リンク故障の依存関係を考慮したネットワーク信頼性評価

代表研究者　　川原　純　　奈良先端科学技術大学院大学 先端科学技術研究科 助教

1 あらまし

ネットワーク信頼性評価とは，ネットワークの各リンクが指定された静的な確率で故障する際に，2 ノード間（または多ノード間）が通信可能となる確率を求める問題である．本研究では，リンクの故障が独立ではない場合を考える．ネットワーク信頼性評価では，論理関数をコンパクトに表現するデータ構造である二分決定グラフを用いる手法が有力な手法の 1 つである．二分決定グラフによってノード間の通信可能性を表現するには，二分決定グラフ上に現れるリンクの順を事前に決定する必要がある．本研究では，リンク故障について，任意の依存関係を扱えるネットワーク信頼性評価法を提案する．また，グラフカットと呼ばれる，取り除くことでネットワークが非連結となるノード集合を用いて，リンク間の依存関係を考慮しながらリンクの順を決定し，ネットワーク信頼性を高速に計算する手法を提案する．

2 研究の背景と既存研究

ネットワーク信頼性評価とは，ネットワークの各リンクが指定された静的な確率で故障する際に，2 ノード間（または多ノード間）が通信可能となる確率を求める問題であり，古くから研究が行われている基礎的な問題である[1]．ネットワーク信頼性の厳密計算の計算複雑さは #P 完全と呼ばれるクラスに属しており[1]，数万ノードの大規模ネットワークに対して厳密計算アルゴリズムを設計するのは困難である．

しかしながら，ノード数が数十から数百程度のネットワークに対しては，現代の計算機性能の向上により，厳密計算が可能となっており，災害時の信頼性評価など，その規模のネットワークに対する厳密計算が必要とされることも多い[2]．厳密計算アルゴリズムとして，分離積和法[3]や因数分解に基づく方法[4]などが知られている．それらの中でも最も有力な手法の 1 つとして考えられているのが，二分決定グラフ（Binary decision diagram，以下 BDD）[5]と呼ばれる，論理関数をコンパクトに表現するデータ構造を用いた手法である．各リンクの故障，非故障を 0 と 1 の値をとる 1 つの論理変数として表し，ネットワークの指定されたノード間が通信可能である場合に 1 を，不可能である場合に 0 を出力する論理関数を考え，それを BDD で表現する．本稿では，ネットワークの接続可能性を表す論理関数を表現する BDD を信頼性 BDD と呼ぶ．信頼性 BDD は，ネットワークの接続，非接続の情報をすべて含んでおり，一度信頼性 BDD を構築できれば，単純な動的計画法に基づくアルゴリズムでネットワーク信頼性の確率値を信頼性 BDD から計算できる．信頼性 BDD の構築法は複数提案されている[6,7,8]．ネットワーク信頼性評価に BDD を用いている最近の研究としては，辺素パスに基づく信頼性計算[9]や，ネットワーク設計を信頼性計算に基づいて行う手法の提案[10]などがある．

ネットワーク信頼性評価の典型的な問題では，各リンクが独立に静的な確率で故障する場合を考えているが，この仮定は必ずしも正しくない状況がある．例えば，Software Defined Networking (SDN)[11]などの仮想ネットワークでは，物理リンクの 1 本が切断されると，複数の仮想的なリンクが同時に切断される．また，道路網をネットワークと考えると，建物の火災が発生すると周辺の道路が同時に通行不能になる．そのような状況では，各リンクが独立に故障するとは言えず，信頼性評価が正確に行えない．

本研究では，各リンクの故障が独立ではない場合を考える．既存研究[12]では，リンクの故障が独立ではない場合として，複数のリンクがすべて同時に故障するか，すべて非故障かどちらかの状態を取る場合について，信頼性評価を行うアルゴリズムを提案している．例えば，e_x と e_y （本原稿では可読性を損なわないため，A_x または A_{x} で x が A の下付き添え字であることを表す）が同時に故障する場合，論理関数（e_x ∧ e_y）∨（¬ e_x ∧ ¬e_y）と信頼性論理関数の論理積を取ることで，リンクの依存関係の制約を課すことができる．2 つの論理関数が BDD として与えられているとき，論理積を表す論理関数を BDD として求めるアルゴリズム[13]を用いることで，この計算を効率よく行うことができる．

本研究で考えるリンク故障の依存関係は，複数のリンクについて，それらの故障，非故障の各組合せが発生する確率が与えられている場合を考える．例えば，k 本のリンクに依存関係が存在する場合，それらの故

障，非故障の組合せは 2^k 通り存在するが，2^k 通りの組合せすべてに対して，発生確率が与えられているものとする．これは，複数のリンクがすべて同時に故障または非故障となる場合の一般化となっている．この一般的な設定では，上述した論理積を取る手法を用いることは困難である．本研究ではこの設定において，全パターンを列挙する自明なアルゴリズム以外では初めて BDD を構築して信頼性評価の計算を行う手法を提案する．

本研究では，リンクの故障が独立ではない場合に，BDD の構築時間を高速化するための変数順決定アルゴリズムも提案する．BDD の構築順序は，BDD に現れる変数の順序に大きく依存することが知られている[14]．論文[15]には，ネットワーク上のある点から幅優先探索順（BFS 順）に変数順序を決めることで BDD の節点数が少なくなるという記述があるが，根拠は示されていない．本研究では，グラフカットに基づく変数順決定のアルゴリズムを提案する．グラフカットとは，取り除くことでネットワークが非連結となるノードの集合である．ネットワークのサイズの小さなグラフカットを求めると，グラフカットの一方の連結成分と他方の連結成分の関係が少なくなる．このとき，一方の連結成分を先にすべて探索を行い，次に他方の連結成分の探索を行う．以上の手順を再帰的に繰り返すことにより，記憶すべき情報量が減り，BDD の節点数が小さくなり，結果として信頼性評価の計算時間が短縮される．

本研究では，ノードの故障は考えないが，ノードの故障が発生し，複数のノードとリンクの故障が独立ではない場合も，同様の手法で計算できると考えられる．ノードも故障し得る場合で故障が独立に起こる場合の信頼性評価は論文［15,16,17]で研究されている．

本稿の残りの構成は以下の通りである．3 章で問題の定義や BDD の導入などの準備を行う．4 章でリンクの故障が独立ではない場合のネットワーク信頼性の計算法を述べる．5 章ではリンクの故障が独立ではない場合の変数順決定のアルゴリズムを提案する．6 章で計算機実験による提案アルゴリズムの評価を行い，7 章で結論を述べる．

3 準備

3-1 問題の定義

入力となるネットワークをグラフとして表し，G=(V, E) と表記する．V は頂点集合であり，V = {v_1,...,v_n} とする．E は辺集合であり，E = {e_1,...,e_m} とする．以下では，ネットワークのノードを頂点，リンクを辺と呼ぶ．各辺の故障確率を以下の通りに定義する．本研究で考える設定では，複数の辺の故障組合せそれぞれに対して，その故障組合せが発生する確率が与えられていると考える．E の分割 D_1,...,D_h が与えられているとする．すなわち，$E = D_1 \cup \cdots \cup D_h$, $D_i \cap D_j \neq \varnothing$ $(1 \leq i < j \leq h)$ を満たす．各 D_i を依存関係集合と呼ぶ．ℓ 個の辺 D_i = {e_{g_i(1)},...,e_{g_i(ℓ)}} $(1 \leq g_i(1) < \cdots < g_i(\ell) \leq m)$ について，p_i(x_1,...,x_ℓ)，$x_j \in \{0, 1\}$ を，その故障組合せが発生する確率，すなわち，各 $j = 1,...,\ell$ について，$x_j = 0$ なら e_{g_i(j)} が故障，$x_j = 1$ なら e_{g_i(j)} は非故障を表すとし，p_i(x_1,...,x_ℓ) はその故障組合せが発生する確率であるとする．$T \subset V$ を頂点集合の部分集合とする．

本研究で考える問題は，入力を，ネットワーク G=(V, E)，依存関係集合 (D_1,...,D_h)，故障組合せが発生する確率 (p_1,...,p_h)，頂点集合 T とし，辺が故障組合せ発生確率にしたがって故障するときに，T のすべての頂点が通信可能，すなわち，T の任意の 2 点について，それらを結ぶ非故障の辺のみを使用したパスが存在する確率を求める問題である．

3-2 二分決定グラフ

BDD の定義を述べる．BDD は 2 つの終端と 1 つの根をもつ，有向非巡回グラフである．BDD の詳細は書籍[18,19]などを参照されたい．以下では，BDD の各ノードと有向辺を節点と枝と呼ぶ．BDD の終端以外の各節点は 2 つの外向きの枝をもち，それぞれ 0-枝と 1-枝と呼ぶ．根節点は内向きの枝をもたない．2 つの終端節点はそれぞれラベル 0 とラベル 1 をもち，それぞれ 0-終端，1-終端と呼ぶ．終端以外の各節点は e_1,...,e_m のいずれかのラベルをもつ．e_i と e_j のラベルをもつ 2 つの節点間に e_i から e_j へ向かう枝が存在するとき，$i < j$ が成り立つ．このとき，この BDD の辺の順序（変数の順序）は e_1,...,e_m の順であるという．

BDD と，m 変数論理関数 f(e_1,...,e_m) について，以下が成り立つとき，BDD は f を表現するという．

任意の変数割当 (a_1,...,a_m), a_j ∈ {0, 1} について，根節点から出発し，節点のラベルが e_j であるとき、a_j-枝を選択してその先に進む．この過程を繰り返すと，0-終端または 1-終端に到達する．このとき，f(a_1,...,a_m) の値が終端のラベルと一致する．BDD の変数順序が異なると，同じ論理関数を表現する BDD でも，節点の数が指数倍異なる例が知られている[14]．このため，適切な BDD の変数順序を見つけることが BDD 構築時間の改善に重要となるが，これは一般には難しい問題である．

G=(V, E), T ⊂ V が与えられたとき，T のすべての頂点の通信可能を表す論理関数を f_G(e_1,...,e_m) とする．すなわち，T のすべての頂点が通信可能であるとき，またそのときに限り f_G(e_1,...,e_m) = 1 となる．ここで，各辺 e_j を論理変数とみなし，e_j = 0 のときは辺 e_j は故障，e_j = 1 のときは辺 e_j は非故障とする．f_G(e_1,...,e_m)を表現する BDD を構築する手法は[6] や[7], [8] によって提案されている．本稿では，それらの手法を単に信頼性 BDD 構築アルゴリズムと呼ぶ．

各リンクの故障が独立に起こる場合，一度信頼性 BDD が構築されれば，それを用いて信頼性確率を計算することができる．この場合，各 i = 1,..., m について，D_i = {e_i} と書け，p_i(e_i) は 0 以上 1 以下の定数となる．q_i = p_i(e_i) とする．すなわち，辺 e_i が非故障である確率は q_i であり，辺 e_i が故障する確率は 1 − q_i である．信頼性確率は，根節点から出発し，節点のラベルが e_i であるとき，確率 q_i で 1-枝を選択し，確率 1 − q_i で 0-枝を選択し，選択した先に進む過程を繰り返したとき，1-終端に到達する確率となる．最初は根節点αから出発する．すなわち，根節点には確率 1 で到達する．α の 0-枝と 1-枝の先をそれぞれ α_0, α_1 とすると，α_0 に到達する確率は，α に到達する確率（この場合は 1）に（1 − q_i）を掛けた値であり，α_1 に到達する確率は，α に到達する確率に q_i を掛けた値である．したがって，α_0, α_1 に計算した値を格納する．以上の操作を，根に近い節点から順に行う．

ある節点が複数の親をもつ場合は，それぞれの親からの到達確率を足し合わせて，節点に記憶させる．最終的に 1-終端に格納された確率値が信頼性確率となる．本研究で考える，各リンクの故障が独立ではない場合はこの方法では計算できない．

4 リンクの故障が独立ではない場合の信頼性確率計算

リンクの故障が独立ではない場合を考える．例えば，D_i = {e_{g(1)},...,e_{g(ℓ)}} について，変数 e_{g(1)} の 0/1 選択に応じて，e_{g(2)} の 0/1 選択の確率が変化する．従って，ラベル e_{g(2)} をもつ節点 α について，到達確率値を計算する際に，その節点に到達するまでに e_{g(1)} への割当が 0 か 1 かを分けて記憶する必要がある．しかしながら，BDD の構築の際，0-枝と 1-枝が指す節点が同じである 2 つの節点は併合されるので，元の BDD では，節点について e_{g(1)} への割当が 0 か 1 かどちらを表しているかを区別できない．

以上の観察を踏まえて，以下のアルゴリズムを提案する．このアルゴリズムはサブセッティング法[20]または BDD 制約探索[21]として知られている技法を用いている．このアルゴリズムはリンクの故障が独立な場合の信頼性 BDD を入力として受け取り，各 D_i の変数の選択の情報を節点に記憶した BDD を出力とする．出力 BDD を用いて，リンクの故障が独立ではない場合の信頼性確率計算を行う．

説明の簡単のため，D_1 = {e_{g(1)},e_{g(2)},e_{g(3)}}, 1 ≤ g(1) < g(2)< g(3) ≤ m という依存関係だけが存在する場合を考える．出力 BDD は根節点から順に幅優先的に構築を行う．最初に根節点αを作成する．根節点には，入力 BDD の根節点へのポインタを記憶する．次に，α の 0-枝の先と 1-枝の先を作成する．j=0,1 について，α の j-枝の先には，α に格納されている入力 BDD 節点（ポインタ）の j-枝の先の節点のポインタを格納する．以下同様に，節点の 0-枝と 1-枝の先として，節点に格納されている入力 BDD 節点のそれぞれ 0-枝と 1-枝の先を格納する．この過程で，a = 0,1 について，入力 BDD 節点のポインタが a-終端に到達した場合，出力 BDD の節点も a-終端とする．

この過程で，ラベル e_{g(1)}の節点α’に到達したとき，各 j=0,1 について，α’の j-枝の先に，e_{g(1)} = j を割り当てたことを記憶させる．この割当情報は，その節点の 0-枝や 1-枝の先の子にも複製して記憶させる．同様に，ラベル e_{g(2)} の節点 α’’ に到達したとき，各 j’ = 0,1 について，α’’ の j’-枝の先に，e_{g(2)} = j’ を割り当てたことを（e_{g(1)}の割当に加えて）記憶させる．ラベル e_{g(3)} の節点 α’’’ に到達したときは，α’’’ の 0-枝と 1-枝の先の節点には，e_{g(1)} と e_{g(2)} の割当情報は記憶させない（情報を消去する）．これは，後で説明する確率の計算時に，α’’’ の次の節点まで到達した時点で，D_1 に関する確率の計算が行われ，e_{g(1)} と e_{g(2)} の割当の情報が不要になるためである．

出力 BDD に関して，節点に記憶している情報（入力 BDD の節点ポインタと変数の割当）がすべて等しいときは，節点は等価であるとみなし，節点の併合を行う．逆に，割り当てが 1 つでも異なる場合は，節点の併合は行わない．節点の併合により，BDD の構築が効率化される．出力 BDD の節点は入力 BDD の節点へのポインタをもち，a=0,1 について，入力 BDD の節点へのポインタが a-終端を指すとき，出力 BDD の節点も a-終端になることから，出力 BDD は入力 BDD と同じ論理関数を表現している．節点の併合により，出力 BDD が正しく構築されることの証明は省略する．

出力 BDD から，辺の故障が独立ではない場合の信頼性確率の値の計算を以下の手順で行う．辺故障が独立な場合と同様，根節点から出発する．その節点のラベルが e_j で，e_j が他の辺と独立に故障するときは，辺故障が独立な場合と同様に，節点に格納された確率値に (1 − q_j)，q_j をそれぞれ掛けた値を 0-枝，1-枝の先の節点にそれぞれ足し合わせる．節点のラベルが e_{g(1)} または e_{g(2)} のときは，その場では何も計算をせず，節点に格納された確率値をそのまま 0-枝，1-枝の先の節点にそれぞれ足し合わせる．ラベルが e_{g(3)} である節点に到達したとき,節点には e_{g(1)} と e_{g(2)} の割当が格納されている.e_{g(1)} = a_1 と e_{g(2)} = a_2 とすると，各 a_3 = 0, 1 について，a_3-枝の先の節点は，e_{g(1)} = a_1, e_{g(2)} = a_2, e_{g(3)} = a_3 が実現されたことを意味する．その実現確率は，問題の定義から p_1(a_1,a_2,a_3) である．したがって，節点に格納された確率値に p_1(a_1,a_2,a_3) を掛けた値を，a_3-枝の先の節点に足し合わせる．以上の過程を繰り返す．最終的に，1-終端に格納された確率値が求めたい値である．

辺故障の依存関係 D_* が複数存在する場合も同様の手順で計算できる．その場合は D_* ごとに，D_* に関する変数割当を記憶する必要がある．この場合も，任意の D_j について，変数への割り当てが 1 つでも異なる場合は節点の併合は行わない．D_i に含まれる変数への割り当てがすべて終了した後に，D_j の変数が出現する場合は，D_i と D_j の変数割当は同時に記憶する必要はないため（D_i に含まれる変数への割り当てがすべて終了した時点で，D_i に関する変数割当の情報は削除される），節点に格納する情報はそれほど多くならないが，そうではない場合，すなわち，D_i の最後の変数より前に，D_j の変数がある場合は，各 D_i, D_j の変数割当を記憶しなければならず，節点に格納すべき情報が大きくなる．ある節点の処理時に記憶しておくべき D_* の変数の情報が多いほど，指数的に記憶すべき情報が増え，その分，節点の併合が行われなくなり，構築される BDD の節点の数が多くなる．最悪の場合にはほとんど 2 分木に近い状態になると考えられる．逆に言えば，同時に記憶しておかなければならない変数割当の数がなるべく少なくなるように，BDD の変数の順を決めることで，構築される BDD の節点数を小さくすることができる．

5 グラフカットを用いた変数順序決定法

5-1 リンクの故障が独立な場合の変数順序決定法

説明の簡単のため、まずはリンクの故障が独立な場合の変数順序決定法について説明する．BDD を構築する際，BDD が表す論理関数の変数の順序をあらかじめ定める必要がある．BDD の根から終端までの任意のパスについて，パスに現れるラベルの変数の順は，定められた変数順に従わなければならない．変数の順序に応じて，BDD の構築時間や節点の数は大幅に変化する．一般に，論理関数 f が与えられたとき，f を表現する BDD の中で，節点の数が最小となるように変数順を求める問題を考えると，この問題の計算複雑さは NP 完全である[22].ネットワーク信頼性 BDD の場合,変数順序はネットワークを用いて決定する手法が用いられる．変数は辺であるため，辺の順を決定する．最もよく使用されている変数順は，ネットワーク上のある頂点からの幅優先探索順 (BFS 順) である．深さ優先順 (DFS 順) よりは BFS 順の方がよいと考えられている．また，NDS と呼ばれる変数順決定法が提案されている[23]．NDS では初めに頂点の順を決定する．開始頂点を 1 つ選択する．次の頂点として，隣接している，既に選択された頂点の個数が最も多くなるように頂点を選択する．個数が同じ場合，隣接している，既に選択された頂点のインデックス（何番目に選択されたか）の和が最小の頂点を選択する．辺の順序は，両端の頂点がともに選択された瞬間に，辺も選択されるとしたとき，辺が選択された順となる．

信頼性 BDD の節点数について，理論的な評価が行われている．論文[6]では，消去波面を用いた解析を行っている．(消去波面は書籍[14]ではフロンティア，論文[8]では境界集合と呼ばれている.) ラベル e_i をもつ BDD の節点は，変数 e_1,...,e_{i-1} までの割当が行われたことを意味している．このとき，ネットワーク G = (V, E) において，辺 e_1,...,e_{i-1} のうち少なくとも 1 本と，辺 e_{i},...,e_{m} のうち少なくとも 1 本が接続している頂点の集合を消去波面と呼ぶ．消去波面の頂点の個数を F_i としたとき，ラベル e_i

をもつ BDD の頂点の個数は F_{i-1} 番目のベル数以下であることが証明されている．ここで，a 番目のベル数とは，(1/e) $\Sigma_{k=0}^{\infty} k^a /k!$ で定義される値である．このことから，i=1,...,m と変化すると消去波面の大きさ F_i が変化するが，F_i が大きくなると，節点の個数が指数的に大きくなり得るといえる．したがって，F_i の i = 1,...,m と変化させたときの最大値をなるべく小さくするように，変数順序を決めることが望ましい．

以上を踏まえて，変数順序を決めるアルゴリズムを提案する．まず初めに，辺の依存関係を考慮せずに変数順序を決める方法を提案する．最初に，ネットワーク上の 2 頂点 s,t について，s と t の最短経路（各辺の長さは 1 とする）が最長となる s,t を求める（この長さはグラフ理論の用語で直径と呼ばれる）．s から開始し，t に向かう方向に順序を付けることになる．次に，グラフの s-t 頂点カットのうち，大きさが最小となるものを求める．s-t 頂点カットとは，頂点集合であり，その各頂点を削除することで，s と t が非連結になるような頂点集合である．s-t 頂点カットの大きさは，構成する頂点の個数で定義される．最小 s-t 頂点カットとは，大きさが最も小さくなる s-t 頂点カットである．最小 s-t 頂点カットは例えば [24] の方法で計算できる．s-t 頂点カットによってネットワークが 2 つ以上に分断される．このとき，以下の通りに変数順を決める．最初は s を含む連結成分の各辺とする．次は頂点カット内の頂点同士を結ぶ各辺とする．その次は s も t も含まない連結成分の各辺とする．最後に，t を含む連結成分の各辺とする．

各連結成分の変数順は再帰的に決定する．s を含む連結成分の場合，頂点カット内のすべての頂点を縮約して 1 つの頂点とし，その頂点を新たな t とする．s は変更しない．このように s と t を決めることで，連結成分内の変数順を再帰的に決定することができる．t を含む連結成分の場合は，頂点カット内のすべての頂点を縮約して 1 つの頂点とし，その頂点を新たな s とする．t は変更しない．s も t も含まない連結成分の場合は，頂点カット内のすべての頂点を縮約して 1 つの頂点とし，その頂点を新たな s とする．t は，s から最短距離が最も長い頂点とする．連結成分の辺の個数が決められた値を下回ったときは（例えば 10 以下），NDS によって変数順序を決める．

5-2 リンクの故障が独立ではない場合の変数順序決定法

リンクの故障が独立ではない場合に，グラフの頂点カットを用いた変数順序の決定アルゴリズムを提案する．s-t 頂点カットによって，複数の連結成分に分けるとき，依存関係集合 D_i の要素の一部が s を含む連結成分に，残りの要素が t を含む連結成分に含まれるとき，s-t 頂点カットは依存関係集合 D_i を分断すると言うことにする．最小の s-t 頂点カットは複数存在し得るが，その中で，依存関係集合 D_1,...,D_h のうち分断されるものの個数が最小になるように，s-t 頂点カットを選ぶ．そのような頂点カットが複数存在する場合は任意に 1 つ選ぶ．

連結成分の辺の個数が決められた値を下回ったとき，NDS によって変数順序を決めるが，このときも依存関係集合 D_1,...,D_h を考慮して変数順序を決定する．各 i = 1,...,h について，D_i の要素が 1 つでも選択されたとき，残りの要素もなるべく早く選択される方がよい．このため，もし，e ∈ D_i について，e の少なくとも一方の端点が，NDS によって既に選択されていた場合，e のもう一方の端点を次に選択する．e の少なくとも一方の端点が，既に選択されている頂点に隣接している場合，その端点を次に選択し，e のもう一方の端点をその次に選択する．候補が複数存在する場合は任意に 1 つを選択する．D_* は分断されにくいように頂点カットが選ばれているため，D_i の要素が 1 つでも選択されたとき，残りの要素が早く選択されやすくなる．

6 計算機実験

提案アルゴリズムの性能を評価するため，計算機実験を行う．BDD 構築は C++ 言語を用いて実装し，BDD 構築のための TdZdd ライブラリ[20]を用いる．コンパイラは g++ で，-O3 最適化オプションを使用している．変数順序決定は Python 言語を用い，s-t 頂点カットの計算やネットワーク（グラフ）の操作のために NetworkX ライブラリ[26]を用いる．計算機環境は Ryzen Threadripper 1920X，メモリ 64GB，OS は Windows 10 / cygwin である．入力ネットワークとして，The Internet Topology Zoo [25] の web サイトに公開されている全ネットワークを辺の数で昇順にソートを行い，その中から辺の数 50 本以上，100 本以下の奇数番目のネットワークと，辺の数 100 本以上，200 本以下の全ネットワークを用いる．ネットワークの頂点数と辺数を表 1 に示す．

表 1 ネットワークの頂点数と辺数，B_1，B_3 の構築時間，確率計算の時間．単位は秒．

ネットワーク	頂点数	辺数	Time 1	Time 3	Time 4
Intranetwork	39	51	<0.01	<0.01	<0.01
Ntelos	47	58	<0.01	<0.01	<0.01
Syringa	74	74	<0.01	<0.01	<0.01
Missouri	67	83	0.05	0.02	0.17
VltWavenet2008	88	92	<0.01	<0.01	<0.01
Intellifiber	73	95	0.05	<0.01	0.03
Oteglobe	83	99	0.03	<0.01	0.06
Interoute	110	146	0.17	0.12	1.33
Ion	125	146	0.60	0.11	1.79
DialtelecomCz	138	151	0.07	0.01	0.14
Deltacom	113	161	15.99	11.28	266.12
TataNld	145	186	0.62	1.30	25.30
UsCarrier	158	189	94.09	0.08	1.42

表 1 は 3 章で提案した，辺の故障が独立ではない場合の信頼性 BDD の構築時間を示している．ネットワークの変数順は BFS により決定している．辺の依存関係は，E = D_1 ∪ … ∪ D_h ∪ D'_1 ∪ … ∪ D'_{h'} と表されるとする．ここで，各 i = 1,...,h について，D_i の大きさは 3 であるとし，i' = 1,...,h' について，D_{i'} の大きさは 1 であるとし，h = 5 とする．すなわち，3 本のリンクの依存関係が 5 組あるとする．確率値は $p_i(e_{g_i(1)}, e_{g_i(2)}, e_{g_i(3)}) = (1 + \sum_{j=0}^{2} 2^{2-j} e_{g_i(j)}) / \sum_{j=0}^{7}(j+1)$ とする（どのような値を用いても計算時間には影響は無いと考えられる）．ランダムに依存関係の生成を 10 回行う．表 1 の計算時間は，10 回の平均を取っている．実験では，まず辺の故障が独立の場合のネットワーク信頼性 BDD を，[6]のアルゴリズムを用いて構築する．この BDD を B_1 とする．次に B_1 の既約化を行う．既約化とは，等価な節点を併合するなど，BDD が表す論理関数を変えずに節点数を最小化する操作である[14]．この BDD を B_2 とする．その次に，B_2 を入力として，3 章の提案手法により，辺の故障に依存関係がある場合の BDD を構築する．この BDD を B_3 とする．最後に，B_3 から，辺の故障に依存関係がある場合の BDD から確率の値を計算する．表の 「Time 1」 は B_1 の構築時間を表す．B_2 の構築時間は全体の時間と比べて無視できるほど小さいので，表には掲載していない．「Time 3」 は B_3 の構築時間を表す．「Time 4」 は最後の確率計算の時間を表す．「<0.01」 は実行時間が 0.01 秒以下であることを示す．表 2 は，B_1，B_2，B_3 の節点数（10 回の試行の平均）を表す．

表 2 B_1，B_2，B_3 の BDD 節点数．

ネットワーク	B_1 節点数	B_2 節点数	B_3 節点数
Intranetwork	1735	131	3156
Ntelos	11576	987	9291
Syringa	1128	99	918
Missouri	222831	5583	117622
VltWavenet2008	32764	444	4501
Intellifiber	236772	3239	27003
Oteglobe	138112	1644	50618
Interoute	772124	19070	740845
Ion	2373650	38074	869148
DialtelecomCz	347451	5325	113653
Deltacom	43508618	647953	45063533
TataNld	2332101	79928	5992231
UsCarrier	222018203	33775	812476

いずれのネットワークの場合も,「Time 4」 は 「Time 3」 の数十倍の時間がかかっている．BDD の構築時間と節点数は比例する傾向にある.BDD の既約化により大幅に節点数が減る場合があり,その場合は「Time 1」 が最も大きい (UsCarrier)．それ以外の場合 (Deltacom や TataNld) は,「Time 4」 が最も大きい．

B_3 の節点数が B_2 の節点数の数十倍になるのは，BDD B_3 に，依存関係の割当の全情報が含まれているためであると考えられる．依存関係集合 D_* の大きさが大きくなると，記憶すべき情報が指数的に増大すると考えられる．

次に，依存関係の組の個数 (h の値) が増大するときの，計算時間と BDD 節点数の分析を行う．ネットワークは Interoute とし，依存関係集合の大きさは 3 で固定し，個数 h を 2 から増大させる．実験結果を表 3 に示す．h が 1 増えると，計算時間や節点数が指数的に増大することが確認できる．

表 3　h を変化させた際の，B_3 の構築時間，確率計算時間，B_3 の節点数．時間の単位は秒．

h	B_3 構築時間	確率計算時間	B_3 節点数
2	<0.01	0.06	65042
3	0.02	0.16	150499
4	0.04	0.49	346748
5	0.11	1.19	740845
6	0.26	3.30	1518133
7	0.72	9.50	3266812
8	1.16	16.27	5018118
9	4.27	68.57	15411660
10	6.87	112.95	23810410
11	17.89	329.69	56886269

最後に，5 章で提案した，頂点 s-t カットを用いた変数順序決定アルゴリズムの評価を行う．本稿では辺の故障が独立な場合の評価のみを行う．表 4 は，BFS と提案手法による BDD 構築時間と BDD 節点数の比較である．提案手法は，BFS に比べて，ほとんどのネットワークに対して，大幅に計算時間が改善されていることが確認できる．

表 4　BFS と提案手法の変数順序決定アルゴリズムの比較．時間の単位は秒．

ネットワーク	BFS 時間	提案手法時間	BFS 節点数	提案手法節点数
Intranetwork	0.05	0.05	1735	633
Ntelos	0.05	0.05	11576	2634
Syringa	0.05	0.05	1128	349
Missouri	0.11	0.05	222831	6260
VltWavenet2008	0.06	0.07	32764	24484
Intellifiber	0.11	0.08	236772	84203
Oteglobe	0.08	0.05	138112	15269
Interoute	0.25	0.06	772124	42778
Ion	0.81	0.06	2373650	24037
DialtelecomCz	0.13	0.05	347451	11040
Deltacom	19.60	1.10	43508618	1171756
TataNld	0.84	0.07	2332101	81456
UsCarrier	113.68	0.05	222018203	12211

7 結論

本稿では，辺の故障が独立に発生するのではなく，依存関係がある場合の BDD を用いた信頼性評価についてアルゴリズムを提案した．また，依存関係を考慮した変数順序の決定アルゴリズムを提案した．計算機実

験では，依存関係はランダムに設定したが，実際の仮想ネットワークや道路網では，何らかの偏りをもった依存関係が存在すると考えられ，そのような場合について，実データを用いた評価が必要であり，今後の課題である．

【参考文献】

[1] L. G. Valiant, "The complexity of enumeration and reliability problems," SIAM Journal on Computing vol. 8, no. 3, pp. 410-421, 1979.

[2] H. Saito, R. Kawahara, T. Fukumoto, "Proposal of disaster avoidance control," in Proc. of Telecommunications Network Strategy and Planning Symposium (Networks), pp. 1-6, 2014.

[3] J. M. Wilson, "An improved minimizing algorithm for sum of disjoint products (reliability theory)," IEEE Transactions on Reliability, vol. 39, no. 1, pp. 42-45, 1990.

[4] A. Satyanarayana, M. K. Chang, "Network reliability and the factoring theorem," Networks, vol. 13, no. 1, pp. 107-120, 1983.

[5] S. B. Akers, "Binary Decision Diagrams," IEEE Transactions on Computers, vol. 100, no. 6, pp. 509-516, 1978.

[6] K. Sekine, H. Imai, S. Tani, "Computing the Tutte polynomial of a graph of moderate size," in Proc. of the 6th International Symposium on Algorithms and Computation (ISAAC), pp. 224-233, 1995.

[7] S.-Y. Kuo, S.-K. Lu, F.-M. Yeh, "Determining terminal-pair reliability based on edge expansion diagrams using OBDD," IEEE Transactions on Reliability, vol. 48, no. 3, pp. 234-246, 1999.

[8] G. Hardy, C. Lucet, N. Limnios, "Computing all-terminal reliability of stochastic networks with binary decision diagrams," in Proc. of the 11th International Symposium on Applied Stochastic Models and Data Analysis, pp. 1468-1473, 2005.

[9] T. Inoue, "Reliability Analysis for Disjoint Paths," IEEE Transactions on Reliability, 2018.

[10] M. Nishino, T. Inoue, N. Yaasuda, S. Minato, M. Nagata, "Optimizing Network Reliability via Best-First Search over Decision Diagrams," Proc. of 37th Annual IEEE International Conference on Computer Communications (INFOCOM), pp. 1817-1825, 2018.

[11] K. Benzekki, A. E. Fergougui, A. E. Elalaoui, "Software-defined networking (SDN): a survey," Security Communication Networks, vol. 9, no. 18, pp. 5803-5833, 2016.

[12] 吉田 拓弥, 川原 純, 井上 武, 笠原 正治, "リンクの故障に依存関係がある場合のネットワーク信頼性評価," 研究報告アルゴリズム, vol. 2017-AL-165, no. 16, pp. 1-7, 2017.

[13] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," IEEE Transactions on Computers, vol. 100, no. 8, pp. 677-691, 1986.

[14] D. E. Knuth, The Art of Computer Programming, Volume 4, Fascicle 1, Addison-Wesley, 2008.

[15] S.-Y. Kuo, F.-M. Yeh, H.-Y. Lin, "Efficient and exact reliability evaluation for networks with imperfect vertices," in IEEE Transactions on Reliability, vol. 56, no. 2, pp. 288-300, 2007.

[16] 園田 晃己, 川原 純, 井上 武, 笠原 正治, 明石 修, 川原 亮一, 斎藤 洋 "フロンティア法によるノードの故障も考慮したネットワーク信頼性評価手法の提案," 電子情報通信学会ネットワークシステム研究会, 信学技報, vol. 115, no. 483, pp. 261-266, 2016.

[17] J. Kawahara, K. Sonoda, T. Inoue, S. Kasahara, "Efficient Construction of Binary Decision Diagrams for Network Reliability with Imperfect Vertices," Reliability Engineering & System Safety, vol. 188, pp. 142-154, 2019.

[18] 藤重 悟 編, "離散構造とアルゴリズムⅤ," 近代科学社, 1998.

[19] 巳波 弘佳, 井上 武, "情報ネットワークの数理と最適化," コロナ社, 2015.

[20] H. Iwashita, S. Minato, "Efficient top-down ZDD construction techniques using recursive specifications," Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-13-69, 2013.

[21] M. Nishino, N. Yasuda, S. Minato, M. Nagata, ″BDD-constrained Search: A Unified Approach to Constrained Shortest Path Problems,″ In Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 1219-1225, 2015.
[22] B. Bollig, I. Wegener, ″Improving the variable ordering of OBDDs is NP-complete,″ IEEETrans. Computers, vol. 45, no. 9, pp. 993-1002, 1996.
[23] Y. Dutuit, A. Rauzy, J. P. Signoret, ″Computing Network Reliability with Réséda and Aralia,″ Proceedings of the European Safety and Reliability Association Conference, pp. 1947-1952, 1996.
[24] N. Garg, V. V. Vazirani, M. Yannakakis, ″Multiway cuts in directed and node weighted graphs,″ In Proc. of the 21st Int. Colloquium on Automata, Languages and Programming, pp. 487-498, 1994.
[25] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, ″The Internet topology zoo,″ IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765-1775, 2011.
[26] A. A. Hagberg, D. A. Schult, P. J. Swart, ″Exploring network structure, dynamics, and function using NetworkX,″ in Proceedings of the 7th Python in Science Conference (SciPy2008), pp. 11-15, 2008.

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月
リンク故障について任意の依存関係を扱えるネットワーク信頼性評価法	電子情報通信学会 信学技報	2019 年 3 月