ネットワーク構成情報表示システムのための自動配置アルゴリズムについて

平川龍*, 中谷真人*, 山路晃徳*, 吉田和幸**
*大分大学工学部, **大分大学総合情報処理センター

ネットワーク管理において、その構成を把握することはとても重要なことである。しかし、近年のネットワークは急速に大規模化・複雑化してきたため、絶えず変化する学内 LAN のような環境において、ネットワークの構成を把握することは非常に難しくなった。我々は、大分大学学内 LAN の構成情報を収集し、IP ネットワーク上で動作するルータ,Layer3 スイッチ,サブネットの接続状況を視覚的に表示するシステムを作成した。ネットワーク構成図の表示において、最終的には、それぞれのルータ,サブネットの場所をマウスで指定するとしても、百数十のルータ,サブネットすべての座標をユーザがここに指定することは不可能である。そのため、自動的に見やすい配置のネットワーク構成図を作成することは重要である。本論文では、自動配置アルゴリズムの改良について述べる。

A Layout Algorithm for Network Topology Visualizing System

Ryo Hirakawa*, Mabito Nakatani*, Akinori Yamaji*, Kazuyuki Yoshida**
*Department of Computer Science and Intelligent Systems, Oita University
**Information Processing Center, Oita University

In the network management, it is very important to grasp the network topology. However, it is very difficult to grasp the network topology like LAN, because the network in recent years becomes large in scale and complicated, and always changes somewhere in the LAN. We designed and implemented the system, which collected information of network topology of LAN in the OITA University, and displays a connection state visually between routers (including Layer3 switches) and subnets, which work on IP-network. In the display of network topology, a user is unable to specify all the coordinates of 100 or more routers and subnets separately, though the place of some routers and subnets are specified with a mouse finally. Therefore, it is important to create the legible network topology graph automatically. In this paper, we describe improvement of the auto layout algorithm for this system.

1. はじめに

近年、ネットワークは急速に普及し、利用者は今まで以上に快適なネットワーク環境を求めるようになった。この要求を満たし、ネットワークを円滑に運用していくためにはネットワーク管理が必要不可欠である。ネットワークを管理するには、ネットワーク構成を常に把握し、その変化が異常であるならば直ちに対処しなければならない。しかしながら、現在のネットワークは爆発的に大規模・複雑化しており、学内 LAN のように絶えず変化する環境においてその構成を常に把握することは非常に困難である。

そこで、我々は上記の問題を解決するために、ネットワークの構成情報を収集し、その情報に基づき IP ネットワーク上で動作するルータ（Layer3 スイッチも含む）とサブネットの接続状況を構成図として視覚化するシステムを作成した[1][2][3][4][5]。

本論文ではこれまでの研究で作成された、構成図上のノードの配置を最適化する自動配置アルゴリズムの改良について述べる。

2. システムの構成

構成情報表示システムはサーバ上で稼動する「構成情報収集部」と、クライアントで使用する「構成情報表示部」から構成される[1][2][3][4][5]。

2.1 構成情報収集部

2.1.1 構成情報の収集

構成情報収集部は SNMP(Simple Network Management Protocol)を用いてネットワークの構成情報を定期的に収集する。構成情報の収集には、各ルータが持つ OSPF(Open Shortest Path Fast)の MIB(Management Information Base)を用いている[7]。

ospfObjectIdentifier.ospfLsdbTable.ospfLsdbEntry.ospfLsdbAdvertisement {1.3.6.1.2.1.14.4.1.8}

これは、OSPF ではエリア内の全てのルータが共通のリンク状態データベース(エリア全体のトポロジが記述されている)を持つため、1 つのルータに問い合せるだけで LAN 全体のトポロジを取得できるからである。

2.1.2 表示データの作成

収集したリンク状態データベース(図 1)には、構成図の描画に必要のない情報も含まれている。

そこでその中から、表示部で利用する、ルータ・サブネット名と、それらの間の接続情報を作成する(図 2)。

図 2 の各行は接続関係を表しており、各行の左は".oita-u.ac.jp"を省略したルータ名、右はサブネットの IP ネットワークアドレスである。各サブネットの表すサブネットマスクは一部例外もあるが、ほとんどが/24 であるので、記述を省略している。また、作成されたデータはサーバ上に保存され、表示部からいつでも参照できる。

2.2 構成情報表示部

2.2.1 表示部の概要

構成情報表示部(図 3)は、構成情報収集部が作成したルータ、サブネット間の接続情報に基づき構成図を描画する。

表示部は Applet で作成されているので、ネットワーク管理者はどこからでも自分の管理するネットワークの構成情報を知ることができる。しかし、管理者以外の人間にネットワークの情報を知られることはあまり好ましくない。そこで本表示部は、パスワードによる認証を用いてシステムの利用者を制限している。

```
14.4.1.8.0.0.0.0.1.2.6.240.162.2.6.240.162 = Hex: 00 02 22
01 02 06 F0 A2 02 06 F0 A2 80 00 03 8C E8 EE 00 48 00 00
00 04 85 25 DE 80 FF FF FF 80 03 00 00 0A 85 25 CD 7E
85 25 CD 7E 02 00 00 0C 85 25 DC 14 85 25 DC 76 02 00
00 0A 85 25 D6 09 85 25 D6 1E 02 00 00 0C
14.4.1.8.0.0.0.0.1.2.18.37.111.2.18.37.111 = Hex: 00 02 22
01 02 12 25 6F 02 12 25 6F 80 00 04 51 8D A6 00 30 00 00
00 02 85 25 90 FE 85 25 90 FD 02 00 00 32 85 25 DC 14 85
25 DC 78 02 00 00 0A
```

‥省略‥

図 1　リンク状態データベース

```
# 2004.xx.xx
s1.net          133.37.240.224
s1.net          133.37.254.0
s1.net          133.37.211.0
s1.net          133.37.203.128
s2.net          133.37.96.0
s2.net          133.37.99.0
‥省略‥
```

図 2　作成したルータ・サブネットの接続情報

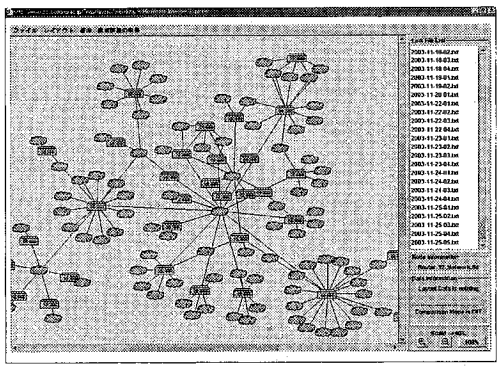


図 3　構成情報表示システム

2.2.2 表示部の機能

表示部は収集部が作成したデータに基づき、ネットワークの構成情報を無方向グラフとして表現する。グラフのノードはルータとそれに繋がるサブネットから構成される。

構成情報表示部には、構成図(構成情報から作成される無方向グラフ)をより見やすくするために以下で説明する 5 つの機能を用意している。

(1) 自動配置機能

収集部が作成したデータからは、ノードの座標

を決定できない。そこでこの自動配置機能は、3章で述べる自動配置アルゴリズムを用いてレイアウト（ノードの配置）を最適化する。

(2) 手動配置機能

前述した自動配置だけでは、必ずしも最適なレイアウトを作成できるとは限らない。そこでこの手動配置機能はノードのアイコンをドラッグすることで、ノードの配置を自由に変更できる。

(3) レイアウトデータの保存

自動配置・手動配置で見やすいレイアウトを作成した時、そのレイアウトを本システムの利用時に何度でも利用できると便利である。そこで、作成したレイアウトを保存し、何度でも使用できる。

(4) スケール変換

表示部で作成した構成図をネットワーク管理の参考にする時、全体的な構成が必要な場合と、一部の詳細な構成が必要な場合がある。そこで構成図のスケールを拡大・縮小できる。

(5) 差分表示機能

ネットワーク管理において、ネットワークの変化をすぐに見つけ、問題があればそれに対処しなければならない。そこで、ネットワークのノード及び、ノード間の接続に変化があれば、それをグラフィカルに表示することができる。

3. 自動配置アルゴリズム

3.1 アルゴリズムの概要

表示部で使用する自動配置アルゴリズムを図4に示す、このアルゴリズムでは、ノード間に目標距離を定め、目標距離と実際のノード間の距離との差から、各ノードに働く引力・斥力を求める。このため、ノード間の距離が目標距離から離れるほど、強い力が働き、目標距離に近づくほど、その力は弱くなるため、ノード間の距離は徐々に目標距離に近づいていく。また、本アルゴリズムでは、接続のないノード間の基本距離 c1、接続の有無に応じたノード間の反発係数c2(接続あり)、c3（接続無し）、ノードの移動距離の計算時に利用する移動履歴の係数 c4、そして、ノードの移動距離を徐々に小さくし、自動配置を停止させる移動距離の緩和係数 c5、を用いてノード間の距離・配置、ノードの散らばり具合を調節し、より見やすい配置を作成する。これらの係数については 3.2 節以降で説明する。

```
double c1,c2,c3,c4,c5;      // 係数
double relax;               // 緩和係数
node[] node;                // ノード
double optimal;             // ノード間の目標距離
double F;                   // ノード間に働く力
Dimension len;              // ノード間の距離
Dimension[] mv;             // 各ノードの移動距離

relax = 1.0;
for( int i=0; i<node.length; i++ ) mv[i] = 0;
While(true){
  // ノードの移動距離を計算する
  for( int i=0; i<node.length; i++ ){
    for( int j=i+1; j<node.length; j++ ){

      len = node[i].distanceOf(node[j]);
      if( node[i].connectTo(node[j]) )
        optimal = node[i].getOptimal(node[j]);
      else
        optimal = c1;
      F = ( optimal - len.abs() ) / len.abs();
      if( node[i].connectTo(node[j]) )
        F = c2 * F;
      else
        F = c3 * F;
      if( (len > optimal && node[i].connectTo(node[j])) ||
        (len < optimal) ){
        mv[i]  -= relax * F * len;
        mv[j] += relax * F * len;
      }
    }
  }
  for( int i=0; i<node.length; i++ ){
    // ノードの座標を更新する(mv[i]分の距離を移動する)
    node[i].move( mv[i] );
    mv[i] *= c4;
  }
  relax *= c5;
}
```

図4　自動配置アルゴリズム

3.2 目標距離の決定

[3][4][5]では、ノード間の目標距離を各ノードのリンク数から決定していた。これにより、リンク数の多いノード同士を遠ざけて表示できたが、逆にリンク数が少ない末端ノード同士の距離が短くなってしまった（図 5-a)。

そこで、接続のないノード間の基本目標距離 c1 を設定し、c1 の値を変更しながら、レイアウトの状態を最も最適化できる c1 について以下の検証を行った（図 5)。結果、c1 の値を 300 と決定した。以下に詳細を述べる。

(1) c1 が 100 以下の場合（図 5-b,c)

c1 の値が小さすぎると、接続のないノード間で斥力が働きにくくなり、ルータに接続されたサブネット等のノードが中央付近で移動を止めてしまうため、中央付近でノードとリンクが重なり非常に見づらい。

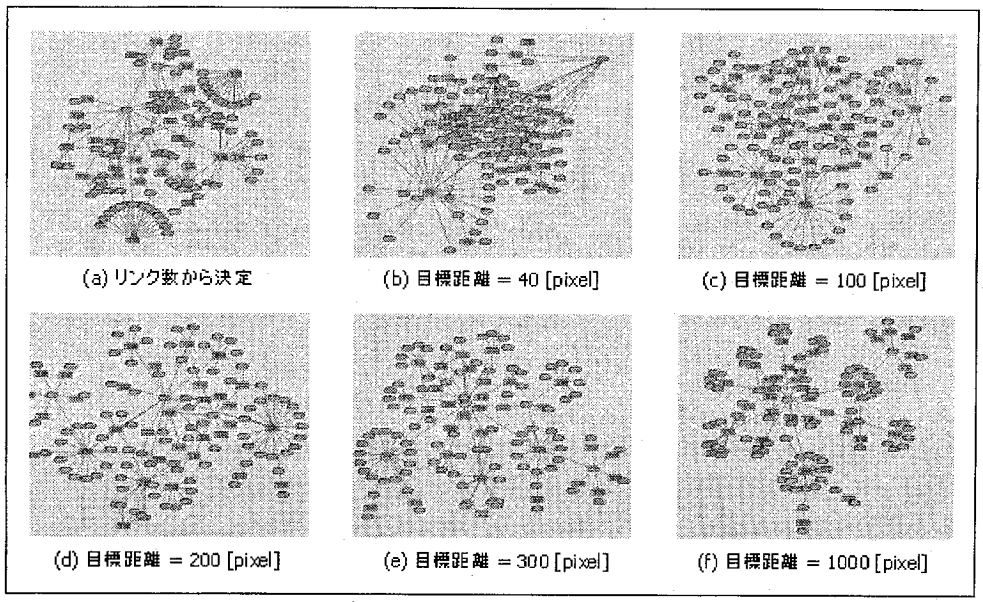


図 5　接続されていないノード間の目標距離によるレイアウトの変化

(2) c1 が 200 ～ 300 程度の場合（図 5-d,e)

c1 の値が大きくなると、接続のないノード同士に働く斥力の有効範囲が十分に広がり、接続のない末端のノード同士が強く遠ざけあい、綺麗なレイアウトを作成できる。特に c1 = 300 の時、ルータに接続するサブネットのノードが程よくまとまる事で、ノードとリンクの重なりが少なくなり、最も見やすいレイアウトを作成できる。

(3) c1 が 1000 の場合（図 5-f)

c1 の値があまりに大きいと接続のないノード間に働く斥力の影響が強くなりすぎ、ルータに接続するスタブノードが重なってしまうため、とても見づらくなる。

3.3 引力・斥力の計算

[3][4][5]では、各ノード間に働く力を、ノード間の距離と目標距離の比率から計算した。

しかし、接続のないノード間の斥力が強すぎると、同じルータに接続されたスタブノード同士で強い反発が起こり、綺麗なレイアウトにならない（図 6-a)。そこで、ノード間の接続関係に応じて反発係数 c2(接続有り), c3(接続無し)を設定し、c2=1.0 の状態で c3 の値を徐々に小さくし、スタブノードを綺麗に配置できる c3 について検証（図 6）を行い、c3 の値を 0.05 と決定した。

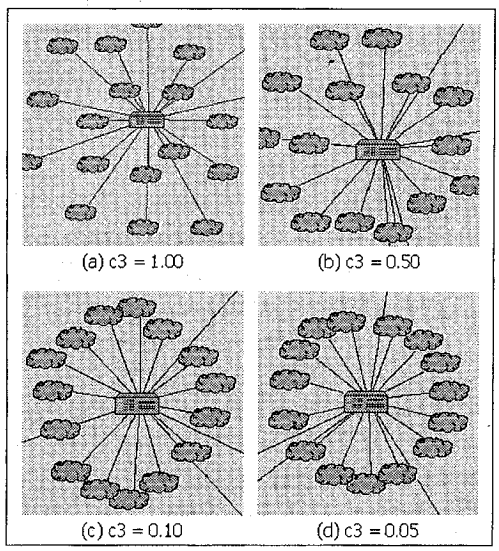


図 6　反発係数によるスタブノードの配置

(1) c3 = 0.50 の場合（図 6-b)

c3 の値を 1/2 にするとやや反発は弱まったが、ノードの配置にまだばらつきがある。

(2) c3 = 0.10 の場合（図 6-c)

c3 の値を 1/10 にすると十分反発は弱まったが、スタブノードの配置は少し歪んでいる。このためスタブノードの数や周囲のノードの配置状況に

よっては、見づらい場合がある。

(3) c3 = 0.05 の場合（図 6-d）

c3 の値が 1/20 になると、スタブノード間の斥力が十分に弱まり、綺麗な円状に配置されるのでとても見やすい。

(4) c3 = 0.01 以下の場合

c3 の値が小さくなりすぎると、スタブノード間の斥力が弱くなりすぎ、スタブノードが重なってしまうため、とても見づらくなる。

3.4 ノードの移動履歴

[3][4][5]では、ノードのレイアウトがほぼ最適化された後も、各ノード間に働く小さな力の影響で、振動状態に陥ってしまった（図 7）。

そこでノードの振動に着目し、移動距離の計算時に毎回ノードの移動の履歴を採り、次の移動距離の計算にその情報を用いた。また、移動履歴が移動距離の計算に及ぼす影響を緩和する係数 c4 を設定し、c4 の値で移動距離がどのように変化するかを検証し、c4 の値を 0.25 と決定した。

(1) c4 = 1.0 の場合（図 7-(1)）

移動距離の計算に前回の移動距離が 100%影響すると、ノードは目標位置で停止できず、結果以前にまして振動が大きくなった。

(2) c4 = 0.5 の場合

従来に比べると振動はやや小さくなったが、まだまだ十分ではない（(1)とほぼ変わりない）。

(3) c4 = 0.25 の場合（図 7-(3)）

この時ノードの振動が最も小さくなり、ノードの振動を抑制できたと判断できる。また、図より振動状態に達するまでの時間も短くなった。

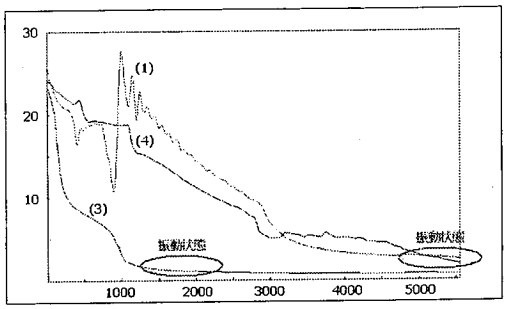
(4) c4 = 0.125 以下の場合（図 7-(4)）

c4 の値がこれ以下になると移動履歴の効果が小さくなりすぎ、移動距離の計算に移動履歴を用いる意味がない。

この結果、振動するノードでは、前回と今回の移動同士が弱めあいノードの振動を抑制できる。逆に移動中のノードは、前回と今回の移動が強めあい、より速くノードの配置を行える。

3.5 移動緩和係数

これまでに述べた方法により、ノードの振動を抑制し、より速くレイアウトを最適化できるようになった。しかしノードの振動が小さくなったが、



*縦軸：移動距離 [pixel], 横軸：計算回数 [times]

図 7　ノードの移動距離と計算回数

緩和係数	0.9980	0.9985	0.9990	0.9995
平均計算回数 [times]	781	1014	1589	3030

表 1　移動緩和係数と自動配置の計算回数

なくなったわけではない。

そこで、ノードの移動距離を徐々に短くするための移動緩和係数 c5 を設定し、c5 の値について検証を行った（表 1）。表に記される値は、緩和係数を各値に設定して、計 10 回実験を行った結果得られた、移動距離の計算回数の平均回数である。また自動配置の計算時間は、以下に示す。

[自動配置の計算時間]

計算時間(≒0) + 計算間隔(10[ms]) = 10[ms]

(1) c5 = 0.9980 の場合

表 1 の計算回数を図 7 と比べると、レイアウトの最適化には十分な計算回数ではない。

(2) c5 = 0.9985 の場合

この計算回数では、まだレイアウトを十分に最適化できない。

(3) c5 = 0.9990 の場合

計算回数は、レイアウトの最適化が終わり、ノードが振動状態に陥るタイミングにほぼ重なる。

(4) c5 = 0.9995 の場合

この時の計算回数を見るとレイアウトの最適化に必要な回数に比べて、明らかに多すぎる。

3.6 自動配置の終了条件

これまでに述べた方法でノードの振動を十分抑制出来るようになった。そこで、ほぼ全て（90%以上）のノードが停止している、又は極めて小さ

な振動中であれば自動配置を終了する、という条件を追加した。結果、ノードの配置が最適化されると自動配置の移動計算を、自動的に終了する。

3.7 評価

本章で述べた目標距離・各係数の設定と移動履歴の参照が有効であるかを確認するために、従来と今回、両方の自動配置に前節で述べた終了条件をつけて検証を行った（表 2）。

	最良	平均	最悪
従来 [times]	5143	5394	5753
今回 [times]	1453	1481	1518

表 2　自動配置終了までの計算回数の比較

この結果から、レイアウトの最適化にかかる計算回数が従来に比べて約 1/3 まで短くできたことが確認できる。

またこれまでは、ノード数が約 130 である大分大学学内 LAN について自動配置を行ってきた。

そこで、あるネットワークで観測した BGP の AS Pass の情報（ノード数が約 500）について、自動配置アルゴリズムでレイアウトを最適化できるか検証した（表 3）。

	最良	平均	最悪
LAN [times]	1453	1481	1518
AS [times]	2364	2672	3127

表 3　LAN と AS Pass での計算回数比較

この結果から、ノード数の多いネットワークでも本自動配置アルゴリズムが有効であることが確認できた。

4. 関連研究

文献[6]ではスプリング手法を用いてノードのレイアウトを最適化する方法を示した。

この手法では、ノードを接続する辺をバネと考え、バネのエネルギーが最小になるようにノード間の距離を決定する。

各ノードに働く力の計算には、本アルゴリズムと同様に、ノード間の距離と目標距離を用いるが、この手法では、直接接続のあるノード間（バネ）に働く力は、ノード間の距離と目標距離の比の対数とし、接続のないノード間に働く力は、ノード間の距離の二乗の逆数としている。

これに対して、本アルゴリズムにおけるノード間に働く力の計算では、直接接続の有無に関わらず、引算一回、割算一回、掛算二回で済むので、計算コストの観点から本アルゴリズムが有効であることがわかる。

5. 終わりに

本論文では、ネットワーク構成情報表示システムで使用する自動配置アルゴリズムの計算方法の改善について述べた。ここで述べた様々な改善の結果、従来と比べて約 1/3 の計算回数でレイアウトを最適化できるようになった。

今後の課題として、表示部のインターフェース、システム内部のデータ構造、構成図の描画方法等を改善し、より軽量で高速なシステムの設計を考えている。

参考文献

[1] 松浦孝典, 奥田慎一, 吉田和幸 ：ネットワーク構成図の自動作成について, 第 55 回情報処理学会全国大会講演論文集(3), pp.579-580, 1997

[2] 吉田和幸, 松浦孝典, 長野聡 ：ネットワーク構成情報 3 次元表示システムの実現, 1999 年情報処理学会マルチメディア通信と分散処理ワークショップ論文集, pp.55-60, 1999

[3] 久多良木亨, 松本匡浩, 山本崇文, 吉田和幸 ：ネットワーク構成情報の自動収集と構成図表示システムについて, 第 55 回電気関係学会九州支部連合大会論文集, pp232, 2002

[4] 久多良木亨, 中谷真人, 山路晃徳, 吉田和幸 ：ネットワーク構成図表示システムとその使用経験について, 2003 年情報処理学会マルチメディア通信と分散処理ワークショップ論文集, pp.61-66, 2003

[5] 中谷真人, 久多良木亨, 平川龍, 山路晃徳, 吉田和幸 ：Applet によるネットワーク構成図表示システムについて, マルチメディア,分散,協調とモバイルシンポジウム論文集, pp635-638, 2004

[6] P.Eades: A heuristic for graph drawing, Congressus Numerantium, vol.42, pp.149-160, 1984

[7] F.Baker, R.Coltum: OSPF Version2 Management Information Base, RFC1850, November 1995