ネットワークの
信頼度評価と設計問題に関する研究

1999年

小　出　　武

目 次

第 1 章

緒論

1999 年 11 月 22 日，埼玉県狭山市柏原の入間川河川敷に自衛隊のジェット練習機 T33A が墜落し，入間川を横切る東京電力の高圧送電線を切断，都心と埼玉県南部の 80 万世帯が停電するという事件が起きた．この事件によって首都圏の私鉄，地下鉄が約 30 分にわたり運転をストップしたほか，信号機やビルのエレベーターも一時停止するなど，市民生活が大きく混乱した．現在我々はこの送配電網をはじめ，電話網や情報ネットワークなど様々なネットワークの恩恵を受けて生活をしている．この事件からもわかるように，これらのネットワークを高信頼度のものにすること，例えばネットワークを構成する機器の故障率を小さくしたり，一部の機器の故障によって生じる損害がなるべく小さくなるようにネットワークの構造を設計したりすることは，我々が生活していく上で必要なことであり，社会的要請でもある．ネットワークの信頼性を向上させるためには，ネットワークの設計や保守の段階において，ネットワークの信頼度を正しく評価することが重要となる．例えばネットワークを設計する場合，ネットワークを正しく評価することにより，様々な制約の下でなるべく信頼度の高いネットワークを設計することが可能となる．また保守を行う場合も，現存のネットワークのどの部分をより重点的に保守すべきかを判断することができるであろう．

一般にネットワーク構造を表現するときにはグラフを用いることが多い．例えば情報ネットワークの場合，各種サーバやルータ・ゲートウェイなどの機器を節点に，それらを接続するケーブル類を枝に対応付けることが多い．更にネットワークの信頼度を問題にする場合には，ネットワークモデルとしてグラフの節点や枝に確率の概念を付加した確率グラフが用いられることが多い．このときネットワークの信頼度は，確率グラフ中の特定の節

点間が正常に機能している枝によって連結である確率として評価することが一般的である．ネットワークの信頼度に関する問題は，古くは1952年にJ. von Neumann[87]が信頼度の低い素子を組み合わせて信頼度の高いシステムを設計するという問題として考え，1956年にはE. F. MooreとC. E. Shannon[55]が信頼性の低い構成要素をうまく組み合わせてシステム全体の信頼性上げることができることを示すという観点から解析している．このようにネットワーク信頼度に関する問題は元々はシステム解析での動機からグラフの問題として定式化されたが，その構造の複雑さからグラフ理論そのものの重要な分野として理論的研究がなされ，組合せ論における一つの重要な研究分野を構成している．

確率グラフを用いてネットワーク信頼度を評価する場合，代表的なネットワーク信頼度として，総合信頼度（全節点間信頼度ともいう），2節点間信頼度，k節点間信頼度などが挙げられる．節点は故障せず，枝の故障する確率は互いに独立である確率グラフに対し，総合信頼度は確率グラフ中の全ての節点が正常に機能している枝によって連結されている確率として定義され，その値を求める問題はネットワーク全体の信頼性を評価する上で最も本質的で基本的な問題である．しかし一般にどのようなネットワークに対しても，そのネットワークが持つ枝の本数に対し多項式時間で総合信頼度を計算することは絶望視されているので，大規模なネットワークの総合信頼度を厳密に計算することは計算量理論の観点から事実上困難である．確率グラフによるネットワークモデルは現実のネットワークの核となる部分を表した極めてシンプルなモデルであるが，その総合信頼度を求める問題は一筋縄ではいかない難しい問題であり，理論的に研究する価値があると考えられている．またこの研究の成果は，現実に利用されている更に複雑なネットワークの信頼性に関する研究の理論的基盤になるという意味で，応用面への発展性に関しても十分に価値があると言える．

総合信頼度に関してこれまでに様々な研究がなされている．それらを類別すると次のようになる．

(a) 総合信頼度を低指数オーダーで計算する研究

(b) 総合信頼度が多項式時間で計算可能なグラフのクラスを拡張する研究

(c) 総合信頼度が最大（最小）のネットワークの構造を追究する研究

(d) 総合信頼度の上界・下界を多項式時間で求める研究

(e) 確率的近似アルゴリズムにより総合信頼度の近似値を計算する研究

(f) 総合信頼度を考慮に入れたネットワーク設計に関する研究

(a) の分野は古くから研究が行われている [8, 14, 71]. 最近では 2 部決定グラフを利用した方法が提案され [32, 73], 例えば枝の故障確率が全て等しい n 点平面グラフに対して $O(2^{O(\sqrt{n})})$ 時間で総合信頼度を求めることができることが示された.

(b) の研究では, 対象のグラフを平面グラフに限定しても, 総合信頼度の真値を計算する問題は#P 完全であることが D. Vertigan[86] により示された. この成果により研究対象は平面グラフに限定され, その対象内において多項式時間で総合信頼度を計算できるグラフのクラスを拡大する研究が盛んである [29, 63, 64].

(c) の研究では, 節点数 n, 枝数 m, 全ての枝が同じ確率 $1-p$ で故障するネットワークのうち, $0 \leq p \leq 1$ の任意の p に対し総合信頼度が最大 (または最小) となるネットワークの構造を追究する研究が特によく行われている [9, 10, 60].

(d) の研究については, 主に信頼度多項式に基づく方法と, エッジ・パッキングの利用による方法の 2 通りのアプローチがなされている. 前者の方法のうち最も有効と言われているのは M. O. Ball と J. S. Provan による方法で, その方法による境界値は Ball-Provan の境界値と呼ばれる [6]. Ball-Provan の境界値は, 全ての枝が同一の確率で故障するネットワークに関する総合信頼度の境界値であり, 特定のネットワーク独自の情報や, グラフ変換, 他の信頼度計算の手法などを組み合わせることにより, 更に精度を上げる研究が数多く行われている [11, 12, 13]. 総合信頼度に関して言えば, 信頼度多項式に基づく方法による境界値の方がエッジ・パッキングの利用による境界値より精度が良い傾向があることが実験的に確認されている [18, 19]. またこの研究分野で得られる境界値は, 後述する (e) や (f) の研究分野で, サンプリングの範囲や解空間を限定して実行時間を小さくするために利用されている.

(e) では, モンテカルロ法などを用いて総合信頼度の推定値をある誤差範囲に収める研究がなされており [3, 38], コンピュータの性能向上に伴い近年盛んに行われている.

(f) については, 総合信頼度を一定値以上とすることを制約条件として費用を最小化するネットワーク設計問題が, 分枝限定法やメタ・ヒューリスティックスを利用して解く研究がなされている [21, 35]. 問題を解く過程で総合信頼度の値を評価するので一般に多くの計算時間を必要とするが, コンピュータの性能向上に伴いこの分野の研究も行われるようになってきた.

上記のどの研究分野においても, モデルの単純化と解析の容易化のため,「枝が故障する確率はネットワーク中の全ての枝について等しい」という仮定をしていることが多い. し

かし現実問題においては，故障確率が異なる枝を有するネットワークの信頼度を評価したり，重要と考えられる枝に対してのみ信頼性を上げて，少ないコストによりネットワーク全体の信頼度を向上させたりするケースは数多く存在する．そのため，より一般的なモデルである故障確率の異なる枝を有するネットワークの信頼性に関して知見を得ることは非常に重要であるが，故障確率が全ての枝について等しい場合と比較すると解析が困難であるため，まだあまり研究されていないのが現状である．

本研究ではまず，故障確率の異なる枝を含むネットワークの総合信頼度の下界を多項式時間で導出する幾つかの方法を提案する（上記研究分野の類別で言えば (d) に該当する）．対象となるネットワークの信頼度が少なくともその値以上であることを保障するという意味で，上界や近似値に比べ，信頼度の下界は現実問題で必要とされることが多い．そのため，精度の良い総合信頼度の下界を実用的な時間で求めることは非常に有用である．次に総合信頼度を考慮に入れたネットワーク設計問題の最適解を探索する方法を提案する（類別 (f) に該当する）．この方法は従来の方法とは異なり，故障確率が異なる枝を含むネットワーク設計問題にも適用可能である．

本研究はこれまでにあまり行われていない枝の故障確率が異なるネットワークの信頼性の研究に関する新たな知見を得て，今後のこの分野の研究における一つの評価基準になることを目的としている．本研究の成果により，ネットワークの信頼度を評価したり，信頼度を考慮してネットワークの構造を設計したりする際に，枝の故障確率が異なるネットワークを対象とすることが可能となり，ネットワークモデルをより現実のネットワークに近いものとすることが可能となる．

本論文の構成は以下の通りである．

第2章では，本論文を展開するために必要なグラフ理論の概念と，組合せ最適化に関する基本概念を説明する．

第3章では，総合信頼度とその下界に関する概念を説明する．まず本論文で扱うネットワークモデルを設定し，その妥当性について言及する．次に総合信頼度を求める問題の#P困難性，すなわちネットワークが有する枝の本数に対し多項式時間で総合信頼度を求めることが絶望視されていることについて述べる．最後に総合信頼度の下界を多項式時間で計算する主な既存手法である信頼度多項式による方法とエッジ・パッキング法を紹介する．

第4章では，故障確率が異なる枝を有するネットワークに対し，極大木によるエッジ・パッキングを構成し，総合信頼度の下界を求めるアルゴリズムを提案する．得られる下界

の精度を上げるためには，エッジ・パッキングの要素数，すなわち枝排反な極大木の本数を増やすことと，極大木自体の総合信頼度を高くすることという互いに相反する戦略が存在することを示し，それぞれの戦略に基づいてエッジ・パッキングを構成するヒューリスティック・アルゴリズムを提案する．それぞれのアルゴリズムの効率性を理論的に示し，更にアルゴリズムが導出する総合信頼度の下界を数値実験により比較する．また総合信頼度を計算する際に用いられる短絡除去，直列縮退，並列縮退という3つのグラフ変換を提案したアルゴリズムに組み合わせると，導出する下界の精度が上がる傾向があることを数値実験により示す．また枝の故障確率が全て等しいネットワークを対象にした場合，エッジ・パッキング法による下界は信頼度多項式に基づく方法による下界より精度が悪いと言われているが，提案したエッジ・パッキング法にグラフ変換を組み合わせた方法が信頼度多項式に基づく方法より精度の良い下界を導出できることを数値実験により示す．

第5章では第4章で提案したエッジ・パッキング法において，エッジ・パッキングの要素を極大木から極大木を含むクラスである直並列グラフへと拡張し，第4章と同様，エッジ・パッキングを構築する2つのヒューリスティック・アルゴリズムを提案する．第4章で提案したアルゴリズムによる下界と数値実験により比較して，直並列グラフへの拡張が下界の精度を上げることに大きく貢献していることを示す．

第6章では，M. O. Ballにより提案されたグラフ変換を用いて対象となるネットワークを有向ネットワークに変換し，その有向ネットワークの信頼度の下界を多項式時間で計算することにより，元のネットワークに対する総合信頼度の下界を多項式時間で計算する方法について述べる．有向ネットワークの信頼度の下界を計算するアルゴリズムとして，アサイクリックグラフを要素とするアーク・パッキングを利用するアルゴリズムを提案する．数値実験によって第5章で提案したアルゴリズムと比較することにより，本章で提案したアルゴリズムが同程度の精度を持つ下界を導出できることを示す．

第7章では，総合信頼度を考慮したネットワーク設計問題の最適解を探索するアルゴリズムを提案する．この問題の特徴は故障確率が異なる枝を含むネットワークを対象とすることである．提案するアルゴリズムは分枝限定法を基盤とし，総合信頼度の上界，R. -H. Janが彼のアルゴリズムで用いた技法などを組み込むことによって実行時間の短縮を図っている．このアルゴリズムによって，より現実的なネットワーク設計問題に対する理論的基礎を与えることができる．

第8章では本論文を総括し，これまでの成果をまとめるとともに，残された課題や今後

の発展について述べる.

第 2 章

グラフ理論と組合せ最適化問題

本章では，準備として本論文を展開するために必要な**グラフ理論** (graph theory) の概念を説明し，グラフ理論に関連する用語や記号の定義を行う [28, 33, 34]．続いて**組合せ最適問題** (combinatorial optimization problem)，およびその関連概念としてアルゴリズムの計算量，分枝限定法について説明する [2, 30, 31]．

2.1 グラフ理論における諸定義

2.1.1 無向グラフにおける基礎概念

グラフ (graph) は，いくつかの**節点** (vertex, node) と 2 節点を接続する**枝** (edge) によって構成される．グラフ G は，その節点集合 V と枝集合 E によって $G = (V, E)$ と表され，節点の個数を $|V|$，枝の本数を $|E|$ と記す．慣例として，$n = |V|$，$m = |E|$ とすることが多い．後述する有向グラフと区別するときは，グラフを**無向グラフ** (undirected graph) と呼ぶ．

節点 u と節点 v $(u, v \in V)$ とを接続する枝 $e \in E$ を $e = (u, v)$ と記す．2 本の枝 e, f が共に同じ 2 節点を両端とするとき，すなわち $e = (u, v)$，$f = (u, v)$ であるとき，枝 e, f は**平行枝** (parallel edges) という．また両端点が同一の節点である枝，すなわち $e = (v, v)$ と表される枝を**自己閉路** (self-loop) という．図 2.1 において，枝 7 と枝 8 は平行枝，枝 4 は自己閉路である．

グラフ $G = (V, E)$ において，節点 $v \in V$ を両端点とする自己閉路が i 本，節点 v を端点

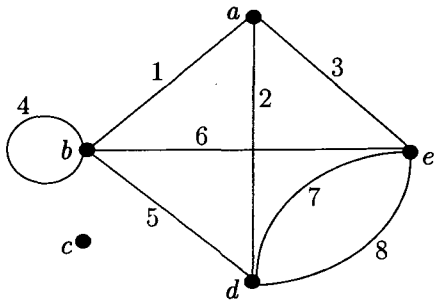


図 2.1 グラフの例

とする自己閉路でない枝が j本存在するとき，節点 vの**次数** (degree) $\delta(v)$ を

$$\delta(v) = 2i + j \tag{2.1}$$

と定義する．図2.1において，$\delta(a) = 3$, $\delta(b) = 5$ である．更にグラフ $G = (V, E)$ の次数を $\min_{v \in V} \delta(v)$ とする．次数が 0 の点を**孤立点** (isolated vertex) という．図 2.1において，節点 c は孤立点である．節点 c が存在しなかった場合，図 2.1のグラフの次数は 3 となる．平行枝も自己閉路も持たないグラフを**単純グラフ** (simple graph) という．また n 個の節点を有する単純グラフ $G = (V, E)$ において，任意の 2 節点間を接続する枝が存在する場合，G を**完全グラフ** (complete graph) といい，K_nで表す．K_nの枝本数は ${}_nC_2$である．

グラフ $G = (V, E)$ において，以下の条件を満足するように節点と枝を交互に並べた系列

$$(v_{i(1)},\ e_{i(1)},\ v_{i(2)},\ e_{i(2)}, \ldots, e_{i(l)},\ v_{i(l+1)}) \tag{2.2}$$

$$\text{ただし } v_{i(j)} \in V\ (j = 1, \ldots, l+1),\quad e_{i(j)} = (v_{i(j)}, v_{i(j+1)}) \in E\ (j = 1, \ldots, l) \tag{2.3}$$

を長さ lの**パス** (path) といい，節点 $v_{i(1)}$を**始点** (start vertex)，節点 $v_{i(l+1)}$を**終点** (end vertex) という．また同じ節点を 2 度以上通らないパスを**単純な** (simple) パスという．始点と終点とが同一であるパスを**サイクル** (cycle) といい，同じ節点を 2 度以上通らない単純なサイクル（ただし始点と終点は同一視して合わせて 1 回とする）を**閉路** (circuit) と呼ぶ．図 2.1 において，$(a, 1, b, 5, d, 2, a)$ は閉路である．

グラフ $G = (V, E)$ 上の 2 節点 $u, v \in V$に対し，節点 u を始点，節点 vを終点とするパスが存在するとき，節点 u は節点 vへ**到達可能** (reachable) であるという．無向グラフの場合，節点 u から節点 vへ到達可能ならば，節点 vから節点 u へも到達可能であるので，節点 u と節点 v間は到達可能ともいう．グラフ $G = (V, E)$ 上の任意の 2 節点間が到達可能

であるとき，グラフ G は**連結** (connected) であるという．連結であるグラフを**連結グラフ** (connected graph) といい，連結でないグラフを**非連結グラフ** (unconnected graph) という．図 2.1において，節点 c は他のどの節点とも到達可能ではないので，図 2.1で表されるグラフは非連結グラフである．仮に節点 c が存在しなければ，図 2.1で表されるグラフは連結グラフである．

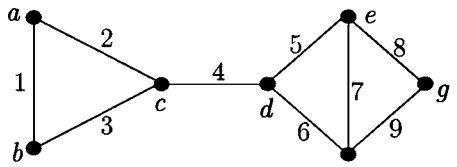


図 2.2 カットの例

節点集合 V を 2 つの部分集合 X と $\overline{X} = V - X$ に分けたとき，枝集合

$$\{(u,v) \in E \,|u \in X,\ v \in \overline{X}\} \tag{2.4}$$

を**カット** (cutset) と呼び，$C(X,\overline{X})$ で表す．カットに属する枝の本数はカットの大きさと呼ばれ，$|C|$ で表される．$|C|$ が最小となるカットを**最小カット** (minimum cut) といい，最小カットの大きさを**枝連結度** (edge-connectivity) という．図 2.2において $X = \{a,b\}$ とすると，$C(X,\overline{X}) = \{2,3\}$ となる．図 2.2のグラフにおける最小カットは $X = \{a,b,c\}$ としたときのカットで（または $X = \{d,e,f,g\}$），$C(X,\overline{X}) = \{4\}$ である．従って図 2.2のグラフの枝連結度は 1 である．

グラフ $G = (V,E)$ において，節点集合 $V' \subset V$，枝集合 $E' \subset E$ が

$$(u,v) \in E' \Longrightarrow u \in V' \text{ かつ } v \in V' \tag{2.5}$$

を満足するとき，グラフ $G' = (V',E')$ をグラフ G の**部分グラフ** (subgraph) という．特にグラフ $G = (V,E)$ が連結グラフであるとき，$V' = V$ なる連結部分グラフを**全域部分グラフ** (spanning subgraph) という．枝の本数が最も少ない全域部分グラフを**極大木** (spanning tree) という．閉路を部分グラフとして持たないグラフを**木** (tree) というが，極大木は G の部分グラフのうち，枝の本数が最も多い木と定義することもできる．グラフ $G = (V,E)$ の極大木が有する枝の本数は $|V|-1$ 本になる．図 2.3のグラフ $G_1,\ldots,G_5$ はいずれもグラフ G の部分グラフである．そのうちグラフ G_1 を除く部分グラフは全域部分グラフで，G_3, G_4, G_5 は極大木である．

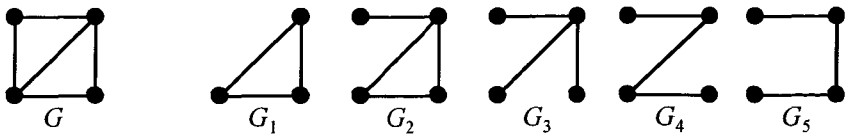


図 2.3 部分グラフの例

グラフ $G=(V,E)$ から枝 $e=(u,v)\in E$をその両端点 u, vを残したまま取り去ることを枝 e の**削除** (deletion) という．グラフ $G=(V,E)$ から枝 e を削除したグラフを $G-e$, グラフ $G=(V,E)$ から枝集合 E'に属する枝を削除したグラフを $G-E'$と表記することにする．すなわち，

$$G-e \equiv (V,E-\{e\}) \tag{2.6}$$

$$G-E' \equiv (V,E-E') \tag{2.7}$$

である．また $G'=(V,E')$ を G の全域部分グラフとすると，特に混同しない限り，グラフ G から枝集合 E'に属する枝を削除したグラフを

$$G-G' \equiv (V,E-E') \tag{2.8}$$

と表記することにする（節点に関する削除は考えない）．逆にグラフ $G=(V,E)$ に2節点 $u,v\in V$を端点とする枝 $e=(u,v)\notin E$を追加したグラフを $G\cup e$ と記す．

グラフ $G=(V,E)$ から枝 $e=(u,v)\in E$の両端点を1つの節点にまとめ，かつ枝 e を削除することを，枝 e の**縮約** (contraction) という．グラフ $G=(V,E)$ に対し枝 $e\in E$を縮約したグラフを $G\cdot e$ と表す．

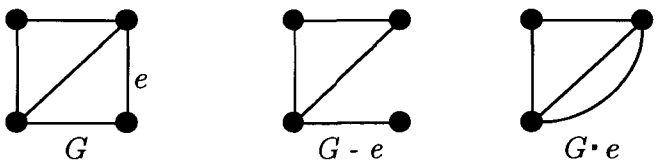


図 2.4 枝の削除と縮約の例

2つのグラフ $G_1=(V_1,E_1)$ と $G_2=(V_2,E_2)$ が，$|V_1|=|V_2|$, $|E_1|=|E_2|$ で，V_1と V_2間および E_1と E_2間にそれぞれ適当な1対1対応が存在して，その対応のもとで両者の接続関係が等しくなるとき，G_1と G_2とは**同形** (isomorphic) であるという．図2.5の G_1と G_2は同形である．

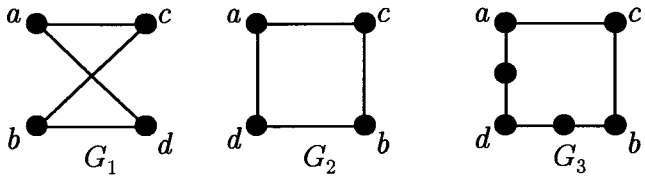


図 2.5 同形と同相の例

グラフ $G=(V,E)$ 中の 1 本の枝 $e=(u,v)$ を 2 つの新しい枝 $e_1=(u,w)$, $e_2=(w,v)$ (wは新しく追加された節点) で置換する操作を枝 e の直列分解と呼ぶことにする．ある 2 つのグラフ G_1, G_2のいずれか一方もしくは両方のグラフに対し，有する枝に直列分解を何度か適用することによって G_1と G_2とが同形になる場合，G_1と G_2は互いに**同相** (homeomorphic) であるという．図 2.5の G_1と G_3（当然 G_2と G_3も）は同相である．節点数 4 の完全グラフ K_4と同相なグラフを部分グラフとして持たないグラフを**直並列グラフ** (series-parallel graph) という．

グラフの節点や枝にスカラーやベクトルを付加したものを一般に**ネットワーク** (network) と呼ぶが，慣例上それらもグラフと呼ぶことも多い．グラフ内の節点や枝が常に存在するのではなく，ある確率で存在しなくなるグラフは一般に節点や枝にその存在確率を付加したネットワークで表現される．このネットワークを**確率グラフ** (probabilistic graph) と呼ぶ．

2.1.2 有向グラフにおける基礎概念

枝に方向性を持たせたものを枝と区別するため**アーク** (arc) と呼ぶ．節点集合 Vとアーク集合 A からなるグラフ Dを**有向グラフ** (directed graph) といい，$D=(V,A)$ と記す．文脈から有向グラフであることが明らかである場合，有向グラフを単にグラフと呼ぶことも多い．節点 $u\in V$を始点，節点 $v\in V$を終点とするアーク $a\in A$ は，$a=(u,v)$ と記される．無向グラフにおけるパスと同様，有向グラフにおけるパスも以下のように定義される．

$$(v_{i(1)},\ a_{i(1)},\ v_{i(2)},\ a_{i(2)},\ldots,a_{i(l)},\ v_{i(l+1)}) \tag{2.9}$$

$$\text{ただし } v_{i(j)}\in V\ (j=1,\ldots,l+1),\quad a_{i(j)}=(v_{i(j)},v_{i(j+1)})\in A\ (j=1,\ldots,l) \tag{2.10}$$

パスに関する他の用語も無向グラフのときと同様に定義できるが，有向グラフの場合，始点と終点が同一である単純なパスを**サイクル** (cycle) ということが多い．図 2.6の有向グラフ D_1において，$(b,3,c,5,d,4,b)$ はサイクルである．サイクルを部分グラフとして持たな

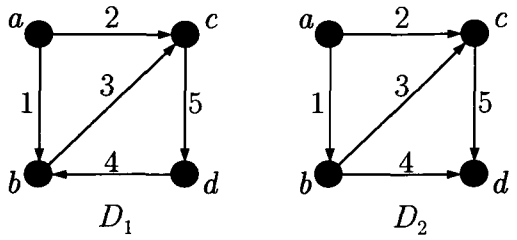


図 2.6 有向グラフの例

い有向グラフを**アサイクリックグラフ** (acyclic graph) という．図 2.6の D_2はアサイクリックグラフである．

有向グラフ $D=(V,A)$ において，ある節点 $s \in V$から他の任意の節点へ到達可能であるとき，節点 s をグラフ Dの**根** (root) という．図 2.7の有向グラフ Dにおいて，節点 s は根である．根を有し，アークの方向性を無視したときに木となる有向グラフを**有向木** (directed tree) という．有向グラフ $D=(V,A)$ の部分グラフで，アークの本数が最も多い有向木を**有向極大木** (arborescence) という．有向グラフを対象としていることが明確な場合は，有向木，有向極大木をそれぞれ単に木，極大木と呼ぶ．図 2.7において，D_1, D_2, D_3はいずれも有向グラフ Dの部分グラフであるが，D_1は有向木ではなく，D_3は有向極大木である．

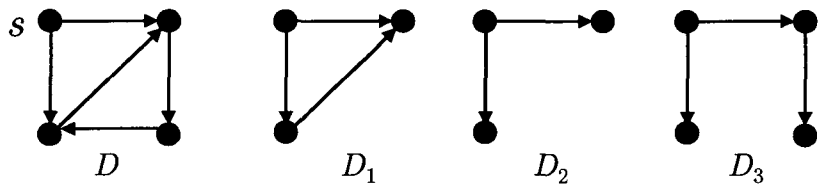


図 2.7 部分グラフ・有向木・有向極大木の例

2.1.3 マトロイド

Eを有限集合とする．Eの部分集合族 $\mathcal{I} \subset 2^E$が以下の条件

$$\phi \in \mathcal{I} \tag{2.11}$$

$$Y \subset X \subset E,\ X \in \mathcal{I} \Longrightarrow Y \in \mathcal{I} \tag{2.12}$$

$$X,\ Y \in \mathcal{I},\ |X| < |Y| \Longrightarrow \exists x \in Y - X;\ X \cup \{x\} \in \mathcal{I} \tag{2.13}$$

を満足するとき，$\mathcal{M}=(E,\mathcal{I})$ を E上の**マトロイド** (matoroid) という．マトロイド $\mathcal{M}=(E,\mathcal{I})$ において，$\mathcal{I}$の要素を**独立集合** (independent set)，極大な独立集合を**基** (base)，独立

集合でない Eの部分集合を**従属集合** (dependent set)，極小な従属集合を**サーキット** (circuit) という．Eの部分集合 Fに対し，Fの部分集合で極大な独立集合の要素数を Fの**階数** (rank) といい，Fと同じ階数を持つ Fを含む極大集合を Fの**スパン** (span) という．

マトロイドの例として，グラフ $G=(V,E)$ の閉路を含まない Eの部分集合の全体を $\mathcal{I}$ とすれば，$\mathcal{M}=(E,\mathcal{I})$ はマトロイドとなる．このようなマトロイドを**グラフ的マトロイド** (graphic matroid) という．連結グラフ $G=(V,E)$ のグラフ的マトロイドにおける独立集合は G の木，基は G の極大木，従属集合は G のサイクル，サーキットは G の閉路になる．図 2.8にグラフ的マトロイドの例を示した．

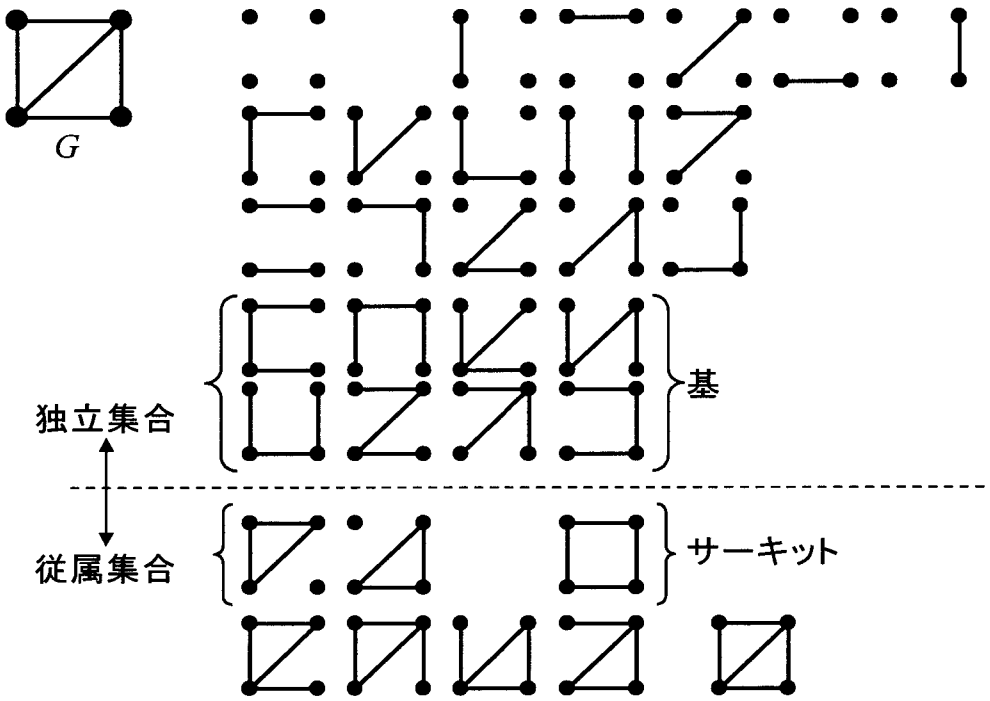


図 2.8 グラフ的マトロイドの例

$\mathcal{M}_1=(E,\mathcal{I}_1),\ldots,\mathcal{M}_k=(E,\mathcal{I}_k)$ を有限集合 E上の k個のマトロイドとする．このとき枝集合 Eに属する枝を k個のマトロイドへと分割する問題，つまり

$$\forall i\in\{1,\ldots,k\};\ E_i\in\mathcal{I}_i,\ \text{ただし}\ E=\bigcup_{i=1}^{k}E_i \tag{2.14}$$

となるように $E_1,\ldots,E_k$を定める問題を**マトロイド分割問題** (matroid partition problem) という．

2.2　組合せ最適化問題とその複雑さ

2.2.1　組合せ最適化問題

一般に**最適化問題** (optimization problem) は以下のように書ける．

$$\left|\begin{array}{l} \text{目的関数 } f(x) \longrightarrow \text{最小（最大）} \\ \text{制約条件 } x \in S, \text{ ただし } S \subset X \end{array}\right. \tag{2.15}$$

ここで変数 x は基礎となる空間 X 内のベクトル，Sはその部分集合で**実行可能領域** (feasible region) と呼ばれる．問題 (2.15) は集合 Sという形式に記述された**制約条件** (constraint) を満たす**実行可能解** (feasible solution) $x \in S$のうち，**目的関数** (objective function) $f(x)$ を最小（最大）にする**最適解** (optimum solution) を見つけることが目的である．ここで $f(x)$ は Sを含む適当な集合上で定義された実数値，あるいは整数値をとる関数である．変数 x が離散的な値をとるとき，問題 (2.15) は**離散的最適化問題**とも呼ばれ，特に空間 Xが有限個の要素の組み合わせから構成される場合は**組合せ最適化問題** (combinatorial optimization problem) と呼ばれる．

最適化問題 (2.15) を解くということは，その実行可能解の中で最も小さい目的関数値を持つものを見つけること，すなわち

$$\forall x \in S;\ f(x^*) \leq f(x) \tag{2.16}$$

を満たす実行可能解 $x^* \in S$を見つけることを意味する．しかしある種の問題においては，全ての実行可能解の中から目的関数値が最小となる解を見つけることが計算時間上著しく困難な場合もある．そのような問題に対しては，

$$\forall x \in S \cap U(x^*);\ f(x^*) \leq f(x) \tag{2.17}$$

を満たすような実行可能解 $x^* \in S$とその近傍 $U(x^*)$ が存在するとき，x^*を問題 (2.15) の解として採用することがある．式 (2.16) を満たす x^*を問題 (2.15) の**大域的最適解** (global optimal solution; **厳密解**ともいう)，式 (2.17) を満たす x^*を**局所最適解** (local optimal solution) という．大域的最適解を求めるアルゴリズムを**厳密アルゴリズム** (exact algorithm; 厳密法ともいう）という．逆に大域的最適解を求める保証がないが，一般に"良い"解である**近似解** (approximate solution) を導出するアルゴリズムを**近似アルゴリズム** (approximate algorithm) という．近似アルゴリズムは大域的最適解を見つける保証がない分，実用的な

短い計算時間で近似解を導出することが要求される．近似アルゴリズムのうち，問題の構造に基づいて解を探索する手法を**ヒューリスティック・アルゴリズム** (heuristic algorithm; ヒューリスティック法，発見的手法ともいう）という．各ステップで，先のことは考えずにその場で最も良いものを選択するヒューリスティック・アルゴリズムを**貪欲法** (greedy algorithm) という．

2.2.2 アルゴリズムの計算量と NP 困難性

一般に○○問題と呼ぶとき，その問題は無限個の**問題例** (problem instance) の全体を指し，実際に解くには問題が有するパラメータを具体的に設定して 1 つの問題例を定める必要がある．従ってその問題を解くアルゴリズムは，これら全ての問題例に対し正しい答えを出力するものでなくてはならない．

アルゴリズムの実行は，加算，乗算，比較などの基本ステップに分解することができるが，これら基本ステップの実行回数を**時間量** (time complexity) という．また計算の途中結果を保持するための記憶領域の広さを**領域量** (space complexity; **記憶量**ともいう)，両者をまとめて**計算量** (complexity) と呼ぶ．アルゴリズムの計算量は個々の問題例によって変化するので，全体的な評価をするには問題例の規模に応じて計算量がどのように変化するのか，その関数形を求めることが大切である．問題例の規模はそれを入力するために必要なデータの長さ Nで評価し，通常 $O(f(N))$ の形式をとる．$O(f(N))$ は**オーダー** $f(N)$ と読み，$T(N) = O(f(N))$ とは，ある正定数 c と N_0が存在して，N_0以上の任意の Nに対し常に

$$T(N) \leq cf(N) \tag{2.18}$$

が成立することを意味する．

ところで計算量を評価するとき，規模 $O(N)$ を持つ問題例は一般に多数存在するため，規模 $O(N)$ の問題例全体の評価として以下の 2 種類の方法がよく用いられる．

最悪計算量 (worst case complexity):

規模 $O(N)$ の全問題例の中で最も大きい計算量

平均計算量 (average complexity):

規模 $O(N)$ の問題例に対し，それぞれの生起確率に基づいて求めた計算量の平均

前者は解析が比較的容易であるが，ごく少数の異常な問題例のために極めて悲観的な評価

になる危険性がある．一方後者は実用的にはより重要であると考えられるが，その導出が容易でない場合が多い．そのため一般には前者を用いることが多い．

問題の複雑さを考える場合，対象の問題を**決定問題** (decision problem) として扱うことが多い．決定問題は，個々の問題例に対しある性質が成立する実行可能解が存在するかどうかを判断し，Yes か No かの回答を要求する問題である．最適化問題は目的関数値に関してある閾値を導入することで,「目的関数値がその閾値以下となる実行可能解が存在するか?」という決定問題に帰着することが可能である．ある決定問題を解く難易度を，時間量が**多項式オーダー** (polynomial order) であるか否かで判断することが多い．多項式オーダーとは，ある定数 kを用いて $O(N^k)$ と書けるという意味である．計算量が多項式オーダーであるアルゴリズムを**多項式（時間）アルゴリズム** (polynomial time algorithm) という．逆に多項式オーダー時間では解けないぐらい難しい問題であることを示すには，どのようなアルゴリズムを用いても不可能であることを示さねばならず，容易ではない．この目的に有効な概念が **NP 困難性** (NP-hardness) である．

ある組合せ問題 Q を解く多項式アルゴリズムが存在するならば，問題 Q は**クラス P**(polynomially solvable の略) に属するという．それに対し，互いに独立なコンピュータが無限個存在して同時に並列処理ができると仮定して，複数の処理を同時に実行するという処理を 1 つのステップと考えたとき，ある組合せ問題 Q を解く多項式アルゴリズムが存在するならば問題 Q は**クラス NP**(non-deterministic polynomial の略) に属するという．明らかにクラス P はクラス NP の部分集合であるが，真部分集合であるかどうかの証明はなされていない．

決定問題 A と Bを考え，A の任意の問題例 Pが多項式時間で完了する処理によって B のある問題例 Q に変換でき，更に

$$(P\text{の解が Yes}) \Longleftrightarrow (Q\text{ の解が Yes}) \tag{2.19}$$

が成立するとき，問題 A は問題 Bに**帰着可能** (reducible) という（厳密には多項式的に帰着可能 (polynomially reducible)）．つまり変換に必要な多項式時間を無視すれば，問題 B は問題 A より難しい問題であると言える．

クラス NP に属する任意の問題 A がある一つの問題 B に帰着可能であるとき，問題 B は **NP 困難** (NP-hard) であるとという．更に NP 困難な問題 B自身がクラス NP に属するとき，問題 Q は **NP 完全** (NP-complete) という．問題 Bが多項式オーダーで解ければ，クラス NP に属する全ての問題が多項式オーダーで解けることになる．従って NP 困難な問

題は，最悪時間量の意味で多項式オーダー時間のアルゴリズムを持つことが絶望視されている．

NP 困難性は決定問題の難易度に関する概念であるが，**数え上げ問題** (counting problem) の難易度に関する概念が**#P 困難性** (#P-hardness) である（"#P"は"Number P"と読む．この"#P"自体には意味はなく，"#"を用いることによって"NP"の概念を数え上げ問題に展開したことを表現している）．数え上げ問題は，個々の問題例においてある性質が成立する実行可能解の数を数え上げる問題である．扱う問題を決定問題から数え上げ問題に変えることで，**クラス#P**，**#P 困難** (#P-hard)，**#P 完全** (#P-complete) を同様に定義することができる．一般に#P 困難な問題は，NP 困難な問題より解くのが難しい問題である．

NP 困難性や#P 困難性などの計算の複雑性に関しては [27] が詳しく，現在までに知られている多くの NP 完全問題のリストを含んでいる．その他 [39, 40] にも NP 完全問題がリストアップされている．

2.2.3 分枝限定法

分枝限定法 (branch-and-bound method) は，主として NP 完全問題に代表されるような難しい問題に適用される手法の 1 つである．基本的なアイデアは，直接解くことが困難な問題をいくつかの**部分問題** (partial problems) に分解し，その全てを解くことで等価的に元の問題を解くというものである．この分解は生成された部分問題にも適用され，原問題は多数の小規模な部分問題に分解されていく．しかし可能な全ての分解を行うと列挙法になってしまうので，それを避けるため各部分問題にある種のテストを加え，その結果部分問題の最適解が求まったり，逆に原問題の最適解を与えないことが判明したりすると，その部分問題を**終端** (terminate) させる．部分問題への分解を**分枝操作** (branching operation)，部分問題のテストによる終端を**限定操作** (bounding operation) という．

例として以下の 0-1 ナップサック問題を考える．

$$P_0: \left| \begin{array}{ll} \text{目的関数} & \sum_{j=1}^{n} c_j x_j \rightarrow \text{最大} \\ \text{制約条件} & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0,1\},\ j = 1,2,\ldots,n \end{array} \right. \tag{2.20}$$

ただし係数 c_j, a_j, b は全て正整数である．この n 変数問題を適当な変数 x_k を 0 と 1 に固定することによって（この変数を**分枝変数** (branching variable) という），2 個の $n-1$ 変

数問題に分解する．生成される2個の部分問題もやはり0-1ナップサック問題であって，

$$P_1: \left| \begin{array}{ll} \text{目的関数} & \sum_{j\neq k} c_j x_j \rightarrow \text{最大} \\ \text{制約条件} & \sum_{j\neq k} a_j x_j \leq b \\ & x_j \in \{0,1\},\ j \neq k \end{array} \right. \tag{2.21}$$

$$P_2: \left| \begin{array}{ll} \text{目的関数} & \sum_{j\neq k} c_j x_j \rightarrow \text{最大} \\ \text{制約条件} & \sum_{j\neq k} a_j x_j \leq b - a_k \\ & x_j \in \{0,1\},\ j \neq k \end{array} \right. \tag{2.22}$$

となる．P_1とP_2の両方を解けば，それぞれの最適解のうち大きい値を持つ方が原問題P_0の最適解を与え，P_0が解けたことになる．

上の分枝操作を生成された部分問題にも適用し可能な限り分解を進めると，図2.9のような**分枝図** (branching diagram; **分枝木**ともいう) を得る．図2.9の最下位層の節点は全変数が固定された部分問題を表し，これらは自明的に解かれる．この分枝操作だけだと実質的に列挙法になるので，全部分問題の一部分のみを実際に処理するように工夫することが必要となるが，以下の性質はこの目的のために有効である．

(1) ある部分問題P_iの最適解が何らかの方法で求まれば，P_iを更に分解する必要はない．

(2) ある部分問題P_i，およびそれから生成される部分問題が原問題P_0の最適解を与えないことが何らかの理由で結論できれば，P_iを更に分解する必要はない．

以上を限定操作といい，(1)，(2) のいずれかに相当すればP_iは終端される．具体的な限定操作は，境界値によるものと優越関係によるものが代表的である．

ここで分枝図における用語を定義する．図2.9で言うP_0に相当する最上位層の節点を**根** (root) という．根からある節点vまでの単純なパスの長さがlであるとき，節点vの**レベル** (level) はl，または節点vはレベルlに位置するという．根のレベルは0である．図2.9におけるP_1とP_2に相当する節点のレベルは1である．また最下位層に位置する節点を**葉** (leaf) という．

レベルlの節点uとレベル$l+1$の節点vが枝で接続されているとき，節点uは節点vの**親** (parent)，節点vは節点uの**子** (child) という．根から節点vへの単純なパス上に存在するvよりレベルの浅い節点を節点vの**先祖** (ancestor)，根からのパスで，節点vをパス上に

持たなければならない単純なパスの終点となり得る節点を節点 v の**子孫** (descendant) という．例えば，根を除く全ての節点は根の子孫である．

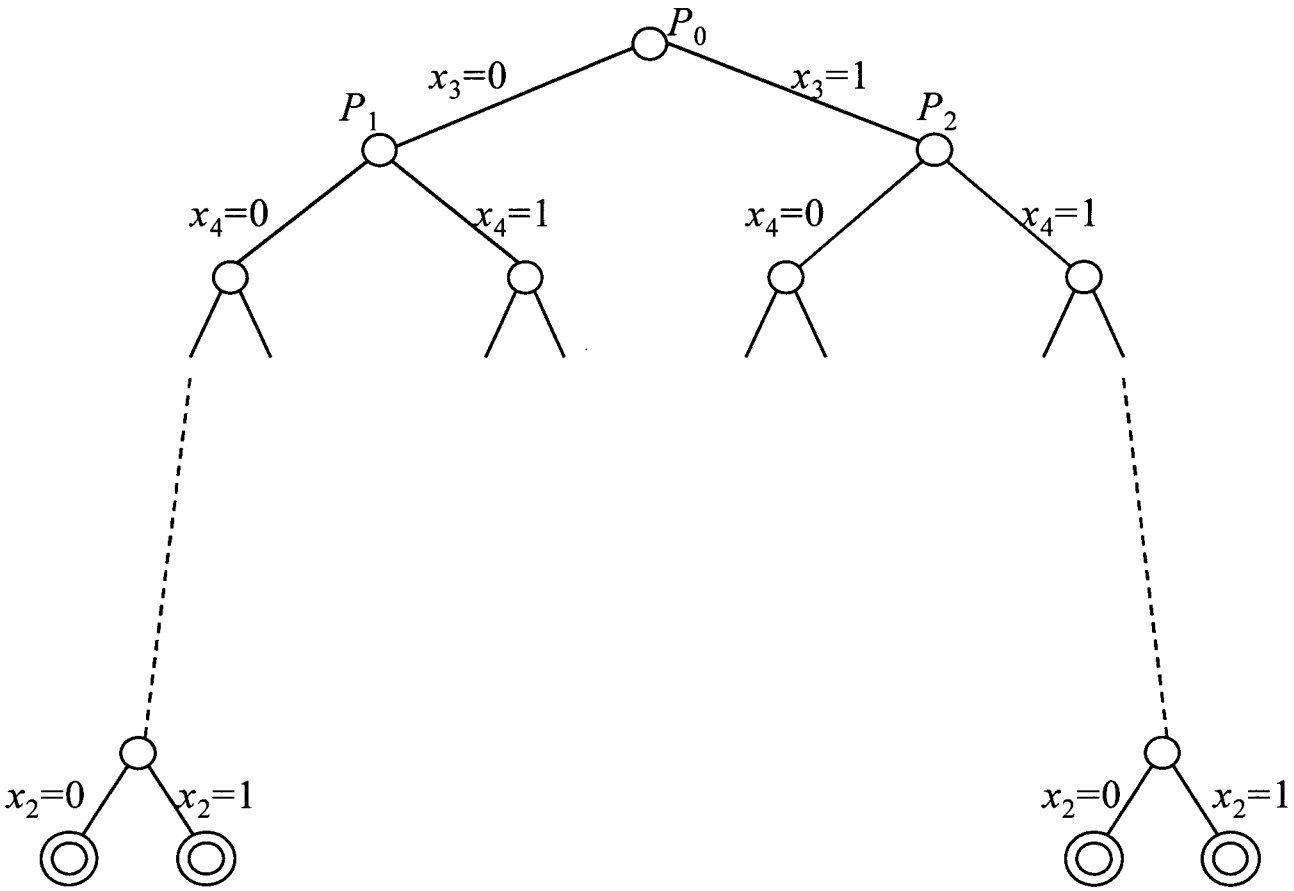


図 2.9 分枝図

第 3 章

総合信頼度とその下界

本章ではまず本論文で扱うネットワークモデルと**ネットワーク信頼度** (network reliability) を定義し，採用したモデルと信頼度の妥当性について考察する．続いて本論文で採用したネットワーク信頼度である総合信頼度の算出方法について触れ，その#P 完全性について述べる．最後に既存の総合信頼度の下界を求める手法をいくつか紹介する．

3.1 モデルの設定と総合信頼度

要素数 n の節点集合 $V = \{v_1, \cdots, v_n\}$，要素数 m の枝集合 $E = \{e_1, \cdots, e_m\}$ からなる単純無向グラフを考える．グラフ中の各枝には正常に機能している状態（**正常状態** : operative state）と故障している状態（故障状態: failed state）の 2 つの状態がある．枝 $e \in E$が正常状態である確率を $p(e)$ とし，各枝の枝正常確率は互いに独立とする．ここで $p : E \to [0, 1]$ である．$p(e)$ を枝 e の**枝正常確率** (edge-operative probability)，あるいは単に**枝確率** (edge probability) と呼ぶ．また節点は常に正常であると仮定する．このような確率グラフを $G = (V, E, p)$，あるいは単に $G = (V, E)$ と表す．確率グラフ G 中の全ての節点が正常な枝によって連結になるとき，G で表現されるネットワークは正常であるとする．これは確率グラフ G が正常な枝からなる極大木を持つことと同値である．このとき，ネットワークが正常である確率を**総合信頼度** (all-terminal reliability : 全節点信頼度ともいう) と呼び，$Rel(G)$ で表す [17].

一般にシステムの信頼度を定義しその妥当性を評価するためには，以下の 3 つの項目を考慮する必要がある [19].

(1) システムはどのような構造をしているのか？

本論文ではネットワーク構造で表現可能なシステムを対象とする．表現不可能な例としては，n 個のうち少なくとも k 個が正常であれば，所定の処理が行えるようなシステム（k-out-of-n システム）があるが，本論文ではそのようなシステムは扱わない．

(2) どのような状態をもって，システムが正常であるとするのか？

正常な枝によってネットワーク中の全ての節点が連結である状態をシステムが正常であるとする．スループットや応答時間などの処理時間に関する問題をとりあげてシステムの信頼性を評価することも多いが，物理的にネットワークが接続されているか否かという連結性に関する問題は，頻度が小さくても起こったときの損害の大きさを考えると非常に重要であるため，本論文では連結性に焦点を絞ることにする．

ネットワークの連結性に関しては，ネットワーク中の特定の 2 節点が正常な枝によって連結であることや，特定の k 節点が正常な枝によって連結であることをシステムの正常状態とすることも多い．それぞれの場合について，システムが正常になる確率は**2 節点間信頼度** (two-terminal reliability)，**k 節点間信頼度** (k-terminal reliability) と呼ばれ，k 節点間信頼度は総合信頼度と 2 節点間信頼度を一般化した信頼度として捕らえることができる [17].

インターネットが多数の LAN を接続しているように，一般にネットワークは階層構造を有する．通常ネットワークの信頼度を考える場合，故障が生じたときの影響度や解析の容易性を考慮して基幹ネットワークのみを対象とし，LAN などの局地的なネットワークを同時に評価することは少ない．その点から言えば，総合信頼度は現実の基幹ネットワークの信頼性を評価するのに最も適した信頼度と思われる．

(3) どのような理由でシステム構成要素が故障するのか？

まず枝の故障は互いに独立であると仮定する．例えば劣化による故障や人為的な破壊による故障などのハードウェア的故障，制御プログラムの誤動作によるソフトウェア的故障など，主なネットワークシステムはこの仮定を満足する．

また枝の故障を考える場合，枝確率が全て同一とは考えにくい．例えば劣化による故障は使用年数や使用状況によって異なるし，テロなどの悪意を持った外部者がネットワークを破壊する場合もその重要性から破壊の標的とする可能性は枝ごとに異なるであろう．このことを踏まえ，本研究では枝確率は枝ごとに異なっていても構わないとする．

一方，節点に相当する機器の故障も現実には考えられる．しかし総合信頼度を評価尺度とする場合，故障する可能性のある節点を等価な枝に変換することにより，全ての節点が故障しないモデルに変換することが可能となる．例えば図3.1の確率グラフ G で，節点 v は確率 p で正常，$1-p$ で故障するとし，その他の節点は故障しないと仮定する．全ての節点が正常，かつ全ての節点が正常な枝によって連結であるとき，確率グラフ G は正常であるとする．一方，確率グラフ G' の節点はこれまでの仮定と同じく常に正常である．このとき確率グラフ G が正常である確率は，確率グラフ G' が正常である確率と等しくなる．

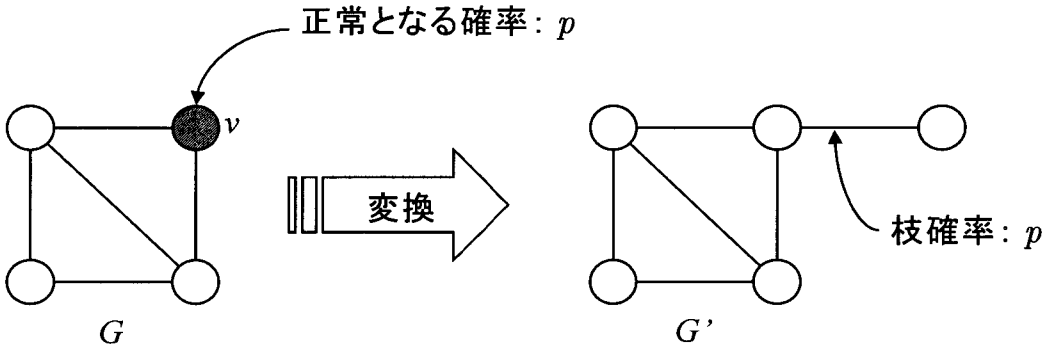


図 3.1 節点故障から枝故障へ変換

3.2 総合信頼度の計算方法

あるネットワーク G の総合信頼度 $Rel(G)$ は次のように求めることが可能である．まずネットワーク G において，枝集合 $E_{op} \subset E$ に属する全ての枝が正常，残りの全ての枝が故障となる状態を考えたとき，各枝の枝確率の独立性からネットワーク G がこの状態になる確率は，

$$\prod_{e \in E_{op}} p(e) \prod_{e \in E - E_{op}} (1 - p(e)) \tag{3.1}$$

となる．$\mathcal{A}$ を G の全域部分グラフの集合とすると，$Rel(G)$ は次式のように表すことができる．

$$Rel(G) = \sum_{G_{op}=(V,E_{op}) \in \mathcal{A}} \{ \prod_{e \in E_{op}} p(e) \prod_{e \in E - E_{op}} (1 - p(e)) \} \tag{3.2}$$

式 (3.2) ではネットワークが正常になる全ての状態について，その確率を計算し合計することによって $Rel(G)$ の値を求めている．よって式 (3.2) の右辺を計算するには，ネットワー

クの全ての状態に対し，ネットワークが正常か否かを調べなければならない．各枝の状態は正常状態と故障状態の2通りであるので，ネットワークの状態は全部で$2^m(m=|E|)$通り存在する．よってこの方法により$Rel(G)$の値を求めるには，枝の本数に対し指数的な計算時間が必要となることがわかる．第1章で記述した通り，さまざまな計算上の工夫を施すことによってこのような単純な方法と比較して少ない計算時間で$Rel(G)$の値を求める研究もなされているが，総合信頼度の値を求める問題が#P完全であることは，J. S. Provan, M. O. Ball[67]とM. Jerrum[37]によって既に示されているので，総合信頼度を枝の本数に対し多項式時間で導出することは絶望視されている．

一般に総合信頼度を計算する際には，次の削除縮約展開式がよく用いられる[56, 54]．

定理 3.1 確率グラフ$G=(V,E)$の任意の枝$e\in E$に対し，以下の式が成立する．

$$Rel(G)=p(e)Rel(G\cdot e)+(1-p(e))Rel(G-e) \tag{3.3}$$

枝eが正常である場合，枝eの両端点は枝eによって接続されているので，確率グラフGが正常である確率は枝eを縮約した確率グラフ$G\cdot e$が正常である確率と等しくなる．逆に枝eが故障している場合は，枝eを削除した確率グラフ$G-e$が正常である確率と等しくなることから式(3.3)を得ることができる．この削除縮約展開式に基づいて総合信頼度の値を算出した場合も，枝の本数に対し指数的な計算時間を必要とする．しかし式(3.3)は再帰構造を持つので，プログラム化したとき非常にシンプルな構造になるため，コンピュータによる数値計算の際によく用いられる．

3.3　主な総合信頼度の下界導出法

総合信頼度を求める問題は#P完全であるため，大規模なネットワークの総合信頼度を厳密に計算することは，計算量理論の観点から非常に困難である．そのため総合信頼度の近似値を現実的な時間で計算することは極めて重要であり，対象とするネットワークが満足している総合信頼度の水準を保証することができるという意味で，総合信頼度の下界を求めることは特に重要である．本節では，これまでに研究された主な総合信頼度の下界導出法を簡単に紹介する．

3.3.1 信頼度多項式による方法

ここでは全ての枝確率が等しく p である場合について考える．このとき，確率グラフ $G=(V,E)$ の総合信頼度 $Rel(G)$ を $Rel(G,p)$ と書くことが多く，グラフ G が自明なときは単に $Rel(p)$ と表すこともある．

ネットワーク G が有する m 本の枝のうち，$m-i$ 本が正常状態，i 本が故障状態となる確率は，

$$p^{m-i}(1-p)^i \tag{3.4}$$

である．ネットワークが正常となる状態のうち，正常な枝の本数が $m-i$ 本，故障している枝の本数が i 本であるような状態の数を F_i とすると，$Rel(p)$ は以下のように表すことができる．

$$Rel(p)=\sum_{i=0}^{m} F_i p^{m-i}(1-p)^i \tag{3.5}$$

式 (3.5) のように，$Rel(p)$ は p の多項式で表わされるので**信頼度多項式** (reliability polynomial) とも呼ばれる [17].

式 (3.5) の係数のうち $F_0,\ldots,F_c$ と $F_d,\ldots,F_m$ は多項式時間で計算可能で，

$$F_i=\begin{cases}\dbinom{m}{i} & i=0,\ldots,c-1\\ \dbinom{m}{i}-C_c & i=c\\ \text{グラフ } G=(V,E) \text{ の極大木の本数} & i=d\\ 0 & i=d+1,\ldots,m\end{cases} \tag{3.6}$$

となる．ただし，c はグラフ G の枝連結度，C_c は大きさ c のカットの数，$d=m-n+1$ である．J. B. Kruskal と G. Katona[41, 48] は，式 (3.5) 中の隣接する係数 F_{i-1} と F_i とに以下のような制約関係があることを示した．

$$0\le F_i^{i-1/i}\le F_{i-1} \qquad \text{(Kruskal-Katona の定理)} \tag{3.7}$$

ここで F_i の肩にかかっている $i-1/i$ は通常の分数計算ではなく lower-pseudopower である (定義と計算方法は [17] を参照)．関係式 (3.7) を利用して，R. M. Van Slyke, H. Frank[84] は $Rel(p)$ について以下の下界を導出した．

$$Rel(p)\ge\sum_{i=0}^{c} F_i p^{m-1}(1-p)i+\sum_{i=c+1}^{d} F_d^{i/d}p^{m-i}(1-p)^i \tag{3.8}$$

この下界を **Kruskal-Katona の下界**という.

また式 (3.5) は以下のように変形できる.

$$Rel(p) = p^{n-1}\sum_{i=0}^{d} H_i(1-p)^i \tag{3.9}$$

$$H_i = \sum_{r=0}^{i} F_r(-1)^{i-r}\begin{pmatrix} d-r \\ i-r \end{pmatrix} \tag{3.10}$$

$$F_i = \sum_{r=0}^{i} H_r\begin{pmatrix} d-r \\ i-r \end{pmatrix} \tag{3.11}$$

式 (3.6), 式 (3.10), 式 (3.11) より, $H_0, \ldots, H_c$と$\sum_{i=0}^{d} H_i$は多項式時間で計算することができる. R. P. Stanley[78] は $1 \leq i < d$ なる i に対し, 式 (3.9) 中の隣接する係数 H_{i-1}と H_iとに以下のような制約関係があることを示した.

$$0 \leq H_{i+1} \leq H_i^{<i+1/i>} \qquad \text{(Stanley の関係式)} \tag{3.12}$$

ここで H_iの肩にかかっている$< i+1/i >$は upper-pseudopower である（定義と計算方法は [17] を参照）. Stanley の関係式 (3.12) を用いて, M. O. Ball, J. S. Provan[6] は $Rel(p)$ に対し以下の下界を導出した.

$$Rel(p) \geq p^{n-1}\sum_{i=0}^{c} H_i(1-p)^i + p^{n-1}\sum_{i=c+1}^{d} \underline{H_i}(1-p)^i \tag{3.13}$$

ただし,

$$\underline{H_i} = \min\{r | \sum j = 0^{i-1}\underline{H_j} + \sum_{j=i}^{d} r^{<j/i>} \geq F_d\} \tag{3.14}$$

式 (3.13) で表される下界を **Ball-Provan の下界**という.

3.3.2　エッジ・パッキング法

まずエッジ・パッキングを定義する.

定義 3.2 グラフ $G = (V, E)$ に対し, 節点集合 V中の全ての節点を有し, 互いに枝排反な G の部分グラフの集合

$$\mathcal{P} = \{G_1, \ldots, G_k | G_i = (V, E_i),\ E_i \subset E,\ E_i \cap E_j = \phi\ (i \neq j)\}$$

を**エッジ・パッキング** (edge-packing) という.

この定義は確率グラフに対しても同様になされる．

確率グラフ $G=(V,E)$ がエッジ・パッキング $\mathcal{P}=\{G_1,\ldots,G_k\}$ を持つ場合を考える．エッジ・パッキングの要素 $G_i,G_j\in\mathcal{P}$ $(i\neq j)$ は互いに共通する枝を持たず，枝集合 Eに属する枝の枝確率は互いに独立であるので，G_iが正常となる確率と G_iが正常となる確率は互いに独立になる．また確率グラフの状態とエッジ・パッキングの要素である部分確率グラフの状態について，明らかに以下のことが言える．

$$\exists i\in\{1,\ldots,k\};\quad G_i\text{が正常}\Longrightarrow G\text{が正常} \tag{3.15}$$

$$\forall i\in\{1,\ldots,k\};\quad G_i\text{が故障}\not\Longrightarrow G\text{が故障} \tag{3.16}$$

式 (3.15) はいずれかの部分確率グラフが正常であれば確率グラフ G は正常であること，式 (3.16) は全ての確率部分グラフが故障であっても，確率グラフ G が故障であるとは言えないことを示している．このことから確率グラフ G が正常である確率は，いずれかの部分確率グラフが正常である確率以上であることがわかる．この関係を定理として証明したのが以下の定理である [17].

定理 3.3 確率グラフ $G=(V,E)$ がエッジ・パッキング $\mathcal{P}=\{G_1,\ldots,G_k\}$ を持つとき，次の不等式が成立する．

$$Rel(G)\geq 1-\prod_{G_i\in\mathcal{P}}(1-Rel(G_i))\ \equiv Low(G,\mathcal{P}) \tag{3.17}$$

式 (3.17) の右辺は確率グラフ G の総合信頼度の下界になっている．このようにエッジ・パッキングを構成し，式 (3.17) を利用して総合信頼度の下界を導出する方法を**エッジ・パッキング法** (edge-packing method) と呼ぶ．

枝確率が全て等しい場合，極大木 $T=(V,E_T)$ の総合信頼度 $Rel(T)$ は，

$$Rel(T)=p^{n-1} \tag{3.18}$$

となる．ここで $n=|V|$ である．従って $\mathcal{P}$を k個の枝排反（互いに共通の枝を持たない）極大木で構成すると，式 (3.17) は次のようになる．

$$Rel(G)\geq 1-(1-p^{n-1})^k \tag{3.19}$$

よって総合信頼度の下界の精度を向上させるためには，枝排反極大木の本数 kを増やせば良いことがわかる．kの値はグラフ $G=(V,E)$ の構造によって取り得る範囲が異なるが，$0 \leq k \leq c$ である．ここで c はグラフ G の枝連結度である．

グラフが有する枝排反極大木の本数に関する研究は古くから関心を持たれ [57, 58, 82]，その後この問題がマトロイド分割問題に帰着できることが J. Edmonds により示された [24]．V. P. Polesskii[62] はそれまでの結果を用いて，グラフ $G=(V,E)$ の枝連結度が c のとき，枝排反極大木の最大本数が少なくとも $\lceil \frac{c}{2} \rceil$ 本であることを示した．ここで $\lceil x \rceil$ は x を越えない最大の整数である．この結果を式 (3.19) に適用すれば，

$$Rel(G) \geq 1-(1-p^{n-1})^{\lceil \frac{c}{2} \rceil} \tag{3.20}$$

となる．式 (3.20) の右辺で表される総合信頼度の下界を **Polesskii の下界**と呼ぶ．

Polesskii の下界は Kruskal-Katona の下界と比較した場合，下界の精度があまり良くないことが [18, 19] によって実験的に確認されている．その後，Kruskal-Katona の下界を凌駕する Ball-Provan の下界が提案されたこともあり，総合信頼度の下界の研究に関しては，エッジ・パッキング法に関する研究は下火になった．しかし Polesskii の下界はあくまでも枝確率が全て等しいネットワークを対象とした下界であり，エッジ・パッキング法自体は枝確率が枝ごとに異なるネットワークに対しても適用できる．これは信頼度多項式に基づく手法にはない大きな特徴である．

第 4 章

極大木によるエッジ・パッキング法

4.1 はじめに

3.3.2節で記したように，信頼度多項式による方法と比較すると下界の精度があまり良くないことが実験的に確認されて以来，総合信頼度の下界に対してエッジ・パッキング法を適用する研究は下火になった．エッジ・パッキング法には，枝確率が異なるネットワークにも適用できるという大きな長所を持っているものの，枝確率が同一の場合と比較すると解析が困難なためこれまであまり研究されていなかった．

本章では枝確率が異なるネットワークを対象とした場合における，極大木によるエッジ・パッキング法が導出する総合信頼度の下界の精度について考察する．その考察を基に精度の良い下界を導出することを目的とした多項式時間エッジ・パッキング構成アルゴリズムを提案し，その性能を数値実験により検証する．また提案したアルゴリズムにあるグラフ変換を組合せて適用することにより，得られる下界の精度を上げることができることを数値実験で示す．このエッジ・パッキング法にグラフ変換を組み合わせた手法は，枝確率が全て等しいネットワークに対して適用しても，Ball-Provan の下界を凌駕することもあるほど精度の良い下界を導出できることを数値例で示す．本研究は枝確率が異なるネットワークの総合信頼度の下界に関する研究の理論的基礎に貢献し，一つの評価基準を与えるだけでなく，これまで枝確率が全て等しいネットワークに対して有効ではないと言われているエッジ・パッキング法がグラフ変換を併用することにより十分に有効な方法になることを示すことを目的としている．

4.2　極大木によるエッジ・パッキング法の特徴

式 (3.17) を再記する.

$$Rel(G) \geq 1 - \prod_{G_i \in \mathcal{P}} (1 - Rel(G_i)) \equiv Low(G, \mathcal{P}) \tag{3.17}$$

枝確率が枝ごとに異なる場合，極大木 $T = (V, E_T)$ の総合信頼度 $Rel(T)$ は,

$$Rel(T) = \prod_{e \in E_T} p(e) \tag{4.1}$$

と書け，$Rel(T)$ は多項式時間で計算可能である．従って $\mathcal{P}$を枝排反極大木で構成すると，式 (3.17) は次のようになる.

$$Rel(G) \geq 1 - \prod_{G_i = (V, E_i) \in \mathcal{P}} (1 - \prod_{e \in E_i} p(e)) \equiv Low(G, \mathcal{P}) \tag{4.2}$$

よって極大木によるエッジ・パッキング $\mathcal{P}$を多項式時間で構成できれば，G の総合信頼度の下界を多項式時間で計算することが可能となる．この極大木によるエッジ・パッキング法を利用して精度の良い，すわなち値の大きな総合信頼度の下界 $Low(G, \mathcal{P})$ を得るために，以下の最適化問題 PST（Packings by Spanning Trees の略）を定式化する.

問題 PST

Maximize　$Low(G, \mathcal{P}_{ST}) = 1 - \prod_{G_i \in \mathcal{P}_{ST}} (1 - \prod_{e \in E_i} p(e))$

Subject to　$\mathcal{P}_{ST}$: G における極大木によるエッジ・パッキング

$G_i = (V, E_i)$

問題 PST は枝確率が同一の場合と異なり，単純に枝排反極大木の本数を増やせばよい訳ではなくなる．この問題は#P 完全であると予想されるが，未だその証明はされていない.

問題 PST の解の精度を上げるためには，$0 \leq Rel(G_i) \leq 1$ に留意すると，

[戦略 1]　：　$Rel(G_i)$ の値を大きくする.

[戦略 2]　：　枝排反な極大木の本数 $|\mathcal{P}|$ を大きくする.

の 2 通りの戦略があることがわかる．しかし $|\mathcal{P}|$ を大きくするほど，全てのエッジ・パッキングの要素が極大木になるように枝を適切に選択する必要があり，個々の極大木の総合信頼度を上げるために枝を適切に選択する妨げになるので，両方の戦略を同時に重視することは一般に不可能である．よって次節にそれぞれの戦略を重視したエッジ・パッキング構成アルゴリズムを提案する.

4.3 提案するアルゴリズム

4.3.1 信頼度最大木に関する貪欲法

前述の [戦略 1]，すなわちエッジ・パッキングの要素となる極大木の総合信頼度を大きくする戦略に基づいて，ネットワーク $G=(V,E)$ の極大木によるエッジ・パッキング $\mathcal{P}_{ST_1}$ を構築するアルゴリズム PST1(Packings of Spanning Trees 1) を以下に示す.

Procedure PST1(G)
begin
1 $\mathcal{P}_{ST_1} \leftarrow \phi$;
2 **while** G is connected **do**
3 $G' \leftarrow$ the most reliable spanning tree in G;
4 $G \leftarrow G - G'$;
5 $\mathcal{P}_{ST_1} \leftarrow \mathcal{P}_{ST_1} \cup \{G'\}$;
6 **end**
end

アルゴリズム PST1 は貪欲法であり，ネットワーク G から総合信頼度最大となる極大木（以下，信頼度最大木と呼ぶ）を摘出してエッジ・パッキングの要素とする，という処理を可能な限り継続する．信頼度最大木は，J. B. Kruskal[47] や R. C. Prim[66] による最大木を探索するアルゴリズムを利用して獲得することができる．以下にグラフ $G=(V,E)$ の最大木 $T=(V,E_T)$ を探索する Kruskal のアルゴリズムを示す.

Procedure KRUSKAL(G)
begin
1 $T \leftarrow \phi$;
2 $Q \leftarrow$ a queue containing all edges in E in non-decreasing order of cost;
3 **while** $|T| < |V| - 1$ **do**
4 take out and delete edge e from the front of Q;
5 **if** $T \cup e$ does not have any circuits **then** $T \leftarrow T \cup e$;
6 **end**
end

Kruskal のアルゴリズムは，枝集合 Eの中からコストの大きい順に枝を取り出し，構築中の木に追加しても閉路ができないようなら追加する，という手順を繰り返す貪欲法であ

る．Kruskal のアルゴリズムの計算オーダーは $O(|E|\log|E|)$ である．一方，Prim のアルゴリズムの計算オーダーは $O(|V|^2)$ である．信頼度最大木は以下の定理により，枝確率をコストとみなしたときの最大木と同一になる．

定理 4.1 確率グラフ $G=(V,E)$ の信頼度最大木は，枝確率をコストと見たときの最大木と同一である．ただし，$\forall e \in E;\ 0<p(e)<1$ とする．

証明

$T=(V,E_T)$ を $G=(V,E)$ における信頼度最大木，$T_0=(V,E_{T0})$ を任意の極大木とすると，log関数の単調増加性から，

$$
\begin{aligned}
Rel(T) \geq Rel(T_0) &\iff \prod_{e\in E_T} p(e) \geq \prod_{e\in E_{T0}} p(e) > 0 \\
&\iff \log\{\prod_{e\in E_T} p(e)\} \geq \log\{\prod_{e\in E_{T0}} p(e)\} \\
&\iff \sum_{e\in E_T} \log\{p(e)\} \geq \sum_{e\in E_{T0}} \log\{p(e)\}
\end{aligned}
$$

従って，信頼度最大木 Tは枝確率の対数をコストとみなしたときの G の最大木になっている．最大木を Kruskal のアルゴリズムによって探索する場合，得られる最大木はキュー Q における枝の順序に依存する．log関数の単調増加性より，枝確率の対数をコストとした場合も枝確率自体をコストとした場合も，キュー Q における枝の順序は同一である．従って Kruskal のアルゴリズムは，どちらの場合も同じ極大木を探索する．

証明終

アルゴリズム PST1 の計算オーダーに関しては以下の定理が成立する．

定理 4.2 グラフ $G=(V,E)$ にアルゴリズム PST1 を適用したときの計算オーダーは $O(c\cdot O(\mathrm{MAXTREE}))$ である．ここで c はグラフ G の枝連結度，$O(\mathrm{MAXTREE})$ は最大木を探索するアルゴリズムの計算オーダーである．

証明

グラフ G が有する枝排反極大木の最大数は c であるので，2行目の **while** ループの繰り返し回数は高々c回である．グラフの連結性を調べるのに必要な計算オーダーは $O(|V|+|E|)$ で十分であり [33]，最大木を探索するアルゴリズムより計算オーダーは一般に小さい．よって，全体では $O(c\cdot O(\mathrm{MAXTREE}))$ となる．

証明終

最大木を見つけるアルゴリズムとしてPrimのアルゴリズムを採用した場合，枝連結度cのグラフにおいては全ての点の次数がc以上であることから，

$$\frac{c\cdot|V|}{2}\leq|E|$$
$$\therefore\ c\leq\frac{2|E|}{|V|}. \tag{4.3}$$

となり，アルゴリズムPST1の計算オーダーは$O(c\cdot O(\mathrm{MAXTREE}))=O(|V|\cdot|E|)$となる．

4.3.2 マトロイド分割に基づく方法

次に[戦略2]，すなわち枝排反極大木の本数を多くする戦略に基づいてアルゴリズムを構築する．J. Edmonds [24]はマトロイド分割のアルゴリズムを提案し，そのアルゴリズムを適用することにより，要素数最大の極大木によるエッジ・パッキングを構成することができることを示した[23]．Kをグラフ$G=(V,E)$に対する極大木によるエッジ・パッキングの最大要素数とし，以下の問題PST'を考える．

問題PST'

$$\begin{aligned}
\text{Maximize}\quad & Low(G,\mathcal{P}_{ST})=1-\prod_{G_i\in\mathcal{P}_{ST}}(1-\prod_{e\in E_i}p(e))\\
\text{Subject to}\quad & \mathcal{P}_{ST}:G\text{の極大木によるエッジ・パッキング}\\
& |\mathcal{P}_{ST}|=K\\
& G_i=(V,E_i)
\end{aligned}$$

問題PST'は問題PSTの部分問題であるが，その計算量についてはやはりまだ何の結果も得られていない．そこで以下に問題PST'に対するヒューリスティック・アルゴリズムを提案する．

アルゴリズムの表記を簡略化するためにいくつかの準備を行う．まず枝eの枝確率に対し，$p(e)\in(0,1)$，すなわち$p(e)\neq0$かつ$p(e)\neq1$とする．$p(e)=0$はその枝が存在しないことと等価であり，$p(e)=1$は枝が故障しないことを表すので枝の両端の節点を同一の節点とみなすことと等価であるので，この仮定は一般性を失わない．

新たに以下の記号を定義する．

K ： グラフGに対する極大木によるエッジ・パッキングの最大要素数

$\mathcal{P}_{ST}$ ： $\{G_1,\ldots,G_K\}$, $G_i=(V,E_i)$は極大木

$$
\begin{aligned}
G_{K+1} &: (V, E_{K+1}), \quad E_{K+1} = E - E_1 - \cdots - E_K \quad (G_{K+1}\text{は極大木ではない}) \\
I &: \{1, \ldots, K, K+1\} \quad (\text{添字集合}) \\
P(e) &: \frac{Rel(G_i)}{p(e)} = \begin{cases} \prod_{f \in G_i - e} p(f) & e \in G_i \in \mathcal{P}_{ST} \\ 0 & e \in G_{K+1} \end{cases} \\
\Delta(e, f) &: \begin{cases} -\dfrac{(p(e) - p(f))(P(e) - P(f))}{(1 - Rel(G_i))(1 - Rel(G_j))} & e \in G_i,\ f \in G_j\ (i, j \in I,\ i \neq j) \\ 0 & e \in G_i,\ f \in G_j\ (i, j \in I,\ i = j) \end{cases}
\end{aligned}
$$

ここで以下の用語を定義する.

定義 4.3 $\mathcal{P}_{ST} = \{G_1, \ldots, G_K\}$ をグラフ $G = (V, E)$ の極大木によるエッジ・パッキングとする. 2本の枝 $e_i \in G_i$ と $e_j \in G_j$ $(i, j \in I,\ i \neq j)$ に対し, $G_i^* = G_i - e_i \cup e_j$, $G_j^* = G_j - e_j \cup e_i$ とする. $\mathcal{P}_{ST}^* = \mathcal{P}_{ST} - \{G_i, G_j\} \cup \{G_i^*, G_j^*\}$ もまた極大木による大きさ K の G のエッジ・パッキングになっているとき, 枝 e_i と枝 e_j は**交換可能である**という.

以下に交換可能な2本の枝に関する定理を与える.

定理 4.4 $\mathcal{P}_{ST} = \{G_1, \ldots, G_K\}$ をグラフ $G = (V, E)$ の極大木によるエッジ・パッキングとする. 枝 $e_i \in G_i$ と枝 $e_j \in G_j$ $(i, j \in I,\ i \neq j)$ が交換可能であるならば以下の等式が成立する. ただし $\mathcal{P}_{ST}^* = \mathcal{P}_{ST} - \{G_i, G_j\} \cup \{G_i^*, G_j^*\}, G_i^* = G_i - e_i \cup e_j,\ G_j^* = G_j - e_j \cup e_i$ である.

$$
Low(G, \mathcal{P}_{ST}^*) - Low(G, \mathcal{P}_{ST}) = \Delta(e_i, e_j)(1 - Low(G, \mathcal{P}_{ST})) \tag{4.4}
$$

証明

$$
\begin{aligned}
Low(G, \mathcal{P}_{ST}) &= 1 - \prod_{k=1}^{K} (1 - Rel(G_k)) \\
&= 1 - (1 - p(e_i)P(e_i))(1 - p(e_j)P(e_j)) \prod_{\substack{k=1 \\ k \neq i,j}}^{K} (1 - Rel(G_k)) \\
&= 1 - \frac{(1 - p(e_i)P(e_i))(1 - p(e_j)P(e_j))}{(1 - Rel(G_i))(1 - Rel(G_j))} (1 - Low(G, \mathcal{P}_{ST})), \\
Low(G, \mathcal{P}_{ST}^*) &= 1 - (1 - p(e_i)P(e_i))(1 - p(e_j)P(e_j)) \prod_{\substack{k=1 \\ k \neq i,j}}^{K} (1 - Rel(G_k)) \\
&= 1 - (1 - p(e_i)P(e_j))(1 - p(e_j)P(e_i)) \prod_{\substack{k=1 \\ k \neq i,j}}^{K} (1 - Rel(G_k))
\end{aligned}
$$

$$= 1 - \frac{(1-p(e_i)P(e_j))(1-p(e_j)P(e_i))}{(1-Rel(G_i))(1-Rel(G_j))}(1-Low(G,\mathcal{P}_{ST})).$$

よって，

$$\begin{aligned}
&Low(G,\mathcal{P}^*_{ST}) - Low(G,\mathcal{P}_{ST}) \\
&= \frac{1-Low(G,\mathcal{P}_{ST})}{(1-Rel(G_i))(1-Rel(G_j))} \cdot \{(1-p(e_i)P(e_j))(1-p(e_j)P(e_i)) \\
&\qquad\qquad -(1-p(e_i)P(e_i))(1-p(e_j)P(e_j))\} \\
&= -\frac{(p(e_i)-p(e_j))(P(e_i)-P(e_j))}{(1-Rel(G_i))(1-Rel(G_j))}(1-Low(G,\mathcal{P}_{ST})) \\
&= \Delta(e_i,e_j)(1-Low(G,\mathcal{P}_{ST})).
\end{aligned}$$

証明終

$\Delta(\cdot,\cdot)$ の定義より，$p(e) > p(f)$ かつ $P(e) < P(f)$ が成立する，または $p(e) < p(f)$ かつ $P(e) > P(f)$ が成立するとき，$\Delta(e,f) > 0$ となる．よって定理 4.4は，エッジ・パッキングを構成する極大木同士の信頼度の差が大きくなるほど，そのエッジ・パッキングによって得られる総合信頼度の下界の精度が上がることを示している．

定理 4.4に基づき，極大木間の信頼度の差が大きくなるようにグラフ G のエッジ・パッキング $\mathcal{P}_{ST_2}$ を構成するヒューリスティック・アルゴリズム PST2(Packings of Spanning Trees 2) を以下に提案する．

Procedure PST2
begin
1 $\mathcal{P}_{ST_2} = \{G_1,\ldots,G_K\}$
  $\leftarrow$ an edge-packing whose cardinality is maximum K by Edmonds's algorithm;
2 **for** each $i \in \{1,\ldots,K\}$ **do** $Rel(G_i) \leftarrow \prod_{e\in G_i} p(e)$;
3 $G_{K+1} \leftarrow G - G_1 - \cdots - G_K$, $Rel(G_{K+1}) = 0$;
4 **for** each edge $e \in E$
5 **if** $\exists i \in I;\ e \in G_i$ **then**
6 $l(e) = i$, $P(e) = Rel(G_i)/p(e)$;
7 **end**
8 **end**
9 $Q \leftarrow$ the queue containing all edges in E in non-decreasing order of edge probability;
10 **while** $|Q| \neq \phi$ **do**
11 Take out and delete an edge e from the top of Q;

```
12    δ ← 0, e_δ ← null;
13    for each f ∈ Q do
14      Compute Δ(e, f);
15      if Δ(e, f) > δ and e and f are exchangeable
16        δ ← Δ(e, f), e_δ ← f;
17      end
18    end
19    if δ > 0 then
20      Swap l(e) and l(e_δ)
21      for all g ∈ G_{l(e)} − e do P(g) = P(g)p(e_δ)/p(e);
22      for all g ∈ G_{l(e_δ)} − e_δ do P(g) = P(g)p(e)/p(e_δ);
23      Rel(G_{l(e)}) = Rel(G_{l(e)})p(e_δ)/p(e), Rel(G_{l(e_δ)}) = Rel(G_{l(e_δ)})p(e)/p(e_δ);
24    end
25  end
26  P_{ST2} ← {G_1, ..., G_K} according to l(e);
end
```

アルゴリズム PST2 の概要を以下に記す．Step 1 では Edmond のアルゴリズムにより，最大要素数の極大木によるエッジ・パッキングを構築している．Step 2 から Step9 は以後に行う枝交換の準備処理である．Step 11 から Step18 では交換することにより下界の精度を上げることの出来る 2 本の枝を探索し，Step 19 以降で交換する．

定理 4.5 アルゴリズム PST2 の計算オーダーは $O(|E|^3)$ である．

証明

行 1 での Edmonds のマトロイド分割に必要な計算時間は $O(|E|^3)$ である [23]．2 つの枝が交換可能であるか確認するためには $O(|E|)$ 時間必要とするため，Step10 からの **while** ループ全体で $O(|E|^3)$ 時間必要となり，その他の処理に必要な計算はこれより低いオーダーである．

証明終

4.4　グラフの次数を増加させるグラフ変換の効果

4.2節で述べたように，エッジ・パッキング法によって得られる総合信頼度の下界の精度を上げるための戦略の一つは，エッジ・パッキングの要素数 $|\mathcal{P}|$ を大きくすることである．

しかし対象のグラフの枝連結度が c のとき $|\mathcal{P}|$ は高々 c であるので，枝連結度が小さなグラフに対してはエッジ・パッキング法で求める下界の精度があまり期待できない可能性がある．そこで枝連結度（正確にはグラフの次数）を上げるために，次に示す 3 つの確率的に等価なグラフ変換を用いる．これらのグラフ変換を任意の確率グラフに適用すると，その総合信頼度を保存しつつ次数を少なくとも 3 以上のグラフに変換することができる．これらのグラフ変換は総合信頼度の真値を計算する際によく用いられる変換である [17]．確率的に等価なグラフ変換としては他にデルタスター変換やトリサブグラフ・K_4変換などがあるが，本研究ではグラフの次数を上げることを目的としているのでこれらの変換は用いていない [29, 63, 64]．

以下にまず 2 つのグラフ変換に関する定理を示す．

定理 4.6 確率グラフ $G=(V,E)$ 中の次数 1 の節点を $v \in V$，この節点を端点とする枝を $e \in E$とする．このとき以下の等式が成立する．

$$Rel(G) = p(e)Rel(G-e) \tag{4.5}$$

確率グラフ G から確率グラフ $G-e$ へのグラフ変換を**短絡除去** (terminal deletion) という．

定理 4.7 確率グラフ $G=(V,E)$ 中の次数 2 の節点を $v \in V$，この節点を端点とする枝を $e=(v,u) \in E$, $f=(v,w) \in E$とする．$g=(u,w) \notin E$とすると，以下の等式が成立する．

$$Rel(G) = \{1-(1-p(e))(1-p(f))\}Rel(G-e-f \cup g) \tag{4.6}$$

ただし，加えられた枝 gの枝確率は

$$p(g) = \frac{p(e)p(f)}{1-(1-p(e))(1-p(f))} \tag{4.7}$$

である．確率グラフ G から確率グラフ $G-e-f \cup g$へのグラフ変換を**直列縮退** (series reduction) という．

定理 4.6より，短絡除去を繰り返し適用すれば，与えられた確率グラフの次数は少なくとも 2 以上になることが分かる．同様に定理 4.7より，直列縮退を繰り返し適用すれば，与えられた次数 2 以上の確率グラフの次数は少なくとも 3 以上になることが分かる．

以下に示す並列縮退自体はグラフの次数を下げるグラフ変換であるが，直列縮退と併用すれば更に直列縮退や短絡除去を適用できる可能性が高まる傾向があることが知られている．

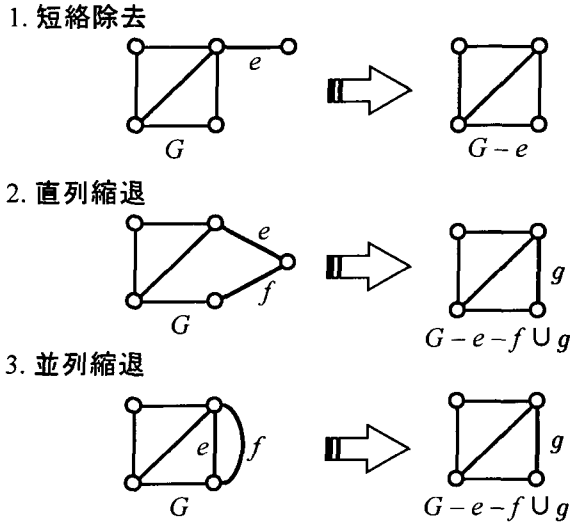


図 4.1 3つのグラフ変換

定理 4.8 確率グラフ $G = (V, E)$ 中の2節点 $u, v \in V$ を両端とする2本の平行枝を $e, f \in E$ とする．$g = (u, v) \notin E$ とすると，以下の等式が成立する．

$$Rel(G) = Rel(G - e - f \cup g) \tag{4.8}$$

ただし，加えられた枝 g の枝確率は

$$p(g) = 1 - (1 - p(e))(1 - p(f)) \tag{4.9}$$

である．確率グラフ G から確率グラフ $G - e - f \cup g$ へのグラフ変換を**並列縮退** (parallel reduction) という．

一般に平行枝を持たない確率グラフに対し上記3つのグラフ変換を適用した場合，変換後の確率グラフの枝連結度は元の確率グラフの枝連結度より大きくなる．従って元の確率グラフにエッジ・パッキング法を適用するよりは，変換後の確率グラフにエッジ・パッキング法を適用した方が，得られる総合信頼度の下界の精度が良くなることが期待できる．また上記3つのグラフ変換により変換された確率グラフは元の確率グラフより枝の本数が少なくなる．変換するために時間が必要であるので必ずしも断定できる訳ではないが，一般にグラフ変換に要する時間よりも下界導出に要する時間の方が大きくなることが多いので，下界を導出する際にグラフ変換を併用した方が全体の計算時間も短くなることが期待できる．

確率グラフ内の枝確率が全て等しく p であるネットワークに直列縮退や並列縮退を適用した場合，定理 4.7，4.8より，新たに加えられる枝の枝確率は p ではなくなる．従って枝確率が全て等しいネットワークに対してのみ総合信頼度の下界を求めることができる手法には，直列縮退や並列縮退を併せて用いることはできない．その点からも，これらのグラフ変換をエッジ・パッキング法と併用することは意義があると考えられる．

4.5 数値実験と考察

この節ではまず Ball-Provan の下界などの総合信頼度による手法に劣ると言われているエッジ・パッキング法が，4.4節で紹介した 3 つのグラフ変換を組み合わせることによって Ball-Provan の下界より精度の良い解を導出することができることを 1 つのネットワークを例に挙げて示す．次にこのグラフ変換がネットワークの構造に依存することなくエッジ・パッキング法の精度を向上させることを数値実験により示す．その後，提案した 2 つのアルゴリズム PST1 と PST2 による総合信頼度の下界の精度を比較する．最後に小さめのネットワークに対し総合信頼度の真値を求め，2 つのアルゴリズムによる下界の精度を検証する．以下簡単のため，各手法によって得られる総合信頼度の下界に名前を付ける．

- L1： アルゴリズム PST1 による下界
- L2： アルゴリズム PST2 による下界
- L1'： 3 つのグラフ変換とアルゴリズム PST1 を併用して得られる下界
- L2'： 3 つのグラフ変換とアルゴリズム PST2 を併用して得られる下界

数値実験は Dell Dimension XPS R400 (CPU:PentiumII 400MHz, Memory:128MB) 上で行い，プログラムは C 言語を用いて作成した．

4.5.1 信頼度多項式による方法との比較

全ての枝に同じ枝確率を付与した図 4.2で表されるネットワーク（$|V| = 14, |E| = 24$）に対し本章で提案したアルゴリズム PST1 と PST2，またそれらに 3 つのグラフ変換を組み合わせた手法によって下界を導出し，信頼度多項式による方法である Kruskal-Katona の

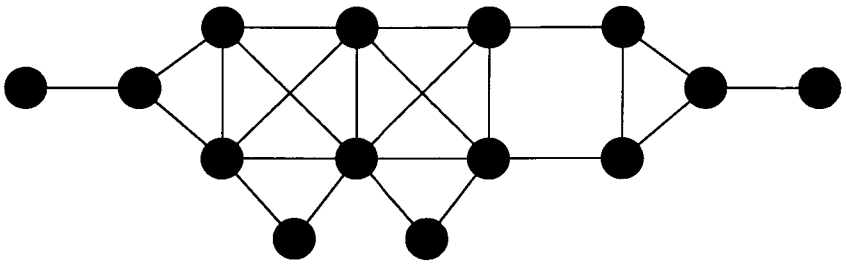


図 4.2 実験対象ネットワーク（枝確率が全て等しい）

下界やBall-Provanの下界と比較した．結果を表4.1に示す．表4.1中のRelは総合信頼度の真値を表し，定理3.1の削除縮約展開式を用いて計算した．

表 4.1 図4.2のネットワークに対する総合信頼度の下界

p	Kruskal-Katona	Ball-Provan	L1,L2	L1',L2'	Rel
0.9	0.566834	0.633056	0.254186	0.715963	0.767377
0.95	0.796558	0.837755	0.513342	0.877137	0.890846
0.99	0.973225	0.976581	0.877521	0.979088	0.979606
0.995	0.988195	0.989111	0.936915	0.989773	0.989901

表4.1中でL1とL2，L1'とL2'が同じ枠に入っているのは，いずれも手法も同じ下界を導出したことを意味する．この結果から，図4.2のネットワークに対しては枝確率が$p = 0.9, 0.95, 0.99, 0.995$のいずれの場合においても，エッジ・パッキング法単体ではKruskal-Katonaの下界ほど精度の良い下界を導出することができなかったが，グラフ変換を組み合わせることによって信頼度多項式による方法で最も優れているBall-Provanの下界よりも精度の良い下界を導出できることがわかる．

次に図4.3で示すネットワークを考える．これは図4.2のネットワークと同じ構造を有するが，枝確率が高い枝と低い枝が混在している．ここでは信頼度の高い枝は，信頼度の低い枝2本から成る平行枝で表現できるとする．すなわち，高い方の枝確率をp_H，低い方をp_Lとしたとき，

$$p_H = 1 - (1 - p_L)^2 \tag{4.10}$$

と書けるとする．信頼度の高い枝を信頼度の低い2本の平行枝に置換すれば全ての枝確率が等しくなるので，このネットワークについては信頼度多項式による手法も適用可能とな

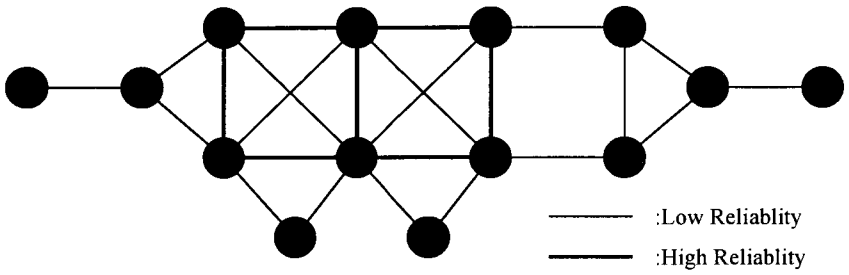


図 4.3 実験対象ネットワーク（2 種類の枝確率を持つ）

る．これに対しエッジ・パッキング法は，枝確率の分布がどのようであっても適用できる．

表 4.2 図 4.3 のネットワークに対する総合信頼度の下界

p_L	Kruskal-Katona	Ball-Provan	L	L'	Rel
0.9	0.543083	0.627801	0.254186	0.763963	0.767752
0.95	0.776182	0.835923	0.513342	0.890418	0.890871
0.99	0.970974	0.976487	0.877521	0.979603	0.979606
0.995	0.987556	0.989087	0.936915	0.989900	0.989901

表 4.2に計算結果を示す．この場合においても，エッジ・パッキング法にグラフ変換を組み合わせた手法が Ball-Provan の下界より精度の良い下界を導出することがわかる．ただし枝確率が等しい，または式 (4.10) の条件を有する任意のネットワークに対し，提案した手法が Ball-Provan の下界より常に有効であるとは言えず，ネットワークの構造と付与する枝確率の値によって優劣が異なる．これを裏付けるため，節点数が 10，枝数が 15〜25 のグラフをランダムに 30 個作成し，枝確率を 0.7, 0.8, 0.9, 0.95, 0.99, 0.995 の 6 通りに付与して数値実験を行った．その結果，20.8%のネットワークに対しては提案した手法の方が優れていた．枝確率が全て等しい場合は Ball-Provan の手法の方が優れる傾向があると言えるが，従来のエッジ・パッキング法に比べれば十分に優れた手法であると言える．

4.5.2 グラフ変換の効果

数値実験の対象ネットワークとして**ランダムグラフ** (random graph) を採用した．節点数 n，枝数 m のランダムグラフとは，枝の両端点を n 個の節点の中からランダムに選択して作成されるグラフで，$R(n,m)$ と記される．両端点をランダムに選択するため平行枝が作

成される可能性があるが，今回は既に同じ両端点を持つ枝が存在するような両端点が選択されたときには，両端点を再設定して平行枝が生じないようにした．ここでは点数 $n = 50$ とし，枝数 $m = 60, 80, \ldots, 200$ とし，それぞれの組合せに対し 50 個ずつ，合計 400 個のネットワークを作成した．枝確率は [0.9,1) 上の一様分布に従うと仮定し，ランダムに設定した．400 個のネットワークのうち，331 個のネットワークについてはその次数が 3 未満であったためグラフ変換を適用することができた．元のネットワークに対しエッジ・パッキング法を適用した場合と，グラフ変換を施した後でエッジ・パッキング法を適用した場合について総合信頼度の下界を求め，その精度を比較した．結果を表 4.3と図 4.4に示す．

表 4.3 3 つのグラフ変換の効果

	Applicable	Average of				Cases on L1			Cases on L2		
m	Networks	c	c'	L1'/L1	L2'/L2	$>$	$=$	$<$	$>$	$=$	$<$
60	50	1.00	3.00	1.736	1.757	50	0	0	50	0	0
80	50	1.00	3.00	1.500	1.512	50	0	0	50	0	0
100	50	1.02	3.00	1.336	1.374	50	0	0	50	0	0
120	50	1.10	3.00	1.271	1.314	50	0	0	50	0	0
140	50	1.48	3.00	1.183	1.148	50	0	0	48	0	2
160	42	1.57	3.02	1.135	1.066	42	0	0	27	0	15
180	28	1.75	3.18	1.097	1.076	27	1	0	25	0	3
200	11	1.82	3.55	1.126	1.073	11	0	0	9	0	2

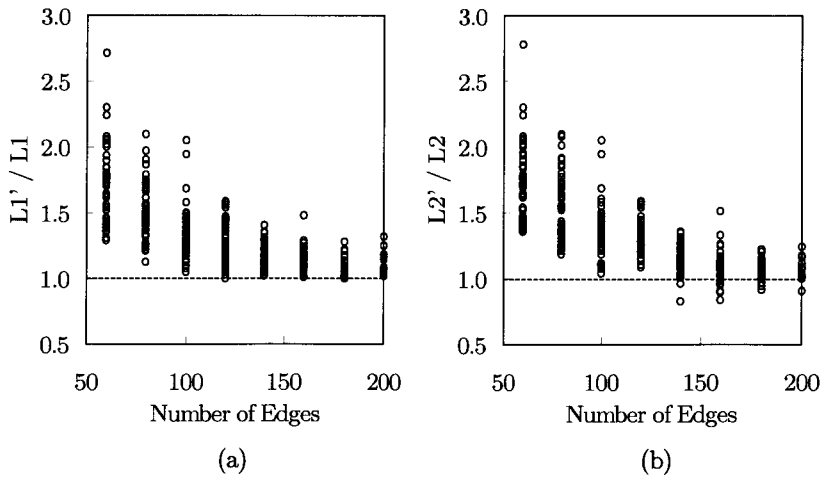


図 4.4 3 つのグラフ変換の効果

表4.3において，Applicable Networkの列はグラフ変換を適用できたネットワークの数，記号>, =, <の列はそれぞれグラフ変換を併用することにより下界の精度が良くなった場合，精度に変化がなかった場合，精度が落ちた場合の数を表している．図4.4(a),(b)における横軸はmの値，縦軸はそれぞれL1'/L1，L2'/L2の値を示している．これらの結果から3つのグラフ変換はほとんどのネットワーク（厳密にはL1については99.7%，L2については93.75%）に対し，下界の精度を向上させることができたことが分かる．

4.5.3 アルゴリズムPST1とPST2との比較

ここでも対象のネットワークとしてランダムグラフ$R(50,m)$を採用し，枝数$m = 100, 200, \ldots, 1000$とした．それぞれの枝数に対し連結で平行枝のないネットワークを50個ずつ，合計500個作成し，枝確率は[0.9,1)上の一様分布に従うとしてランダムに設定した．下界を導出するとき3つのグラフ変換も併用したが，$m \geq 300$については次数3未満のグラフがなかったので，グラフ変換を適用することはできなかった．数値実験の結果を表4.4と図4.5に示した．

表 4.4 PST1とPST2の比較

	Average of								Cases that		
m	c	$\lvert\mathcal{P}_{ST_1}\rvert$	$\lvert\mathcal{P}_{ST_2}\rvert$	L1'	L2'	L1'/L2'	t_1	t_2	>	=	<
100	1.0	1.3	2.0	0.30842	0.31740	0.97415	***	***	8	15	27
200	2.9	2.5	3.3	0.55756	0.54511	1.02459	***	***	27	11	12
300	6.0	4.2	5.6	0.75950	0.74772	1.01639	***	1.7	36	0	14
400	8.8	6.2	7.8	0.87743	0.87115	1.00747	***	3.1	30	0	20
500	12.5	8.4	10.0	0.93462	0.93384	1.00083	***	4.9	22	0	28
600	15.7	10.4	11.9	0.96329	0.96268	1.00065	***	7.1	15	0	35
700	20.0	12.3	14.0	0.97980	0.97998	0.99981	***	9.7	6	0	44
800	24.5	14.4	16.0	0.98961	0.98975	0.99986	***	13.0	3	0	47
900	29.0	16.3	18.0	0.99423	0.99430	0.99993	***	16.8	2	0	48
1000	33.6	18.6	20.0	0.99695	0.99698	0.99997	***	21.5	1	0	47

*** : Uncomputable (less than 1 sec.)

表4.4において，t_1, t_2はアルゴリズムPST1とPST2の実行時間を，記号>, =, < はそれぞれL1'>L2', L1'=L2' , L1'<L2' となるケースの数を表している．図4.5中の横軸はネットワークの枝数m，縦軸はL2/L1の値を示している．これらの結果を見ると枝の本数が

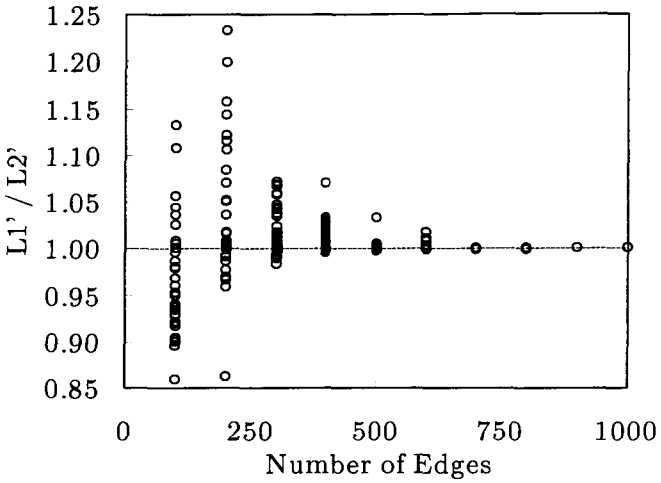


図 4.5 PST1 と PST2 との比較

大きくなるに従って，アルゴリズム PST2 の方がアルゴリズム PST1 より精度の良い下界を導出する傾向があることが分かる．また次に示す更に小型のネットワークに対する数値結果と合わせると，グラフ変換が適用できるぐらい疎なネットワーク，または非常に密なネットワークほど，アルゴリズム PST2 の方が精度の良い下界を導出する傾向があることがわかる．ただし枝本数が多い密なネットワークに対しては，アルゴリズム PST2 による下界は PST1 による下界より精度は劣るものの，その差は極めて小さいことが分かる．一方，実行時間についてはアルゴリズム PST1 は PST2 と比べると，圧倒的に短い時間で下界を導出できることがわかる．

次に総合信頼度の真値と比較することにより，各アルゴリズムによる下界の精度を検証した．大規模なネットワークの総合信頼度の真値は実用的な時間内で計算することができないので，やや小型である節点数 10，枝数 $m = 20, 25, \ldots, 45$ のランダムグラフを対象ネットワークとし，それぞれの枝数に対し 50 個のネットワーク，合計 300 個のネットワークを作成した．ランダムグラフに対するその他の条件はこれまでのものと同等にした．これらのネットワークに対し定理 3.1の削除縮約展開式を用いて総合信頼度の真値を計算し，エッジ・パッキング法による下界と比較した．その結果を，表 4.5と図 4.6に示す．

図 4.6(a), (b) の横軸はネットワークの枝数 m，縦軸は下界と真値との比を表している．枝確率が異なるネットワークの総合信頼度の下界に関する研究はこれまでほとんどなされていないので，これらの結果から我々の提案したアルゴリズムによる下界が精度の良いも

のか否かを結論付けることは難しい．しかし図4.6を見る限り，提案したアルゴリズムが実用上十分に有用である．また枝の本数が増えるにつれ下界の精度が良くなっていることから，ネットワーク中の枝の本数が多いほど実用的な時間内で総合信頼度の真値を計算するのは困難になるが，提案したアルゴリズムが導出する下界の精度は良くなることが期待できる．

表 4.5 総合信頼度の真値との比較

	Average of				
m	L1'	L2'	Rel	L1'/Rel	L2'/Rel
20	0.9082	0.9192	0.9872	0.9202	0.9312
25	0.9361	0.9361	0.9993	0.9319	0.9367
30	0.9599	0.9687	0.9997	0.9601	0.9690
35	0.9775	0.9810	1.0000	0.9775	0.9810
40	0.9899	0.9921	1.0000	0.9899	0.9921
45	0.9952	0.9971	1.0000	0.9952	0.9971

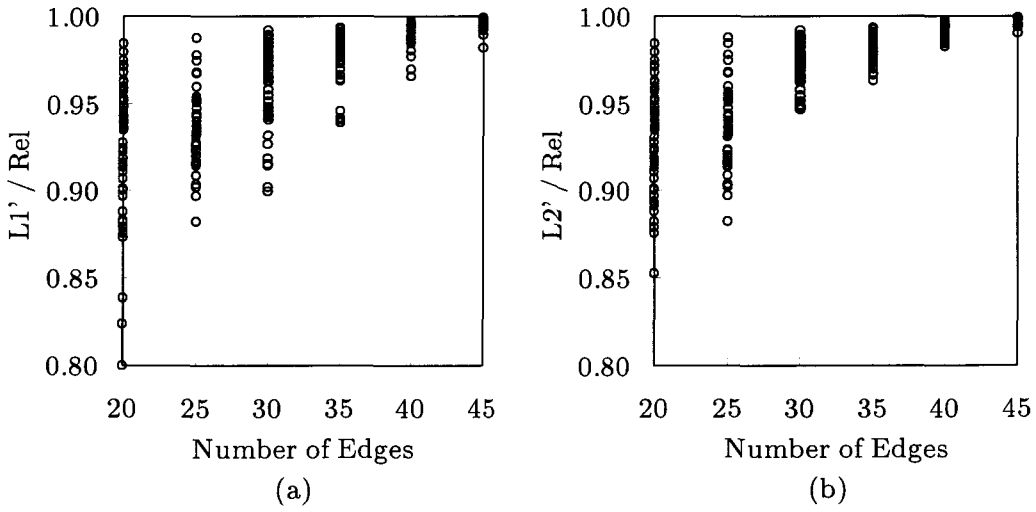


図 4.6 総合信頼度の真値との比較

4.6 おわりに

本章では極大木によるエッジ・パッキング法を利用して，枝確率が異なるネットワークの総合信頼度の下界を多項式時間で導出する2つのアルゴリズムを提案した．これらのアルゴリズムを適用する際に，短絡除去・直列縮退・並列縮退という3つのグラフ変換を併用することにより，得られる下界の精度を向上させる傾向があることを示した．極大木によるエッジ・パッキング法にグラフ変換を組み合わせる手法は枝確率が等しいネットワークに対しても有効で，信頼度多項式による方法の中で最も優れているBall-Provanの下界よりも精度の良い下界を導出することができることを1つのネットワークを例に挙げて示した．このことにより，エッジ・パッキング法は枝確率が等しいネットワークに対しては総合信頼度による手法よりは劣ると言われているが，他の手法と組み合わせることによって十分に有効な手法になることを示した．

個々の極大木の信頼度を大きくする戦略に基づくアルゴリズムPST1と，枝排反極大木の数をなるべく大きくする戦略に基づくアルゴリズムPST2とで一概にどちらが優れているとは言えないが，グラフ変換が十分に適用できるぐらい枝本数の少ない疎なネットワークや枝本数が非常に多い密なネットワークに対しては，アルゴリズムPST2の方が精度の良い下界を導出する傾向があることを数値実験により示した．下界を導出する時間に関しては，ネットワーク中の枝数が大きくなるほどアルゴリズムPST1の方が圧倒的に短い．そのため，応用面を考慮した場合は下界の精度と計算時間について，どちらをどのぐらい重要視するのかによって，良いアルゴリズムの評価が異なるであろう．

問題PSTや問題PST'は#P完全であることが予想されるため，それらの問題に対し多項式時間で最適解を求めるアルゴリズムは恐らく存在しないと考えられる．しかし極大木によるエッジ・パッキング法による総合信頼度の下界の限界を評価するためにも，できるだけ短い計算時間で最適解を求めるアルゴリズムを開発する必要がある．

これまでに枝確率が異なるネットワークの総合信頼度の下界を評価する研究がほとんどなされていないため，提案したアルゴリズムによる下界の精度を評価するのは難しいが，この分野における今後の研究における一つの評価基準になるという点で，その成果は有意義であると考えている．また枝確率が異なるネットワークに対して極大木以外の部分グラフを用いてエッジ・パッキングを構築する際に，本研究の成果は理論的基礎になると思われる．

第 5 章

直並列グラフによるエッジ・パッキング法の改良

5.1 はじめに

エッジ・パッキング法によって多項式時間で総合信頼度の下界を導出するための条件の一つとして，エッジ・パッキングの要素の総合信頼度が多項式時間で計算可能である，という条件がある．第 4章ではエッジ・パッキングの要素として極大木を採用したが，本章では極大木を含むクラスである直並列グラフを採用する．直並列グラフは短絡除去・直列縮退・並列縮退の 3 つのグラフ変換を繰り返し適用することにより孤立点 1 点に変換できるグラフとして定義されるが，直並列グラフで表されるネットワークの総合信頼度は枝数に対し多項式時間で計算できる．

本章では，第 4章で極大木によるエッジ・パッキングについて考察した内容を直並列グラフによるエッジ・パッキングについて拡張することを考え，第 4章と同様，エッジ・パッキングを構築する 2 つのヒューリスティック・アルゴリズムを提案する．数値実験により第 4章で提案したアルゴリズムと比較し，極大木から直並列グラフへの拡張によって導出される下界が改良されたことを示す．

5.2 直並列グラフによるエッジ・パッキング法

直並列グラフは節点数4の完全グラフ K_4 と同相なグラフを部分グラフに持たないグラフとして定義されるが，短絡除去・直列縮退・並列縮退の3つのグラフ変換を用いると，以下のように定義できる [22].

定義 5.1 短絡除去・直列縮退・並列縮退の3つのグラフ変換を繰り返し適用することによって，孤立点1点に変換できるグラフを直並列グラフという.

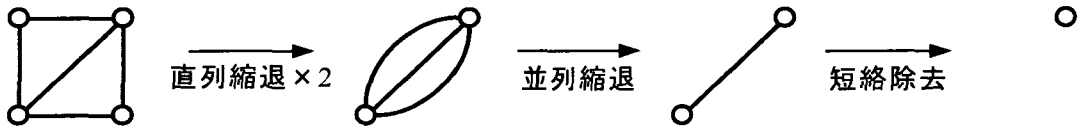


図 5.1 直並列グラフの例

定義5.1から，以下の結果を得る [72].

補題 5.2 直並列グラフの総合信頼度は，計算オーダー $O(m)$ で求めることが可能である.

補題5.2とエッジ・パッキングに関する式 (3.17) より，直並列部分グラフによるエッジ・パッキングを多項式時間で構築すれば，元のグラフの総合信頼度の下界を多項式時間で計算することが可能であることがわかる．そこで全ての点を含む直並列部分グラフ（以下特に明示する必要がない限り，単に直並列グラフと表記する）によるエッジ・パッキングを構成するアルゴリズムを提案する.

5.3 提案するアルゴリズム

まず4.2節で述べた [戦略 1]，すなわち，各 $Rel(G_i)$ が大きな直並列グラフをエッジ・パッキングの要素とする戦略に基づいたアルゴリズムを考える．しかし，総合信頼度最大の直並列グラフを見つける効果的アルゴリズムはまだ開発されていない．そこで代替として，なるべく信頼度の大きな直並列グラフによって構成される，確率グラフ $G=(V,E)$ のエッジ・パッキング $\mathcal{P}_{SP_1}$ を作成するヒューリスティック・アルゴリズム PSP1(Packings of Series-Parallel graphs 1) を以下に提案する.

Procedure PSP1(G)

```
begin
1   Apply the three graph transformations to G repeatedly as long as possible;
2   P_SP1 ← φ;
3   while G is connected do
4     Q ← the queue containing all edges in E in non-decreasing order of edge probability;
5     G' ← φ;
6     while |Q| > 0 do
7       take out and delete edge e from the front of Q;
8       if G' ∪ e is a series-parallel graph then G' ← G' ∪ e;
9     end
10    G ← G − G';
11    P_SP1 ← P_SP1 ∪ {G'};
12  end
end
```

アルゴリズム PSP1 は，4.3.1節で記したアルゴリズム PST1 において木の判定を行っている箇所で直並列グラフであるかどうかの判定を行っているところが異なるだけで，その他の部分はアルゴリズム PST1 と同じ構造になっている．ただし第4章で確認したように，3 つのグラフ変換を組み合わせたアルゴリズムの方が精度の良い下界を導出することが期待できるので，アルゴリズム PSP1 は 3 変換を内部に組み込んでいる．

アルゴリズム PSP1 の計算オーダーについては，次の定理が成立する．

定理 5.3 確率グラフ $G = (V, E)$ にアルゴリズム PSP1 を適用したときの計算オーダーは $O(c|E|^2)$ である．ここで c は G の枝連結度である．

証明

枝のソートには $O(|E| \log |E|)$ 必要である．直並列グラフ Tに 1 本の枝 e を追加してもなお直並列グラフであるか否かの判断には，補題 5.2より $O(|E|)$ の時間が必要である．Step 6 から 9 までの **while** ループの繰り返し回数は $|E|$ 回以下であるので，**while** ループ全体では $O(|E|^2)$ となり，$O(|E| \log |E|)$ より計算時間が大きい．Step 3 の **while** ループの繰り返し回数は，最終的に得られるエッジ・パッキングの要素数に等しいので c 以下になる．従って，Step 3 から Step 12 までの処理には $O(c|E|^2)$ 必要である．

証明終

次に4.2節の [戦略 2] を重視したアルゴリズム，すなわち要素数（直並列グラフの数）がなるべく大きいエッジ・パッキングを構築するアルゴリズムを考える．ここで以下の定理を利用する．

定理 5.4 直並列グラフからなる任意のエッジ・パッキング $\mathcal{P}_{SP}$に対し，

$$|\mathcal{P}_{SP}| \leq |\mathcal{P}_{ST_2}|$$

ここで，$\mathcal{P}_{ST_2}$とはアルゴリズム PST2 により得られるエッジ・パッキングである．

証明

極大木は最も枝本数の少ない全域直並列部分グラフであることから明らか．

証明終

つまり直並列グラフによるエッジ・パッキングの要素数は高々$|\mathcal{P}_{ST_2}|$ であるので，アルゴリズム PST2 によるエッジ・パッキング $\mathcal{P}_{ST_2}$を利用すればよいことがわかる．以下に確率グラフ $G = (V, E)$ の直並列グラフによるエッジ・パッキング $\mathcal{P}_{SP_2}$を作成するアルゴリズム PSP2(Packings of Series-Parallel graphs 2) を示す．

Procedure PSP2(G)
begin
1　Apply the three graph transformations to G repeatedly as long as possible;
2　$\mathcal{P}_{SP_2} \leftarrow$ the edge-packing constructed by calling PST2(G);
3　Compute all-terminal reliability of each spanning tree in $\mathcal{P}_{SP_2}$;
4　Let $\mathcal{P}_{SP_2} = \{G_1, \ldots, G_{|\mathcal{P}_{SP_2}|}\}$ where $Rel(G_i) \geq Rel(G_j)$ for $i < j$;
5　$Q \leftarrow$ the queue containing all edges in $G - G_1 - \cdots - G_{|\mathcal{P}_{SP_2}|}$
　　in non-decreasing order of edge probability;
6　**while** $|Q| > 0$ **do**
7　　Take out and delete an edge e from the front of Q;
8　　**for** $i = 1, \ldots, |\mathcal{P}_{SP_2}|$ **do**
9　　**if** $G_i \cup e$ is a series-parallel graph **then**
10　　$G_i \leftarrow G_i \cup e$;
11　　**break** for-loop;
12　　**end**
13　　**end**
14　**end**
end

アルゴリズム PSP2 は，アルゴリズム PST2 によって作成されたエッジ・パッキング $\mathcal{P}_{ST_2}$で利用されていない枝を，直並列グラフである範囲で $\mathcal{P}_{ST_2}$の要素に追加している．アルゴリズム PSP1 と同様，アルゴリズム PSP2 でも 3 つのグラフ変換を内部に含む構造になっている．

アルゴリズム PSP2 の性能については次の定理が成立する．

定理 5.5 アルゴリズム PST2 と PSP2 により構築されたエッジ・パッキングをそれぞれ $\mathcal{P}_{ST_2}, \mathcal{P}_{ST_2}$とする．このとき次式が成立する．

$$Lower(G, \mathcal{P}_{ST_2}) \leq Lower(G, \mathcal{P}_{SP_2})$$

証明

アルゴリズム PSP2 はアルゴリズム PST2 の結果得られた $\mathcal{P}_{ST_2}$の要素に枝を追加して $\mathcal{P}_{SP_2}$を構成する．$\mathcal{P}_{ST_2} = \{G_1, \cdots, G_{|\mathcal{P}_{ST_2}|}\}$ とし，$G_i \in \mathcal{P}_{ST_2}$に枝を追加してできた直並列グラフを $G'_i \in \mathcal{P}_{SP_2}$とすると，

$$\begin{aligned} \mathcal{P}_{SP_2} &= \{G'_1, \cdots, G'_{|\mathcal{P}_{SP_2}|}\}, \\ Rel(G_i) &\leq Rel(G'_i) \quad (i = 1, \cdots, |\mathcal{P}_{SP_2}|), \\ |\mathcal{P}_{SP_2}| &= |\mathcal{P}_{ST_2}| \end{aligned}$$

となるので，(3.17) 式から導くことができる．

証明終

アルゴリズム PSP2 の計算オーダーについては，次の定理が成立する．

定理 5.6 確率グラフ $G = (V, E)$ にアルゴリズム PSP2 を適用したときの計算オーダーは $O(|E|^3)$ である．

証明

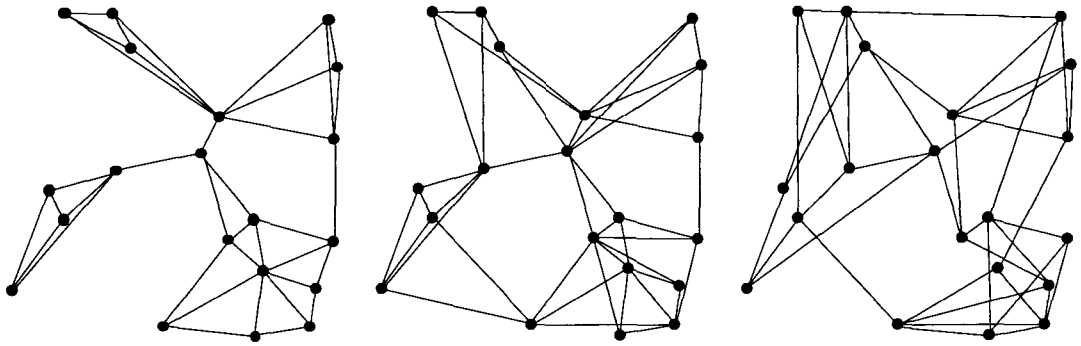
Step 2 には $O(|E|^3)$ の計算時間が必要である．Step 6 の **while** ループの繰り返し回数は高々$|E|$ 回，Step 8 の **for** ループは高々$|\mathcal{P}_{SP_2}|$ 回しか繰り返さない．直並列グラフの判定には $O(|E|)$ 必要であるので，Step 6 から始まるループ全体では，$O(|E|^2 \cdot |\mathcal{P}_{SP_2}|)$ となる．明らかに $|\mathcal{P}_{SP_2}| < |E|$ であるので，全体では $O(|E|^3)$ となる．

証明終

5.4 数値実験と考察

5.4.1 極大木によるエッジ・パッキング法との比較

近接グラフ (near neighbor graph) は，通信ネットワークの形状をモデル化するために考案されたグラフである [38]．近接グラフとは，正方形内に一様分布する n 点に対し，各点について a 番目までの近接点（距離の近い点）を求め，その中の b 点をランダムに選んで枝で結んだグラフで，$N(n,a,b)$ と表せる．近接グラフ $N(n,a,b)$ には nb 本の枝が生成されるが，グラフ中に平行枝が存在する可能性がある．本数値実験では生成された平行枝を1本の枝として扱い，近接グラフ中から平行枝が存在しないようにしたため，枝の数は必ずしも nb 本にはならない．図5.2に近接グラフの例を示した．この図からもわかる通り，近接点数 a が小さいほど，点と枝が密集している密な部分と密集していない疎な部分に分かれる傾向がある．



(a) $N(20,3,3)$　(b) $N(20,5,3)$　(c) $N(20,8,3)$

図 5.2 近接グラフの例

現実的な通信ネットワークを想定して，点数50の大規模な近接グラフに対し，第4章で提案したアルゴリズム PST1，PST2 と本章で提案したアルゴリズム PSP1，PSP2 を適用し，得られた下界を比較した．以下，アルゴリズム PST1 により計算される下界を PST1 とするように，アルゴリズム名とそのアルゴリズムにより得られる下界を同じ用語を用いる．計算対象のネットワークは $N(50,a,b)$; $a=8,16,24,32,40,48$; $b=3,4,5,6,7,8$ の近接グラフで，全36通りの近接点数 a と選択点数 b の組合せに対し，ネットワークを100個ずつ，合計3600個のネットワークを構築した．枝確率は，開区間 $(0.9,1)$ 上の一様分布に

従いランダムに設定した．前述の通り，近接グラフの構築において平行枝が生成された場合は1本の枝と見なすため，各 (a,b) の組に対する100個のネットワークの枝数は同一にはならない．そこで，表5.1に各 (a,b) の組に対する100個のネットワークの枝数の平均値を示した．

表 5.1 $N(50,a,b)$ の平均枝数 $(a=8,16,\cdots,48;\ b=3,4,\cdots,8)$

$b\backslash a$	8	16	24	32	40	48
3	128.2	139.7	143.0	145.0	145.2	145.9
4	160.5	180.0	187.1	190.3	192.0	191.9
5	188.7	219.6	230.1	234.2	236.5	237.0
6	210.8	254.7	269.9	276.8	280.6	281.8
7	228.1	288.1	309.9	318.6	232.0	324.7
8	241.4	320.2	347.6	359.0	365.7	367.4

表 5.2 $N(50,a,b)$ に対する各手法による下界の平均値 $(a=8,16,\cdots,48;\ b=3,4,\cdots,8)$

PST1	PST2
PSP1	PSP2

$b\backslash a$	8		16		24		32		40		48	
3	0.346	0.355	0.393	0.399	0.405	0.402	0.407	0.393	0.417	0.402	0.412	0.393
	0.856	0.799	0.786	0.803	0.764	0.736	0.766	0.668	0.769	0.687	0.769	0.646
4	0.455	0.441	0.509	0.512	0.531	0.538	0.542	0.547	0.561	0.556	0.552	0.550
	0.912	0.690	0.896	0.856	0.898	0.871	0.892	0.873	0.901	0.836	0.904	0.863
5	0.545	0.544	0.620	0.619	0.637	0.638	0.650	0.651	0.655	0.657	0.653	0.656
	0.955	0.892	0.958	0.865	0.952	0.904	0.950	0.919	0.956	0.926	0.962	0.928
6	0.592	0.589	0.689	0.680	0.718	0.716	0.724	0.723	0.737	0.740	0.747	0.748
	0.971	0.798	0.981	0.815	0.981	0.909	0.978	0.927	0.984	0.946	0.984	0.952
7	0.637	0.640	0.743	0.744	0.779	0.773	0.791	0.790	0.801	0.802	0.802	0.802
	0.984	0.923	0.991	0.929	0.992	0.903	0.992	0.937	0.993	0.952	0.993	0.957
8	0.665	0.665	0.790	0.790	0.821	0.816	0.838	0.837	0.844	0.845	0.845	0.846
	0.991	0.900	0.997	0.942	0.997	0.889	0.997	0.933	0.998	0.955	0.997	0.959

表5.2の各欄における左上，右上，左下，右下の順にPST1，PST2，PSP1，PSP2の平均値を示した．全ての (a,b) の組に対し，本論文で提案した手法によるPSP1，PSP2が先に提案したPST1，SPT2を大幅に改善したことがわかる．また $(a,b)=(16,3)$ の場合を除き，PSP1の方がPSP2より平均的に優れていることがわかる．

次に各ネットワークにおいて4つの下界PST1，PST2，PSP1，PSP2をそれぞれ2者比較した結果を表5.3に示した．表の各欄における最上段に100個のネットワークのうちPSP1>PSP2となった個数，以下同様に2段目は左からPSP1>PST1の個数，PSP2>PST1の個数，3段目は左からPSP1>PST2の個数，PSP2>PST2の個数を記した．なお括弧の中の数値は比較した二つの下界が同値であったネットワークの個数を表している．例えば，$(a,b)=(24,3)$の場合，PSP1とPSP2とを比較した場合，45個のネットワークでPSP1，残り55個のネットワークではPSP2の方が良い精度を示したことを表す．またPSP2とPST2を比較した場合は，93個のネットワークでPSP2が良い精度を示し，7個については二つが同値で，PST2の方が良い精度を示したネットワークはないことを表している．

表 5.3 $N(50,a,b)$に対する各下界の1:1比較結果 $(a=8,16,\cdots,48;\ b=3,4,\cdots,8)$

PSP1 > PSP2	
PSP1 > PST1	PSP2 > PST1
PSP1 > PST2	PSP2 > PST2

$b\backslash a$	8	16	24	32	40	48
3	82 100 100 100 100	38 100 100 100 100	45 100 90 100 93(7)	56 100 74 100 82(18)	49 100 80 100 87(13)	62 100 71 100 81(19)
4	98 100 100 100 100	78 100 100 100 100	79 100 100 100 100	64 100 99 100 100	71 100 87 100 91(9)	66 100 94 100 97(3)
5	82 100 94 100 95(5)	99 100 100 100 100	94 100 100 100 100	89 100 100 100 100	89 100 100 100 100	96 100 100 100 100
6	100 100 100 100 100	100 100 99 100 99(1)	100 100 100 100 100	100 100 100 100 100	99 100 100 100 100	97 100 100 100 100
7	100 100 100 100 100	98 100 99 100 100	100 100 100 100 100	100 100 100 100 100	100 100 100 100 100	100 100 100 100 100
8	99 100 95 100 97(3)	100 100 100 100 100	100 100 94 100 96(4)	100 100 100 100 100	100 100 100 100 100	100 100 100 100 100

提案したアルゴリズムによる2つの下界のうちPSP1については，全てのネットワーク

に対し PST1 と PST2 の両方より良い精度を示すことができた．一方 PSP2 は定理 5 で示した通り，PST2 に対しては全て同等以上の精度を示すが，PST1 に対しては必ずしも精度が上回るとは言えない．

また今回提案した 2 つの下界 PSP1 と PSP2 の優劣に関しては，全体的には PSP1 の方が精度が良い．b が小さいほど PSP2 に劣る割合が増す傾向があるが，a に関して主な傾向が見られないことを考慮すると，表 5.1より，枝数が多いほど PSP1 が優位であるとは言い切れない．そこで 3600 個のネットワークに対し，PSP1/PSP2 の値を枝余剰度に関してプロットした．結果を図 5.3に示す．ここで枝余剰度は以下のように定義した．

$$(\text{枝余剰度}) = \frac{(\text{PST2 で利用されなかった枝の本数})}{(\text{PST2 で作成された枝排反極大木の本数})}$$

枝余剰度が大きいほど，アルゴリズム PSP2 において，極大木 1 本当たりに追加する候補となる枝の本数が多いことになる．例えば枝余剰度が 0 の場合は，PST2 において全ての枝が利用されて PSP2 において追加する枝がなく，$\mathcal{P}_{SP_2} = \mathcal{P}_{ST_2}$となることを表している．

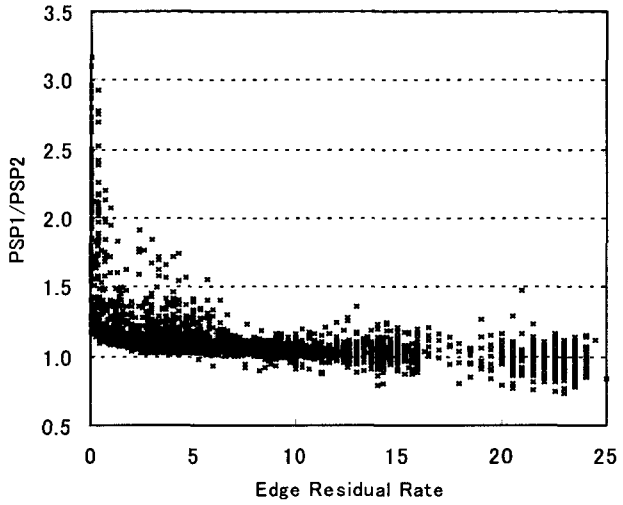


図 5.3 枝余剰度に対する PSP1/PSP2 の値

図 5.3から枝余剰度が大きいほど PSP2 の相対的な精度が上がることが確認できる．従って，与えられたグラフに対しアルゴリズム PST2 を適用した結果，枝余剰度が大きくなるようなグラフほど，PSP2 が PSP1 より精度が良くなる傾向が大きくなると期待できる．しかし一般に枝余剰度の大きさはグラフの構造に依存するため，与えられたグラフに対し，PSP1 と PSP2 のどちらの方が精度が良いかを予測することは難しい．

5.4.2 総合信頼度の真値との比較

先の実験で対象としたネットワークに対し総合信頼度の真値を計算した．総合信頼度の計算は式 (3.3) の削除縮約展開式をプログラム化して行った．このとき再帰的に $Rel(\cdot)$ の計算をする処理を呼び出すので，$Rel(\cdot)$ の呼び出し回数が 10^6 を越えたとき，計算対象のネットワークの総合信頼度は実用的な時間内で計算することは不可能と判断し，計算不可能として計算処理を中断した．実際に計算不可能と判断するまでの時間はネットワークの構造により異なるが，およそ 210〜250 秒であった．

総合信頼度の真値を求めることができたネットワークに対し，本論文で提案した 2 つのアルゴリズム PSP1，PSP2 を適用し，得られる下界を真値と比較することにより，アルゴリズムの性能を評価した．また，枝確率が枝ごとに異なるネットワークに対する総合信頼度の下界を求める具体的なアルゴリズムは我々による手法以外に提案されていないため，ネットワーク中の全ての枝確率を同一の 0.95（開区間 (0.9,1) の一様分布の平均値）に設定した場合の Ball-Provan の下界を参考のためにあわせて計算した．ネットワークの構造は同じでも枝確率の値が異なるため，一概に Ball-Provan の下界と比較して精度の優劣を判定することはできないが，本章で提案したアルゴリズムの精度を評価するひとつの目安になるであろう．評価結果を表 5.4，5.5にまとめた．表中の $Rel(G)$ は総合信頼度の真値を，$Low(G)$ はアルゴリズム PSP1，PSP2 それぞれにより得られた下界のうち，より精度の良かった方の下界，B-P は Ball-Provan の下界を表している．$Low(G)$ が Ball-Provan の下界と比較して，ほぼ同程度の精度を示していることがわかる．

表 5.4 $N(20, a, 3)$ における提案した方法による下界の評価

	Number of	Average of		
a	computable networks	$Rel(G)$	$Low(G)$	B-P
5	50	0.99890	0.96133	0.97803
6	35	0.99930	0.96647	0.98187
7	11	0.99927	0.96086	0.98531
8	6	0.99944	0.96707	0.98446
Average		0.99911	0.96338	0.98051

表 5.5 $N(15, a, 4)$ における提案した方法による下界の評価

	Number of	Average of		
a	computable networks	$Rel(G)$	$Low(G)$	B-P
6	48	0.99998	0.98927	0.99889
7	28	0.99999	0.98911	0.99907
8	28	0.99998	0.98837	0.99904
9	9	0.99999	0.98744	0.99899
Average		0.99998	0.98887	0.99898

5.5 おわりに

本章では，直並列部分グラフによるエッジ・パッキングを用いて総合信頼度の下界を多項式時間で求める 2 つのアルゴリズムを提案した．提案したアルゴリズムは第 4章で提案した極大木によるエッジ・パッキング法を拡張させたものであるので，各枝の枝確率が同一でないネットワークに対しても総合信頼度の下界を導出できる．数値実験により，極大木によるエッジ・パッキング法が求める下界と比較して，極大木から直並列グラフへの拡張によって導出する下界の精度が大幅に向上したことを確認した．総合信頼度の真値を計算できる小さめのネットワークに対し，提案したアルゴリズムを適用し下界を求めて真値および Ball-Provan の下界と比較したところ，下界として十分な精度を示すことを確認できた．一方，提案した 2 つのアルゴリズムの性能を比較すると，要素となる直並列部分グラフ自体の総合信頼度をなるべく大きくするようにエッジ・パッキングを構築する手法であるアルゴリズム PSP1 の方が，要素となる直並列部分グラフの数をなるべく多くなるようにエッジ・パッキングを構築する手法であるアルゴリズム PSP2 よりも精度の良い下界を導出する傾向があることを確認した．しかし，この 2 つのアルゴリズムの優劣は対象となるネットワークの構造に依存するので，一概にどちらの手法が優れているとは言えない．

本章では，エッジ・パッキングを構成する部分グラフとして，極大木を含むグラフのクラスである直並列グラフを採用した．同様な発展として，直並列グラフを含むグラフのクラスで，多項式時間でその総合信頼度を計算できるものをエッジ・パッキングの要素として採用することが考えられる．実際に多項式時間で計算可能なグラフのクラスを拡張する研究は盛んに行われている [29, 64, 65]．ただし多項式時間で下界を計算するためには，エッ

ジ・パッキングの構成も多項式時間で行う必要がある．一般により大きなクラスに属する総合信頼度を計算する場合，多項式時間で計算するために特別な構造のデータを構築する必要があることが多く，理論上は多項式時間であってもプログラムにした場合にはかなりの時間がかかることが多い．より大きなクラスに属するグラフによるエッジ・パッキング法は更に計算時間がかかることが予想されるが，本章で提案した方法による下界よりも精度が良いことが期待できるので理論的に非常に興味深く，今後更なる研究を進めていきたい．

第 6 章

有向ネットワーク信頼度によるアプローチ

6.1 はじめに

本章では，有向ネットワークにおける信頼度を利用することにより，総合信頼度の下界を導出する方法について述べる．前章までは，枝に向きのない無向ネットワークを対象に話をすすめていたが，一般にネットワーク信頼度を考慮する場合，枝に向きがある有向ネットワークを対象とする場合もある．有向ネットワークの信頼度を測る尺度の一つに reachability がある．M. O. Ball はあるグラフ変換により無向ネットワークを有向ネットワークに変換したとき，元の無向ネットワーク対する総合信頼度と，構築された有向ネットワークに対する reachability が同じ値になることを示した [5]．すなわち，構築された有向ネットワークに対する reachability の下界は，元のネットワークに対する総合信頼度の下界になる．この定理を利用して本章では，M. O. Ball により提案されたグラフ変換によって構築された有向ネットワークに対し，アサイクリックグラフを要素とするアーク・パッキングを構築し，有向ネットワークに対する reachability の下界を算出するアルゴリズムを提案する．提案するアルゴリズムと第 5章で提案した直並列グラフによるエッジパッキング法に基づくアルゴリズムの性能を数値実験により比較する．

6.2 総合信頼度と reachability

無向グラフに対して確率グラフを定義したように，節点集合 $V = \{v_1, \ldots, v_n\}$，アーク集合 $A = \{a_1, \ldots, a_m\}$ を有する有向グラフ $D = (V, A)$ に確率を付加したものを**確率有向**

グラフ (probabilistic digraph) という．確率有向グラフは有向ネットワークを表現するモデルの一つとして利用される．また有向ネットワークを対象としていることが自明であるとき，確率有向グラフを単に確率グラフということも多い．確率有向グラフが表す有向ネットワークに対しても，節点は常に正常で，アークはある確率で故障すると仮定する．3.1節で無向ネットワークに対して行ったその他の仮定も有効ネットワークに対し同様に行う．

確率有向グラフ $D=(V,E)$ におけるアーク $a \in A$ が正常である確率をアーク a の**アーク確率** (arc probability) といい，$p(a); p: A \to [0,1]$ で表す．節点集合 V 中のある特定の節点 s から他の全ての節点に，正常なアークによって到達可能である確率をネットワーク D の節点 s からの **reachability** といい，$Conn(D,s)$ で表す．このとき，節点 s を**根** (root) という．根 s が自明であるとき，$Conn(D,s)$ は単に $Conn(D)$ と記される．reachability は有向ネットワークの信頼度の一つであり，総合信頼度と同様，その値を求める問題は#P 完全である [17]．

無向ネットワークと有向ネットワークに対する最も基本的なネットワーク信頼度である総合信頼度と有向信頼度には，以下の定理6.2に示す関係があることが知られている．まず有向化変換の定義を行う．

定義 6.1 確率グラフ $G=(V,E)$ が有する枝 $e=(u,v)$ を 2 本のアーク $e_1=(u,v)$, $e_2=(v,u)$ に置換する変換を考える．ただし，$p(e_1)=p(e_2)=p(e)$ とする．この変換を枝集合 Eに属する全ての枝について適用して得られる確率有向グラフを $D=(V,A)$ とする．この確率グラフ $G=(V,E)$ から確率有向グラフ $D=(V,A)$ への変換を**有向化変換** (directed transformation) という．

定義6.1より，明らかに $|A|=2|E|$ である．M. O. Ball[5] は有向化変換に関する以下の定理を証明した．

定理 6.2 $D=(V,A)$ を確率グラフ $G=(V,E)$ に有向化変換を施して得られた確率有向グラフとする．このとき節点集合 V 中の任意の節点 s に対し，次式が成立する．

$$Rel(G) = Conn(D,s). \tag{6.1}$$

定理6.2によって，無向ネットワーク G の総合信頼度 $Rel(G)$ の下界は，G に有向化変換を施して構築した有効ネットワーク Dの reachability である $Conn(D)$ の下界と等しいこ

とが分かる．つまり $Rel(G)$ の下界を求めるのに直接計算する代わりに，$Conn(D)$ の下界を計算することにより $Rel(G)$ の下界を求めることができる．以下では $Conn(D)$ の下界を計算する方法について検討する．

6.3 アーク・パッキングによる総合信頼度の下界導出法

確率グラフに対してエッジ・パッキングを定義したように，確率有向グラフに対して同様な概念であるアーク・パッキングを定義することができる．

定義 6.3 有向グラフ $D = (V, A)$ に対し，点集合 V 中の全ての点を有し，互いにアーク排反な A の部分グラフの集合

$$\mathcal{P} = \{D_1, \ldots, D_k | D_i = (V, A_i),\ A_i \subset A,\ A_i \cap A_j = \phi\ (i \neq j)\}$$

を**アーク・パッキング** (arc-packing) という．

この定義は確率有向グラフに対しても同様になされる．アーク・パッキングについて，以下の不等式が成立する [17]．

定理 6.4 確率有向グラフ $D = (V, A)$ がアーク・パッキング $\mathcal{P} = \{D_1, \ldots, D_k\}$ を持つとき，$\forall s \in V$ に対し次の不等式が成立する．

$$Conn(D, s) \geq 1 - \prod_{D_i \in \mathcal{P}} (1 - Conn(D_i, s)) \equiv Low(D, s, \mathcal{P}) \tag{6.2}$$

式 (6.2) を利用して reachability の下界を導出する方法を**アーク・パッキング法** (arc-packing method) と呼ぶ．アーク・パッキング法もエッジ・パッキング法と同じく，枝確率が枝ごとに異なっているようなネットワークに対しても適用可能であるという大きな特徴を有している．

式 (6.2) より，任意の $D_i \in \mathcal{P}$ に対し，$Conn(D_i, s)$ が多項式時間で計算できるようなアーク・パッキング $\mathcal{P}$ を多項式時間で構築できれば，$Conn(G)$ の下界 $Low(D, \mathcal{P})$ を多項式時間で計算できる．多項式時間で reachability を計算可能な有向グラフのうち，最も構造が単純なのが有向極大木である．節点 s を根とする有向極大木 $T = (V, A_T)$ の reachability は，

$$Conn(T, s) = \prod_{a \in A_T} p(a) \tag{6.3}$$

と書ける.

サイクルを含まない有向グラフを**アサイクリックグラフ** (acyclic graph) という. アサイクリックグラフ $D=(V,A)$ において, 根 $s \in V$が他の全ての節点に到達可能であるとき, Dの reachability は次式で与えられる [6] :

$$Conn(D,s) = \prod_{v \in V-\{s\}} \{1 - \prod_{a \in A(D,v)} (1-p(a))\}. \tag{6.4}$$

ここで $A(D,v)$ は, 節点 vを終点とするアークの集合である. 以降, 根 s が他の全ての節点へ到達可能であるアサイクリックグラフを **s-全域アサイクリックグラフ** (s-spanning acyclic graph) と呼ぶ. $\mathcal{P}_{SA}$を s-全域アサイクリックグラフによるアーク・パッキングとすると, $Low(D,s,\mathcal{P}_{SA})$ は多項式時間で計算可能である.

我々の目標は確率有向グラフ Dの reachability である $Conn(D,S)$ の下界 $Low(D,s,\mathcal{P}_{SA})$ の精度が良くなる（値が大きくなる）ように, s-全域アサイクリックグラフによるアーク・パッキング $\mathcal{P}_{SA}$ を多項式時間で構築することである. この目標を以下の最適化問題 PSA(Packings of s-Spanning Acyclic graphs) の形式で表す.

問題 PSA

Maximize　$Low(D,s,\mathcal{P}_{SA}) = 1 - \prod_{D_i \in \mathcal{P}_{SA}} \{1 - Conn(D_i,s)\}$

Subject to　$\mathcal{P}_{SA}$: s-全域アサイクリックグラフによる Dのアーク・パッキング

問題 PSA は#P 困難であると見られているが, 現在のところその証明はされていない. 問題 PSA に対しても第4章と同様, 以下の 2 つの戦略が考えられる.

[戦略 1] : $Conn(D_i,s)$ を大きくする.

[戦略 2] : $|\mathcal{P}_{SA}|$ を大きくする.

やはりこれら 2 つの戦略も一般には互いに両立しない.

今回対象としている確率有向グラフは, 確率グラフを有向化変換して構築したグラフである. 枝連結度 c の確率グラフ G に有向化変換を施して確率有向グラフ Dが構築した場合, [戦略 2] に関しては $|\mathcal{P}_{SA}| \leq c$ という結果が示されている [17].

6.4 提案するアルゴリズム

6.4.1 Ramanathan と Colbourn による方法の具体化

A. Ramanathan と C. J. Colbourn は，s-全域アサイクリックグラフからなる $D=(V,A)$ のアーク・パッキングを構築する 2 つのアプローチを記した [69]．以下にそれらをアルゴリズム形式で記述する．

Approach 1

1. Dから s を根とする reachability 最大の有向極大木を抜き出し，
   $\mathcal{P}_{SA}$に追加する．
2. サイクルができない範囲で，D中のアークをアーク確率の高い順に
   $\mathcal{P}_{SA}$中の要素に追加する．追加したアークは Dから削除する．
3. 有向極大木が見つからなくなるまで Step 1,2 を繰り返す．

Approach 2

1. Dから c 個の s を根とするアーク排反極大木を抜き出す．
2. サイクルができない範囲で，D中のアークをアーク確率の高い順に
   $\mathcal{P}_{SA}$中の要素に追加する．

Approach 1 で reachability が最大となる有向極大木を見つける必要があるが，無向グラフの場合と同様に定理 4.1を利用することによって，最大有向木を見つけることに帰着可能である．最大有向木は R. E. Tarjan のアルゴリズム [81] などにより得ることができる．また Approach 2 における c は，有向化変換を施す前の確率グラフ G の枝連結度である．Y. Shiloach のアルゴリズム [75] を用いると，s を根とする c 個のアーク排反有向極大木を抜き出すことが可能である．

上記のアプローチには 2 つの不明な点がある．1 つはアーク・パッキング $\mathcal{P}_{SA}$に D中のアークを追加する方法が不明確なことである．追加するアークについてはアーク確率の大きい順であるが，追加されるアーク・パッキング中の要素についてはどの要素に対し追加するのか全く記述されていない．そこで彼らの論文 [69] 内で行われている数値例を参考にして，reachability の大きな要素ほど優先的にアークを追加される，という基準で彼らは考えていると類推した．

次の不明な点は根 s の選択基準である．定理6.4では s は V 中の任意の節点としているが，実際にアルゴリズムを記述するためには具体的な選択基準を明記する必要がある．更にできれば得られる下界 $Low(D,s,\mathcal{P})$ の精度が良くなるように選択したい．s-全域アサイクリックグラフは根 s を終点とするアークを有することはできない．よって仮に根 s を定めたとすると，s を終点とするアークはアーク・パッキング中のどの s-全域アサイクリックグラフにも含まれない．[戦略 1] より，s-全域アサイクリックグラフの reachability が高くなるほど下界 $Low(D,s,\mathcal{P})$ の精度は良くなる．アーク確率の高いアークはなるべくアーク・パッキングを構成するアークとして利用したいので，なるべくアーク確率の低いアークを終点とする節点を根 s にした方が良いことが予想される．そこで根 s の選択基準を以下のように設定した．

$$\forall v \in V;\ \frac{\sum_{a\in A(D,s)} p(a)}{|A(D,s)|} \leq \frac{\sum_{a\in A(D,v)} p(a)}{|A(D,v)|} \tag{6.5}$$

ここで $A(D,v)$ は Dにおいて節点 vを終点とするアークの集合である．すなわち自分を終点とするアークの平均アーク確率が最も低い節点を根 s とする．

上記の事項を考慮に入れて，A. Ramanathan と C. J. Colbourn による Approach 1 と 2 を具体的なアルゴリズム RC1，RC2（RC は Ramanathan と Colbourn の頭文字）として以下に記述する．

Procedure RC1(D)
begin
1. Determine the root s by (6.5);
2. $\mathcal{P}_{SA} \leftarrow \phi$;
3. **while** D has a s-rooted arborescence **do**
4. $D' \leftarrow$ the maximum s-rooted arborescence of D;
5. $\mathcal{P}_{SA} \leftarrow \mathcal{P}_{SA} \cup \{D'\}$;
6. $D \leftarrow D - D'$;
7. **call** DISTRIBUTE-RC($\mathcal{P}_{SA}$, D, s);
8. **end**

end

Procedure RC2(D, c)
begin
1. Determine the root s by (6.5);
2. $\mathcal{P}_{SA} = \{D_1, \ldots, D_c\} \leftarrow c$ s-rooted arc-disjoint arborescences;

3. **call** DISTRIBUTE-RC($\mathcal{P}_{SA}$, $D - D_1 - \cdots - D_c$, s);
end

Procedure DISTRIBUTE-RC($\mathcal{P}, D_{\text{remain}}, s$)
begin
1. $Q \leftarrow$ the queue containing all arcs in D_{remain} in non-decreasing order of arc probability;
2. Sort all s-rooted acyclic graphs in $\mathcal{P}$
   in non-decreasing order of reachability and index them again;
3. **while** $|Q| \neq \phi$ **do**
4. Take out and delete an arc a from the top of Q;
5. **for** $i = 1, \ldots, |\mathcal{P}|$ **do**
6. **if** $D_i \cup a$ has no cycle **then**
7. $D_i \leftarrow D_i \cup a$;
8. $D_{\text{remain}} \leftarrow D_{\text{remain}} - a$;
9. **break** the for-loop
10. **end**
11. **end**
12. **end**
end

アルゴリズム DISTRIBUTE-RC 内の D_iにアークを追加する処理では，$D_i \cup a$ がサイクルを持たなければ追加を容認する，という手続きを行っている．アーク・パッキング $\mathcal{P}$は s-全域アサイクリックグラフである必要があるが D_iは既に s-全域アサイクリックグラフであるので，$D_i \cup a$ がサイクルを持たなければ s-全域アサイクリックになる．

これらのアルゴリズムの計算量に関して以下の定理が成立する．

定理 6.5 $\mathcal{P} = \{D_1, \ldots, D_{|\mathcal{P}|}\}$ を確率グラフ $D = (V, A)$ のアーク・パッキング，$D_{\text{remain}} = D - D_1 - \cdots - D_{|\mathcal{P}|}$とすると，アルゴリズム DISTRIBUTE-RC($\mathcal{P}, D_{\text{remain}}, s$) の計算オーダーは $O(|A_{\text{remain}}|(|\mathcal{P}| \cdot |V| + |A|))$ である．ここで，$D_{\text{remain}} = (V, A_{\text{remain}})$ である．

証明

$D_i = (V, A_i)$ とすると，$D_i \cup a$ が s-全域アサイクリックグラフであるか判断する処理は $O(|V|+|A_i|)$ で行える．アーク・パッキングの要素は互いにアーク排反であるので，**for** ループの処理は高々$O(|\mathcal{P}| \cdot |V|+|A|)$ である．従って **while** ループの処理は $O(|A_{\text{remain}}||\mathcal{P}|(|V|+|A|))$ で，その他の処理はこの処理より低いオーダーで実行可能である．

証明終

定理 6.6 確率グラフ $G = (V, E)$ の枝連結度を c, G に有向化変換を施して構築された確率有向グラフを $D = (V, A)$ とする．このとき，アルゴリズム RC1(D) の計算オーダーは $O(c|A|^2)$ である．

証明

最大有向極大木を見つけるアルゴリズムは $O(|A| \log |A|)$ で実行可能である．また **while** ループは高々c 回しか繰り返さず，また $|E| \geq c|V|/2$ より，$|A| \geq c|V|$ を考慮に入れると，アルゴリズム DISTRIBUTE-RC の部分は高々$O(|A|^2)$ となる．従って **while** ループ全体で $O(c|A|^2)$ となる．

証明終

定理 6.7 確率グラフ $G = (V, E)$ の枝連結度を c, G に有向化変換を施して構築された確率有向グラフを $D = (V, A)$ とする．このとき，アルゴリズム RC2(D) の計算オーダーは $O(\max\{c^2|V|(|V| + |A|), |A|^2\})$ である．

証明

c 本のアーク排反有向極大木は $O(c^2|V|(|V| + |A|))$ で求めることができる．その後に行うアルゴリズム DISTRIBUTE-RC において，$|\mathcal{P}| = c$, $|D_{\text{remain}}| = |A| - c(|V| - 1)$ であるので，処理時間は高々$O(|A|^2)$ となる．

証明終

6.4.2　残アーク分配アルゴリズムの改良

アルゴリズム DISTRIBUTE-RC は下界 $Low(D, s, \mathcal{P})$ の精度を上げるため，D_{remain}に残っているアークをアーク・パッキング $\mathcal{P}$の要素である s-全域アサイクリックグラフにサイクルができない範囲で追加している．その際なるべくアーク確率の高いアークを，なるべく reachability の高い要素に追加しているが，その選択基準には何の正当性も保証されていない．ここでは追加するアーク $a_0 \in D_{\text{remain}}$と追加される要素 $D_0 \in \mathcal{P}$の選択基準を提案する．具体的には下界 $Low(D, s, \mathcal{P})$ の値を最も大きくするようなアーク・パッキングの要素と追加するアークの組合せ (D_0, a_0) を選択する基準を提案する．

以下の定理の前提として，確率有向グラフ $D = (V, A)$ 中の任意のアーク a について $0 < p(a) < 1$，すなわち $p(a) \neq 0,\ 1$ とする．

定理 6.8 $\mathcal{P}$を確率有向グラフ $D=(V,A)$ の s-全域アサイクリックグラフによるアーク・パッキング，$v_e(a)$ をアーク $a\in A$ の終点，$A(D,v)$ を Dにおける節点 vを終点とするアーク集合とする．また，

$$Q(D,v)=\prod_{a\in A(D,v)}\{1-p(a)\} \tag{6.6}$$

とする．このとき，アーク・パッキング中の一要素 $D_0\in\mathcal{P}$にあるアーク a_0を追加することよって増加する Dの *reachability* の下界 $Low(D,s,\mathcal{P})$ は，次の項に比例する．

$$p(a_0)\cdot\frac{Conn(D_0,s)}{1-Conn(D_0,s)}\cdot\frac{Q(D_0,v_e(a_0))}{1-Q(D_0,v_e(a_0))}.$$

証明

まず以下のように記号を定義する．

$$\delta(D_0,a_0)=p(a_0)\cdot\frac{Q(D_0,v_e(a_0))}{1-Q(D_0,v_e(a_0))}, \tag{6.7}$$

$$\Delta(D_0,a_0)=\delta(D_0,a_0)\cdot\frac{Conn(D_0,s)}{1-Conn(D_0,s)}. \tag{6.8}$$

式 (6.4) より，$Conn(D_0,s)$ と $Conn(D_0\cup a_0,s)$ の差は $v_e(a_0)$ に関する項のみである．よって以下の式が成立する．

$$\begin{aligned}
Conn(D_0\cup a_0,s) &= \left[\prod_{v\in V-\{s,v_e(a_0)\}}\{1-Q(D,v)\}\right]\cdot\{1-Q(D_0\cup\{a_0\},v_e(a_0))\}\\
&= \frac{Conn(D_0,s)}{1-Q(D_0,v_e(a_0))}\cdot[1-\{1-p(a_0)\}Q(D_0,v_e(a_0))]\\
&= Conn(D_0,s)+Conn(D_0,s)\cdot p(a_0)\cdot\frac{Q(D_0,v_e(a_0))}{1-Q(D_0,v_e(a_0))}\\
&= Conn(D_0,s)+Conn(D_0,s)\cdot\delta(D_0,a_0)
\end{aligned}$$

アーク・パッキングの要素 D_0にアーク a_0を追加したときの reachability の下界を Low_0とすると，

$$\begin{aligned}
Low_0 &= 1-\left[\prod_{D\in\mathcal{P}-\{D_0\}}\{1-Conn(D,s)\}\right]\cdot\{1-Conn(D_0\cup a_0,s)\}\\
&= 1-\frac{1-Low(D,s,\mathcal{P})}{1-Conn(D_0,s)}\cdot\{1-Conn(D_0,s)-Conn(D_0,s)\cdot\delta(D_0,a_0)\}\\
&= Low(D,s,\mathcal{P})+\{1-Low(D,s,\mathcal{P})\}\cdot\delta(D_0,s)\cdot\frac{Conn(D_0,s)}{1-Conn(D_0,s)}\\
&= Low(D,s,\mathcal{P})+\{1-Low(D,s,\mathcal{P})\}\cdot\Delta(D_0,a_0)
\end{aligned}$$

アーク・パッキングの要素 D_0にアーク a_0を追加したときに，reachability の下界の増加分は $Low_0 - Low(D, s, \mathcal{P})$ であるので，$\Delta(D_0, a_0)$ の値に比例することが分かる．

証明終

定理6.8を考慮に入れ，最も精度が良くなるような (D_0, a_0) の組合せを選択してアークを追加するアルゴリズム DISTRIBUTE を以下に示す．なお，A_{remain}は D_{remain}の有するアーク集合である．

Procedure DISTRIBUTE($\mathcal{P}$, D_{remain}, s)
begin
1　**for** each $D_0 = (V, A_0) \in \mathcal{P}$ **do**
2　　Calculate $Conn(D_0, s)$;
3　　**for** each $v \in V$ **do** Calculate $Q(D_0, v)$;
4　**end**
5　**for** each $a_0 \in A_{\text{remain}}$ **do** and each $D_0 = (V, A_0) \in \mathcal{P}$ **do**
6　　Calculate $\Delta(D_0, a_0)$;
7　**end**
8　**while** D_{remain} has an arc **do**
9　　$\Delta(D^*, a^*) \leftarrow \max \Delta(D, a)$ where $D \in \mathcal{P}$ and $a \in A_{\text{remain}}$ by using $Q_{\Delta(\cdot)}$;
10　　**if** $\Delta(D^*, a^*) = 0$ **then break**;
11　　**if** $D^* \cup a^*$ has a cycle **then**
12　　　$\Delta(D^*, a^*) \leftarrow 0$;
13　　　move $\Delta(D^*, a^*)$ to the bottom of $Q_{\Delta(a^\wedge)}$;
14　　**end**
15　　**else**
16　　　$D^* \leftarrow D^* \cup a^*$;
17　　　$D_{\text{remain}} \leftarrow D_{\text{remain}} - a^*$;
18　　　$Conn(D^*, s) \leftarrow Conn(D^*, s) \cdot \left[1 + p(a^*) \cdot \dfrac{Q(D^*, v_e(a^*))}{1 - Q(D^*, v_e(a^*))}\right]$;
19　　　$Q(D^*, v_e(a^*)) \leftarrow \{1 - p(a^*)\} \cdot Q(D^*, v_e(a^*))$;
20　　　**for** each $a_0 \in A_{\text{remain}}$ **do**
21　　　　**if** $\Delta(D^*, a_0) \neq 0$ **then** Calculate $\Delta(D_0, a_0)$;
22　　　**end**
23　　**end**
24　**end**
end

今，アーク a_0を s-全域アサイクリックグラフ D_0に追加することを考える．すると，

$$\begin{aligned} Q(D_0 \cup a_0, v_e(a_0)) &= \{1 - p(a_0)\} \cdot Q(D_0, v_e(a_0)), \\ Conn(D_0 \cup a_0, s) &= Conn(D_0, s) + Conn(D_0, s) \cdot \delta(D_0, a_0) \\ &= Conn(D_0, s)\{1 + \delta(D_0, a_0)\} \end{aligned}$$

より，while ループ内における $Conn(D^*, s)$ と $Q(D^*, v_e(a^*))$ の値に対する更新処理の正当性が確認できる．D_{remain}中のアーク a^*がアーク・パッキングの要素に追加されるのは高々一度であるので，アーク・パッキング $\mathcal{P}_{SA}$は常にアーク排反であり，アークの追加は常にサイクルができない範囲で行うので，$\mathcal{P}_{SA}$の要素は常に s-全域アサイクリックグラフである．以上により，アルゴリズム DISTRIBUTE の正当性を確認できた．

アルゴリズム RC1(D) と RC2(D, c) において，アルゴリズム DISTRIBUTE-RC をアルゴリズム DISTRIBUTE に変更したものをそれぞれアルゴリズム PSA1(D)，PSA2(D, c) と定義する．計算オーダーに関して以下の定理が成立する．

定理 6.9 アルゴリズム DISTRIBUTE($\mathcal{P}, D_{\text{remain}}, s$) の計算オーダーは $O(|\mathcal{P}|^2|A_{\text{remain}}|^2)$ である．ここで，$D_{\text{remain}} = (V, A_{\text{remain}})$ である．またアルゴリズム PSA1(D)，PSA2(D) の計算オーダーはそれぞれ $O(c^3|A|^2)$，$O(c^2|A|^2)$ である．

証明

Step8 からの **while** ループ内で最も時間のかかる処理は Step12 の$\Delta(D, a)$ の最大値を見つける処理で，その計算オーダーは高々$O(|\mathcal{P}| \cdot |A_{\text{remain}}|)$ である．**while** ループ内の処理を一度行うと D_{remain}の要素が 1 つ減るか，ある正なる$\Delta(D, a)$ が 0 になるかのいずれかである．よって **while** ループの繰り返し回数は高々$O(|\mathcal{P}| \cdot |A_{\text{remain}}|)$ 回であるので，**while** ループ全体の処理時間は高々$O(|\mathcal{P}|^2|A_{\text{remain}}|^2)$ で，**while** ループ以外の処理はこれよりも低いオーダーで実行可能である．よってアルゴリズム DISTRIBUTE($\mathcal{P}, D_{\text{remain}}, s$) の計算オーダーは $O(|\mathcal{P}|^2|A_{\text{remain}}|^2)$ となる．アルゴリズム PSA1，PSA2 の計算オーダーは $|\mathcal{P}| \leq c$, $|A_{\text{remain}}| \leq |A|$ より導出できる．

証明終

6.5 数値実験と考察

本節では前節までに提案したアルゴリズムの性能を数値実験により評価する．数値実験の対象ネットワークとして，5.4.1節と同様に近接グラフ $N(n, a, b)$ を採用した．以下簡単

のため，アルゴリズム RC1，RC2 による総合信頼度の下界をそれぞれ RC1，RC2 とし，2 つの下界のうち精度の良い方を RC とする．同様に PSA1，PSA2 をアルゴリズム PSA1，PSA2 による下界とし，PSA を精度の良い方とした．第5章では直並列グラフによるエッジ・パッキングを構築するアルゴリズムアルゴリズム PSP1，PSP2 を提案したが，それらのアルゴリズムによる総合信頼度の下界のうち，精度の良い方を PSP とする．また総合信頼度の真値を Rel とする．数値実験のためのプログラムは C 言語で記述され，Dell Dimension XPS R400 (CPU:PentiumII 400MHz, Memory:128MB) 上で実験を行った．

6.5.1　下界の精度の評価

総合信頼度の真値と比較して各アルゴリズムが導出する下界の精度を評価するため，対象のネットワークの大きさは適当な時間内で総合信頼度が計算できるよう小規模なネットワークを対象とした．節点数 $n = 15$，$b = 3$ に対し $a = 3, 6, 9, 12$，$b = 4$ に対し $a = 4, 6, 8$ とした．それぞれの場合に対し 50 個のネットワーク，合計 350 個のネットワークを作成した．枝確率は開区間 (0.9,1) 上の一様分布に従うものとし，ランダムに設定した．それぞれのネットワークに対し各アルゴリズムによる下界，および総合信頼度の真値を計算した．ネットワークが小規模であるので，各アルゴリズムの実行時間は非常に短く比較することは不可能であった．実行時間については 6.5.2節において，大規模なネットワークに対して各手法を適用して評価する．

RC1 と RC2，PSA1 と PSA2 の比較

350 個のネットワークに対する RC1 と RC2 との比較結果を図 6.1に示した．横軸はネットワーク中の枝数を表し，縦軸は RC1/RC2 の値を表す．点線の水平線は 2 つの下界が等しい値を持つ場合を表している．350 個のネットワークに対し，RC1 が優れていたのが 129 個 (36.9%)，RC2 が優れていたのが 220 個 (62.9%) で，同値を示したネットワークが 1 つあった．今回の実験ではやや RC2 の方が RC1 より優れている傾向を示しているが，ネットワークの枝数が増えるほど，このようにどちらの方が優れているという傾向は薄くなる．またランダムに選択した幾つかのネットワーク構造に対し，枝確率の分布を何度かランダムに設定し直して RC1 と RC2 を計算することにより，2 つの下界の優劣がネットワーク構造に強く依存しているのか否かを評価したが，特定のネットワーク構造に対し一方の下界がどんな枝確率の分布に対しても精度が良い，ということはなかった．これらの結果か

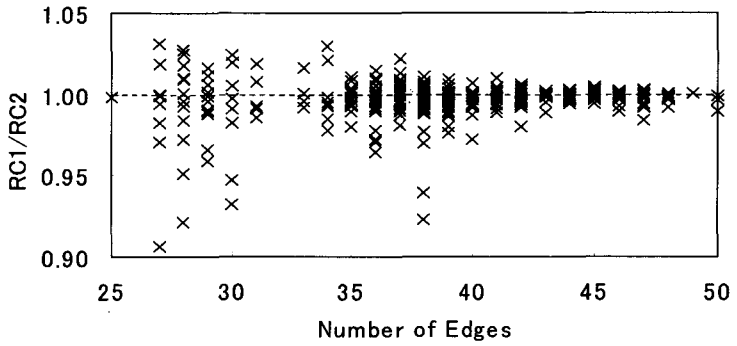


図 6.1 $N(15, a, b)$ における RC1 と RC2 の比較

ら，2 つの下界 RC1 と RC2 の優劣はネットワークの構造よりむしろ枝確率の分布状態に強く依存すると結論付けた．PSA1 と PSA2 についても同様の結論となった．

Procedure DISTRIBUTE の効果

Procedure DISTRIBUTE の効果を評価するため，上記 350 個のネットワークに対し RC1 と PSA1，RC2 と PSA2 を比較した．前者の比較結果を図 6.2に，後者の比較結果を図 6.3に示した．図 6.2と図 6.3の縦軸はそれぞれ PSA1/RC1，PSA2/RC2 の値を表している．点線の水平線は比較する 2 つの下界が同値であるケースを表している．どちらの図を見ても，大多数のネットワークに対し Procedure DISTRIBUTE が下界の精度を上げていることが分かる．厳密に記せば，図 6.2においては 339 個 (96.9%) のネットワーク，図 6.3においては，337 個 (96.3%) のネットワークに対し精度を上げている．$N(n, a, b)$ における a の値が大きくなるほどこの傾向は強くなり，$N(15, 12, 3)$ や $N(15, 4, 8)$ の全てのネットワークに対し，Procedure DISTRIBUTE が下界の精度を上げたことを確認した．

PSA，PSP，Rel との比較

まず本章で提案したアルゴリズム Procedure PSA1 と PSA2 による下界のうち精度の良い方である PSA と，第 5章で提案した 2 つの方法で得られる下界のうち精度の良い方である PSP とを比較する．350 個のネットワークに対し比較した結果を図 6.4に示した．縦軸は PSA/PSP の値を表す．今回提案した方法による下界 PSA は PSP と比較して，172 個 (49.1%) のネットワークについては優れ，178 個 (50.9%) のネットワークに対しては劣っていた．また図 6.4から，ネットワークの枝数が増加するにつれ，PSA の方が PSP より優れ

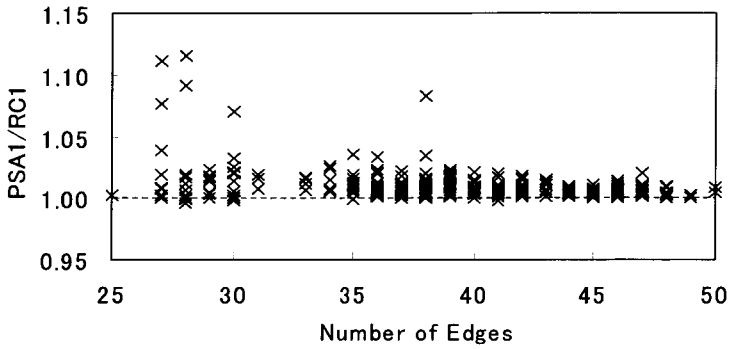


図 6.2 $N(15,a,b)$ における RC1 と PSA1 との比較

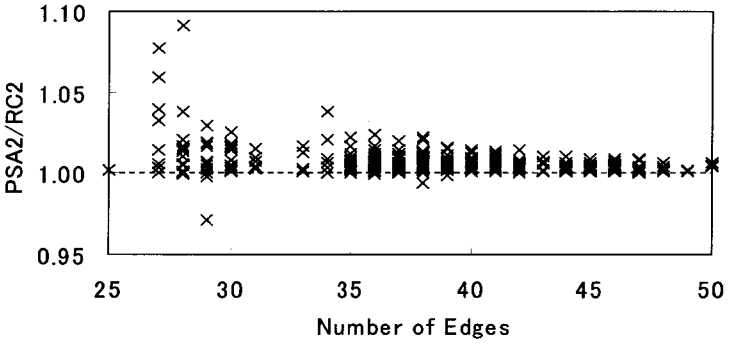


図 6.3 $N(15,a,b)$ における RC2 と PSA2 との比較

る傾向があることが見て取れる．また更なる解析により，$N(n,a,b)$ における $a-b$ の値が大きくなるにつれ，PSA が PSP を凌駕する傾向があることが分かった．これは言い換えれば，PSA はネットワーク内に疎の部分と密な部分が明確に分かれているようなネットワークに対しては，PSP ほどの精度の下界を求めることができない傾向がある，ということになる．これらの結果から，本章で提案した手法は第5章で提案した手法と同程度有効であると結論付けた．

図 6.5は総合信頼度の真値と比較することにより，PSA の精度を評価したものである．縦軸は PSA/Rel の値，点線の水平線は PSA/Rel の値の平均値で 0.981 である．他の下界と同様，ネットワーク中の枝数が増加するほど，下界の精度が良くなる傾向がある．

表 6.1に $N(15,a,b)$ に対する各手法の精度に関するまとめを示した．表中の数値は各 $N(n,a,b)$ に対する 50 個のネットワークに対する値の平均値である．

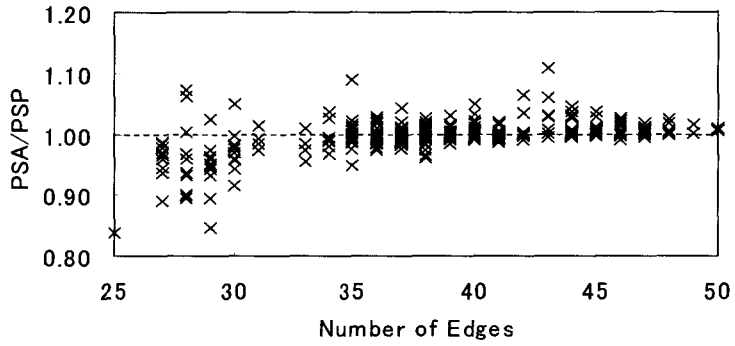


図 6.4 $N(15, a, b)$ における PSA と PSP の比較

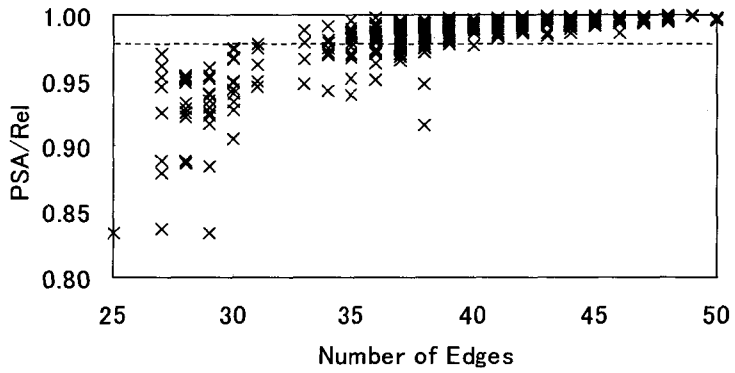


図 6.5 $N(15, a, b)$ における PSA の精度

表 6.1 $N(15, a, b)$ に対する各下界と真値

n	b	a	RC	PSA	SP	Rel	PSA/Rel
15	3	3	0.90829	0.92077	0.95882	0.98640	0.93346
		6	0.97433	0.98091	0.98134	0.99960	0.98131
		9	0.98294	0.98916	0.98726	0.99975	0.98941
		12	0.98588	0.99190	0.98156	0.99981	0.99209
	4	4	0.97192	0.97894	0.98577	0.99918	0.97975
		6	0.98933	0.99247	0.98787	0.99830	0.99416
		8	0.99310	0.99666	0.99038	0.99999	0.99667

6.5.2 計算時間

計算時間を測定するため，ネットワークの大きさを先のものより大きめにし，$n = 50$, $a = 5, 15, 30, 49$, $b = 3, 5, 10, 20, 40$（ただし $a \geq b$）とした．それぞれの $(50, a, b)$ に対し50個のネットワークを作成し，枝確率は開区間 (0.9,1) 上の一様分布に従うと仮定してランダムに設定した．$a \geq b$ なる組合せは全14通りで，合計700個のネットワークを作成した．それぞれの $N(50, a, b)$ に対し作成した50個のネットワークにおける枝数 m と枝連結度 c の平均値を表6.2に示した．

表 6.2 枝数と枝連結度の平均値

$b\backslash a$	5		15		30		49	
3	115.7	2.5	139.4	3.0	145.1	3.0	145.6	3.1
5	153.4	4.1	217.3	5.1	233.6	5.3	237.1	5.7
10			366.3	10.3	432.5	11.1	449.0	12.8
20					725.2	21.6	795.5	26.1
40							1183.9	44.4

m　c

それぞれのネットワークにおける各アルゴリズムの計算時間の平均を表6.3に示した．どの (a, b) の組に対しても，RC2 は RC1 より，PSA2 は PSA より多くの実行時間を必要としている．また PSA1 と RC1 より，PSA2 は RC2 より時間がかかる傾向があるが，RC2 と RC1,PSA2 と PSA1 との差と比較するとその差は非常に小さい．これらの結果より，余りのアークを分配する処理と比較すると，Procedure RC2 や PSA2 でアーク排反有効極大木を見つけるために行う Shiloach のアルゴリズム [75] は，計算オーダー的には小さいものの，実際の処理においては多くの時間を必要とすることがわかる．

6.6 おわりに

本章では対象のネットワークを有向ネットワークに変換し，有向ネットワークの信頼度である reachability の下界を求めることで間接的に対象のネットワークに対する総合信頼度の下界を求めた．A. Ramanathan と C. J. Colbourn が提案したアサイクリックグラフによるアーク・パッキング法のアプローチに対しアルゴリズムとして不明確な部分を補い，具体的なアルゴリズムとして再構築し，その計算量を評価した．また reachability の下界

表 6.3 計算時間

$b\backslash a$	5		15		30		49	
3	0.03	0.17	0.03	0.33	0.03	0.37	0.03	0.37
	0.03	0.23	0.00	0.30	0.00	0.40	0.03	0.40
5	0.10	0.33	0.17	0.77	0.03	0.97	0.13	0.97
	0.13	0.37	0.23	0.80	0.07	0.97	0.23	1.10
10			0.20	2.70	0.53	3.50	0.57	4.10
			0.47	2.80	0.60	4.00	0.67	4.27
20					1.13	11.80	1.27	15.63
					1.90	13.23	2.13	16.83
40							0.03	44.00
							0.03	48.70

RC1	RC2
PSA1	PSA2

の精度を上げるために，アーク・パッキングの要素である既存のアサイクリックグラフにアークを追加するが，その際にどのアークをどのアサイクリックグラフに追加すれば最も下界の精度が大きく向上するかを判断するための定理を証明した．この定理を考慮に入れてアルゴリズムを修正すると，ほぼ全てのネットワークに対し reachability の下界の精度を向上させることができることを確認した．この手法による総合信頼度の下界は，第 5章で提案した手法による下界と比較して，ほぼ同等の精度を持つことを確認した．

本章ではアーク・パッキングを構成する部分有向グラフとしてアサイクリックグラフを採用したが，アサイクリックグラフが有向極大木を含むクラスであるように，アサイクリックグラフをクラスとして含む有向グラフのうち多項式時間で reachability を計算できるものがあれば，その有向グラフによるアーク・パッキング法を評価する価値があるであろう．しかし総合信頼度を多項式時間で計算可能なクラスに関する研究は盛んであるのに対し，reachability に関してはまだほとんどなされていないようである．今回の研究によって reachability に関する研究成果が総合信頼度に関して有用であることを確認できたので，総合信頼度に応用する意味でも reachability の研究にも今後取り組んでいきたい．

第 7 章

総合信頼度を考慮したネットワーク設計問題

7.1 はじめに

ネットワークを新たに構築する場合，通常ネットワークの設計者はなるべく少ない費用で，応答反応時間や接続性などのネットワーク性能を測るいくつかの基準を満足するようにネットワークを設計する．3.1節で述べたとおり，基幹ネットワークの設計をする上で最も重要な要素の一つが信頼性である．本章では，なるべく少ない費用で総合信頼度がある水準以上になるネットワークを設計する問題を扱う．総合信頼度を求める問題自体が#P完全であるので，この問題の最適解を多項式時間で求めることは絶望視されており，大規模なネットワークを対象とする場合は非常に膨大な時間が必要となる．そのため，この問題は 1980 年代から広く研究され多くの手法が提案されているが，そのほとんどが近似解を導出するものである [1, 16, 85]. 最近では総合信頼度に加え他の基準も考慮に入れるモデルに対し，遺伝的アルゴリズムやタブー探索などのメタ・ヒューリスティックスと呼ばれる手法を用いて近似解を求める研究も盛んになされている [21, 61]. そのような状況の中，R. -H. Jan ら [35] は分枝限定法をベースとする最適解導出アルゴリズムを提案した．実用的な時間内で最適解を求めることができないという意味で，彼らのアルゴリズムは決して大規模なネットワークには適用できないが，近似解の精度を評価する意味でもその存在意義は大きい．ただし彼らのアルゴリズムには，ネットワーク中の枝確率が全て同一であるという条件が必要で，異なる枝確率の枝を有するネットワークには適用できない．

本章では枝確率が異なる枝を有するネットワークにも適用できるようにR. -H. Janらのアルゴリズムを拡張する．また彼らのアルゴリズムが，制約条件となる総合信頼度の水準によって計算時間が極端に異なることを示し，その問題点を解決する方法を示す．また更に計算時間を短縮するためのいくつかの工夫も合わせて提案する．

7.2 モデルと定式化

節点集合 $V = \{v_1, \ldots, v_n\}$ と採用候補枝の集合 $E = \{e_1, \ldots, e_m\}$ からなるグラフを $G = (V, E)$ とする．各枝 $e \in E$には向きがなく，枝確率 $p(e) \in (0, 1)$ とその枝を採用するために必要なコスト $c(e) > 0$ という2つのパラメータを持つ．このネットワークに対し，以下の仮定を行う．

1. 各節点の位置は固定されている．
2. 各節点は故障せず常に正常状態．枝は故障する．
3. 異なる枝の枝確率は互いに独立である．
4. 枝確率は枝ごとに異なっていても構わない．
5. Eには自己閉路は含まない（総合信頼度に何の寄与も持たないため）．
6. $G = (V, E)$ は連結である（非連結なら問題が自明となる）．

変数 x_iを枝 e_iに関する変数とし，枝 e_iを採用するときに1，採用しないときに0をとるとする．構築するネットワークの総合信頼度が満たすべき水準を $R_0(> 0)$ とすると，本章の目的は以下のように定式化される．

問題 MP

$$\text{Minimize} \quad \sum_{e_i \in E} c(e_i) \cdot x_i \quad \equiv z^* \tag{7.1}$$

$$\text{Subject to} \quad Rel(G_x) \geq R_0 \tag{7.2}$$

$$x_i \in \{0, 1\},\ i = 1, \ldots, m \tag{7.3}$$

ここで $G_x = (V, E_x)$ は，節点集合 Vと採用された枝の集合 $E_x = \{e_i | x_i = 1, i = 1, \ldots, m\}$ からなるグラフである．問題MPの最適値を z^*とする．式(7.2)を評価するためには $Rel(G_x)$ を求めなければならず，総合信頼度を求める問題自体#P完全なため，問題MPを多項式時間で解くことは絶望視されている．

7.3 Jan のアルゴリズムとその拡張

R. -H. Jan ら [35] は部分問題への分割と分枝限定法をベースとする問題 MP の最適解を求めるアルゴリズムを提案した．彼らのアルゴリズムは，枝確率が全ての枝について同一であるネットワークに対してのみ適用可能であり，枝確率が異なる枝を有するネットワークには適用できない．本節では枝確率が異なる枝を有するネットワークに対しても適用できるように，彼らのアルゴリズムを拡張する方法について述べる．

7.3.1 Jan のアルゴリズム

本節では枝確率が全ての枝について等しいと仮定する．以下に Jan らのアルゴリズムの概要を示す．

Procedure JAN
begin
1. compute a^*
2. $l \leftarrow a^*, z^* \leftarrow \infty$
3. **do**
4. compute $z(l)$ by executing Algorithm JANSUB(l)
5. **if** $z(l) < z^*$ **then** $z^* \leftarrow z(l)$
6. $l \leftarrow l+1$
7. **while** $\underline{z}(l) < z^*$
8. **output** the optimal value z^*

end

Step 1 で算出する a^*は，制約条件 (7.2) を満たすために必要な最小枝本数である．この a^*を計算するために，Jan らは以下の問題 R(l) を考えた．

問題 R(l)

$$\text{Maximize} \quad Rel(G_x) \equiv r(l) \tag{7.4}$$

$$\text{Subject to} \quad \sum_{e_i \in E} x_i = l \tag{7.5}$$

$$x_i \in \{0,1\},\ i = 1, \ldots, m \tag{7.6}$$

問題 R(l) の最適値を $r(l)$ とすると，グラフ G_xが連結であるためには少なくとも $n-1$ 本の枝が必要であるので，$l = 0, \ldots, n-2$ に対しては明らかに $r(l) = 0$ となる．また Jan

ら [36] は枝確率が全ての枝について等しい場合において，$r(n-1), r(n), r(n+1)$ の値と $n+2 \leq l \leq m$ に対する $r(l)$ の上界を多項式時間で求めるアルゴリズムを提案した．これらの値が計算できると，a^*は以下の式 (7.7) を満足するような値を 2 分探索法を用いて求めることができる．

$$\overline{r}(a^*-1) < P_0 \leq \overline{r}(a^*) \tag{7.7}$$

ただし，

$$\overline{r}(l) = \begin{cases} 0 & l = 0, \ldots, n-2 \\ r(l) & l = n-1, n, n+1 \\ r(l) \text{ の上界} & l = n+2 \ldots, m \end{cases} \tag{7.8}$$

Step 4 の $z(l)$ は次の問題 P(l) の最適値であり，後述のアルゴリズム JANSUB(l) によって求められる．

問題 P(l)

$$\text{Minimize} \quad \sum_{e_i \in E} c(e_i) \cdot x_i \quad \equiv z(l) \tag{7.9}$$

$$\text{Subject to} \quad Rel(G_x) \geq P_0 \tag{7.10}$$

$$\sum_{e_i \in E} x_i = l \tag{7.11}$$

$$x_i \in \{0, 1\},\ i = 1, \ldots, m \tag{7.12}$$

Step 7 の$\underline{z}(l)$ は採用候補枝の集合 Eの中から l本選択したときの最小コストで，コストの小さな方から枝を l本選択し，それらのコストを合計することにより簡単に求めることができる．$\underline{z}(l) < z^*$が成立するということは，l本の枝からなるネットワークには現時点での最適解のコスト z^*より低いコストで構築できるものは存在しないことを表すので，この条件が成立するとアルゴリズム JAN は終了する．

アルゴリズム JANSUB(l) は分枝限定法を適用して問題 P(l) の解を求めるアルゴリズムである．図 7.1にネットワーク例と，そのネットワークに対する問題 P(3) の分枝木を示した．図 7.1(a) で示すネットワークは 4 個の節点と 5 本の採用候補枝を持つ．枝はコストに関して昇べきの順に並んでいるとする．つまり $i < j \Longrightarrow c(e_i) \leq c(e_j)$ とする．図 7.1(b) には，ある節点から子供への枝に付くラベルは，親からその節点への枝に付けられているラベルより小さくてはいけない，という規則がある．分枝木において，枝上のラベルは採用した枝の番号を表し，節点は枝の選択方法の一つを表している．例えば，一番頂上にある根から黒い節点へのパス上の枝には 1 と 3 のラベルがあるので，黒い節点は枝 e_1と e_3を

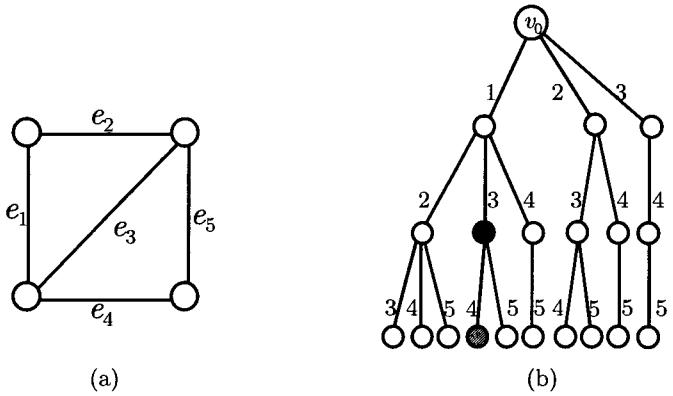


図 7.1 (a): ネットワーク例 (b): 問題 P(3) に対する分枝木

採用し，枝 e_2, e_4, e_5は採用しないという選択を表している．図 7.1(b) の分枝木は問題 P(3) に対する図であるので，深さ 3 より深い節点は存在しない．分枝木における葉は問題 P(l) の解となり，葉の総数は${}_mC_l$となる．分枝木における節点が表す枝の選択をしたときにかかるコストを，その節点のコストと呼ぶことにする．

以下にアルゴリズム JANSUB(l) を示す．

Procedure JANSUB(l)
begin
1. *live node list* $\leftarrow$ {the root v_0}
2. $UC \leftarrow \infty$
3. **while** *live node list* $\neq \phi$
4. $v \leftarrow$ the node v with the minimum value of $g(v)$ from *live node list*
5. **if** $g(v) \geq UC$ **then** exit while-loop
6. **if** node v is not at level l **then**
7. add the first child and the next brother of node v to *live node list*
8. **else**
9. **begin** /* Feasibility testing */
10. **if** $G(v)$ is not connected **then** $flag \leftarrow$ Infeasible
11. **else if** an upper bound of $Rel(G(v)) < P_0$ **then** $flag \leftarrow$ Infeasible
12. **else if** $Rel(G(v)) < P_0$ **then** $flag \leftarrow$ Infeasible
13. **else** $flag \leftarrow$ Feasible
14. **if** $flag =$ Feasible **then** $w \leftarrow v, UC \leftarrow g(v)$
15. **else** add the next brother of node v to *live node list*
16. **end**

17.　remove node v from *live node list*
18. **end**
19. **output** the optimal network layout $G(w)$ and the optimal value UC.
end

分枝木中の節点 vに対し，$g(v)$ は vの子孫の間での最小コストを表す．例えば，節点 vを図 7.1(b) の黒い節点とすると，図 7.1では枝がコストの昇順に並んでいるので，$g(v)$ は斜線のかかった節点のコスト，すなわち $c(e_1)+c(e_3)+c(e_4)$ となる．$G(v)$ は節点 vの表す枝の選択によって作成されるネットワークを表す．例えば斜線の節点を vとすると，$G(v)=(V,E_x)$, $E_x=\{e_1,e_3,e_4\}$ となる．

総合信頼度を求める問題は#P 完全であるので，総合信頼度を計算する処理が最も時間のかかる処理であると想像できる．よって実行可能性を評価する度に式 (7.2) を直接評価すると，その度に総合信頼度 $Rel(G_x)$ を計算する必要があるため，計算時間がかかることになる．グラフ G_xの連結性や総合信頼度 $Rel(G_x)$ の上界を評価することにより実行可能性を判断できることがあるため，それらを考慮することにより $Rel(G_x)$ を計算する回数を減少させ，計算時間を短縮することを狙っている．$Rel(G_x)$ の上界は式 (7.8) で定義された $\bar{r}(l)$ を用いる [35]．$\bar{r}(l)$ は枝確率が全て等しい l本の枝で構成される最大総合信頼度であるので，G_xの枝本数が l本であるとき，$\bar{r}(l)$ は $Rel(G_x)$ の上界になる．ただしこの方法は$\bar{r}(l)$ を利用するので，枝確率が全ての枝について同一でなければ適用できない．

7.3.2　Jan のアルゴリズムの拡張

Jan のアルゴリズムを枝確率が異なるネットワークにも適用できるように拡張するためには，彼のアルゴリズムの中で枝確率が同一であることが必要である処理を枝確率が異なっていても適用できるようにすればよい．具体的には，a^*の計算と総合信頼度の上界の計算に関する処理である．

a^*を計算するためには，$\bar{r}(l)$ を求めることが必要である．$l=0,\ldots,n-2$ については枝確率の分布に関係なく，$\bar{r}(l)=0$ である．$n-1$ の場合については，$r(n-1)$ は総合信頼度最大の極大木となるので，[47] や [66] などの最大木を求めるアルゴリズムを使って計算することができる．$l\geq n$ に対しては，枝確率が異なる場合 $r(l)$ を求める効率的なアルゴリズムは見つけられていないので，総合信頼度の単調性を利用して $r(l)$ の上界を求めることにする．総合信頼度の単調性に関する補題を以下に示す．

補題 7.1 確率グラフ $G=(V,E)$ に対し，枝 $e \in E$の枝確率 $p(e)$ を $p(e)+\delta$ $(\delta > 0)$ に変更したときの確率グラフを $G'=(V,E)$ とする．このとき次式が成立する．

$$Rel(G') > Rel(G) \tag{7.13}$$

証明

確率グラフ G において，枝 e に対し削除縮約展開式 (3.3) を適用すると，

$$Rel(G) = p(e)Rel(G \cdot e) + (1-p(e))Rel(G-e),$$

となる．同様に確率グラフ G'についても展開すると，

$$\begin{aligned} Rel(G') &= (p(e)+\delta)Rel(G' \cdot e) + (1-p(e)-\delta)Rel(G'-e) \\ &= (p(e)+\delta)Rel(G \cdot e) + (1-p(e)-\delta)Rel(G-e) \\ &= Rel(G) + \delta(Rel(G \cdot e) - Rel(G-e)) \end{aligned}$$

となる．$G-e$ が正常状態のとき，$G \cdot e$ は必ず正常状態になるので，枝 e が自己閉路でないことから，$Rel(G \cdot e) > Rel(G-e)$ となる．従って証明すべき式が成立する．

証明終

補題 7.2 確率グラフ $G=(V,E)$ の有する枝のうち，最大の枝確率を p_{max}とする．全ての枝の枝確率を p_{max}に変更したときの確率グラフを $G''=(V,E)$ とすると，以下の式が成立する．

$$Rel(G'') \geq Rel(G) \tag{7.14}$$

証明

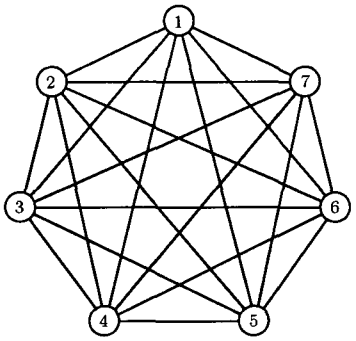
補題7.1より明らか．なお，等号は確率グラフ G の有する枝の枝確率が全て等しいときのみ成立する．

証明終

補題7.2にて作成された確率グラフ G''において枝確率は全て等しいので，7.3.1節で示した枝確率が全て等しいときに適用できる手法を適用することが可能となり，a^*や総合信頼度の上界を計算することが可能である．この手法により，アルゴリズムJANは枝確率が異なるネットワークに対しても適用することが可能となる．この拡張されたアルゴリズムをアルゴリズムM-JAN(Modified-JAN)と呼ぶことにする．

7.4 計算時間短縮のための工夫

7.4.1 アルゴリズムの構造



From\ To	2	3	4	5	6	7
1	97	60	65	54	71	82
2		49	72	31	38	81
3			9	5	82	62
4				1	74	52
5					89	28
6						59

図 7.2 対象ネットワーク K_7とそのコスト分布

本節ではまずアルゴリズム JAN の計算時間に関するある特徴を示す．節点数 7 の完全グラフ (枝数 21) を数値実験対象とした．枝確率は全ての枝について等しく 0.9，コストは閉区間 [1,100] 間の整数値としてランダムに設定した（図 7.2）．満たすべき総合信頼度の水準である R_0の値を 0.9 から 1.0 まで 0.001 刻みで変化させ，それぞれの R_0値に対しアルゴリズム JAN が最適解を求めるまでの計算時間を測定した．その結果を図 7.3に示す．横軸は R_0の値，縦軸は計算時間を表す．数値計算は SUN Ultra 60 (CPU: 360MHz, Memory: 128MB) 上で行った．

図 7.3より，アルゴリズム JAN は大抵の場合は短時間で最適解を見つけることができるが，R_0の値によっては非常に時間がかかることがわかる．この現象が生じるのは，アルゴリズム JANSUB(l) が実行可能解を持たない問題 $P(l)$ に対して最適解を探索するときである．例えば $R_0 \in [0.960, 0.969]$ の場合，

$$r(8) = 0.9265,\ \bar{r}(9) = 0.9700$$

より $a^* = 9$ となり，アルゴリズム JAN はアルゴリズム JANSUB(9) を呼び出す．しかし，$r(9) = 0.9598$ であるので問題 $P(9)$ は実行可能解を持たない．アルゴリズム JANSUB(9) は実行可能解が存在しない問題 $P(9)$ に対する分枝木の中を探索することになる．後述の通り，分枝限定法では暫定解の存在が計算時間の短縮に大きく貢献するため，実行可能解がない場合は計算時間が非常に大きくなることが多い．計算時間が大きくなったその他の R_0の値についても，実行可能解の存在しない部分問題を解いていたことが原因であった．

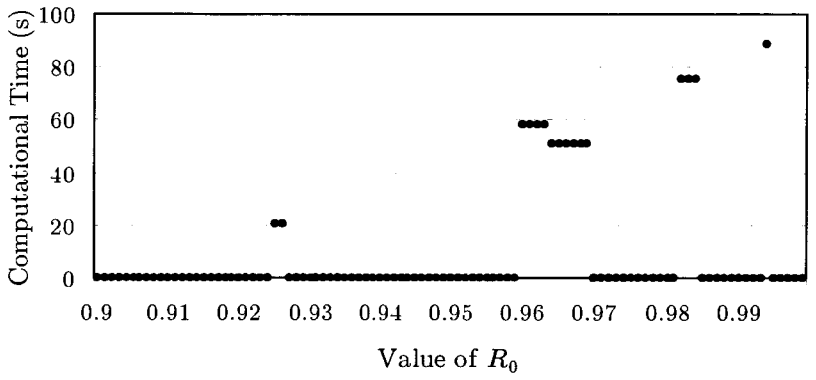


図 7.3 アルゴリズム JAN の R_0に対する計算時間

なお $R_0 \in [0.960, 0.963]$ に対する計算時間が，$R_0 \in [0.964, 0.969]$ に対するものより大きいのは，後述する総合信頼度の上界による限定操作のためである．一般に R_0の値が大きくなるほど，総合信頼度の上界が R_0より小さくなる頻度が高くなる．今回のそれぞれの場合について総合信頼度の真値を計算した回数を調べると，52087 回と 19935 回であった．すなわち後者の方が総合信頼度の上界が R_0より小さくなる頻度が高く，より多くのケースに対して総合信頼度の真値を計算することなく制約条件を満足しないことが判明したことが分かる．

結論として計算時間が非常に大きくなるのは，$r(l)$ の値と上界とのギャップが原因である．図 7.3は枝確率が全て等しい場合での結果であるが，枝確率が枝ごとに異なる場合にアルゴリズム M-JAN を適用した場合，一般に $r(l)$ の上界の精度は劣化するのでこの現象が起こる頻度が高くなることが予想できる．R_0の設定の仕方によって極端に計算時間が変化することは，アルゴリズムの利用者にとって計算終了時間を予想できないという意味であまり好ましいことではない．そこでアルゴリズム JAN が採用している部分問題へ分割する方法を却下し，新しいアルゴリズムの構造を構築することにする．そのために，分枝木内の深さ kにある節点 vのコスト評価関数 $g(v)$ を以下のように再定義する．

$$g(v) = \begin{cases} \text{アルゴリズム JAN における } g(v) & k \leq a^* \\ \text{節点 } v\text{のコスト} & k \geq a^* \end{cases} \tag{7.15}$$

以下に構造を変更した新しいアルゴリズム TOPO(TOPOlogical optimization) を以下に示す．

Procedure TOPO
begin
1. compute a^*
2. $z^* \leftarrow \infty$, *live node list* $\leftarrow$ {the root v_0}
3. **while** *live node list* $\neq \phi$
4. $v \leftarrow$ the node v with the minimum value of $g(v)$ from *live node list*
5. **if** $g(v) \geq z^*$ **then** exit while-loop
6. **if** node v is above level a^* **then** $flag \leftarrow$ Infeasible
7. **else**
8. **begin** /* Feasibility testing */
9. **if** $G(v)$ is not connected **then** $flag \leftarrow$ Infeasible
10. **else if** an upper bound of $Rel(G(v)) < P_0$ **then** $flag \leftarrow$ Infeasible
11. **else if** $Rel(G(v)) < P_0$ **then** $flag \leftarrow$ Infeasible
12. **else** $flag \leftarrow$ Feasible
13. **end**
14. **if** $flag =$ Feasible **then** $w \leftarrow v, z^* \leftarrow g(v)$
15. **else** add the first child and next brother of node v to *live node list*
16. remove node v from *live node list*
17. **end**

end
output the optimum network layout $G(w)$ and the optimal value z^*

アルゴリズムの中で最も時間がかかる処理は総合信頼度を計算する処理であると予想できるので，アルゴリズム M-JAN とアルゴリズム TOPO における総合信頼度の計算回数を評価した．総合信頼度の計算回数について，以下の補題を証明した．

補題 7.3 アルゴリズム TOPO における総合信頼度の計算回数は，アルゴリズム M-JAN における回数よりも少ない．ただし $g(v)$ を最小とする節点集合から節点を抜き出す処理や総合信頼度の上界を計算する処理は同一であるとする．

証明

現在アルゴリズム TOPO が深さ k $(\geq a^*)$ にある節点 vの実行可能性を評価しているとする．$S_T(v)$ と $S_J(v)$ をそれぞれアルゴリズム TOPO，アルゴリズム M-JAN によって節点

vより先に実行可能性を評価された節点の集合とする．深さa^*，およびそれより深いレベルにある実行不可能な節点に対し，アルゴリズム M-JAN はその節点の次の弟を探索対象に加えるが，アルゴリズム TOPO は更にその節点の第1子も追加する．よって2つのアルゴリズムとも *live node list* から次の節点を抜き出す処理が同一であるため，$S_T(v) \supseteq S_J(v)$となる．節点vが現在評価されているということは，$S_T(v)$には節点vのコストより小さなコストをもつ実行可能な節点は存在せず，$S_T(v) \supseteq S_J(v)$より$S_J(v)$にも存在しない．よって，節点vはアルゴリズム JAN においても実行可能性を評価されることになる．2つのアルゴリズムとも同一の総合信頼度の上界を求める処理を行う仮定から，アルゴリズム TOPO が実行可能性を評価するために$Rel(G(v))$を計算するならば，アルゴリズム M-JAN も計算することになる．

証明終

補題7.3より，アルゴリズム TOPO はアルゴリズム M-JAN より総合信頼度の計算回数は少なくなるので，計算時間についても短くなることが期待できる．実際に図7.2のネットワークにアルゴリズム TOPO を適用したところ，R_0の値に関わらず0.1秒以下で最適解を探索することができた．更に詳しい解析は7.5節で行う．

7.4.2 総合信頼度の上界

補題7.1と Jan らによる方法で，枝確率が異なるネットワークに対する総合信頼度の上界を求めることができることを示した．しかし Jan らによる方法は元々枝確率が同一であるネットワークに対する方法であるため，あまり精度の良い上界が得られない．特に枝確率の分布範囲が大きい場合はその傾向が強くなる．そこで総合信頼度の上界を求める新たな方法として，cut basis を用いる方法を以下に提案する．

定義 7.4 2つのカット$(X, \overline{X})$と$(Y, \overline{Y})$に対し，$X \cap Y$, $X \cap \overline{Y}$, $\overline{X} \cap Y$, $\overline{X} \cap \overline{Y}$の全てが空集合でなく要素を持つとき，2つのカット$(X, \overline{X})$と$(Y, \overline{Y})$は**交わる** (*crossing*) という．逆に，$X \cap Y$, $X \cap \overline{Y}$, $\overline{X} \cap Y$, $\overline{X} \cap \overline{Y}$のうち1つでも空集合があるとき，2つのカットは**交わらない** (*non-crossing*) という．互いに交わらない$n-1$個のカットを *cut basis* という．

確率グラフGの cut basis を構築したとき，以下の定理が成立する [52].

定理 7.5 確率グラフGが cut basis $\{C_1, \ldots, C_{n-1}\}$をもつとき，次式が成立する．

$$Rel(G) \leq \prod_{i=1}^{n-1} (1 - \prod_{e \in C_i} (1 - p(e))). \tag{7.16}$$

式 (7.16) の右辺は Lomonosov–Polesskii の上界と呼ばれ，枝確率が異なるネットワークに対しても計算可能な上界の一つである．

総合信頼度の上界はアルゴリズム中で非常に頻繁に計算するため，短時間で cut basis を構築する必要がある．簡潔な cut basis の構築法を以下に提案する．$C(v)$ を節点 vを終点とする枝の集合とすると，$C(v)$ は以下の式より 1 つのカットになっていることがわかる．

$$C(v) = (\{v\}, \overline{\{v\}}) \tag{7.17}$$

異なる 2 節点 u と vに対し，$\{u\} \cap \{v\} = \phi$より $C(v)$ と $C(u)$ は交わらないので，異なる $n-1$ 個の節点に対する $C(v)$ を集めたものは cut basis になる．上界の精度をなるべく良くするために，以下の手順で cut basis を構築する．

1. 全ての $v \in V$に対し以下の値を計算する．

$$p(C(v)) \equiv 1 - \prod_{e \in C(v)} (1 - p(e)) \tag{7.18}$$

2. $p(C(v))$ が最大である節点を除き，全ての節点に関する $C(v)$ を集める．

この手法により，短時間で cut basis を構築することができる．

7.4.3　総合信頼度の計算時間の短縮

分枝木内の節点 vの親を u とし，それぞれの節点が表すネットワークを $G(v) = (V, E(v))$, $G(u) = (V, E(u))$ とする．節点 vは節点 u の子だから，$E(v) = E(u) \cup \{e\}$ と書くことができるので，削除縮約展開式 (3.3) より，

$$\begin{aligned} Rel(G(v)) &= p(e)Rel(G(v) \cdot e) + (1 - p(e))Rel(G(v) - e) \\ &= p(e)Rel(G(v) \cdot e) + (1 - p(e))Rel(G(u)). \end{aligned} \tag{7.19}$$

が成立する．従ってもし $Rel(G(u))$ が既に計算済みであるなら，$Rel(G(v))$ は $Rel(G(v) \cdot e)$ を計算することによって求めることができる．$G(v)$ と $G(v) \cdot e$ との枝本数の差はたかが 1 であるが，総合信頼度を求める問題は#P 完全であるので，アルゴリズム全体で考えた場合にはこの差による計算時間の短縮も重要になると考えられる．

7.5 数値実験と考察

7.5.1 総合信頼度の目標水準に対する計算時間の変化

本節では枝確率が異なるネットワークに対し7.4.1節で行った数値実験，すなわち総合信頼度の目標水準R_0を変化させたときに，アルゴリズムM-JANとアルゴリズムTOPOが最適解を求めるのに必要な計算時間がどのように変化するのかを確認する．対象のネットワークは図7.2で示したグラフで，枝確率は閉区間[0.85,0.95]間の小数第3位の小数としてランダムに設定した(表7.1).

表 7.1 枝確率分布

	1	2	3	4	5	6	7
1		0.914	0.914	0.853	0.853	0.896	0.866
2			0.931	0.896	0.918	0.885	0.935
3				0.936	0.879	0.858	0.941
4					0.898	0.868	0.932
5						0.910	0.888
6							0.937

R_0の値を0.9から1.0まで0.001刻みで変化させ，それぞれのR_0の値に対しそれぞれのアルゴリズムが最適解を見つけるまでの計算時間を測定した．アルゴリズムM-JAN，アルゴリズムTOPOに関する実験結果をそれぞれ図7.4，図7.5に示した．それぞれの図において，横軸はR_0の値，縦軸は計算時間を表す．なおこの数値実験はSUN Ultra 60 (CPU: 360MHz, Memory: 128MB) 上にて行った．

7.4.1節で，$r(\cdot)$の真値と上界とのギャップにより，アルゴリズムJANが時折非常に大きな計算時間を必要とすることについて述べた．図7.4を見ると図7.3よりも，多くの計算時間を必要とする頻度が大きくなっていることがわかる．このことは枝確率が同一の場合に比べ，枝確率が異なる場合は$r(\cdot)$の真値と上界とのギャップがより大きくなることを表している．図7.5を見るとR_0の値が極めて1に近い場合を除けば，R_0の値に対しさほど影響を受けずに一定時間内で最適解を求めていることがわかる．この結果により，アルゴリズムTOPOはアルゴリズムM-JANと比較して，R_0の値に関わらず短時間で最適解を探索することができると結論できる．

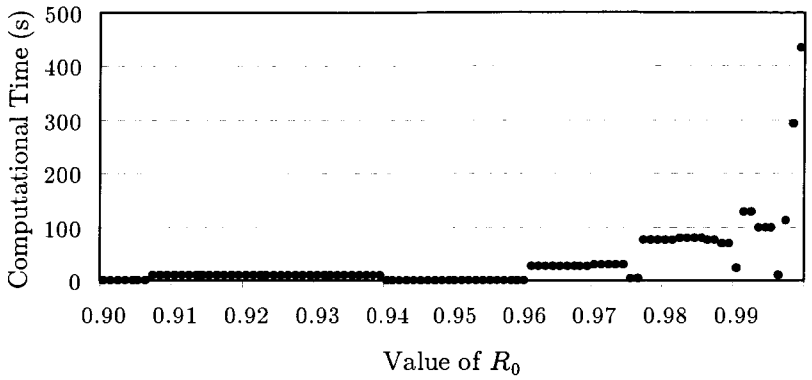


図 7.4 アルゴリズム M-JAN の計算時間

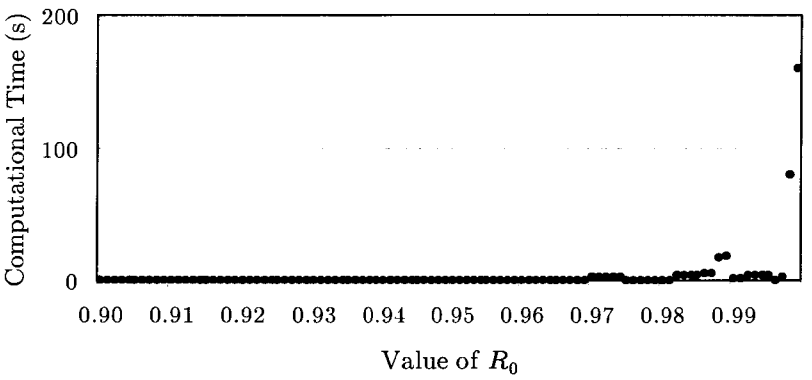


図 7.5 アルゴリズム TOPO の計算時間

7.5.2　枝確率の分布に対する計算時間の変化

次に様々な枝確率の分布に対して，2 つのアルゴリズムの計算時間がどのように変化するのかを数値実験により確認した．対象となるグラフは前節と同じく K_7で，コストは閉区間 [1,100] 間の整数値としてランダムに，また枝確率の元となる基数を [-1,1] 間の実数としてランダムに設定したネットワークを 20 個作成した．枝確率については 2 つの方針に基づいて設定した．第 1 の方針は枝確率分布の中心値の変化に対する実行時間の変化を評価するために，$p = 0.85, 0.9, 0.95$ なる p に対し枝確率の分布範囲を $[p - 0.05, p + 0.05]$ とした．枝確率は枝に設定された基数を元に設定される．例えば $p = 0.85$ の場合，枝確率の分布範囲は $[0.8, 0.9]$ となり，基数が 0.4 に設定された枝の枝確率は $0.85 + 0.4 \times 0.05 = 0.87$ と設定される．第 2 の方針は枝確率分布の範囲の変化に対する実行時間の変化を評価する

ために，$\delta = 0.01, 0.03, 0.05, 0.1$ なるδに対し枝確率の分布範囲を $[0.9 - \delta, 0.9 + \delta]$ とした．この場合の枝確率も基数を元に設定され，例えば$\delta = 0.05$ で基数が 0.4 の場合，枝確率は $0.9 + 0.4 \times 0.05 = 0.92$ となる．ただしいずれの場合も，枝確率は小数第 2 位までになるように第 3 位で四捨五入した．例えば第 2 の方針の場合で$\delta = 0.01$ のときは，枝確率は 0.89, 0.90, 0.91 の 3 通りしかない．従って第 1 の方針に基づく枝確率の分布範囲が 3 通り，第 2 の方針に基づく分布範囲が 4 通りで合計 7 通り，それぞれの分布範囲と枝に付加した基数に基づき枝確率を設定してネットワークを作成するので，$7 \times 20 = 140$ 種のネットワークを作成した．それぞれのネットワークに対し，総合信頼度の目標水準 R_0は 0.90, 0.95, 0.97, 0.99 の 4 通りの場合を設定し，最適解を求めるまでの計算時間を測定した．計算結果を表 7.2と表 7.3にまとめた．

表中の新規パラメータは以下のように定義した．

- $\hat{a^*}$; a^*の平均値
- $\hat{m}_x$; 最適解となるネットワークの有する枝本数の平均値
- $\hat{N}_f$; 分枝木において実行可能性を評価された節点数の平均値
- $\hat{N}_R$; 総合信頼度を計算した平均回数
- $\hat{t}$; 平均計算時間 (CPU 秒).

表 7.2, 7.3より，アルゴリズム M-JAN と比べてアルゴリズム TOPO は枝確率の分布状態に関係なく常に短い時間で最適解を探索していることがわかる．更にアルゴリズム TOPO において$\hat{N}_R$の値は$\hat{N}_f$より遥かに少ないことから，cut basis による総合信頼度の上界が総合信頼度の計算回数を減らすことに極めて大きく貢献していると結論付けることができる．

いずれのアルゴリズムも以下の条件のときに計算時間がかかることが予想できる．

1. $m_x - a^*$ が大きいとき．
2. a^*や m_xが $m/2$ に近いとき．
3. a^*や m_x が大きいとき．

アルゴリズムは分枝木の深さ a^*から深さ m_xまで（またはそれ以深まで）探索する必要があるため，条件 1 が成立すると計算時間は増大する．また m 本の枝の中から k本の枝を選択する場合の数は${}_mC_k$だから，条件 2 が成立するとアルゴリズムがより広い選択肢の中か

表 7.2 枝確率分布の中心値を変化させた場合の計算時間

				Algorithm TOPO			Algorithm M-JAN		
p	P_0	$\hat{a}^*$	$\hat{m}_x$	$\hat{N}_f$	$\hat{N}_R$	$\hat{t}$	$\hat{N}_f$	$\hat{N}_R$	$\hat{t}$
0.80	0.90	9.0	10.4	5576	17	0.09	296169	57239	57.29
	0.95	10.0	11.8	15347	10	0.65	423894	20767	106.12
	0.97	11.0	12.9	18833	7	1.04	534667	32886	190.42
	0.99	13.5	14.7	13794	4	1.17	154931	11832	87.96
0.85	0.90	8.0	9.2	1713	20	0.01	193603	83537	24.12
	0.95	9.0	10.2	3850	9	0.05	294155	56435	56.60
	0.97	10.0	11.1	3168	8	0.03	302150	75392	98.49
	0.99	11.0	12.9	26026	3	1.94	544702	26798	180.68
0.90	0.90	7.0	8.0	950	7	0.00	54870	31936	3.04
	0.95	7.5	8.9	2592	11	0.03	138185	49717	10.99
	0.97	8.0	9.8	4886	13	0.05	202764	12454	19.45
	0.99	9.5	10.9	8970	4	0.22	287464	52248	66.61
0.95	0.90	6.7	7.1	163	11	0.00	5639	1706	0.10
	0.95	7.0	7.5	271	5	0.00	330	166	0.00
	0.97	7.0	7.6	699	4	0.00	13104	7402	0.61
	0.99	7.0	8.8	4848	6	0.05	141242	63344	10.14

ら最適解を探索する必要があるため，計算時間が増大すると考えられる．最後に条件3が成立すると，総合信頼度の計算時間が増大するために最適解を求める計算時間も増大すると考えられる．表7.2より，枝確率分布の中心値が減少する，かつ／またはR_0が大きくなると条件3が成立するので計算時間が増大したと考えられるが，結果的に条件2も成立するので条件2の影響も考えられる．表7.3より，枝確率分布の範囲が広くなるとa^*の精度が悪くなるので条件1が起こりやすくなり，計算時間が増大したと考えられる．結論として，アルゴリズムTOPOに対しては条件1が，アルゴリズムM-JANに対しては条件2が計算時間の増大に最も強く影響すると考えられる．アルゴリズムの構造を変えることなく計算時間を短縮する最も効果的な方法は，a^*の精度を良くすることである．

総合信頼度の上界の精度を上げることも計算時間の短縮に大きく関わってくるが，上界の計算は非常に頻繁に行うため，上界の精度を上げるために必要な少しの時間が結果的に非常に多くの時間を必要とすることも考えられる．更に分枝木内の節点を探索する順番が

表 7.3 枝確率の分布範囲を変化させた場合の計算時間

				Algorithm TOPO			Algorithm M-JAN		
δ	P_0	$\hat{a}^*$	$\hat{m}_x$	$\hat{N}_f$	$\hat{N}_R$	$\hat{t}$	$\hat{N}_f$	$\hat{N}_R$	$\hat{t}$
0.01	0.90	8.0	8.1	99	2	0.00	64	39	0.00
	0.95	9.0	9.1	65	6	0.00	44	8	0.00
	0.97	9.0	10.0	889	11	0.01	293944	52085	55.63
	0.99	11.0	11.0	241	1	0.00	137	1	0.00
0.03	0.90	7.3	8.1	832	3	0.00	78146	45853	4.40
	0.95	8.0	9.0	882	17	0.00	160605	8908	14.27
	0.97	9.0	10.0	677	12	0.00	108621	22050	19.85
	0.99	10.0	11.0	4219	4	0.06	336966	5433	67.17
0.05	0.90	7.0	8.0	950	7	0.00	54870	31936	3.04
	0.95	7.5	8.9	2592	11	0.03	138185	49717	10.99
	0.97	8.0	9.8	4886	13	0.05	202764	12454	19.45
	0.99	9.5	10.9	8970	4	0.22	287464	52248	66.61
0.10	0.90	7.0	7.9	672	11	0.00	7281	3946	0.24
	0.95	7.0	8.5	3728	12	0.04	71096	41762	4.04
	0.97	7.0	9.0	9923	8	0.20	175774	115516	17.15
	0.99	7.2	10.2	43056	5	2.68	414151	162490	62.41

異なれば暫定解の精度や見つかるまでの時間が変わるため，最終的な計算時間も変化する．その意味でも分枝限定法を用いたアルゴリズムの計算時間に関する評価を系統的に行うのは非常に難しい．

7.6 おわりに

本章では総合信頼度を考慮に入れた，枝確率が異なるネットワークの設計問題を扱った．R. -H. Jan らはその問題の最適解を導出するアルゴリズム JAN を提案したが，そのアルゴリズムは枝確率が異なるネットワークには適用できず，枝確率が同一である場合にしか適用できないという短所を持っていた．そこで総合信頼度の単調性を利用して，彼らのアルゴリズムを枝確率が異なる場合にも適用できるように拡張する方法を提案した．また彼らのアルゴリズムには総合信頼度の目標水準が少し変化するだけで，計算時間が急激に増大

するという欠点があることを示し，アルゴリズムの構造を変更することによってその短所を補えることを示した．更に総合信頼度の上界を求める新たな手法と，総合信頼度の計算時間を短縮するための工夫を提案し，それらが計算時間の短縮に大きく貢献することを数値実験により示した．提案したアルゴリズムは枝確率の分布状態に関係なく，アルゴリズムJANと比較すると短い計算時間で最適解を探索することができることを数値実験により確認した．枝確率が同一の場合も異なる場合も，制約条件を満足するために必要な最低枝本数の上界a^*の精度を上げることが，アルゴリズムの構造を変えることなく計算時間を短縮する最も効果的な方法であると考えられる．

本章で扱ったネットワーク設計問題については様々な発展形が考えられる．総合信頼度以外のネットワーク信頼度を考慮したり，枝の故障だけでなく節点の故障を追加してネットワークモデルを変更したりすることもできる．またネットワーク信頼度だけでなく，枝に長さの要素を与えて特定の（または任意の）節点間の距離についての制約条件を追加したり，節点間にデータ転送需要量を定め，枝に転送速度を設定して，データ転送時間についての制約条件を追加したりすることもできる．実際にこれらのモデルに対して研究が行われているが，それらの研究において最終的に導出するのは最適解ではなく近似解である[49, 61]．本研究の成果を見ても分かるように，モデルを複雑にすればするほどその最適解を求めるための時間は非常に大きくなることを覚悟する必要があるが，本研究のように現在の処理能力で実用可能な時間内で最適解を計算可能なモデルを追究することは非常に有意義であると考えている．現実社会に様々なネットワークシステムが導入されるにつれ，ネットワークシステムの形態も多様化するであろう．その意味からも，理論面を重視して実用可能なモデルを拡張していく研究は今後もますます有用であると言えよう．

第 8 章

結論

8.1 本研究のまとめ

本論文ではネットワーク信頼度の一つである総合信頼度に関する研究を行った．従来の研究では枝の故障する確率が全ての枝について等しいネットワークを対象とすることが多かったが，本研究では故障確率が異なる枝を含む，より一般的なネットワークを対象とした．本研究は，解析が困難なためこれまであまり研究されていなかった枝の故障確率が異なるネットワークの総合信頼度に関して新たな知見を得たと同時に，今後この分野を研究する上での一つの評価基準を与えたと言える．本研究の成果は，今後より現実的なネットワークモデルへ拡張する上での大きな足がかりになるであろう．

第 4〜6 章では総合信頼度の下界を多項式時間で導出する方法について述べ，導出する下界の精度に関するいくつかの定理を導出した．それらの定理に基づき，総合信頼度の下界を多項式時間で導出するアルゴリズムを提案し，アルゴリズムの計算量を評価し，数値実験によりそれらの性能を確認した．第 4 章で提案した極大木によるエッジ・パッキング法はグラフ変換を組み合わせることによって，枝確率が全て等しいネットワークに対しても Ball-Provan の下界などの信頼度多項式による方法より精度の良い下界を導出する可能性があることを示した．この結果をもとに，更に精度の良い下界を導出するために研究を進めた結果，第 5 章で提案した直並列グラフによるエッジ・パッキング法と，第 6 章で提案したアサイクリックグラフによるアーク・パッキング法が同程度の精度を持つ下界を導出する手法を示した．枝の故障確率が異なるネットワークについて総合信頼度の下界を評価する研究はこれまでほとんどなされていないため，本研究により導出された下界の精度

の良し悪しを厳密に結論付けるのは難しい．しかし全枝の故障確率を全枝の平均値と仮定して Ball-Provan の手法と比較したり，総合信頼度の真値と比較した結果，本研究による総合信頼度の下界は十分な精度を有していると判断できる．第4章で提案した手法は，導出する下界の精度という点では第5章や第6章で提案した手法に劣るものの，極大木がグラフ的マトロイドにおける基になっている点で理論的追究が容易であるので，本論文での成果を他分野への応用することを考えた場合，その可能性は最も高いであろう．

第7章では，総合信頼度を考慮したネットワーク設計問題の厳密解を探索するアルゴリズムを提案した．従来の手法とは異なり，枝確率が異なるネットワークにも適用できることが最大の利点である．本手法によって厳密解を導出することができるため，この問題に対する近似アルゴリズムが求める近似解の精度を正確に評価することが可能になる．また応用面を考慮して更にモデルを拡張した問題に対して厳密解や近似解を求めるアルゴリズムを開発するとき，本研究の成果は理論的基盤として十分に有用なものとなるであろう．

8.2　今後の展開

ネットワーク信頼性の研究分野において本研究で扱った内容はほんの一部であるが，中心の核になる部分である．よって本研究を基に様々な方向へ研究を展開することができるであろう．以下に本研究の今後の展開について述べる．

本研究では総合信頼度を扱ったが，同じネットワークモデルにおける連結性を考慮した尺度として2節点間信頼度，k節点間信頼度があり，いずれの信頼度においてもエッジ・パッキング法を適用することができる．総合信頼度においてエッジ・パッキングの要素として必要である条件は，全域部分グラフであることと総合信頼度が多項式時間で計算可能であることであった．同様に2節点間信頼度においては特定の2節点 s と t 間が到達可能であることと，2節点間信頼度が多項式時間で計算可能であることが必要となる．総合信頼度において上記条件を満足する最小枝の部分グラフは極大木であり，枝排反極大木についてはマトロイド分割問題を応用することにより様々な結果を利用することができた．その意味では，2節点間信頼度においてはあまり系統的な結果が得られていないので，必然的にヒューリスティックな手法が望まれる．k節点間信頼度は総合信頼度と2節点間信頼度を含むクラスであるので，更に難しい問題である．これらの信頼度に関する理論的追究を進める一方，精度の良い下界を導出するヒューリスティック手法を考案することは非常に重要である．その際，本研究で扱った定理や概念などが応用できると思われる．

本研究で扱ったネットワークモデルでは節点は故障しないとした．例えばコンピュータ・ネットワークを考慮した場合，節点は各種サーバやルータ・ゲートウェイなどの接続機器，枝はそれらを結ぶケーブルになるので，節点が故障せず枝が故障するモデルよりもむしろ，枝が故障せず節点が故障するモデルの方がより現実的であると考えることもできる．節点の故障を考えたネットワークの正常状態を，全節点が正常でかつ，全節点が正常な枝によって連結である状態とした場合は，3.1節で述べたように節点故障を枝故障へ変換することが可能である．それに対し，全節点ではなく正常な節点が正常な枝によって連結である状態をネットワークの正常状態とすることも考えられ，この場合は単純に節点故障のないモデルには変換できない．実際にこのモデルについても若干研究が進められている[20, 80]．しかし枝の故障のみを考慮する場合は単に故障した枝を除去した状態を考えればよいが，節点が故障した場合はその節点を端点とする枝も機能しなくなるので解析が一段と難しくなる．このモデルは節点について互いに独立であるため，エッジ・パッキング法と同様の手法を展開することにより本研究の成果を応用できる可能性がある．また本研究で扱ったネットワークモデルでは枝確率の独立性を前提としたが，枝確率が相互依存するモデルも考えられる．このモデルに対しては本研究の成果を単純に発展させることは簡単ではなさそうであるが，以下に述べるように時間の要素を取り入れることで解決できるのではないかと考え，現在着手しつつある．

本研究では枝確率を既知の一定値としたが，現実のモデルでは故障する確率は時間と共に変化することが多く，一般のシステム信頼性に関する研究においては時間の要素を考慮することが通常である．そこで枝確率を時間に関する関数で表し，ネットワークの連結性が時間と共にどのように変化するのかを調べる研究も興味深い．また故障した機器を修復することを許容するモデルも考えられる．時間の要素を追加した場合，比較的単純な構成をしているシステムに関しても故障確率や修復時間に関して様々なモデルが展開でき，現在でも広く研究されている．ネットワークシステムをモデルにした場合は更に解析が困難になることが容易に想像できるが，マルコフ連鎖モデルを応用する方法など研究され始めている[50, 68]．エッジ・パッキングの構成を時間と共に動的に変化させるなどの工夫を凝らせば，このモデルに対して本研究の成果を応用できると思われる．

第7章で扱ったネットワーク設計問題については，応用面を考慮に入れた様々な発展形が考えられる．実際にこれらの研究を行うにあたっては，理論面を重視するのか，応用面を重視するのかを明確化する必要がある．理論面を重視する場合は，現在のコンピュータ

環境において現実的な時間内に解（厳密解，または近似解）を求めることができるモデルを開発し，そのモデルに対し効果的なアルゴリズムを提案することが重要である．一方応用面を重視する場合は，対象となるネットワークシステムを表現する適切なモデルを構築することが重要であり，手法の形式は問わないが，そのモデルの解を実用的な時間で導出することが重要になる．応用面を重視する場合も当然，どのような手法が効果的に解を求めることができるのかを知っておく必要がある．その点からも本研究の成果を一つの足がかりとして，更に現実的なモデルに対して解を導出するアルゴリズムを開発したい．

謝 辞

本論文は大阪大学大学院工学研究科応用物理学専攻において筆者が行ったネットワークの信頼性に関する研究の成果をまとめ上げたのである．本学大学院博士前期課程修了後，東レ株式会社にて勤務をしておりました筆者に再び本学での研究の機会を与えて下さっただけでなく，本研究を遂行しまとめるにあたり，終始懇切なご指導ご助言を賜りました本学大学院工学研究科教授石井博昭先生に深く感謝の意を表すとともに，厚く御礼申し上げます．

本学大学院工学研究科教授 八木厚志先生，川上則雄先生，同助教授 西島国介先生，小松雅治先生には本論文作成にあたり細部にわたりご指導いただき，また貴重なご助言をいただきましたことに，心から感謝申し上げます．

本学大学院博士前期課程入学時からこれまで永きにわたり細部にわたるご指導いただいたばかりか，研究者の先輩として数々のご助言や激励のお言葉をいただきました鹿児島大学助教授（前 本学大学院工学研究科助手）新森修一先生に心から深く御礼申し上げます．

神戸学院大学教授 塩出省吾先生，本学大学院講師 齋藤誠慈先生，本学大学院助手 都田艶子先生，流通科学大学講師 伊藤健先生，神戸学院大学講師 毛利進太郎先生には本研究の遂行，および本論文の作成において，心あるご指導ご助言をいただきました．ここに深く御礼申し上げます．

最後に学友であります本学大学院工学研究科 片桐英樹氏をはじめとする石井・齋藤研究グループの皆様には互いに切磋琢磨し，大変な中とても楽しく研究ができる環境を提供していただいたことに感謝致します．

参考文献

[1] K. K. Aggarwal, Y. C. Chopra and J. S. Bajwa, "Topological layout of links for optimizing the overall reliability in a computer communication system", *Microelectonics & Reliability*, **22** (1982) 347-351.

[2] A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading Mass. (1974); 野崎，野下 (訳), アルゴリズムの設計と解析 I, II, サイエンス社 (1977).

[3] N. Alon, A. Frieze and D. Welsh, "Polynomial time randomized schemes for the Tutte polynomial of dense graphs", *Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science* (1994) 24-35.

[4] 荒木，ネットワークの信頼度の下界に関する一考察, 大阪大学工学部応用物理学科卒業研究論文 (1990).

[5] M. O. Ball, "Complexity of network reliability computations", *Networks*, **10** (1980) 153-165.

[6] M. O. Ball and J. S. Provan, "Bounds on the reliability polynomial for shellable independence systems", *SIAM Journal on Algebraic and Discrete Methods*, **3** (1982) 166-181.

[7] M. O. Ball and J. S. Provan, "Calculating bounds on reachability and connectedness in stochastic networks", *Networks*, **13** (1983) 253-278.

[8] M. O. Ball and J. S. Provan, "Disjoint products and efficient computation of reliability", Technical Report ORSA/TR-86/13, University of North Carolina (1986).

[9] F. T. Boesch, A. Satyanarayana and C. L. Suffel, "Least reliable networks and the reliability domination", *IEEE Trans. Commun.*, **38** (1990) 2004-2009.

[10] F. T. Boesch, X. Li and C. Suffel, "On the existence of uniformly optimally reliable networks", *Networks*, **21** (1991) 181-184.

[11] T. B. Brecht and C. J. Colbourn, "Improving reliability bounds in computer networks", *Networks*, **16** (1986) 369-380.

[12] T. B. Brecht and C. J. Colbourn, "Multiplicative improvements in network reliability bounds", *Networks*, **19** (1989) 521-529.

[13] J. I. Brown, C. J. Colbourn and J. S. Devitt, "Network transformations and bounding network reliability", *Networks*, **23** (1993) 1-17.

[14] J. A. Buzacott, "A recursive algorithm for finding reliability measures related to the connection of nodes in a graph", *Networks*, **10** (1980) 311-327.

[15] S. -T. Cheng, "Topological optimization of a reliable communication network", *IEEE Trans. Reliability*, **47** (1988) 225-232.

[16] Y. C. Chopra, B. S. Sohi, R. K. Tiwari and K. K. Aggarwal, "Network topology for maximizing the terminal reliability in a computer communication network", *Microelectronics & Reliability*, **24** (1984) 911-913.

[17] C. J. Colbourn, Combinatorics of Network Reliability, Oxford University Press, New York (1987).

[18] C. J. Colbourn, "Edge-packings of graphs and network reliability", *Discrete Math.*, **72** (1988) 49-61.

[19] C. J. Colbourn, "Combinatorial aspects of network reliability", *Ann. Oper. Res.*, **33** (1991) 3-15.

[20] C. J. Colbourn, A. Satyanarayana, C. Suffel and K. Sutner, "Computing residual connectedness reliability for restricted networks", *Discrete Appl. Math.*, **44** (1993) 221-232.

[21] B. Dengiz, F. Altiparmak and A. E. Smith, "Efficient optimization of all-terminal reliable networks, using an evolutionary approach", *IEEE Trans. Reliability*, **46** (1997) 18-26.

[22] R. J. Duffin, "Topology of series-parallel network", *J. Math. Anal. Appl.*, **10** (1965) 303-318.

[23] J. Edmonds, "Minimum partition of a matroid into independence subsets", *Journal of Research of the National Bureau of Standards*, **69B** (1965) 67-72.

[24] J. Edmonds, "Matroid partition", *Mathematics of the Decision Science*, American Mathematics Society (1968) 335-345.

[25] J. Edmonds, "Edge-disjoint branchings", in: *Combinatorial Algorithms* (R. Rustin, editor), Algorithmic Press (1972) 91-96.

[26] L. Fratta and U. G. Montanari, "A Boolean algebra method for computing the terminal reliability in a communication network", *IEEE Trans. Circuit Theory*, **CT-20** (1973) 203-211.

[27] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of NP-completeness", W. H. Freeman and Company, San Francisco (1979).

[28] F. Harary, Graph Theory, Addison-Wesley, Reading Mass.; 池田 (訳), グラフ理論, 共立出版社 (1971).

[29] 林, "双対グラフを用いたネットワークの信頼度解析", 信学論 A, **J74-A** (1991) 72-81.

[30] 茨木, 組合せ最適化－分枝限定法を中心として, 産業図書 (1983).

[31] 茨木 他, FORTRAN77 最適化プログラミング, 岩波書店 (1991).

[32] 今井, "ネットワーク信頼度計算の周辺－組合せ数え上げの新展開", 離散構造とアルゴリズム V(藤重悟編), 近代科学社 (1998) 1-50.

[33] 伊理 他, 演習グラフ理論, コロナ社 (1983).

[34] 石井, 応用代数, 京都コンピュータ学院教科書出版会 (1984).

[35] R. -H. Jan, F. -J. Hwang and S. -T. Chen, "Topological optimization of a communication network subject to a reliability constraint", *IEEE Trans. Reliability*, **42** (1993) 63-70.

[36] R. -H. Jan, "Design of reliable networks", *Computers and Operations Research*, **20** (1993) 25-34.

[37] M. Jerrum, "On the complexity of evaluating multivariate polynomials", Ph.D. thesis, Department of Computer Science, University of Edinburgh, 1981; also Report CST-11-81, Computer Science, University of Edinburgh (1981).

[38] D. Karger and R. P. Tai, "Implementing a fully polynomial time approximation scheme for all terminal network reliability", *Proceedings of the SIAM-ACM Symposium on Discrete Algorithms* (1997).

[39] R. M. Karp, "Reducibility among combinatorial problems", in: *Complexity of Computer Computations* (R. Miller and J. Thatcher, editors) Plenum Press (1972) 85-103.

[40] R. M. Karp, "On the computational complexity of combinatorial problems", *Networks* **5** (1975) 45-68.

[41] G. Katona, "A theorem of finite sets", in: *Theory of Graphs* (P. Erdös and G. Katona, editors) Akademia Kiadó, Budapest (1966) 187-207.

[42] S. Kiu and D. F. McAllister, "Reliability optimization of computer-communication networks", *IEEE Trans. Reliability*, **37** (1988) 433-440.

[43] 小出，新森，石井, "直並列グラフによるネットワーク信頼度の下界", 日本応用数理学会論文誌, **9** (1999) 15-35.

[44] T. Koide, S. Shinmori and H. Ishii, "The evaluation on lower bound of all-terminal reliability by arc-packing for general networks", *IEICE Trans.*, **E82-A** (1999) 784-791.

[45] T. Koide, S. Shinmori and H. Ishii, "A new algorithm for lower bounds of all-terminal reliability", *Mathematica Japonica* (accepted).

[46] T. Koide, S. Shinmori and H. Ishii, "Topological optimization of networks considering a reliability constraint", to be submitted to *Disc. Appl. Math.*

[47] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proc. Amer. Math. Soc.*, **71** (1956) 48-50.

[48] J. B. Kruskal, "The number of simplices in a complex", in: *Mathematical optimization Techniques* (R. Bellman, editor) University of California Press (1963) 251-278.

[49] A. Kumar, R. M. Pathak and Y. P. Gupta, "Genetic-algorithm-based reliability optimization for computer network expansion", *IEEE Trans. Reliability*, **28** (1995) 63-72.

[50] V. K. P. Kumar, S. Hariri and C. S. Raghavendra, "Distributed program reliability analysis", *IEEE Trans. Software Eng.*, **SE-12** (1986) 42-50.

[51] E. L. Lawler, Combinatorial optimization: networks and matroids, Holt, Rinehart and Winston (1976).

[52] M. V. Lomonosov and V. P. Polesskii, "An upper bound for the reliability of information networks", *Problems of Information Transmission*, **7** (1971) 337-339.

[53] M. V. Lomonosov and V. P. Polesskii, "Lower bound of network reliability", *Problems of Information Transmission*, **8** (1972) 229-236.

[54] H. Mine, "Reliability of physical systems", *IRE Trans. Circuit Theory*, **CT-6** (1959) 138-151.

[55] E. F. Moore and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays. Part I, II", *Journal of the Franklin institute*, **262** (1956) 191-208, 281-297.

[56] F. Moskowitz, "The analysis of redundancy networks", *AIEEE Trans. Commun. Electron*, **39** (1958) 627-632.

[57] C. St. J. A. Nash-Williams, "Edge-disjoint spanning trees of finite graphs", *Journal of the London Mathematical Society*, **36** (1961) 445-450.

[58] C. St. J. A. Nash-Williams, "Decomposition of finite graphs into forests", *Journal of the London Mathematical Society*, **39** (1964) 12.

[59] K. T. Newport and P. K. Varshney, "Design of survivable communications networks under performance constraints", *IEEE Trans. Reliability*, **40** (1991) 433-440.

[60] L. Petingi, J. T. Saccoman and L. Schoppmann, "Uniformly least reliable graphs", *Networks*, **27** (1996) 125-131.

[61] S. Pierre and A. Elgibaoui, "A tabu-search approach for designing computer-network topologies with unreliable components", *IEEE Trans. Reliability*, **46** (1997) 350-359.

[62] V. P. Polesskii, "A lower boundary for the reliability of information networks", *Problems of Information Transmission*, **7** (1971) 165-171.

[63] T. Politof and A. Satyanarayana, "A linear time algorithm for computing k-terminal reliability in series-parallel networks", *SIAM J.Comput.*, **14** (1985) 818-832.

[64] T. Politof and A. Satyanarayana, "Network reliability and inner-four-cycle-free graphs", *Math. Oper. Res.*, **11** (1986) 484-505.

[65] T. Politof, A. Satyanarayana and I. Tung, "An $O(n \cdot \log(n))$ algorithm to compute the all-terminal reliability of $(K_5, K_{2,2,2})$ free networks", *IEEE Trans. Reliability*, **41** (1992) 512-517.

[66] R. C. Prim, "Shortest connection networks and some generalizaions", *Bell System Tech. J.*, **36** (1957) 1389-1401.

[67] J. S. Provan and M. O. Ball, "The complexity of counting cuts and computing the probability that a graph is connected", *SIAM J. Comput.*, **12** (1983) 777-788.

[68] C. S. Raghavendra, V. K. P. Kumar and S. Hariri, "Reliability analysis in distributed systems", *IEEE Trans. Comput.*, **37** (1988) 352-358.

[69] A. Ramanathan and C. J. Colbourn, "Bounds on all-terminal reliability via arc-packing", *Ars Combinatoria*, **23A** (1987) 229-236.

[70] C. R. Reeves(editor), Modern heuristic techniques for combinatorial problems, Oxford Blackwell scientific publications (1993); 横山 他（訳）, モダンヒューリスティックス, 日刊工業新聞社 (1997).

[71] A. Satyanarayana and M. K. Chang, "Network reliability and the factoring theorem", *Networks*, **13** (1983) 107-120.

[72] A. Satyanarayana and R. K. Wood, "A liner-time algorithm for computing k-terminal reliability in series-parallel networks", *SIAM J. Comput.*, **14** (1985) 818-832.

[73] K. Sekine, H. Imai and S. Tani, "Computing the Tutte polynomial of a graph of moderate size", *Lecture Notes in Computer Science*, **1004** (1995) 224-233.

[74] D. R. Shier, Network reliability and algebraic structures, Oxford University Press (1991).

[75] Y. Shiloach, "Edge-disjoint branching in directed multigraphs", *Infor. Process. Lett.*, **8** (1979) 24-27.

[76] E. Shimon, Graph Algorithms, Computer Science Press (1979).

[77] 新森，小出，石井, "エッジ・パッキングによるネットワーク信頼度の下界", 日本応用数理学会論文誌, **5** (1995) 19-31.

[78] R. P. Stanley, "Cohen-Macaulay complexes", in :*Higher Combinatorics* (M. Aigner, editor) Reidel (1977) 51-64.

[79] I. M. Soi and K. K. Aggarwal, "Reliability indices for topological design of computer communication networks", *IEEE Trans. Reliability*, **30** (1981) 438-443.

[80] K. Stuner, A. Satyanarayana and C. Suffel, "The complexity of the residual node connectedness reliability problem", *SIAM J. Comput.*, **20** (1991) 149-155.

[81] R. E. Tarjan, "Finding optimum branchings", *Networks*, **7** (1977) 25-35.

[82] W. T. Tutte, "On the problem of decomposing a graph into n connected factors", *Journal of the London Mathematical Society*, **36** (1961) 221-230.

[83] W. T. Tutte, Connectivity in Graphs, Oxford University Press (1966).

[84] R. M. Van Slyke and H. Frank, "Network reliability analysis: part I", *Networks*, **1** (1972) 279-290.

[85] A. N. Venetsanopoulos and I. Singh, "Topological optimization of communication networks subject to reliability constraints", *Problem of Control and Information Theory*, **15** (1986) 63-78.

[86] D. Vertigan, The computational complexity of Tutte invariants for planar graphs, Mathematical Institute, University of Oxford, England (1990).

[87] J. von Neumann, Probabilistic logics, California Institute of Technology (1952); also published in : "Automata Studies" (C. E. Shannon and J. McCarthy, eds.), Princeton University Press (1956).

著者発表論文

(1) 小出武, 新森修一, 石井博昭, "直並列グラフによるネットワーク信頼度の下界", 日本応用数理学会論文誌, **9** (1999) 15-35.

(2) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "The evaluation on lower bound of all-terminal reliability by arc-packing for general networks", *IEICE Trans.*, **E82-A** (1999) 784-791.

(3) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "A new algorithm for lower bounds of all-terminal reliability", *Mathematica Japonica* (accepted).

(4) 新森修一, 小出武, 石井博昭, "エッジ・パッキングによるネットワーク信頼度の下界", 日本応用数理学会論文誌, **5** (1995) 19-31.

(5) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "Topological optimization of networks considering a reliability constraint", to be submitted to *Disc. Appl. Math.*

関連報告

(1) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "A lower bound of all-terminal reliability by arc-packing", *Proceedings of the 3rd Conference of the Association of Asia-Pacific Operational Research Societies* (1995) 277-284.

(2) 小出武, 新森修一, 石井博昭,"直並列グラフを利用した all-terminal reliability の下界導出法", 京都大学数理解析研究所講究録, **1043** (1998) 128-134.

(3) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "Decomposition and transformations for network reliability calculations", *Proceeding of the 1st Euro-Japanese Workshop on Stochastic Risk Modeling for Finance, Insurance, Production and Reliability* (1998).

(4) Takeshi Koide, Shuichi Shinmori, Hiroaki Ishii, "Topological optimization of networks considering a reliability constraint", *The Proceedings of the 1st Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications* (1999) 199-204.