April 1989).

[13] Nielsen, J.: The Art of Navigating through Hypertext, *Communications of the ACM*, Vol. 33, No. 3, pp. 296–310 (March 1990).

[14] 高田喜朗, 辻野嘉宏, 都倉信樹: 最適なメニュー階層構造を求めるアルゴリズム, 情報処理学会研究報告, Vol. 93, No. 88, pp. 11–18 (1993-10).

[15] Walker, N. and Smelcer, J. B.: A Comparison of Selection Times from Walking and Pull-Down Menus, *Proc. of the CHI '90 Conference on Human Factors in Computing Systems*, pp. 221–225 (April 1990).

[16] 山本康友, 魚井宏高, 辻野嘉宏, 都倉信樹: 横配置型メニュー方式の提案とその評価, 電子情報通信学会論文誌, Vol. J74-D-II, No. 12, pp. 1748–1755 (1991-12).

[17] 山本康友, 魚井宏高, 辻野嘉宏, 都倉信樹: リモートプルダウンメニュー方式の提案とその評価, 電子情報通信学会論文誌, Vol. J76-D-I, No. 7, pp. 364–370 (1993-07).