情 報 学 基 礎 26－2
（1992．7．14）

参照関係を用いた大規模マニュアルの自動構造化

佐古 慎二　原 良憲　千村 浩靖
日本電気(株) C&C情報研究所

本報告では、大規模マニュアルがモジュールという小単位の集合で構成されており、モジュール間に参照関係があることに着目し、マニュアルを自動的に構造化する手法について述べる。
また、本手法はドキュメンテーションエンジニアリングの一手法のみならず、ハイパーテキスト的構造のアグリゲーションとも関係が深い。本手法で用いたアルゴリズムの特性を評価するとともに、構造の自動抽出という意味での自己組織化との関係についても言及する。

Structurizing for Large-scale Manual Using Cross References.

Shinji Sako , Yoshinori Hara and Hiroyasu Chimura

C&C Information Technology Research Labs.,
NEC Corporation

4-1-1, Miyazaki, Miyamae-ku, Kawasaki,
Kanagawa 216, Japan

This paper describes the structurizing method for large-scale manuals that consist of the set of the modules and have a reference structure. This method is related to the documentation engineering as well as the aggregation of hypertext structures. We evaluate the characteristics of the algorithm of proposed method. And we refer to the relationship between the method and self-organization.

1 はじめに

近年、マニュアルの品質改善・作成効率化に対する要望が高まっている。筆者らはこの問題に対処すべく、マニュアルの作成・管理の支援を目的とするドキュメンテーションエンジニアリングの研究を行なっている [1][2][3]。

総ページ数が数万にも及ぶような大規模マニュアルでは、全体をどのように分割し、分冊を構成するかが重要な問題となる。しかし、このような大規模マニュアルの構成を把握するのはマニュアルの利用者はもとより、製作者にも困難であるため、使いやすいマニュアル構成の基準および構成手法が必要となる。

本報告では、大規模マニュアルがモジュールという小単位の集合で構成されており、モジュール間に参照関係があることに着目し、マニュアルを自動的に構造化する手法について述べる。また、本手法はドキュメンテーションエンジニアリングの一手法のみならず、ハイパーテキスト的構造のアグリゲーション [4,5,6] とも関係が深い。アルゴリズム面での特性を評価するとともに、構造の自動抽出という意味での自己組織化との関係についても言及する。

2 大規模マニュアルの特徴と構成の要件

2.1 大規模マニュアルの特徴

本論で対象としている大規模マニュアルは次のような特徴をもつ。

2.1.1 リファレンス型のマニュアルである

対象を理解するために読むチュートリアル型のマニュアルと異なり、対象の個々の機能や状態について何か知りたいことがあったとき、それに関するトピック（モジュール）を利用者が自分で探して読むためのマニュアルである。例えば、故障が起きたときに原因や対処法を見つけるための保守マニュアルがその例である。具体的には局用電子交換機 NEAX61E の保守マニュアルを想定している。

2.1.2 モジュール化されている

ここでいうモジュールとは 5 ～ 10 ページから成る説明単位で、一つのトピックが一つのモジュールに対応している。NEAX61E の保守マニュアルの場合、これが約 2000 個ある。

2.1.3 モジュール間には出現順序を決定する依存関係がない

チュートリアル型のマニュアルでは、項目（モジュール）を読み進んで行くといった読み方をするために、上位概念のモジュールが下位の前にあるほうがわかりやすいのでモジュールの順序が規定されるが、リファレンス型のマニュアルでは必要なモジュールから読むので、順序は重要ではない。

2.1.4 あるモジュールが別のモジュールを参照していることがある

利用者の知りたい事柄によっては、一つのモジュールに記述してある内容だけでは不十分で、さらに多くの情報が必要となることもある。このため、関連する項目として参照するモジュールを記述しておく。利用者が最初に選んだモジュールで必要な情報が得られなかった場合には、このモジュールの参照先から真に必要な情報を見つけ出すことができる。

2.1.5 多くの分冊で構成される

例えば、NEAX61E では 2000 モジュールもの分量があるため、1 冊あたり平均 50 個のモジュールを入れるとしても 40 分冊必要となる。このため、参照先のモジュールが別の分冊になることも多く生じる。

2.1.6 複数の人間により作成される

マニュアルの対象となる装置・システム自体が複数の人間により作成されるので、大規模マニュアルの作成も複数の人間の共同作業になる。

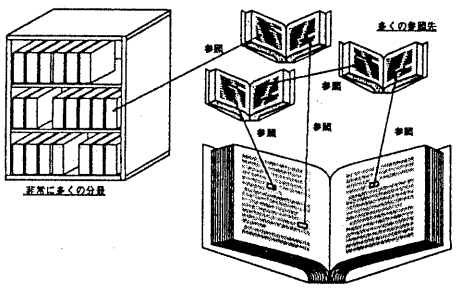

図 1: 大規模マニュアル

2.2 大規模マニュアル構成時の要件

上記の特徴をもとに大規模マニュアルの要件をまとめると以下の様になる。

2.2.1 分冊を構成するモジュールの選択指針

あるモジュールを読んでいるとき、欲しい情報がない場合、あるいは十分でない場合には、そのモジュールが参照しているモジュールを読む必要が生じる。このとき、分冊を構成するモジュールがうまく選択されていないと、参照先のモジュールを含む分冊を広げなければならない。このようなことが頻繁に起こると、利用者にとって大きな負担となり、分冊を読むための

場所も必要で、マニュアル参照が効率良く行なえない。

これを避けるためには、参照関係で結ばれたモジュール同士はなるべく同じ分冊内に収める必要がある。一方、一冊のページ数には物理的・実用的限界があるので、参照関係のあるモジュールすべてを一つの分冊に入れることはできない。そこで、一つの分冊のページ数の拘束を満たしつつ、一つの分冊から他の分冊への参照関係がなるべく少なくなるようにモジュールを分冊に分配する手法が必要となる。

2.2.2 分冊中のモジュールの順序

関連するモジュールが同じ分冊にあっても、モジュール同士が分冊中で連続していないと、関連するモジュールの位置を憶えておいたり、しおりを用いたりする必要がある。また、二つのモジュールを見るために頻繁にページをめくらなければならなくなる。

この問題を解決するためには、一つの分冊の中で、関係の深いモジュール同士が隣接しているように分冊を配置する手法が必要となる。

2.2.3 キャビネットでの分冊の配置

マニュアルの分冊を収納するキャビネットの中で、関係の深い分冊同士が離れていると、関係する分冊を取りに行くためには多くの距離を歩かなければならず、不便である。また、関連する分冊がまとまっていると、キャビネット中の分冊の位置を憶えるのも容易になるので、関係の深い分冊同士が近くになる様に分冊を配置する手法が必要となる。

3 大規模マニュアル構造化手法

従来は上述した要件はマニュアル製作者が考慮していたが、モジュールが多数あるため、満足のいく構造化を行なうことはできなかった。そこで今回、参照関係を十分考慮した構造化を計算機により行なう手法を提案し、実験・評価を行う。

3.1 基本的な考え方

2.2 で述べたように、構造化を行なう際の目標は、「関係の深い（参照関係のある）モジュール同士が近く（同じ分冊）にあること」である。そのために、参照関係のあるモジュール同士を集め、そうでないモジュールから分けなければならない。つまり、構造化はモジュールをクラスタリングする問題で、クラスタリングの基準は、クラスタ間の参照関係を最小にすることである。

いま、モジュールが上の基準とは無関係にクラスタリングされているとき、これを修正して上の基準を満たすクラスタにする方法を考える（図 2）。クラスタ A のモジュール m_1 がクラスタ B のモジュールと参照関係があり、クラスタ A の他のどのモジュールとも参照関係がないときには、m_1 はクラスタ A よりクラスタ B にあるほうが望ましい。また、クラスタ B のモジュール m_2 がクラスタ B に含まれる他のどのモジュールとも参照関係がないならば、クラスタ B から m_2 を取り除いて m_1 をクラスタ B に加えたほうがいい。このことは、モジュール間の参照関係によって決まる力が存在し、クラスタ B のモジュールと m_1 の間に引き付け合う力が、また、クラスタ B のモジュールと m_2 の間に反発する力が働いている、と考えられる。

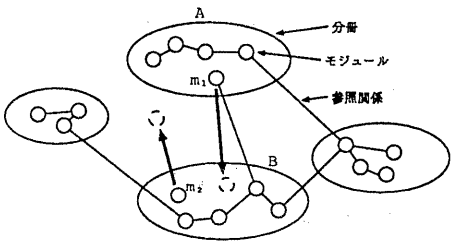

図 2: モジュールのクラスタリング

一方、「一冊の分冊に含まれるモジュールの数（ページ数）は有限」という制約条件があるので、参照関係の有無に関わらず、すべてのモジュール間に斥力が働くこととし、参照関係のあるモジュール同士が引力によって一つのクラスタに集まり過ぎることを防ぐ。一方、モジュールすべてによって一つのマニュアルを構成するのだから、参照関係のないモジュールが斥力によって無限に離れてしまわないように（参照関係によるものよりはかなり小さい）引力を与える。結局、2 つのモジュール間に働く力は、

$$\begin{cases} \cdot \text{一定の斥力} \\ \cdot \text{参照関係に依存する引力} \end{cases}$$

の和になる。

このような力をモジュール間に定義し、すべてのモジュールに働く力が 0 となる状態を求めると、参照関係があるモジュール同士は集まり、そうでないものは離れている。このとき、直接参照関係がないモジュール同士（m_a, m_b）でも、共通に参照関係のある別のモジュール（m_c）があるときには、m_a, m_b には（直接の場合よりは弱い）関係があると考えられるが、このことは、m_c が m_a, m_b を引き付け、m_a, m_b の距離が近付くことにより表現される。（図 3-(1)）。逆に、参照関係があるモジュール同士（m_d, m_e）でも、m_d と他のモジュールとの参照関係の方が強ければ、m_e は他のモジュールからの斥力により、m_d との間の本来の（他のモジュールの影響を受けないときの）距離よりも離される（図 3-(2)）。

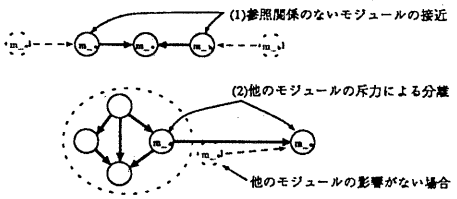

図 3: モジュール間の距離

この状態で距離が近いモジュール同士をまとめて一つの分冊とすれば、参照関係の強いモジュールは同じ分冊になるので、他の分冊への参照が少ない分冊化が実現できる。

上述した概念に基づいた二つのステップからなる構造化の手法について、次節以下で述べる。

3.2 モジュールの安定な位置
– 参照構造の空間的表現 –

3.1の考え方に基づき、モジュール間(距離 r)に働く力の大きさ(F)を、次式で定義する。

$$F = G/r^2 - L$$

ここで、

$$G: \text{斥力定数} = 1$$

$$L: \text{引力定数} = \begin{cases} 0.05 & : \text{参照関係なし} \quad (1) \\ 1 & : \text{1方向の参照関係} \quad (2) \\ 2 & : \text{相互参照関係} \quad (3) \end{cases}$$

Lの値は経験的に決めたもので、定性的に(3) > (2) >> (1)となるように設定している。

二つのモジュール間の安定点($F = 0$の点)は$r = \sqrt{G/L}$である。求めたいのはモジュール間の相対的な距離の比なので、$G = 1$とする。これにより、二つのモジュール間の安定時の距離は、(1)-$\sqrt{20}$, (2)-1, (3)-$\sqrt{0.5}$となる。

3.2.1 力学的シミュレーション

モジュール間に働く力は厳密に定義されているが、対象とするマニュアルのモジュール数が多いので、代数的にすべてのモジュールに働く力が0となる点(安定点)を求めるのは非常に難しい。そこで、参照関係を物理的な力として表したように、モジュールを3次元空間上の実体(質点)と考え、安定点までに至る過程をモジュールの物理的な運動としてシミュレートすることにより、安定点を求める。

モジュールに働く力は、他のモジュールとの間に働く力の総和である。モジュール$p, q (p \neq q)$間に働く力$\mathbf{F}_{pq}$は、モジュールp, qの位置を$\mathbf{P}_p, \mathbf{P}_q$とすると、

$$\mathbf{F}_{pq} = (1/r^2 - L) \cdot \frac{\mathbf{P}_p - \mathbf{P}_q}{r}, \quad r = |\mathbf{P}_p - \mathbf{P}_q|$$

で表され、モジュールiに働く力は、

$$\mathbf{F}_i = \sum_{k \neq i} \mathbf{F}_{ik}$$

で求められる。$\mathbf{F}_{pq} = -\mathbf{F}_{qp}$であるから、すべての$\mathbf{F}_i$をもとめるには、モジュールの個数を$n$とすると、$n(n-1)/2$回の計算が必要となる。

しかしモジュールに働く力がこれだけだと、モジュール間の力によるポテンシャルエネルギーと運動エネルギーとの変換が繰り返されるだけで、モジュールは振動して安定しない。そこで、運動を減衰させるために、運動中のモジュールiに速度に比例する抵抗が働くものとする。速度を$\mathbf{V}_i$とすると抵抗$\mathbf{fl}_i$は、

$$\mathbf{fl}_i = k\mathbf{V}_i, \qquad k < 0$$

となる。

モジュールiの時間Δ後の新しい位置$\mathbf{P}'_i$, 速度$\mathbf{V}'_i$は、$\mathbf{V}_i, \mathbf{F}_i$が$\Delta$の間一定と近似して、

$$\begin{aligned} \mathbf{P}'_i &= \mathbf{P}_i + \mathbf{V}_i \cdot \Delta, \qquad \Delta: \text{微小時間} \\ \mathbf{V}'_i &= \mathbf{V}_i + (\mathbf{F}_i + \mathbf{fl}_i) \cdot \Delta \end{aligned}$$

となる。$\mathbf{P}'_i$を新しい$\mathbf{P}_i$として、この処理を

$$\max_i |\mathbf{F}_i| \approx 0$$

となるまで繰り返すことによりモジュールの安定点を求めることができる。

3.2.2 ニュートン法による安定点の計算

前節では、モジュールの安定点を求めるために、運動をシミュレートしたが、必要なのは安定点で、途中の運動の状態は記述する必要はない。そこで、$\mathbf{F} = 0$となる安定点だけを求めるために、ニュートン法による近似解法を用いる。

ニュートン法では、ある点x_iでの微分係数$f'(x_i)$をもとに、

$$x_{i+1} = x_i - f(x_i)/f'(x_i)$$

を繰り返し、$f(x) = 0$となるxを近似的に求める。これに対して、モジュールpに働く力$\mathbf{F}_p$は3次元空間上の点に対するものであるから、

$$\mathbf{F}_p(x,y,z) = \begin{pmatrix} f_p^x(x,y,z) \\ f_p^y(x,y,z) \\ f_p^z(x,y,z) \end{pmatrix}$$

と表され、偏微分係数

$$\mathbf{F}' = \begin{pmatrix} \frac{\partial f_p^x}{\partial x} & \frac{\partial f_p^x}{\partial y} & \frac{\partial f_p^x}{\partial z} \\ \frac{\partial f_p^y}{\partial x} & \frac{\partial f_p^y}{\partial y} & \frac{\partial f_p^y}{\partial z} \\ \frac{\partial f_p^z}{\partial x} & \frac{\partial f_p^z}{\partial y} & \frac{\partial f_p^z}{\partial z} \end{pmatrix}$$

をすべて求めなければならない。しかし、モジュール間に働く力の方向は、モジュールを結ぶ方向であり、

変位した方向以外の方向の力の変化は小さいので、対角成分のみを考える。よって、

$$\left.\begin{aligned} x_{p,i+1} &= x_{p,i} - 1/2\frac{f_p^x(x_{p,i},)}{\frac{\partial f_p^x}{\partial x}} \\ y_{p,i+1} &= y_{p,i} - 1/2\frac{f_p^y(,y_{p,i},)}{\frac{\partial f_p^y}{\partial y}} \\ z_{p,i+1} &= z_{p,i} - 1/2\frac{f_p^z(,,z_{p,i})}{\frac{\partial f_p^z}{\partial z}} \end{aligned}\right\} \cdots (*)$$

とする。ここで、係数1/2は、安定点に行くためにモジュールpが1/2、他のモジュールが1/2動くことを意味している。モジュール間の相対的な位置だけが必要なので、このようにモジュールの変位を分担しても問題はない。

モジュールp,qに働く力のx成分F_{pq}^xは、前節の式から、

$$F_{pq}^x = (G/r^2 - L)\cdot\frac{x - x_q}{r},\quad r = \mid \mathbf{P}_p - \mathbf{P}_q \mid \quad x_q : \mathbf{P}_q \text{ の } x \text{ 成分}$$

となり、モジュールpに働く力のx成分F_p^xは、

$$F_p^x = \sum_{k \neq p} F_{pk}^x$$

となる。同様にしてy成分、z成分を求め、それぞれx,y,zで偏微分し、(*)式にあてはめる。この式を使い、

$$\max_i \mid \mathbf{F}_i \mid \approx 0$$

となるまで$x_{p,i}, y_{p,i}, z_{p,i}$を求めることを繰り返す。

この手法を用いることにより、近くに来るべきモジュールが遠く離れているとき、つまり微分係数があまり変化しないときの、モジュールの移動量を大きくできるので、3.2.1の方法よりはやく安定点に近付けることができる。

ここで、一つのモジュールに働く力は二つのモジュール間だけに働く力の場合と異なり、極大や極小の部分が存在するため、この付近では偏微分係数が小さいため、1ステップごとの変位が大きくなりすぎて、逆に安定点から遠ざかってしまう。そこで、実際の処理を行なうときは、偏微分係数が小さいモジュールの変位の大きさを一定の値以下に制限している。

3.3 分冊化
－モジュールのクラスタリング－

モジュールが安定な配置になったものとして、空間上での距離の近いモジュール同士を集めて一つの分冊を作る手順を以下に述べる。

クラスタリング（分冊化）の考え方は以下の通りである。

距離の近いモジュール同士を核として、それに近いモジュールを次々に付け加えて、モジュールのグループを作る。二つのグループがあって、一方のグループの中のモジュールが他方のグループのモジュールと近いときには、グループ同士がくっついて一つのグループになるものとする。このグループが一分冊を構成できる大きさになったときに、これを分冊とする。

3.3.1 クラスタリングアルゴリズム（図4）

1. 安定した状態での全てのモジュール間の距離を求め、モジュールの組を距離が近い順にソートし、スタックにいれる。

2. スタックの上からモジュールの組を取り出し、このモジュールと作業スペース上のクラスタのモジュールと共通するものがあるかを調べる。

   - ない場合－この組を作業スペース上のクラスタとする。
   - この組のうちどちらか一方または両方ともが含まれているクラスタが一つある場合－このクラスタに加える。
   - この組の一方があるクラスタに、他方が別のクラスタにある場合－二つのクラスタを一つにする。

   ただし、モジュールをクラスタに加えたり、クラスタを融合したとき、そのクラスタの大きさが分冊の大きさの限界を越えるときには、以上の操作は行なわない。

3. 作業スペース上にすべてのモジュールが取り出されたら、（すべてのモジュール対という意味ではなく、スタックにモジュール対はまだ残っているが、すべてのモジュールがどこかのクラスタに属している状態）、作業スペース上の2つのクラスタの組み合わせのなかで、一分冊のページ数の制限を越えないものがあるかどうかを調べる。そのような組み合わせがあれば2を繰り返す。

   すべてのモジュール対がスタックされているのでクラスタは融合を繰り返して大きくなっていく（ページ数の制限がなければすべてのモジュールを含む一つのクラスタになる）。そして、どのクラスタを組み合わせても一分冊のページ数の制限を越えるようになる、つまりすべてのクラスタが一分冊として独立できる大きさになったら分冊化は終了する。

4 実験・評価

電子交換機NEAX61Eのマニュアルのモジュール488個について実験を行ない、提案した手法の有効性を検討した。このマニュアルのモジュール間の参照数は412、分冊外への参照数は256である。

モジュールの初期位置を、まず図5,6のように格子状配列し、この状態から3.2.2で述べた方法にしたがって

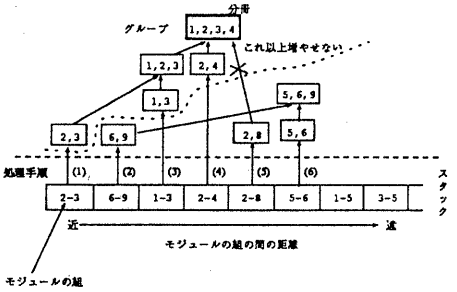

図 4: 分冊化のアルゴリズム

モジュールを動かし、安定状態に達したのが図 7,8である。図では、3次元空間を $x-y, x-z$ の二つの平面に分けて示している。

安定状態から 3.3の方法にしたがって分冊をやり直した結果、人手によって分冊を行なったときの分冊外への参照数 256 が、82 に減少し、手法の効果が認められた。

5 他の手法との比較

以上に述べた手法を採用するにあたって、その他に検討した手法とその特徴、および提案した手法との比較について述べる。

5.1 全探査

モジュールを分冊に配分するすべての方法を生成し、それぞれの分冊結果に対して他の分冊への参照の数を計算し、この数が最小のものを選ぶ。この手法では外部参照の数を最小にする分冊方法が必ず求められる。

一つの分冊に含まれるモジュールの数がすべて同じである場合を例にとって計算量を見積もる。モジュールの個数を n、一分冊に含まれるモジュールの個数を v、分冊の数を p とすれば分冊の方法の数は、

$$ {}_nC_v \times {}_{n-v}C_v \times \ldots \times {}_{n-(p-1)v}C_v \div p! = \frac{n!}{v!^p \cdot p!} $$

となる。ここで $n = 2000, p = 40, v = 50(= 2000/40)$ とすると、およそ 10^{3108} になる。

したがって、この手法はモジュールの数が少ないときにしか使えない。

5.2 Kernighan-Lin の方法 [7]

$2n$ 個のノード持つ無向グラフを、cut edge が最少になる様に n 個ずつの二つのサブグラフに分割する方法である。

Kerninghan-Lin のアルゴリズムを以下に示す。

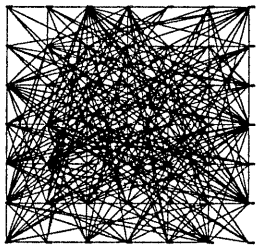

図 5: 初期状態 $x-y$ 平面

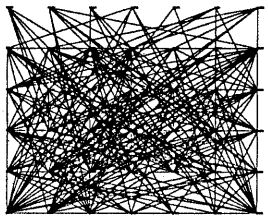

図 6: 初期状態 $x-z$ 平面

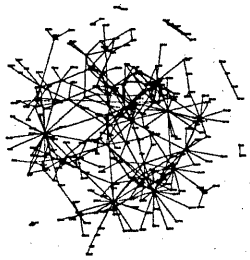

図 7: 安定状態 $x-y$ 平面

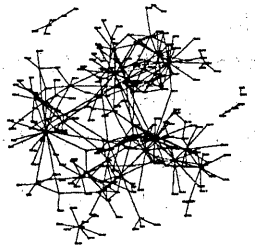

図 8: 安定状態 $x-z$ 平面

(1) 全ノードを n 個ずつのサブセット S と T に任意に分割する;
repeat
(2) for i:=1 to n/2 do begin
(3) 要素の交換が cut の最大の減少または最少の増加になるような未選択のノードの対 (S の u_i と T の v_i) を見つける;
(4) $cost_i := cost$ の変化分;
(5) u_i と v_i を選択する
end;
(6) $\sum_{j=1}^{l} cost_i (0 \leq l < n/2)$ が最大となる l を見つける;
(7) $u_1, \cdots, u_l$ を S から T に移し、$v_1, \cdots, v_i$ を T から S に移す
{ もし、$l = 0$ なら何もしない }
(8) until $l = 0$

このアルゴリズムを分冊化に使うためには、m 個のサブセットに分割するように拡張する必要がある。この場合、(3) で S, T からひとつずつノードを選択していたのを、m 個のサブセットから 2 つを選択し、さらにその中からノードを選択することになる。

この方法での (2) ～ (5) の繰り返しを行う回数は $n/2$ であり、(3) でのノードの選択の回数は、モジュール数: n 個の場合、

$$n = m \times b$$

$$\begin{cases} m & \text{分冊数} \\ b & \text{1分冊のモジュール数} \end{cases}$$

とすると、

$$\begin{array}{lll} b & \times & b(m-1) \\ & + & \\ b & \times & b(m-2) \\ & + & \\ & \vdots & \\ b & \times & b \end{array} = \frac{b^2(m^2-m)}{2} = \frac{n^2 - b^2 m}{2}$$

となって、(2) ～ (7) を行うときの計算量のオーダは n^3 になる。(1) ～ (8) の繰り返しの回数はグラフの性質に依存するので一概にはいえないが、仮に n に依存せずに一定だとしても、全体のオーダは n^3 になる。

また、この方法では一分冊のモジュール数が一定であることを前提にしているが、ある分冊のモジュール数が少ない方が参照数が減る場合や、一つのモジュールの大きさのばらつきにより一つの分冊に含むことができるモジュールの数が変わる場合があるので、実際にはさらに多くの場合を考える必要がある。

5.3 本手法との比較

以下に、いくつかのデータに対してグラフの分割を行ったときの、Kernighan-Lin の方法、今回提案した手法、全探査した場合について示す。

1. ノード数:20 、1 クラスタのノード数:5 のランダムに作成したデータ 7 例

手法	cut edge 数 / 全 edge 数						
K-L	10	7	8	8	7	8	7
本手法	5	7	8	8	7	8	7
全探査	5	7	8	8	7	8	7
	50	60	55	59	60	59	60

2. 実際のマニュアル: ノード数 489、1 クラスタのノード数 29

手法	cut edge 数	/	全 edge 数
K-L	108	/	411
本手法	83	/	411

以上で示された様に Kernighan-Lin の方法では最適解からかなり乖離する場合があるが、本方式ではノード数 20 のときは全て最適解が得られている。また、実際のマニュアルに適用した場合も、計算時間はかかるが、本方式の方が良い結果が得られている。

6 自己組織化との関係

本論ではマニュアルの分冊化について取り上げたが、今回用いた手法は、ある意味単位の集合とそれらの関係が与えられたときに、全体の構造を見つける手法とも考えられる。この点で、構造の自動抽出という意味での自己組織化とも関係が深い。

いま、要素の集合により構成される対象というものを考え、要素間の関係が何らかの手段で得られたとき、本手法を適用することにより、このような対象が持つ階層構造を要素間の距離によって求めることができる。つまり、本手法で、ある一定の距離以下のモジュール同士を一つの分冊とした様に、ある一定の距離以下の要素の集合により最下位の階層が作られ、この距離よりいくらか大きい距離以下の要素の集合で一つ上の階層が作られる。このように本手法を活用することによって、要素間の関係といったミクロなデータから全体の階層構造というマクロなデータを取り出すことが可能となる。

この方法は、個々の要素から全体を作り上げるといった、bottom up による作成の場合に適用することが可能で、構成を自動化したり、構成を行う際の指針とすることができる。

7 今後の課題

7.1 分冊規準の改善

今回は分冊の基準として分冊間の参照関係を最小にする、つまり参照関係があるモジュールを一つの分冊に集めることのみを用いた。しかし、一般的には他にも分冊の基準がある。例えば、しばしば参照されるモジュールは一つの分冊にまとめるというような、モジュールの参照頻度を用いるものもある。また、いくつかの基準を組み合わせることも考えられる。このとき、分冊を構成するための基準が、二つのモジュール間の関係の強さに写像できるものであれば、今回の例と同様に処理できる。また、いくつか基準がある場合は、モジュール間に働く力を複数の（基準により決定される）力の和として定義し、本手法を適用できる。

7.2 並列処理による高速化

3.2.1で述べたように、全モジュール間の力を求めるには、モジュールの数を n とすると、$n(n-1)/2$ 回の計算が必要となる（ニュートン法でも同じ）。しかし、2つのモジュール間の力は他のモジュールの影響を受けないので、モジュールごとに独立して計算できる。そのため、一つのモジュールに一つのプロセッサを割り当て、個々のモジュールに働く力（ニュートン法の場合は偏微分も）を並列に計算すれば、n 回で一度のモジュールの移動量が求められる。

8 終わりに

大規模マニュアルの編集・運用に際し、モジュールの存在とモジュール間に参照関係があることに着目したマニュアルの自動構造手法を提案した。本手法は力学的アナロジーを利用した準最適化する手法であり、実際のマニュアルを対象として実験を行い、手法の有効性を確認した。また、より高次の意味的情報を自動抽出できる可能性が示された。

アルゴリズムの精緻化と同時に、構造の自動抽出という意味での自己組織化手法へと一般化して行くことが今後の課題である。

謝辞

本研究にあたり、多くの有益な助言を頂いた笠原部長を始めとする情報応用研究部の皆様に感謝致します。

参考文献

[1] 佐古・千村 (1990)“参照構造に着目した大規模マニュアル分冊化技法”、情報処理学会第41回全国大会、2N-7

[2] 千村・加藤・佐藤 (1987) “マニュアル分析・評価技法”、情報処理学会情報学基礎研究会、7-3.

[3] 三谷・千村・藤原 (1988) “参照構造の把握に着目した大規模マニュアル管理技法”、情報処理学会第36回全大、5S-6.

[4] 田村 (1992) “記号間の力学に基づく概念マップ生成システム SPRINGS”、情報処理学会論文誌 Vol.33 No. 4 pp.465-470

[5] Hara, Y. et al. “Implementing Hypertext Database Relationships through Aggregations and Exceptons.” ACM HYPERTEXT, 1991, pp.75-90.

[6] Botafogo, R. A. et al. “Identifying Aggregations in Hypertext structures.” ACM HYPERTEXT, 1991, pp.63-74.

[7] Kernighan, B. W., and Lin, S. “An efficient heuristic procedure for partitioning graphs,” Bell Sys. J., Vol. 49, No. 2, 1970, pp.291-307.