修士論文

カスケード故障を抑制するための
迂回ルーティングにおける順序制御の検討

1710200　見雪 雄哉

主指導教員　林 幸雄
審査委員主査　林 幸雄
審査委員　上原 隆平
小谷 一孔
吉高 淳夫

北陸先端科学技術大学院大学
先端科学技術研究科
(情報科学)

平成 31 年 2 月

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究背景

現実に存在する多くのシステムは, 点を「ノード」, 繋がりを「リンク」に置き換えるとネットワークとして表せる. 例えば, 人間関係では人を「ノード」, 人々との繋がりを「リンク」と置き換えることができる. このようにあらゆるシステムをネットワークとして表現すると, 通信網と電力網といったように全く違うネットワークでも驚くほど共通の性質が存在することが 1990 年代から次々と発見された [1]. そのスケールフリー (SF:Scale-Free) ネットワークが発見されると同時に, 多くの結合を持つハブノードを除去すると連結成分が極端に分離されることが判明した. 意図的な攻撃はまさしく次数の高いノードを選ぶため, 攻撃に対しては極端に脆弱性であることが深刻な問題となった.

近年, このような現実の多くに共通する SF ネットワークにおける脆弱性を打破した玉葱状ネットワークが提案されている [2]. その中でも文献 [3] には, 逐次成長の中でループを強化することで, 正の次数相関を持つ玉葱状ネットワークを構築できる新たな方法が提案されている.

しかしながら, カスケード故障と呼ばれる連鎖的な機能不全が, ネットワーク全体へ拡散する現象 [4] が問題となっている.

身近な例では, 電力崩壊や通信障害, 交通の大渋滞などが挙げられる. このような過負荷が連鎖するカスケード故障の防止策として, いけにえ法 [5] によるものが提案されているが, 負荷に注目した迂回ルーティングは更に有効であることが示されている [6].

但し, 従来手法 [6] では送信元 s と受信先 t を一様ランダム順に選択しており, s, t の選び方の順序に検討の余地がある.

本論文では, s, t の選び方の順序にコストに基づいた優先順位を付けることを提案し, シミュレーションによりその効果を調査する.

1.2 研究目的

送信元 s と受信先 t を一様ランダム順に選択していた従来手法 [6] に対して, 負荷の充足率による重みを考慮した「コスト」に基づいて順序を付けることで, カスケード故障の抑制効果を改善することを目指す. また, 従来手法 [6] では困難であった（後述する）耐久性パラメータ α の小さな部分でもカスケード故障を抑制することができたことをシミュレーションによって示す.

対象としたネットワークは現実のネットワークを模擬したスケールフリーネットワーク（BA モデル）と, このネットワークの脆弱な部分を克服した玉葱状ネットワークとする.

1.3 本論の構成

本論文の主な構成を以下に示す.

第 2 章

本論文で扱うネットワークの種類, カスケード故障, 負荷の定義, カスケード故障の抑制効果を測るための指標, カスケード故障の抑制に対する従来の対策について述べる.

第 3 章

提案手法について, 特に s, t の選び方の順序に負荷率に基づいた優先順位を付ける方法について説明する.

第 4, 5 章

提案手法をシミュレーションを用いて調査する. 第 4 章では初期故障ノードが 1 個の場合を, 第 5 章では大規模障害を想定した複数初期故障ノードのカスケード故障に対してカスケード故障の抑制効果を調査する.

第 6 章

本論文の結果を整理する.

第2章　従来研究

本章では, 2.1 節にて本論文で扱うネットワークの種類を, 2.2 節にてカスケード故障と本論文で扱う負荷の定義を, 2.3 節にてカスケード故障の抑制効果を測るための指標を, 2.4 節にてカスケード故障の抑制に対する従来の対策について述べる.

2.1　ネットワークの種類

本節では, SF ネットワーク（Barabási-Albert モデル）と玉葱状ネットワークについて説明する.

2.1.1　SF ネットワーク（Barabási-Albert モデル）

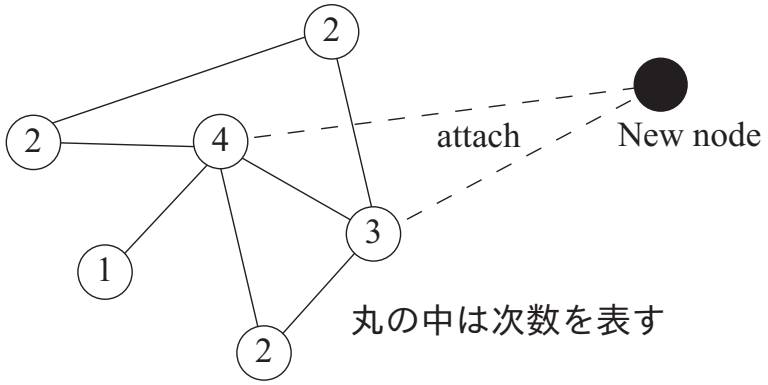


図 2.1: BA モデルにおけるネットワークの生成 ($m = 2$)

現実に存在する通信網や電力網などは, 端末や需要点などをノードとし, それらの繋がりをリンクと見なせばネットワークとして表現できる. ネットワークとして表現することで, 特性の違うネットワーク間でも驚くほど共通する構造的性質がこれまでに数多く発見された [1]. これは, SF 構造と呼ばれ, べき乗則に従う結合分布 $P(k) \sim k^{-\gamma}$ を持つ. スケールフリーネットワークの代表的な生成モデルとして, BA（Barabási-Albert）モデル [7] が挙げられる. BA モデルの特徴としては, 自分にとって有利な接続相手を利己的に選択する優先的選択がある.

BA モデルは以下の手順に従ってスケールフリーネットワークを生成する [7].

Step 1 N_0 個のノードが連結した初期構成を用意する. ここで, $N_0 \geq m$ でなければいけないため, 通常は完全グラフを用いることが多い.

Step 2 $t = 1, 2, 3, ...$ の毎時刻に新たなノードを 1 つ追加し, その新ノードから m 本のリンクを既存ネットワークに対して接続する. その際, ノードはその次数 k_i に比例した確率 $\frac{k_i}{\sum_j k_j}$ で選択される. また, 多重リンク（同じノードを新ノードのリンク先として複数回選択すること）は禁止して再試行する.

Step 3 成長させたいノード数 N になるまで Step 2 を繰り返す.

スケールフリーネットワークは, ランダムな故障に対しては耐性があるが, ハブを狙った意図的な攻撃には極端に脆弱であることが分かっている [7].

2.1.2 玉葱状ネットワーク

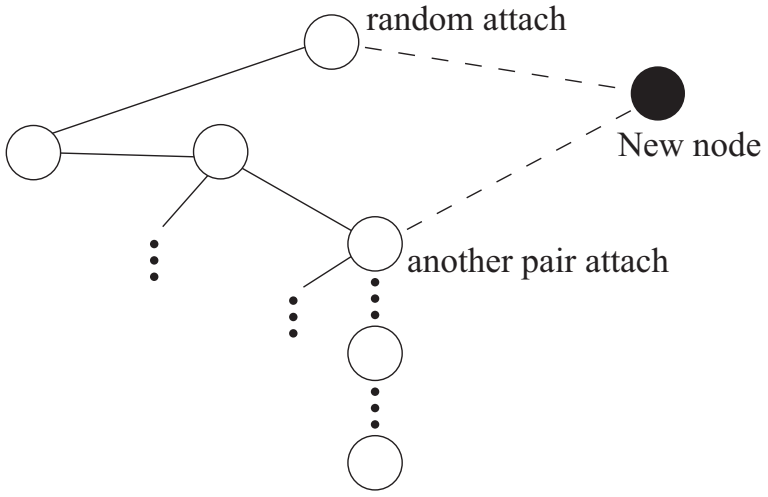


図 2.2: 玉葱状ネットワークの生成 $(m = 2, \mu = 3)$

上記の脆弱性を解決するべく, 逐次成長の中で仲介によりループを強化することで, 正の次数相関を持つ玉葱状ネットワークを構築できる新たな方法が提案されている [2][3]. 玉葱状ネットワークは, 現状において最適な攻撃耐性を持つ. さらに文献 [6] では, 単一及び同時複数の攻撃から引き起こされる過負荷故障の伝播を抑える為に, フロー制御に対する経路方策を通常の最短経路基準から負荷基準に変更することでバイパス的に過負荷ノードを回避できることが明らかにされた. 特に, 玉葱状ネットワークはレジリエンス（しなやかな復活力）における適応力を獲得でき, その抑制効果は多くの現実のシステムに存在するスケールフリーネットワークの場合より強いことが示されている.

玉葱状ネットワークは以下の手順に従って生成される.

Step 1 N_0 個のノードが連結した初期構成を用意する. ここで, $N_0 \geq m$ でなければいけないため, 通常は完全グラフを用いることが多い.

Step 2 $t = 1, 2, 3, ...$ の毎時刻に新たなノードを 1 つ追加し, その新ノードから偶数 m 本をリンクする. ペアの片方を一様ランダムに接続, もう 1 本をランダム接続したノードから仲介数として μ ホップ離れた中で次数最小のノードを選んで接続する. この時, μ ホップ先が存在しない場合, 最もホップ数の遠い（大きい）中で次数最小のノードを選んで接続する. また, 次数最小ノードが複数存在する場合は, ランダムに 1 つ選択し, 多重リンクは禁止する.

Step 3 Step 2 を $m/2$ 回繰り返す.

Step 4 成長させたいノード数になるまで Step 2, Step 3 を繰り返す.

2.2　カスケード故障と負荷の定義

本節では, カスケード故障の説明と, 負荷の定義について説明する.

2.2.1　カスケード故障

カスケード故障 [4] とは, あるノードが故障を起こし, その影響が周辺へ拡散することで連鎖的に引き起こされる現象である. 具体的には図 2.3 のように, 初期故障ノードが発生すると周辺ノードは①:負荷を流し続けるために他の経路へ迂回しようとするが, ②:その迂回先のノードが過負荷になり③:機能不全を引き起こすことでまた別の経路へ迂回しようとすることを繰り返す.

カスケード故障の代表的な例として, 2003 年 8 月 14 日に北米北東部で引き起こされた停電事故が挙げられる [8][9]. また, 2018 年 9 月 6 日に発生した北海道胆振東部地震による北海道全域の大規模停電（ブラックアウト）も一種のカスケード故障である [10].

また電力網だけに限らず, 通信網や航空網なども図 2.3 のサイクルが揃えばカスケード故障は発生する.

図 2.3: カスケード故障

カスケード故障は, 次のプロセスにより引き起こされる.

Step 1 カスケード故障を引き起こすトリガーノードが $t = 0$ で発生したとする. トリガーノード発生後, トリガーノードとそれに関連するリンクは除去される.

Step 2 次の時刻 $t \leftarrow t + 1$ において, 攻撃の引き金または繰り返し続く過負荷ノード故障で経路を変更することにより, 影響を受けたノードの負荷 $L_i(t)$ が更新される. いくつかのノードが多くの負荷を受けてそれ自身の容量を超えると, その過負荷になったノードは機能不全として取り除かれる.

Step 3 過負荷故障が伝播されなくなるまで, Step 2 を繰り返す. すなわち, 更新された負荷が残りのすべての N' 個のノードについて $L_k(T) \leq C_k$ を満たすときに時刻 T で停止する.

ここで, 3.2.3 節にて後述する送信元 s と受信先 t に対する経路を探索した時刻 τ におけるノード k の負荷 $L(\tau)$ は, s,t 以外のノード k を通過する割合を辺の重みとして,

$$L_k(\tau) \leftarrow L_k(\tau - 1) + \frac{\sigma_{st}(k)}{\sigma st} \tag{2.1}$$

と定義する. また, 上記の Step 2 において, ノード k を通過する経路が 1 本増加する度に $+1$ する.

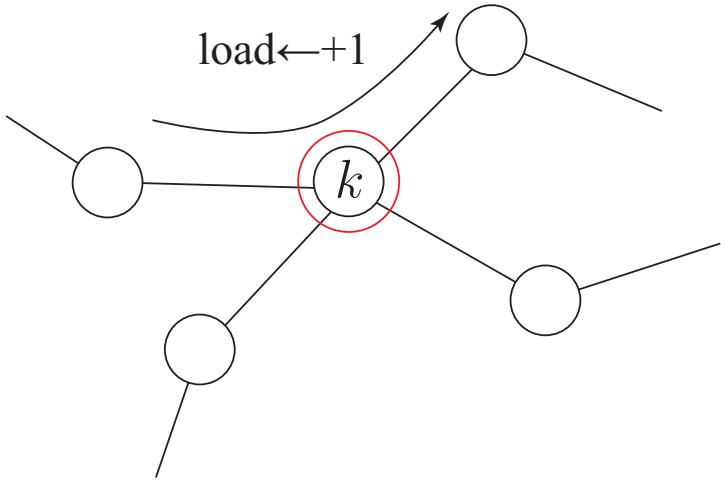


図 2.4: 負荷 $L(\tau)$ 増加

全ての s,t の組合せについての和を考えた式 (2.1) 右辺第 2 項は, 媒介中心性 [11] と呼ばれる. この指標は, ホップ数で測った最短経路がノードやリンクを経由する “頻度” を表す. 具体的には, 送信元ノード s と受信先ノード t を結ぶ σ_{st} 本の経路がノード v を経由するホップ数で測った最短経路の本数を $\sigma_{st}(v)$ とすると, 媒介中心性は,

$$B(v) \stackrel{\text{def}}{=} \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.2}$$

で定義される.

また, ノード k の許容負荷 C_k は, 初期負荷 $L_k(0)$ を用いて,

$$C_k \stackrel{\text{def}}{=} (1+\alpha)L_k(0) \tag{2.3}$$

とする [4]. ここで, 定数 $\alpha \geq 0$ は耐久性パラメータと呼ばれ, C_k にどれだけ余裕を与えるかを決める.

2.3　カスケード故障の抑制効果を測る指標

2.2.1 節で述べたカスケード故障をどれだけ抑制することができたかを測る指標として, 最大連結成分比 G とネットワークの通信効率 E を用いる. 最大連結成分比 G によってどれだけネットワークがバラバラになったかを, ネットワークの通信効率 E によって経路がどれだけ長くなったかを測る.

2.3.1　最大連結成分, 最大連結成分比 G

ネットワーク内で, 複数ノードがリンクで接続された互いに通信が可能な連結成分が複数存在するとき, それらの中で最も多くのノードを含むものを最大連結成分と呼ぶ. 初期時刻 0 における最大連結成分のサイズ N に対して 3.2.3 節の時刻 τ での最大連結成分のサイズ $N(\tau)$ の比として, 最大連結成分比 G を考える.

$$G = \frac{N(\tau)}{N} \quad (0 \leq G \leq 1) \tag{2.4}$$

この値が 0 に近ければ近い程, 孤立した小さな連結成分に分断されている.

2.3.2　ネットワークの効率 E

ネットワークの通信効率を現す指標の 1 つ [12] として,

$$E = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{D_{ij}} \tag{2.5}$$

を用いる. D_{ij} は, ネットワーク内に存在するノード i, j 間で最小ホップとなる最短経路長である.

2.4　カスケード故障に対する従来対策

2.4.1　生贄を用いたノード除去による対策

2.2.1 節のカスケード故障に対するアプローチとして, 生贄を用いたノード除去による対策 [5] が提案されている. この対策は, 初期故障ノードが発生すると負荷が他の経路へ流れようとするが, 中継に寄与しないノードを意図的に複数除去して負荷を減らすことでカスケード故障を抑制する.

除去するノードは, ノード i にかかる式 (2.1) の負荷からノード i 自身で生成される負荷

$$L_i^g = \sum_j (D_{ij} + 1) \tag{2.6}$$

を引いた

$$\Delta = L_i - L_i^g \tag{2.7}$$

の小さい順によって決定される.D_{ij} は, ネットワーク内に存在するノード i, j 間で最小ホップとなる最短経路長である. 文献 [5] の手法は一定の効果は認められるものの, 生贄となるノードが必要となり問題がある.

2.4.2　ノード負荷を考慮したルーティングによる対策

上記の生贄となるノードを必要としない対策の 1 つとして, ノード負荷を考慮したルーティング法 [6] が提案されている. これは, 初期故障ノードが発生した際に残りのノード全てが過負荷になるわけではない点に着目し, 許容負荷に余力のあるノードへ優先的に負荷を分散させることで, 過負荷故障を抑制している.

具体的には, 経路選択においてノードの余力の有無は, ある時刻 τ におけるノード k の負荷率

$$W(\tau) = \frac{L_k(\tau)}{C_k} \tag{2.8}$$

を用いて, ノード s からノード t までの経路 $P(s \to t) \leftarrow s \equiv v_0, v_1, ..., v_{n-1}, v_n \equiv t$ に含まれる中継ノードの負荷率の総和が最小になる経路を選択する.

$$W\left(P(s \to t) : \tau\right) = \sum_{i=1}^{n-1} \left(1 + \frac{L_{v_i}(\tau)}{C_{v_i}}\right) \tag{2.9}$$

第3章　提案手法

本章では, 2.4.2 節で述べた迂回ルーティングにおけるカスケード故障の抑制手法に, s, t の選び方に順序制御を付ける方法を追加したアルゴリズムを考える.

3.1　迂回ルーティングにおける順序制御による カスケード故障の抑制手法

3.1.1　前提とする条件

以降, 説明するアルゴリズムは以下を前提とする.

- 攻撃を受け, 機能停止したノードは 1 つである.
- 機能停止したノードは, ネットワークから除去されたと仮定する.
- 故障前のネットワークは連結し, ノード N 個全て 1 つのネットワークに必ず繋がっているとする.
- ノード id と呼ばれる, 各ノードを全て識別ができるよう名前が付けられている.
- ノード id は重複しておらず, 全てユニークである.
- 各ノード間のリンクの重みを各端点ノードは直接把握できる.
- ノード間のやり取りは受動的で, 受信に対して返信のみ行う.
- BellmanFord 法 [13] を用いて,
  - コストの仮ラベル状態から確定コスト状態になるかの決定は Step 3-1 におけるスタートノード s_i が判断する.
  - 各ノードのラベルの初期状態スタートノードは確定ノードで負荷率 0, その他ノード未確定ノードで ∞ である.

  とする. BellmanFord 法は, その分散版が RIP(Routing Information Protocol) でも使われている

3.2　迂回ルーティングにおける順序制御についてのアルゴリズム

本節では, 提案する順序制御を含んだアルゴリズムを説明する.
アルゴリズムは, 概略として大きく分けて以下の 4 ステップで構成される.

Step 1-1〜Step 1-3
連結領域内を通常（通信）モードから探索モードへ切り替える.

Step 2-1〜Step 2-3
連結領域内を機能不全=除去されたノードの隣接が把握するために末端からそれに向かってメッセージを返信する.

Step 3-1〜Step 3-3
ノード s の指示に従って連結成分内の式 (2.8) のコストを調査する.

Step 4-1〜Step 4-5
調査した連結成分内のコストから, 式 (2.9) における最小負荷率のノードを選択し, 確定する.

3.2.1　Step 1:連結領域内を通常モードから探索モードへ切替

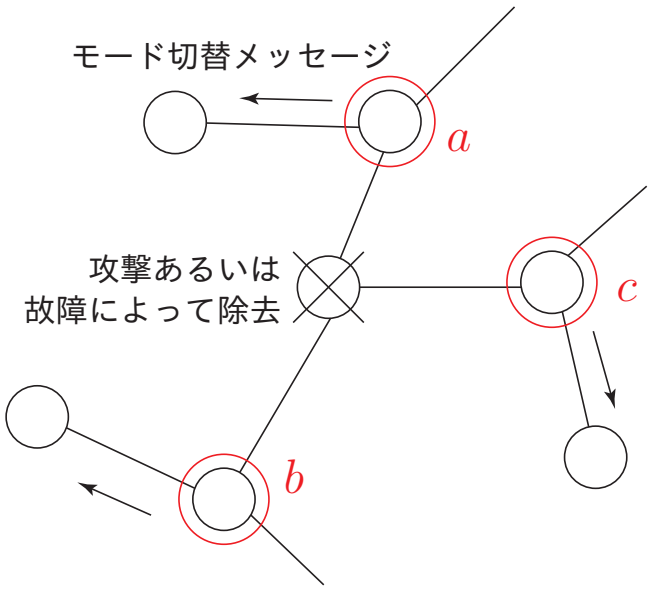


図 3.1: 故障ノードの発生

Step 1-1 故障ノードの発生

図 3.1 のように, 攻撃を受けたノードの隣接ノード（a, b, c）が無応答などで故障を検知すると, それらのノードは通常（通信）モードから探索モードへ切り替わる.

この時, a, b, c が根となって, それぞれの連結領域内を探索モードへ切り替える処理を始める.

a, b, c それぞれの隣接ノードから更にその隣接ノード, という順にノード故障が発生したことを「モード切替メッセージ」を送信して知らせる.

「モード切替メッセージ」の中身は,

- それぞれの連結領域内を探索モードへ切り替えるためのメッセージ
- この「モード切替メッセージ」を発信したノードのノード id（図 3.1 の例では a, b, c）

とする.

ノード a, b, c は連結成分を知る必要があるため, 「モード切替メッセージ」に a, b, c 自身それぞれのノード id を入れる. これにより, 後述する Step 2-3 “「到達メッセージ」がノード a, b, c に到着” の処理に従って連結成分を集めることが可能となる.

Algorithm 1 when 検知 (故障)

1: **when** 検知 (故障) **do**
2: 「モード切替メッセージ」へ自身のノード id を格納
3: 「モード切替メッセージ」を隣接ノードへ送信
4: **end when**

Step 1-2 「モード切替メッセージ」を受信したノードの動作

図 3.2 に示すノード a（ノード a を “仮親ノード” と呼ぶ）より「モード切替メッセージ」を受け取ったノード i は, 以下に従って処理する. ここで, 仮親ノードとは各メッセージの送信元ノードのことであり, 返信メッセージを返信するために用いる.

I. 「モード切替メッセージ」を受信したのが初回の場合（「仮親ノード id 記憶領域」が空の場合）

図 3.2 の j, k, l へさらに「モード切替メッセージ」を送信元以外の隣接ノードに送信（転送）する.

その後, 「モード切替メッセージ」を送ってきたノード（図 3.2 のノード a）のノード id を「仮親ノード id 記憶領域」へ記録する.

「仮親ノード id 記憶領域」は全てのノードが保持しており, その中身は,

- “仮親ノード” のノード id

とする.

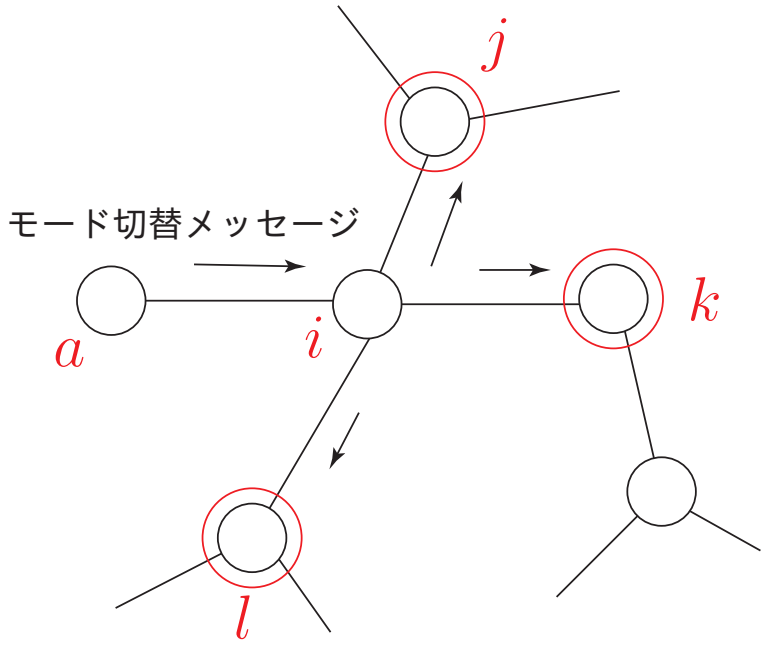


図 3.2: 「モード切替メッセージ」を受信したノードの動作

II. 「モード切替メッセージ」を 2 回目以降に受信した場合

図 3.3 に示すノード m のようにノード j のツリーに含まれているノードの場合, ノード m へ青矢印と黄矢印のように「モード切替メッセージ」の到着が重なるとノード m は既に「モード切替メッセージ」を保持しているため, これと新しく受信した「モード切替メッセージ」のいずれかを選択しなければならない. そこで, ノード id が最小のノードの「モード切替メッセージ」を採用し, 「仮親ノード id 記憶領域」を更新する. ノード id の大小関係を $j < k < l$ とすると, ノード m はノード k の「モード切替メッセージ」を採用する.

また, これに伴い「モード切替メッセージ」を隣接ノードへ転送する.

直径程度以上の連結成分全体へ伝わる時間待った後, ノード m は「仮親ノード id 記憶領域」の更新内容に応じた「返信メッセージ」を仮親ノードへ送信する.

「返信メッセージ」の中身は,

- 「仮親ノード id 記憶領域」を更新していれば「返信メッセージ」の中身を “1” , 更新していなければ “0”

とする.

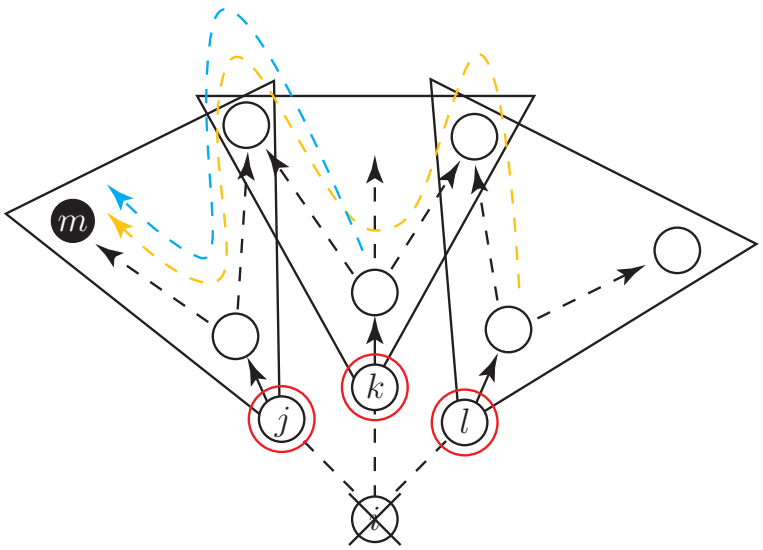


図 3.3: 「到達メッセージ」を送信

Algorithm 2 when 受信（「モード切替メッセージ」）

1: **when** 受信（「モード切替メッセージ」） **do**
2: **if** 「仮親ノード id 記憶領域」=空 *or* 「仮親ノード id 記憶領域」 > （「モード切替メッセージ」を発信したノード id） **then**
3: 「モード切替メッセージ」を隣接ノードへ送信
4: 「モード切替メッセージ」を発信したノード id を「仮親ノード id 記憶領域」へ記録
5: **if** 「仮親ノード id 記憶領域」を更新 **then**
6: 「返信メッセージ」=1
7: **else**
8: 「返信メッセージ」=0
9: **end if**
10: 「返信メッセージ」を仮親ノードへ送信
11: **else if** 「仮親ノード id 記憶領域」 < （「モード切替メッセージ」を発信したノード id） **then**
12: none.
13: **end if**
14: **end when**

Step 1-3 終端ノードの判断

図 3.4 に示すノード m は終端ノードであり, これが終端であると判断をすることで Step 2-1[「到達メッセージ」を送信] へ動作が移行する.

ノード m 自身が終端ノードと判断するには, まず先述の流れと同様に受け取った「モード切替メッセージ」を図 3.4 の隣接ノード p, q, r へ転送しようとする.

もし, 隣接ノード p, q, r が既に「モード切替メッセージ」を受信済の時は, 「仮親ノード id 記憶領域」の更新は無く, 「返信メッセージ」は更新無しとノード m へ返ってくる. この場合, ノード m は隣接ノードが全て「モード切替メッセージ」を受信したことが分かり, 且つノード m が終端ノードと自身で判断できる.

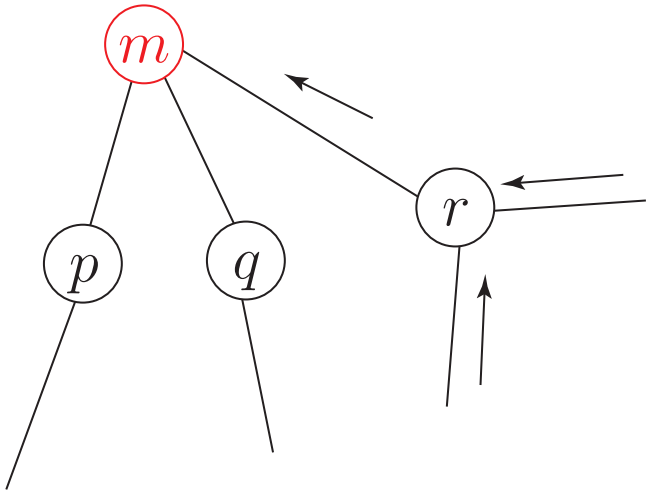


図 3.4: 終端ノードの判断

Algorithm 3 when 判断（終端）

```
1: when 判断（終端） do
2:     隣接ノードへ「モード切替メッセージ」を送信
3:     if 「返信メッセージ」=0 then                    ▷ Algorithm 2 参照
4:         終端ノードと判断
5:     else
6:         return “when 受信（「モード切替メッセージ」）”
7:     end if
8: end when
```

3.2.2 Step 2:連結領域の抽出

Step 2-1 「到達メッセージ」を送信

ノード m が終端と判断した後, ネットワークの端から端までメッセージが届くまで直径程度（充分）の時間を待ち, 新しくメッセージの到着が無いことを確認してから「到達メッセージ」をノード a に向かって送信する. すなわち, 他の連結領域ノードからメッセージが到着する可能性があるため, 一定時間待ってから「到達メッセージ」を送信する.

「到達メッセージ」の中身は,

- 終端まで「モード切替メッセージ」が到達したことをノード a に知らせるメッセージ
- 終端からノード a までのルート情報

とする.

ノード m は「到達メッセージ」を作成する時に, 自身のノード id を「到達メッセージ」に含めて自身の仮親ノードへ送信する.

Step 2-2 「到達メッセージ」を受信

「到達メッセージ」を受け取ったノードは, 更に自身の仮親ノードに転送する. その際, 「到達メッセージ」に自身のノード id を追加する.

Algorithm 4 when 受信（「到達メッセージ」）

```
1: when 受信（「到達メッセージ」） do
2:     「到達メッセージ」へ自身のノード id を格納
3:     仮親ノードへ「到達メッセージ」を転送
4: end when
```

「モード切替メッセージ」を発信したノード a に到達するまで繰り返す.

Step 2-3 「到達メッセージ」がノード a, b, c に到着

「到達メッセージ」を受け取った各ノード a, b, c は, 「到達メッセージ」 内に含まれているルート情報を読み取る. 但し, 例えば a と b が連結している時は, ノード id が最小のノードの連結成分に所属することとする. 読み取った連結成分はノード a, b, c それぞれが持っている「連結成分リスト」に保存される.

「連結成分リスト」の中身は,

- 「到達メッセージ」内に含まれる連結成分情報をリスト形式で保存できる

とする.

3.2.3 Step 3:ノード s の指示に従って連結成分内の負荷率を調査

Step 3-1 3.3 節で後述する 3.3 で決定した順序 $s_1, s_2, \cdots$ に従って 1 番目の送信元ノード s_1 を決定する.

Step 3-2 s_1 と決まったノード（図 3.5 のノード a）は隣接ノードの式 (2.8) のコストを知るため,「負荷率調査メッセージ」を各隣接ノードへ送る.

「負荷率調査メッセージ」の中身は,

- 各ノードの負荷率を要求するメッセージ
- 送信元ノード a へ各ノードの式 (2.8) のコストの情報を集めるために, 各ノードの隣接ノードの式 (2.8) のコストの調査を指示するメッセージ
- この「負荷率調査メッセージ」を発信したノードのノード id（図 3.5 の例では a）
- 送信元ノード a が持っている Step 2-3 の「連結成分リスト」

とする.

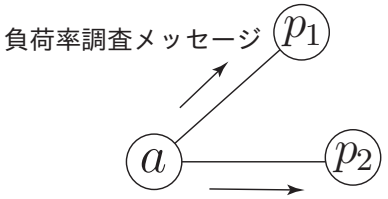


図 3.5: 送信（「負荷率調査メッセージ」）

Algorithm 5 when 送信（「負荷率調査メッセージ」）

```
1: when 送信（「負荷率調査メッセージ」） do
2:     「負荷率調査メッセージ」へ自身のノード id を格納
3:     「負荷率調査メッセージ」を隣接ノードへ送信
4: end when
```

Step 3-3 「負荷率調査メッセージ」を受け取ったノード p_i はまず,「負荷率調査メッセージ」内に含まれている「連結成分リスト」に自身が含まれているかを確認し, 含まれていれば自身の負荷率を「負荷率返信メッセージ」に格納し, 仮親ノード（図 3.6 のノード a）に返信する.

「負荷率返信メッセージ」の中身は,

- 「負荷率調査メッセージ」を受け取ったノード p_i のノード id
- ノード id に対応する式 (2.8) のコスト

とする.

また, ノード p_i は図 3.7 のように自身の隣接ノードへ「負荷率調査メッセージ」を転送する.

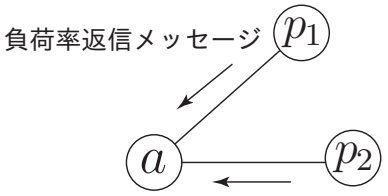


図 3.6: 返信（「負荷率返信メッセージ」）

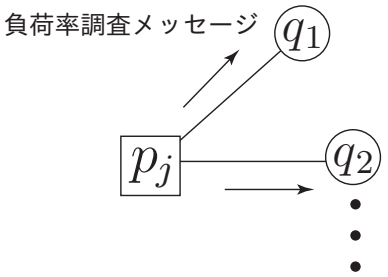


図 3.7: 転送（「負荷率調査メッセージ」）

Algorithm 6 when 受信（「負荷率調査メッセージ」）

```
1: when 受信（「負荷率調査メッセージ」） do
2:     if 「連結成分リスト」に自身が含まれている then
3:         自身の負荷率を計算
4:         「負荷率返信メッセージ」に計算した自身の負荷率を入れる
5:         仮親ノードへ「負荷率返信メッセージ」を返信
6:     else
7:         none.
8:     end if
9: end when
```

3.2.4 Step 4:調査した連結成分内の負荷率から, 最小負荷率のノードを通る経路を選択・確定

Step 4-1 隣接ノードから「負荷率返信メッセージ」を受け取った送信元ノード s_i（図 3.5 のノード a）は「負荷率返信メッセージ」に含まれている負荷率を送信元ノード a 自身の「候補リスト」に追加する.

「候補リスト」の中身は, 「負荷率返信メッセージ」に含まれる

- ノード p_i のノード id
- ノード id に対応する式 (2.8) のコスト

がリスト形式（配列）で保存されている.

I. 「負荷率返信メッセージ」に含まれるノードがまだ「候補リスト」に追加されていないとき

「負荷率返信メッセージ」に含まれている負荷率を送信元ノード a 自身の「候補リスト」に追加する.

II. 「負荷率返信メッセージ」に含まれるノードが既に「候補リスト」に追加されているとき

既にリストに追加されている負荷率と「負荷率返信メッセージ」に含まれる負荷率を比較し, 負荷率が最小になる方を残す.

Algorithm 7 when 受信 (「負荷率返信メッセージ」)

```
1: when 受信 (「負荷率返信メッセージ」) do
2:     「候補リスト」に「負荷率返信メッセージ」内の各ノード負荷率を追加
3:     if (「候補リスト」内のノード負荷率) > (新しく受信した「負荷率返信メッセージ」内のノード負荷率) then
4:         新しく受信した「負荷率返信メッセージ」内の負荷率へ「候補リスト」を更新
5:     else
6:         none.
7:     end if
8: end when
```

Step 4-2 ノード a は, 負荷率の更新が無くなるまで直径程度（充分）の時間待った後, 「候補リスト」の中から最小負荷率ノードを選択し, 負荷率を確定する.

次に, 負荷率が確定したノード p_j へ確定情報を伝えるため, ノード p_j へ「負荷率確定メッセージ」を送る.

「負荷率確定メッセージ」の中身は,

- 負荷率が確定したことを知らせるメッセージ
- 負荷率が確定したノード p_j のノード id

とする.

Algorithm 8 when 送信（「負荷率確定メッセージ」）

```
1: when 送信（「負荷率確定メッセージ」） do
2:     「候補リスト」内から最小負荷率ノードを選択
3:     「負荷率確定メッセージ」に確定メッセージを入れる
4:     「負荷率確定メッセージ」にノード p_j のノード id を入れる
5:   p_j へ「負荷率確定メッセージ」を送信
6: end when
```

Step 4-3 「負荷率確定メッセージ」を受け取った p_j は, 自身が確定したことが分かるよう, 「負荷率確定フラグ」を変更する. 「負荷率確定フラグ」の中身は,

- 「負荷率確定メッセージ」を受信したら中身を “1” , 受信していなければ “0”

とする.

次に, ノード p_j の負荷率が確定した後の隣接ノードの負荷率を知るため, 「負荷率調査メッセージ」を各隣接ノードへ送る.

Algorithm 9 when 受信（「負荷率確定メッセージ」）

```
1: when 受信（「負荷率確定メッセージ」） do
2:     「負荷率確定フラグ」=1 ▷「負荷率確定メッセージ」を受信=1, 未受信=0
3:     「負荷率調査メッセージ」を隣接ノードへ送信
4: end when
```

Step 4-4 図 3.7 のように「負荷率調査メッセージ」を受け取った q_i は自身の式 (2.8) のコストを「負荷率返信メッセージ」で返す.

図3.6のように隣接ノードから「負荷率返信メッセージ」を受け取った p_j はそれらを a へ伝えるために, a へ「負荷率返信メッセージ」を転送する（Algorithm 6 参照）.

Step 4-5 隣接ノードから「負荷率返信メッセージ」を受け取った a は Step 4-1 の動作を行う.

3.3 s, t の選択方法

Step 3-1 における s, t の選択方法は図 3.8 に示すように,

I: 1 つの s_i に対して $N-1$ 個の t_j を 1 つずつ順に選択した後, 次の s_i+1 も $N-1$ 個の t'_j を選択する方法.

II: **I** にて互いに経路が交わらない部分木内の t_j を同時刻で選択する方法.

III: ある定まった $s_1, s_2, ..., s_N$ のノード順に巡回して, 各 s_i に対しては 1 つの t_i を選び, 次の一巡ではペアとなる別の t_i を選ぶことを繰り返す方法.

IV: **II** と同様に **III** に対して, 経路が交わらないものを同時刻で決定する方法.

の 4 種類を考えた.

特に **II** と **IV** に関しては, 図 3.8 内のノード $u_1, u'_1, ...$ のように根が異なる木構造で互いに独立していれば, $t_1, t'_1, ...$ の経路は互いに交わらず, 互いに負荷の影響を考えず, 同時刻にて決定して集約することができる.

また, 上記の **I:**から **IV:**のそれぞれについて,

最長経路優先選択： 2.4.2 節の式 (2.8) による $P(s \to t)$ 間に含まれる各ノード v_k の負荷率 $W(\tau)$ の和が “最大” になる経路から優先的に t_j を決定する.

これは, $W(\tau)$ の和が大きくなると, 図 3.9 で示したように中継するノード数が更に多くなると考えられ, 負荷率は経路を通過する毎にノード v_k の負荷 $L_{v_k}(\tau)$ は $+1$ されるため, 影響を受けやすいために, 経路決定の際は初期に決定することでそれを回避できると考えられる.

最短経路優先選択： 2.4.2 節の式 (2.8) で示した $P(s \to t)$ 間に含まれる各ノード v_k の負荷率 $W(\tau)$ の和が “最小” になる経路から優先的に t_j を決定する.

これは, 始点に近いほど経路の候補が元々限られており, 初期の方で決めておくべきと考えられる.

の 2 種類を考えた.

これらの組合せを, 表 3.1 に示すパターン 1〜8 と名付けた.

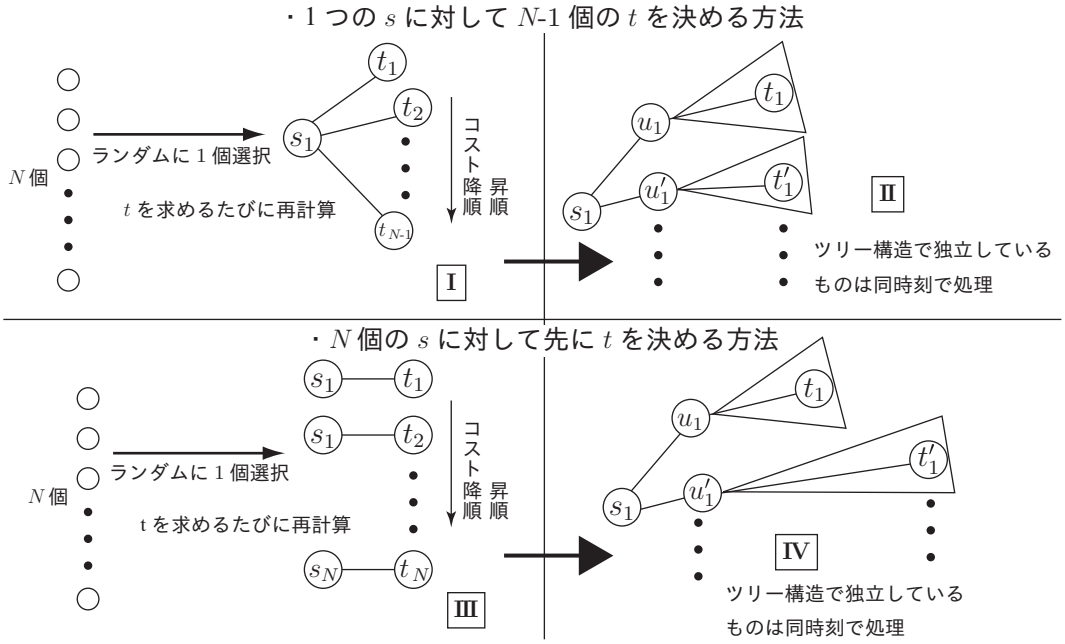


図 3.8: パターン図

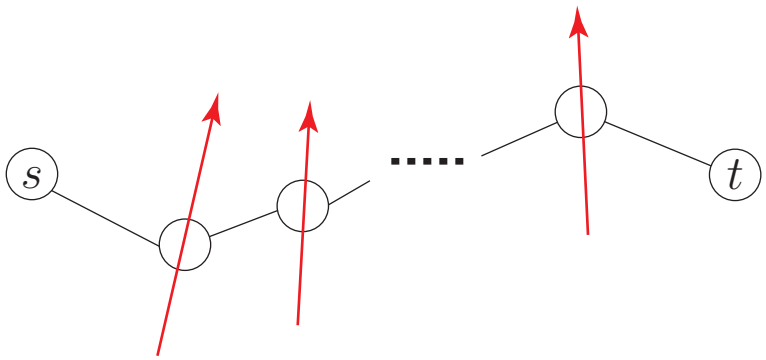


図 3.9: 最長経路優先選択における他経路（赤矢印）の影響

表 3.1: s, t の選択方法

図 3.8	最大コスト経路優先選択	最小コスト経路優先選択
I	パターン 1	パターン 5
II	パターン 2	パターン 6
III	パターン 3	パターン 7
IV	パターン 4	パターン 8

第4章　1個の初期故障に対するカスケード故障の抑制効果

本章では, 順序制御を考慮した迂回ルーティングにおけるカスケード故障の抑制に対して, シミュレーションを通して1個の初期故障（トリガー）ノードによるカスケード故障の抑制効果を調査する.

対象としたネットワークは, 2.1.1節のBAモデルと2.1.2節の玉葱状ネットワークを, ノード数は $N = 10^3$, 追加リンク数は $m = 4$ で生成した.

4.1　SFネットワークに対するカスケード故障抑制効果の評価

図4.1から図4.4は, 2.1.1節のBAモデルを用いて生成したSFネットワークに対するカスケード故障のシミュレーション結果である.

以下, 4.1.1節の最大コスト経路優先選択と4.1.2節の最小コスト経路優先選択に分けて説明する.

4.1.1　最大コスト経路優先選択

図4.1, 4.2は最大コスト経路優先選択で行ったシミュレーション結果であり, 図4.2は式(2.3)の耐久性パラメータ α が小の部分 $(0 \leq \alpha \leq 0.4)$ を拡大したものである. SFネットワークではパターン1と2が従来手法[6]と同等の最大連結成分比 G, ネットワークの効率 E を保てているが, パターン3と4においては最大連結成分比 G, ネットワークの効率 E が下がっている. これは, 主にパターン3と4の特徴である各 s_i に対して最長経路から t_j を決定してしまうことが, かえって他の経路を迂回させてしまい, 結果として連鎖故障を引き起こしてしまったためと考えられる. パターン4は最長経路を更に同時刻にて複数決定するもので, $\alpha = 0.1$ においても $G = 0$ となっている.

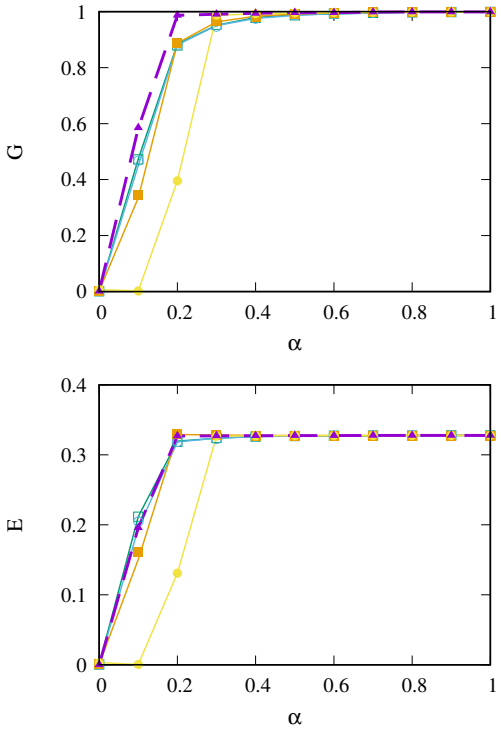


図 4.1: SF, 初期故障ノード:1 個, 最大コスト経路優先選択

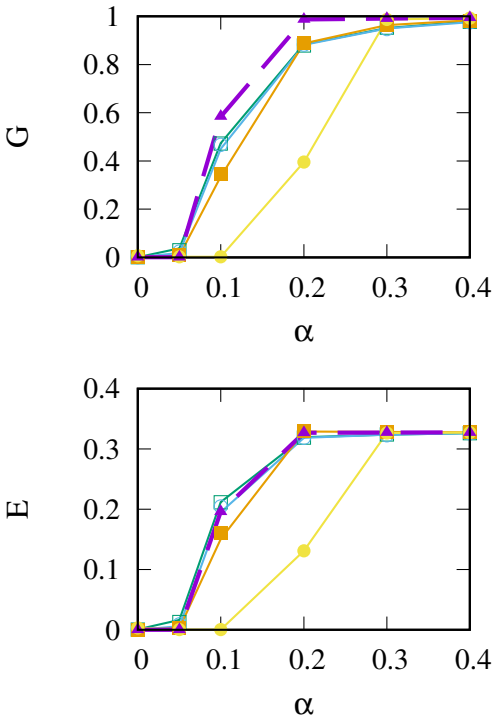
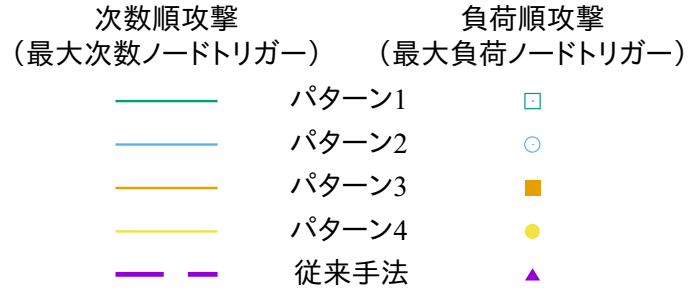


図 4.2: SF, 初期故障ノード:1 個, 最大コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$



4.1.2 最小コスト経路優先選択

図 4.3, 4.4 は最小コスト経路優先選択で行ったシミュレーション結果であり, 図 4.4 は式 (2.3) の耐久性パラメータ α が小の部分 $(0 \leq \alpha \leq 0.4)$ を拡大したものである. SF ネットワークでは特に, パターン 7 と 8 が従来手法 [6] と同等の最大連結成分比 G, ネットワークの効率 E を保てているが, パターン 5 と 6 においては最大連結成分比 G, ネットワークの効率 E が下がっている.

パターン 7 と 8 が従来手法 [6] と同等 G, E を保てた理由として, 始点に近いほど経路の候補が元々限られているため, 初期の方で決めておけたことが影響したと考えられる.

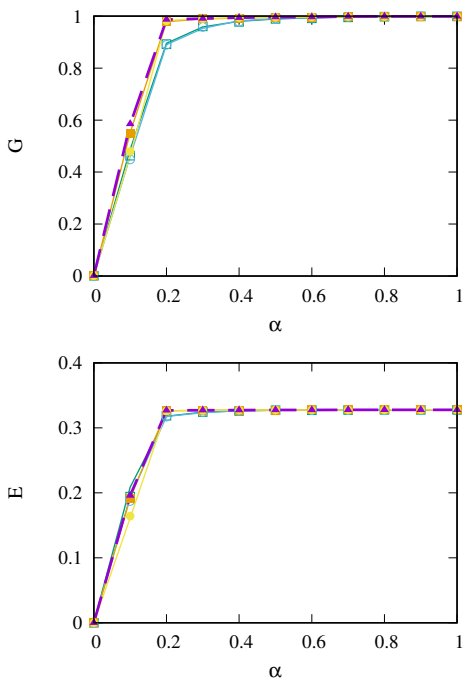


図 4.3: SF, 初期故障ノード:1 個, 最小コスト経路優先選択

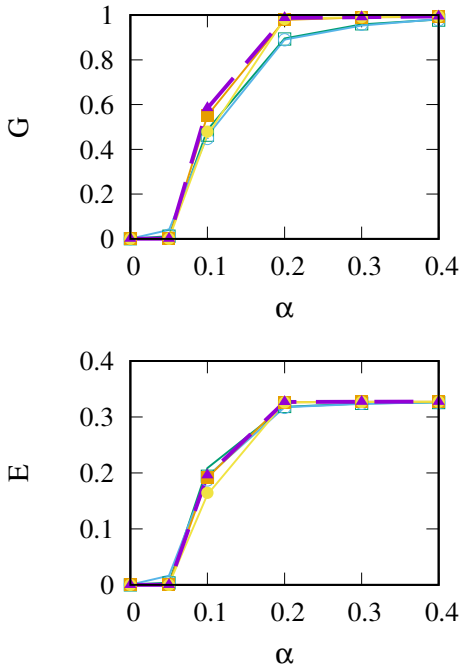
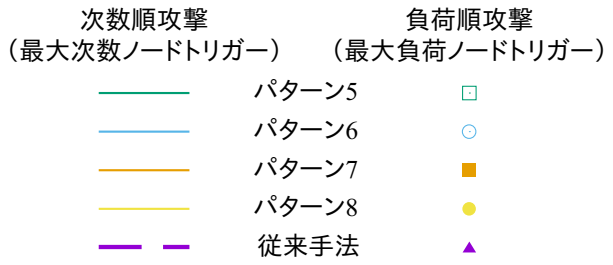


図 4.4: SF, 初期故障ノード:1 個, 最小コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$



4.1.3　ネットワークのノードサイズ N の違いにおける変化

図 4.5, 4.6 は SF ネットワークのノードサイズ N が変わった時の変化を確認するために, $N = 10^3$ のデータに $N = 500$ を加え, 再整理したものである. ノードサイズ N が変化しても, 全パターンの傾向は同じであることが確認できる.

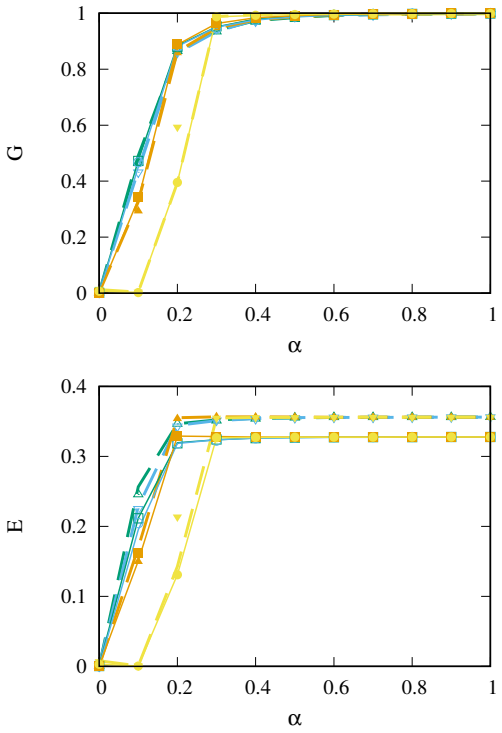


図 4.5: SF, 初期故障ノード:1 個, 最大コスト経路優先選択, $N = 500, 10^3$

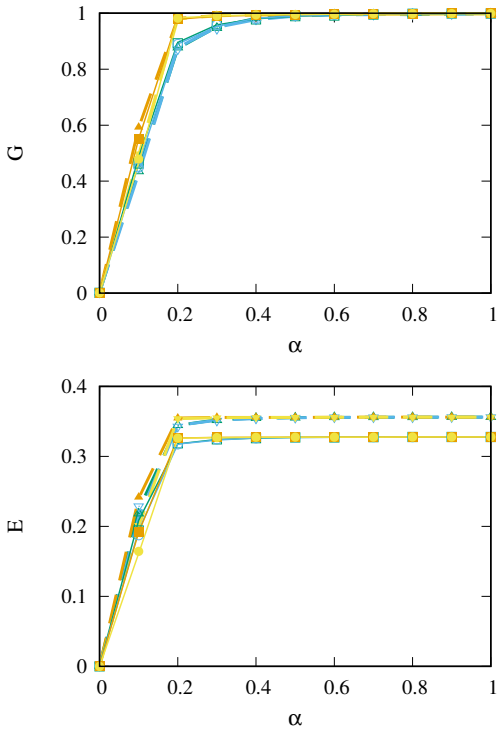


図 4.6: SF, 初期故障ノード:1 個, 最小コスト経路優先選択, $N = 500, 10^3$

4.2 玉葱状ネットワークにおけるカスケード故障の抑制効果

図 4.7 から図 4.18 は, 2.1.2 節の最適耐性を持つ玉葱状ネットワークに対するカスケード故障のシミュレーション結果である. 図 4.7, 4.8, 4.13, 4.14 は仲介数 $\mu = 1$, 図 4.9, 4.10, 4.15, 4.16 は仲介数 $\mu = 3$, 図 4.11, 4.12, 4.17, 4.18 は仲介数 $\mu = 5$ の μ が小・中・大のネットワークである.

4.2.1 最大コスト経路優先選択

図 4.7 から図 4.12 は最大コスト経路優先選択で行ったシミュレーション結果であり, 図 4.8, 4.10, 4.12 は式 (2.3) の耐久性パラメータ α が小の部分 $(0 \leq \alpha \leq 0.4)$ を拡大したものである. 先の 4.1 節の SF に対しては見られなかった特筆すべき結果として, 玉葱状ネットワークでは耐久性パラメータ $\alpha = 0$ において従来手法 [6] では最大連結成分比 G が 0 となり, ネットワークの連結成分を維持できなくなっていたが, 本研究の提案手法では各ノードに全く余裕が無い $\alpha = 0$ の状況においてもパターン 1 と 2 あるいは 3 において G を 2 割〜4 割程度維持できている. これは, 玉葱状ネットワークの特徴である多数の絡まったループ構造によって迂回路ができ, 過負荷故障を起こさなかったと考えられる. 特にパターン 4 は 4.1.1 節のスケールフリーネットワークの場合と同様に最長経路を更に同時刻にて複数決定するもので, $\alpha = 0.1$ においても $G = 0$ となっている.

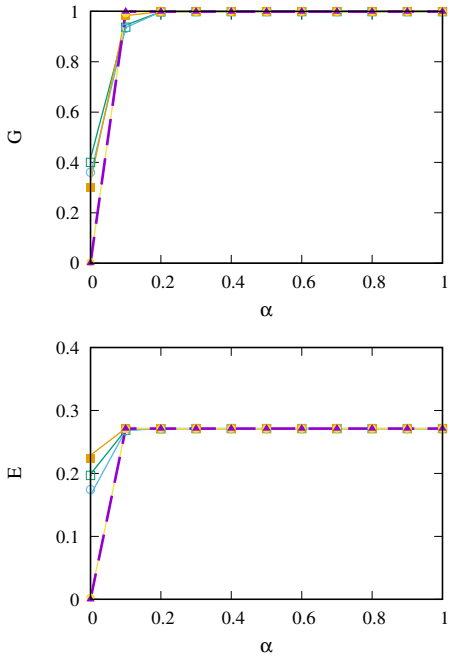


図 4.7: $\mu = 1$, 初期故障ノード:1 個, 最大コスト経路優先選択

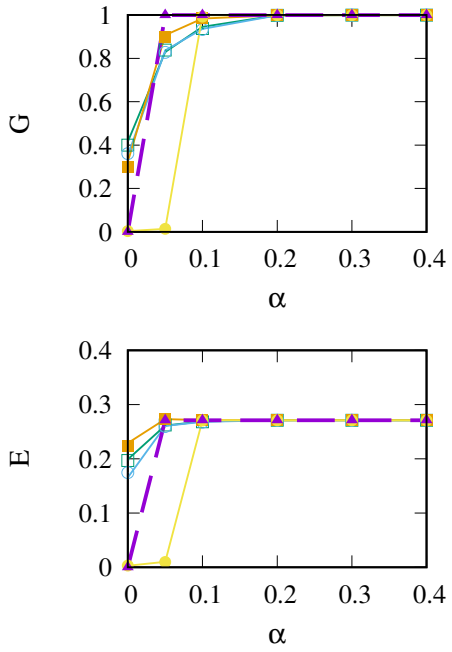


図 4.8: $\mu = 1$, 初期故障ノード:1 個, 最大コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$

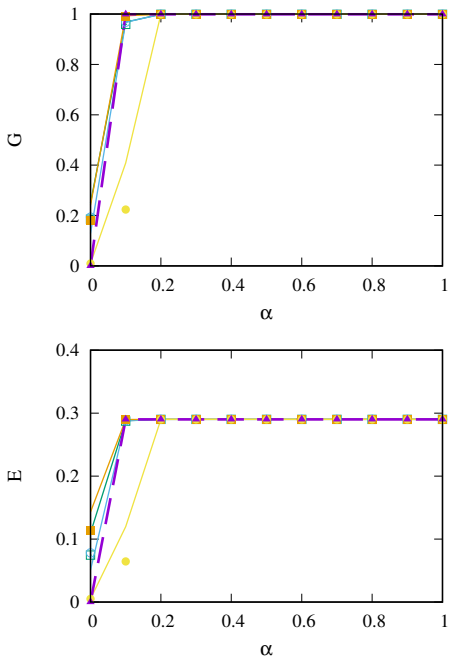


図 4.9: $\mu = 3$, 初期故障ノード:1 個, 最大コスト経路優先選択

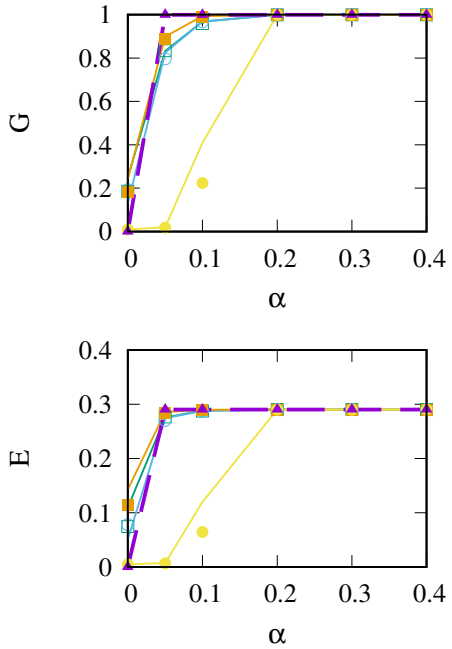


図 4.10: $\mu = 3$, 初期故障ノード:1 個, 最大コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$

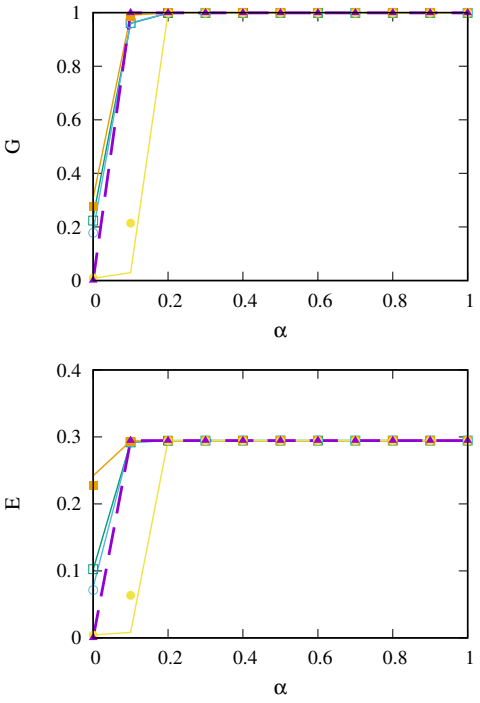


図 4.11: $\mu = 5$, 初期故障ノード:1 個, 最大コスト経路優先選択

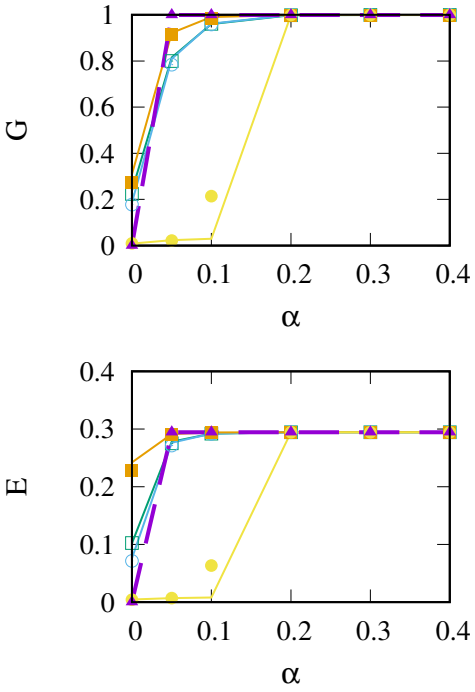
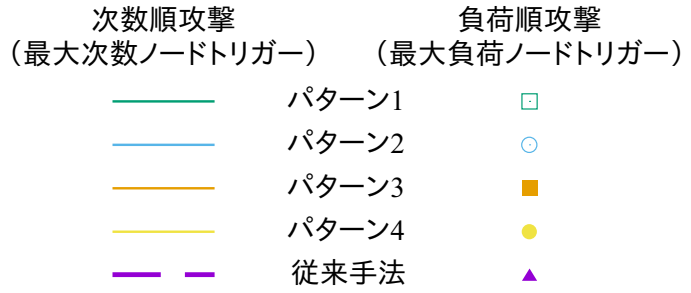


図 4.12: $\mu = 5$, 初期故障ノード:1 個, 最大コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$



4.2.2　最小コスト経路優先選択

図 4.13 から 4.18 は最大コスト経路優先選択で行ったシミュレーション結果であり, 図 4.14, 4.16, 4.18 は式 (2.3) の耐久性パラメータ α が小の部分 $(0 \leq \alpha \leq 0.4)$ を拡大したものである. 4.2.1 節の最大コスト経路優先選択と同様に, 玉葱状ネットワークでは耐久性パラメータ $\alpha = 0$ において従来手法 [6] では最大連結成分比 G が 0 となり, ネットワークの連結成分を維持できなくなっていたが, 本研究の提案手法では各ノードに全く余裕が無い $\alpha = 0$ の状況においてもパターン 5 と 6 において G を 2 割〜4 割程度維持できている. これは, 4.2.1 節と反して, 全てのノードに対して最小コスト経路優先選択をすることで短いものは短いままで経路確定をできたと考えられる.

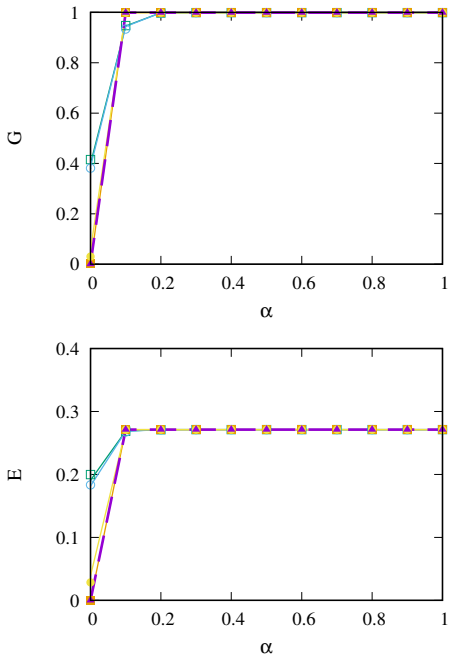


図 4.13: $\mu = 1$, 初期故障ノード:1 個, 最小コスト経路優先選択

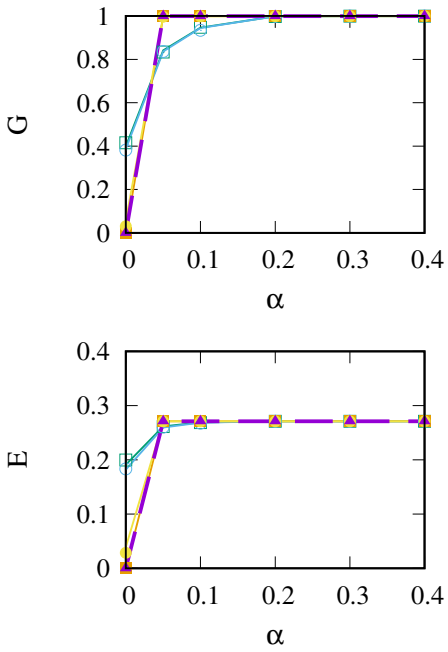
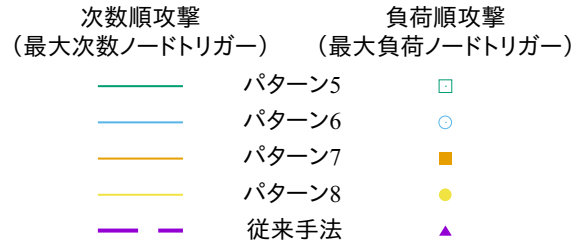


図 4.14: $\mu = 1$, 初期故障ノード:1 個, 最小コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$



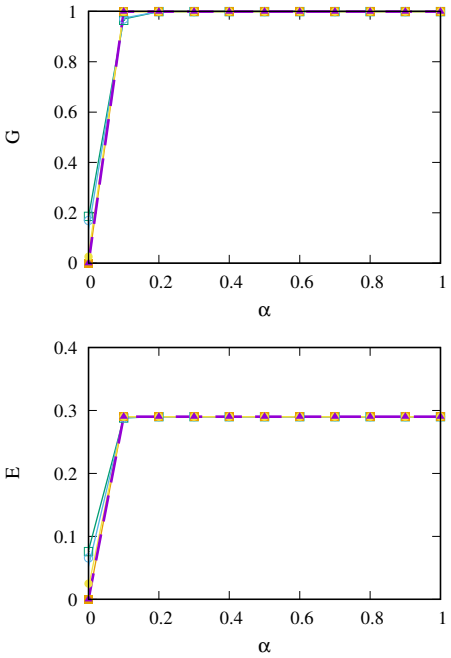


図 4.15: $\mu = 3$, 初期故障ノード:1 個, 最小コスト経路優先選択

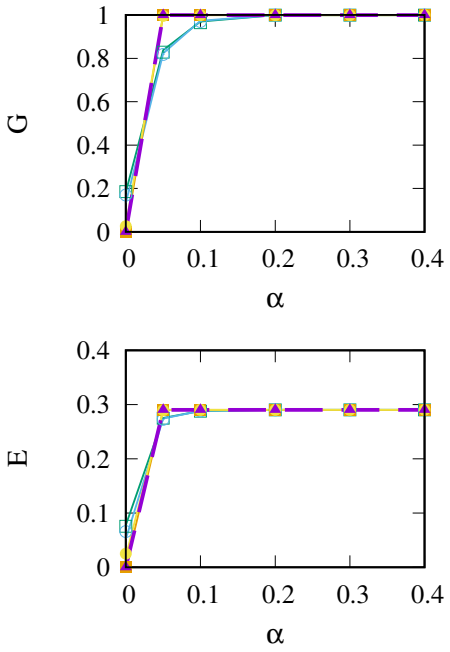


図 4.16: $\mu = 3$, 初期故障ノード:1 個, 最小コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$

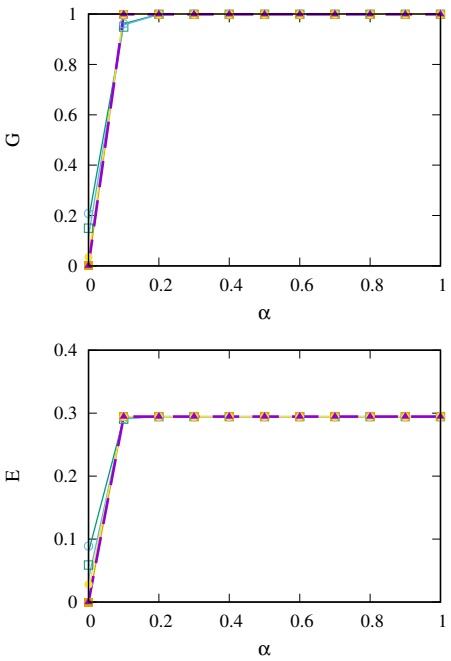


図 4.17: $\mu = 5$, 初期故障ノード:1 個, 最小コスト経路優先選択

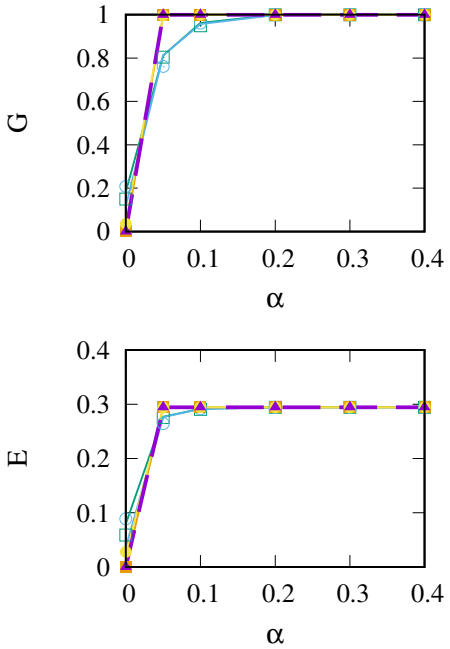


図 4.18: $\mu = 5$, 初期故障ノード:1 個, 最小コスト経路優先選択 $(0 \leq \alpha \leq 0.4)$

4.2.3　ネットワークのノードサイズ N の違いにおける変化

図 4.19 から図 4.24 は SF ネットワークのノードサイズ N が変わった時の変化を確認するために, $N = 10^3$ のデータに $N = 500$ を加え, 再整理したものである. ノードサイズ N が変化しても, 全パターンの傾向は同じであることが確認できる.

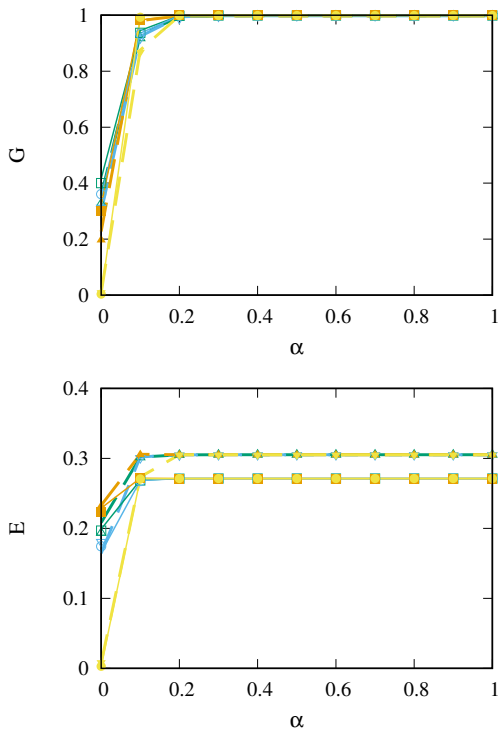


図 4.19: $\mu = 1$, 初期故障ノード:1 個, 最大コスト経路優先選択, $N = 500, 10^3$

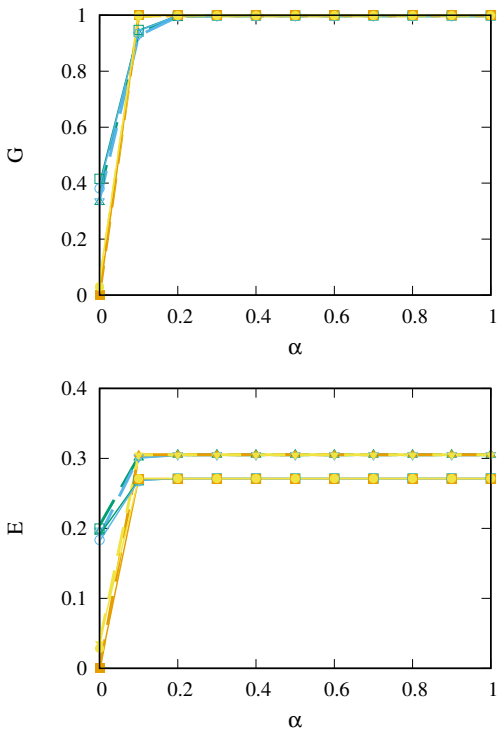


図 4.20: $\mu = 1$, 初期故障ノード:1 個, 最小コスト経路優先選択, $N = 500, 10^3$

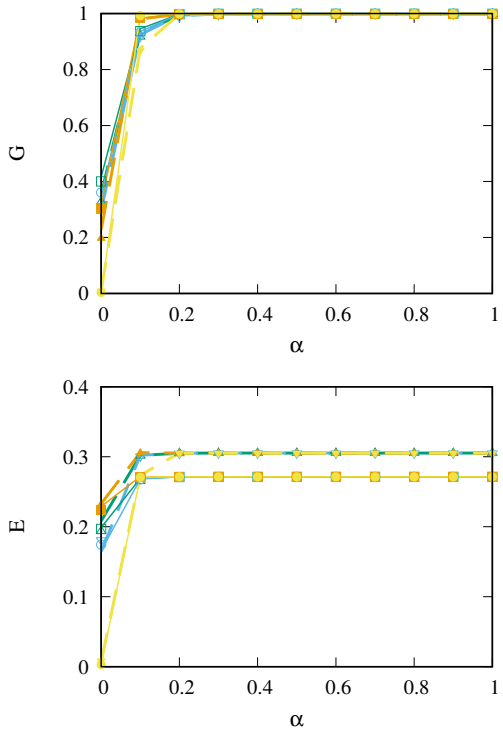


図 4.21: $\mu = 3$, 初期故障ノード:1 個,
最大コスト経路優先選択,
$N = 500, 10^3$

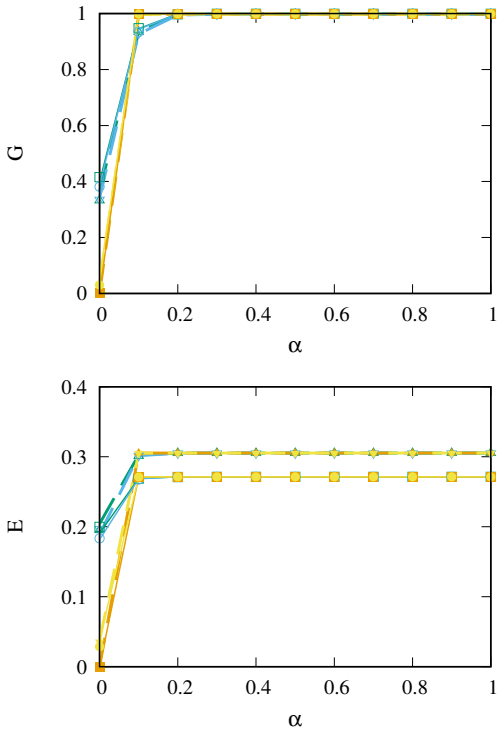


図 4.22: $\mu = 3$, 初期故障ノード:1 個,
最小コスト経路優先選択,
$N = 500, 10^3$

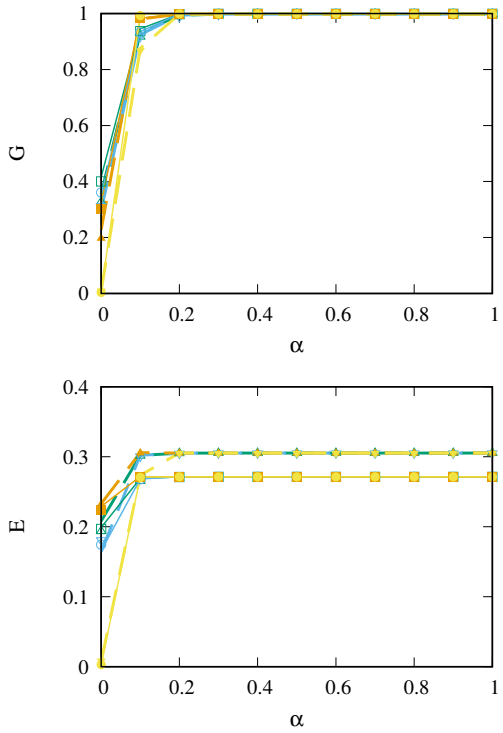


図 4.23: $\mu = 5$, 初期故障ノード:1 個, 最大コスト経路優先選択, $N = 500, 10^3$

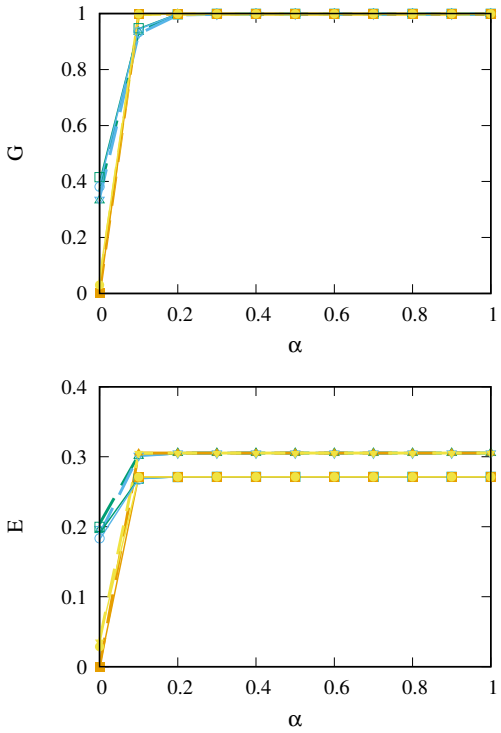


図 4.24: $\mu = 5$, 初期故障ノード:1 個, 最小コスト経路優先選択, $N = 500, 10^3$

4.3　各パターンにおける経路探索本数の比較

本節では, 図 3.8 および表 3.1 の **II** と **IV:**で, 経路探索回数がどのくらい削減できているかを計測した結果を示す. ここで, 対象としている経路探索の部分は, 第 3 章の 3.2.3, 3.2.4 節であり, Step 3-1 における s, t の選択方法は最大コスト経路優先選択と最小コスト経路優先選択で行った. また, ノード数は $N = 500$ のネットワークで行った.

4.3.1　経路探索本数の比較

図 4.25 から図 4.40 は, 各ネットワークモデルにおける経路探索本数を示したものである. スケールフリーネットワークでは式 (2.3) の耐久性パラメータ α がほぼすべての範囲において経路探索本数が多いが, 玉葱状ネットワークでは $\alpha = 0.1$ を除いて経路探索本数が 10^8 程度である.

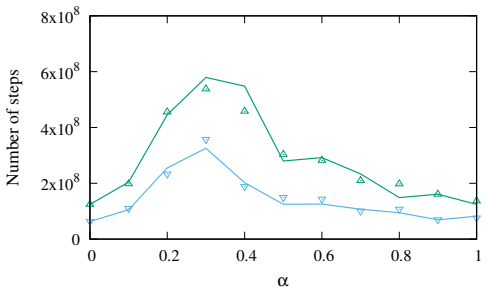


図 4.25: SF, パターン 1・2

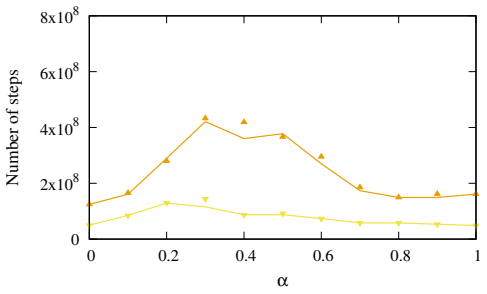


図 4.26: SF, パターン 3・4

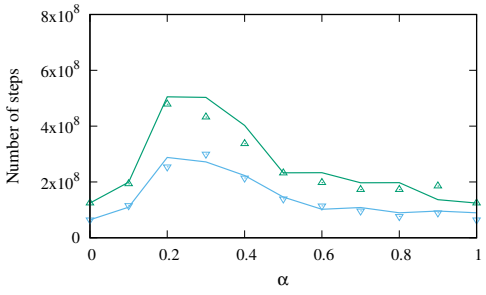


図 4.27: SF, パターン 5・6

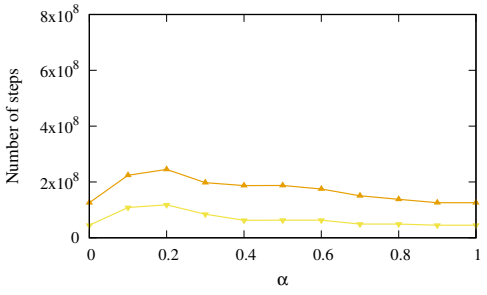
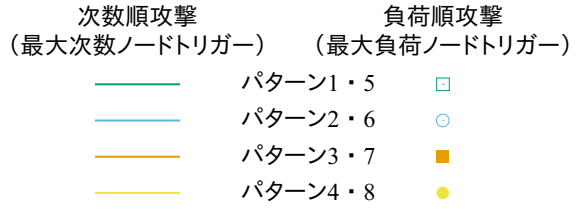


図 4.28: SF, パターン 7・8



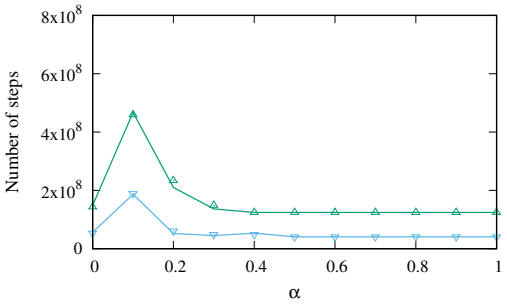


図 4.29: Onion-like($\mu = 1$), パターン 1・2

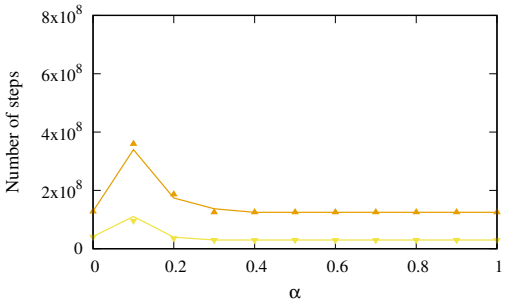


図 4.30: Onion-like($\mu = 1$), パターン 3・4

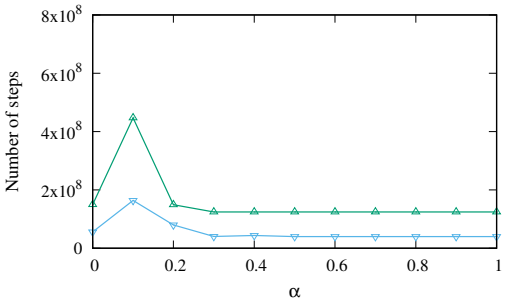


図 4.31: Onion-like($\mu = 1$), パターン 5・6

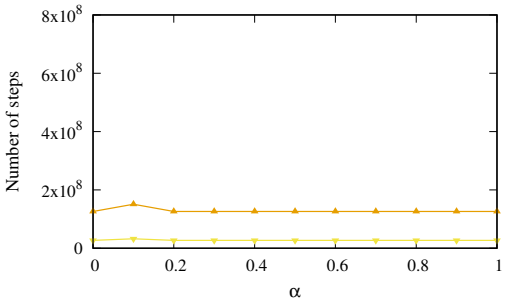
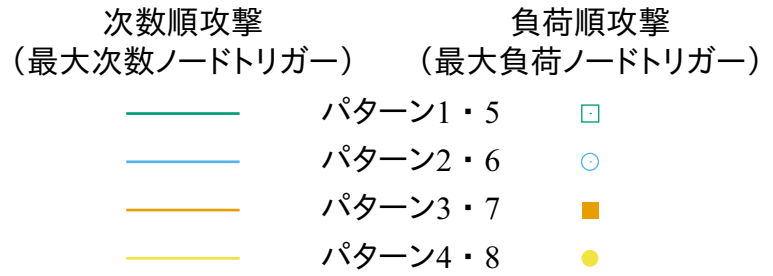


図 4.32: Onion-like($\mu = 1$), パターン 7・8



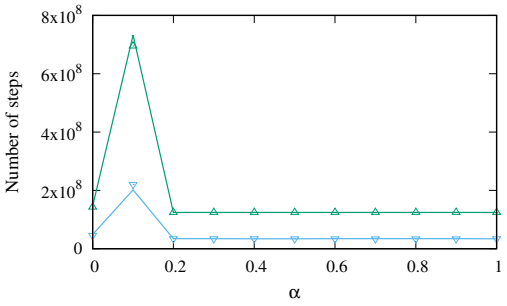


図 4.33: Onion-like($\mu = 3$), パターン 1・2

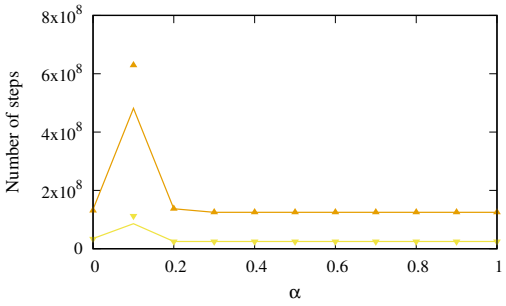


図 4.34: Onion-like($\mu = 3$), パターン 3・4

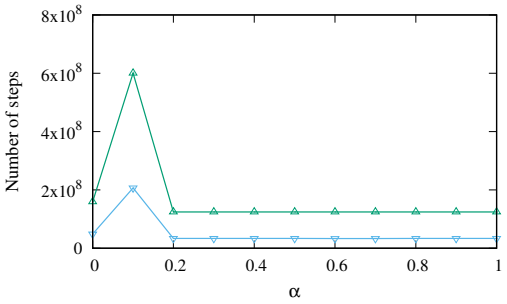


図 4.35: Onion-like($\mu = 3$), パターン 5・6

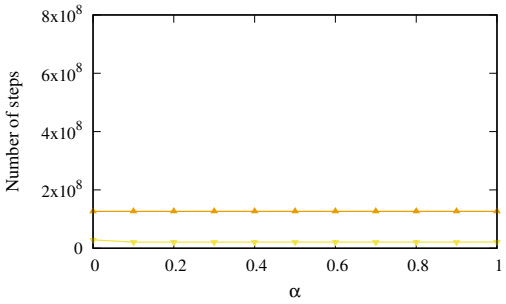
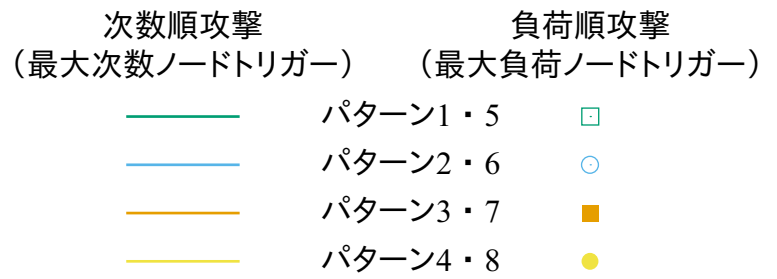


図 4.36: Onion-like($\mu = 3$), パターン 7・8



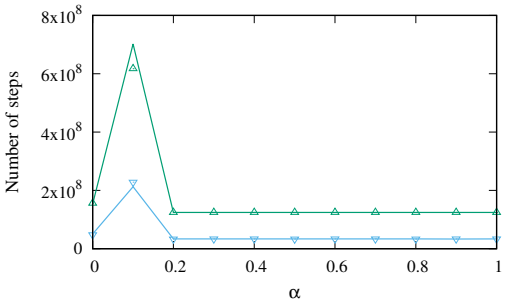


図 4.37: Onion-like($\mu = 5$), パターン 1・2

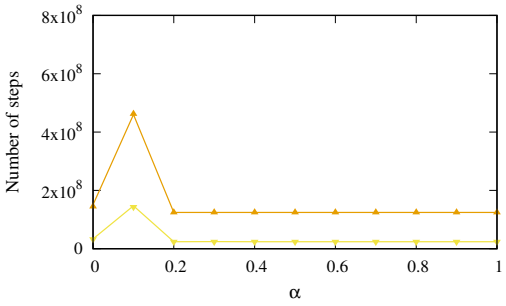


図 4.38: Onion-like($\mu = 5$), パターン 3・4

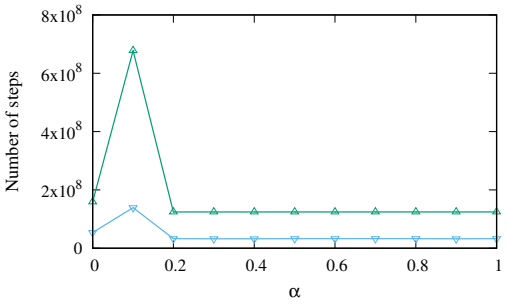


図 4.39: Onion-like($\mu = 5$), パターン 5・6

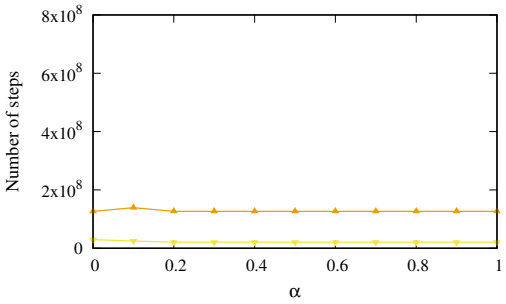
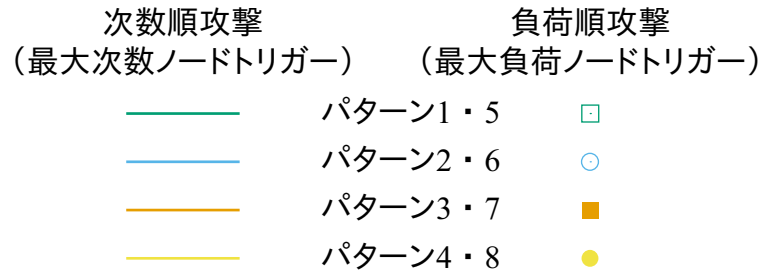


図 4.40: Onion-like($\mu = 5$), パターン 7・8



4.3.2　経路探索本数の割合での比較

図 4.41 から図 4.44 は 4.3.1 節のデータを割合（図 4.41 はパターン 2/パターン 1, 図 4.42 はパターン 4/パターン 3, 図 4.43 はパターン 6/パターン 5, 図 4.44 はパターン 8/パターン 7）で示して再整理したものである.

パターン 1〜8 すべてにおいて玉葱状ネットワークでは経路探索本数の削減量が少ない部分でも 50%以下, 特に最も経路探索本数の削減量が多かった $\mu = 3$, $\mu = 5$ に関しては約 20%まで経路探索本数の削減をすることが可能になっている. また, パターン 1〜8 を比較するとパターン 3 と 4, パターン 7 と 8 の方が, 同時処理を行った結果経路探索本数の削減量が多いことが言える.

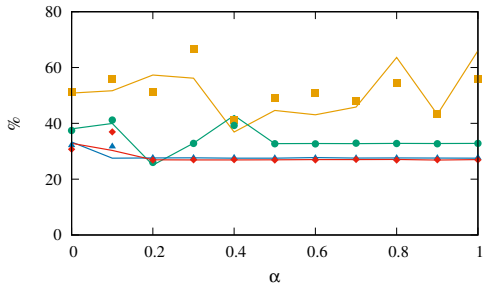


図 4.41: パターン 1・2[%]

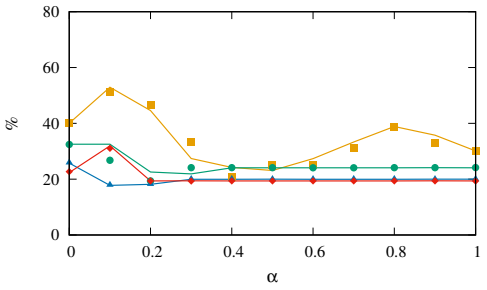


図 4.42: パターン 3・4[%]

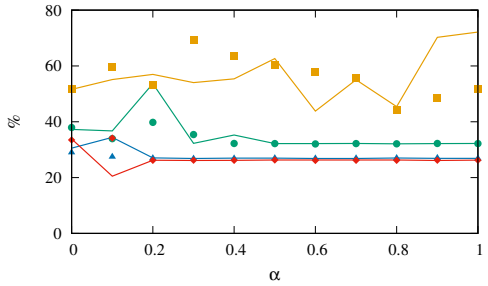


図 4.43: パターン 5・6[%]

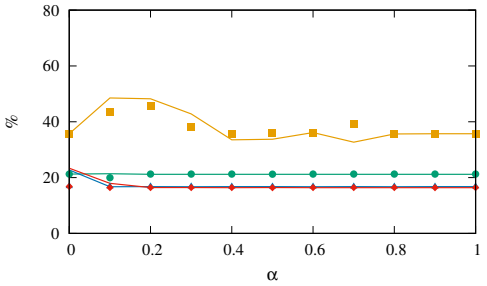
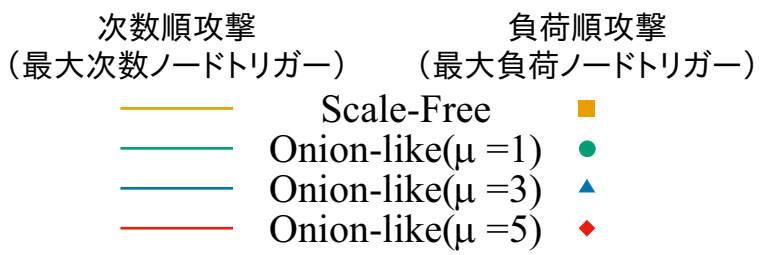


図 4.44: パターン 7・8[%]



第5章　複数の初期故障に対するカスケード故障の抑制効果

本章では, 順序制御を考慮した迂回ルーティングにおけるカスケード故障の抑制に対して, シミュレーションを通して複数の初期故障（トリガー）ノードによるカスケード故障の抑制効果を調査する.

シミュレーションの対象としたネットワークは, 4 章と同じく 2.1.1 節の BA モデルと 2.1.2 節の玉葱状ネットワークを, ノード数は $N = 10^3$, 追加リンク数は $m = 4$ で生成した.

また初期故障ノードの選択方法は, 3.2 節の Step 1-1 において, 最大次数又は最大負荷の大きい順に複数個選択する.

5.1　スケールフリーネットワークに対するカスケード故障抑制効果の評価

5.1.1　最大コスト経路優先選択

図 5.1 から 5.5 は最大コスト経路優先選択で行ったシミュレーション結果である. 図 5.3 の初期故障（トリガー）ノードが 8 個までに対しては抑制できていることが確認できるが, 図 5.4 の初期故障（トリガー）ノードが 16 個以降は連結成分がほとんど失われており, ネットワークとして機能していないことが分かる.

また傾向としては, 初期故障（トリガー）ノードが増加しても各パターンにおける特性は 4.1.1 節から変化が見られない.

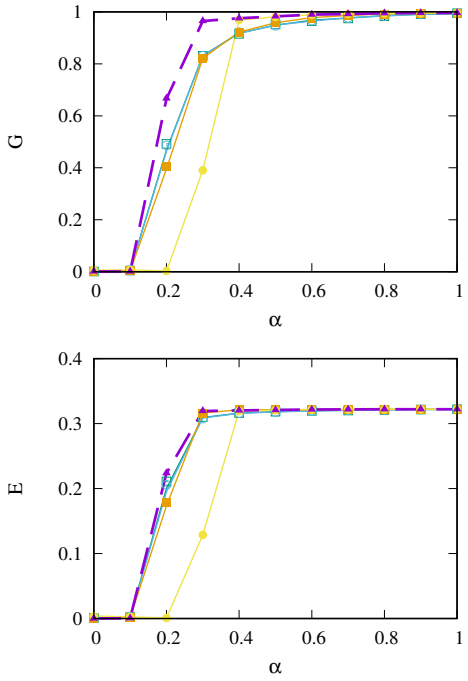


図 5.1: SF, 初期故障ノード:2 個, 最大コスト経路優先選択

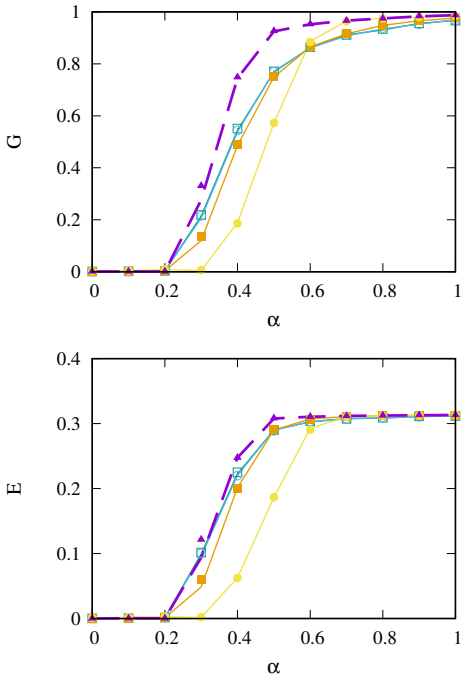


図 5.2: SF, 初期故障ノード:4 個, 最大コスト経路優先選択

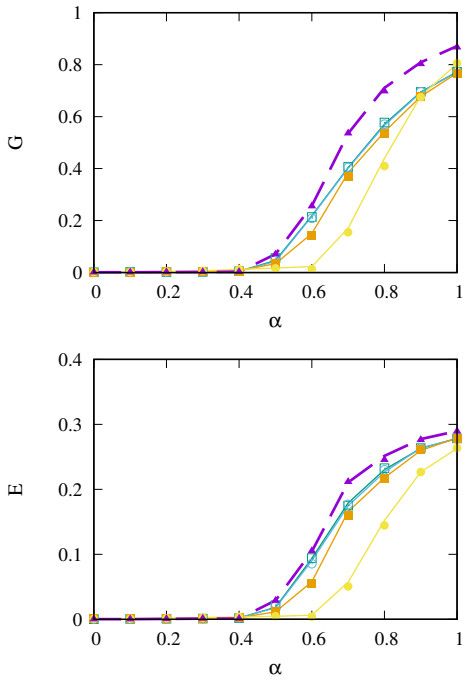


図 5.3: SF, 初期故障ノード:8 個, 最大コスト経路優先選択

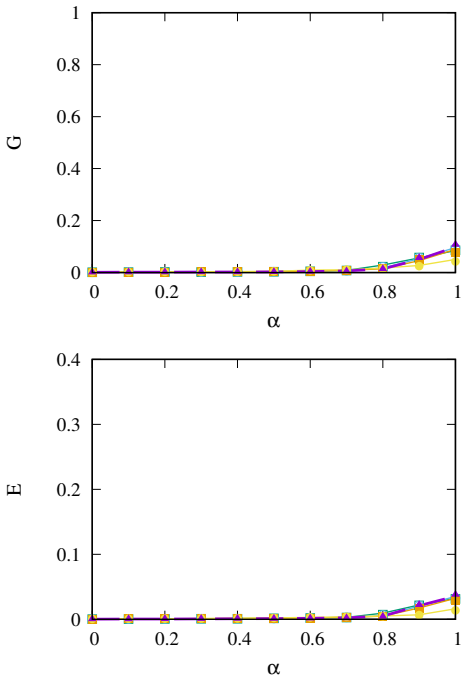


図 5.4: SF, 初期故障ノード:16 個, 最大コスト経路優先選択

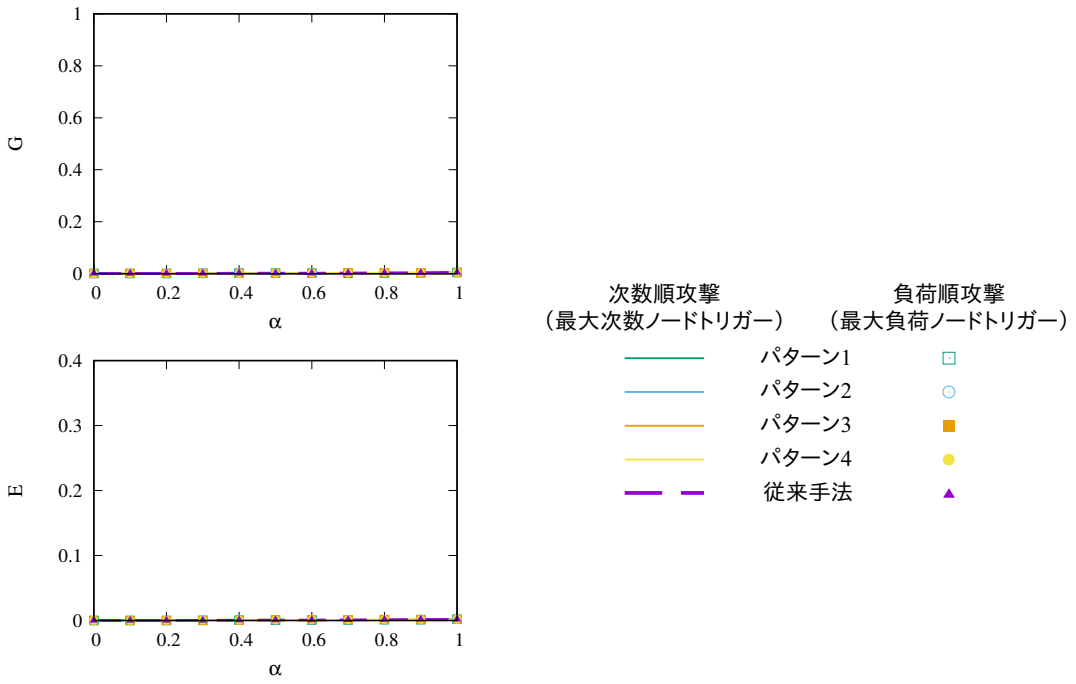


図 5.5: SF, 初期故障ノード:32 個,
最大コスト経路優先選択

5.1.2　最小コスト経路優先選択

図 5.6 から 5.10 は最大コスト経路優先選択で行ったシミュレーション結果である．最小コスト経路優先選択でも 5.1.1 節と同じく図 5.8 の初期故障（トリガー）ノードが 8 個までに対しては抑制できていることが確認できるが, 図 5.9 の初期故障（トリガー）ノードが 16 個以降は連結成分がほとんど失われており, ネットワークとして機能していないことが分かる．

また傾向としても 5.1.1 節同様に, 初期故障（トリガー）ノードが増加しても各パターンにおける特性は 4.1.2 節から変化が見られない．

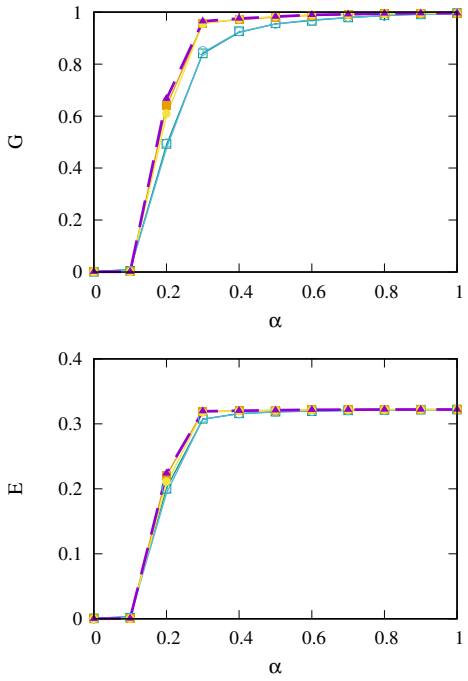


図 5.6: SF, 初期故障ノード:2 個, 最小コスト経路優先選択

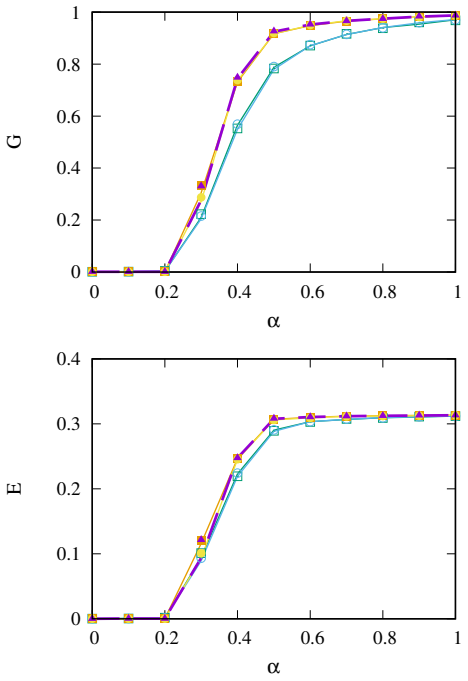


図 5.7: SF, 初期故障ノード:4 個, 最小コスト経路優先選択

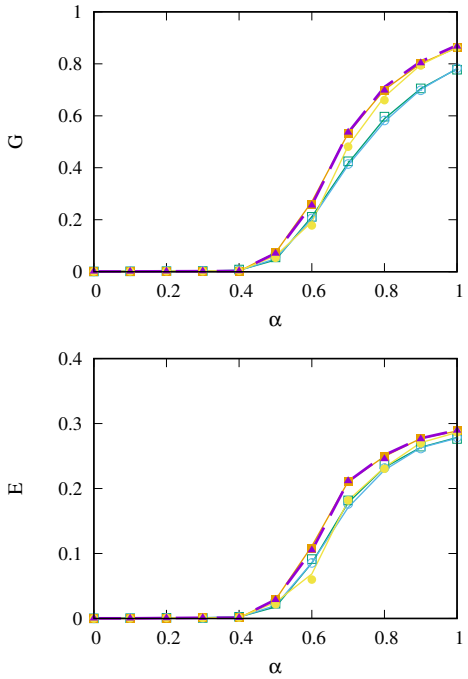


図 5.8: SF, 初期故障ノード:8 個, 最小コスト経路優先選択

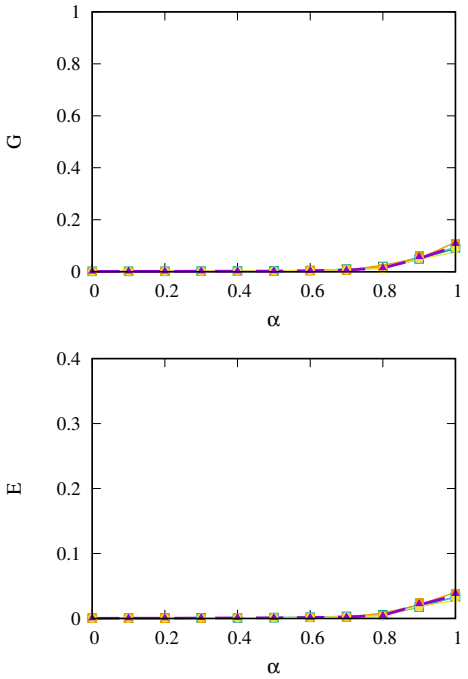


図 5.9: SF, 初期故障ノード:16 個, 最小コスト経路優先選択

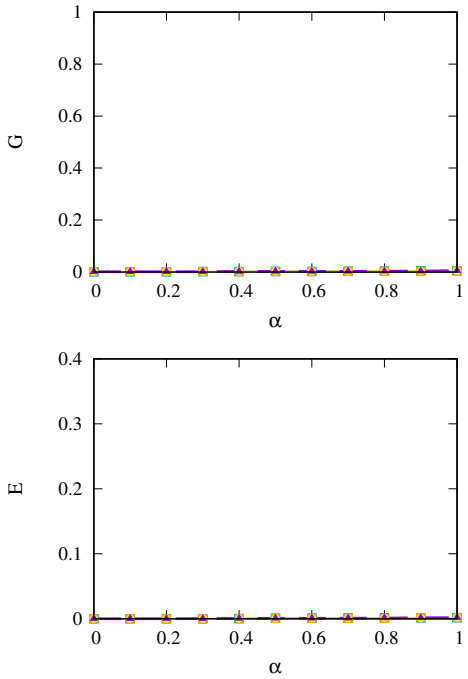
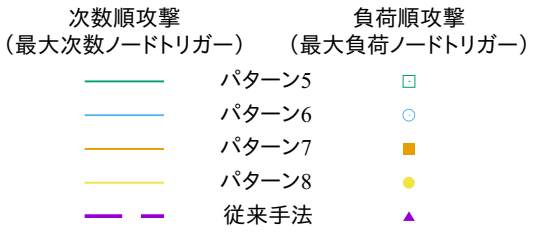


図 5.10: SF, 初期故障ノード:32 個, 最小コスト経路優先選択



5.2 玉葱状ネットワークにおけるカスケード故障の抑制効果

図 5.11 から図 5.52 は, 2.1.2 節で述べた最適耐性を持つ玉葱状ネットワーク [6] におけるカスケード故障のシミュレーション結果である. 図 5.11 から図 5.17, 図 5.32 から図 5.38 は仲介数 $\mu = 1$, 図 5.18 から図 5.24, 図 5.39 から図 5.45 は仲介数 $\mu = 3$, 図 5.25 から図 5.31, 図 5.46 から図 5.52 は仲介数 $\mu = 5$ の μ が小・中・大のネットワークである.

5.2.1 最大コスト経路優先選択

図 5.11 から 5.31 は最大コスト経路優先選択で行ったシミュレーション結果である. $\mu = 1$, $\mu = 3$, $\mu = 5$ すべてにおいて最大連結成分比 G が維持できており, カスケード故障を抑制できていることが確認できる. また, $\mu = 1$ では図 5.17 でも従来手法と同等のカスケード故障抑制効果が確認できるが, $\mu = 3$, $\mu = 5$ においては図 5.21, 5.28 の初期故障ノード数が 16 個以降で本提案よりも従来手法 [6] の方がカスケード故障の抑制効果が高いことが確認できる.

またパターン 4 については, $\mu = 1$, $\mu = 3$, $\mu = 5$ すべてにおいて特性としては良くないが, $\mu = 3$, $\mu = 5$ では初期故障（トリガー）ノードが増加するにつれて他パターンがパターン 4 に近付いていくことが確認できる.

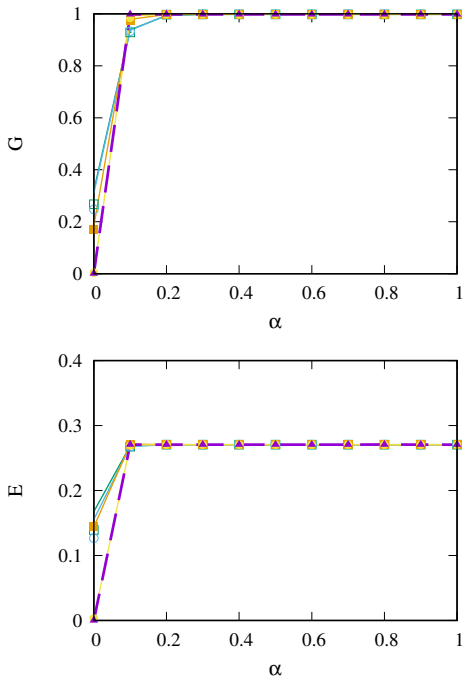


図 5.11: $\mu = 1$, 初期故障ノード:2 個, 最大コスト経路優先選択

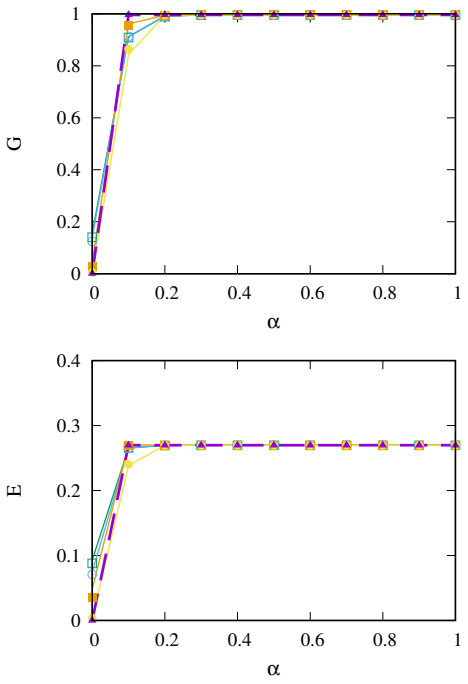


図 5.12: $\mu = 1$, 初期故障ノード:4 個, 最大コスト経路優先選択

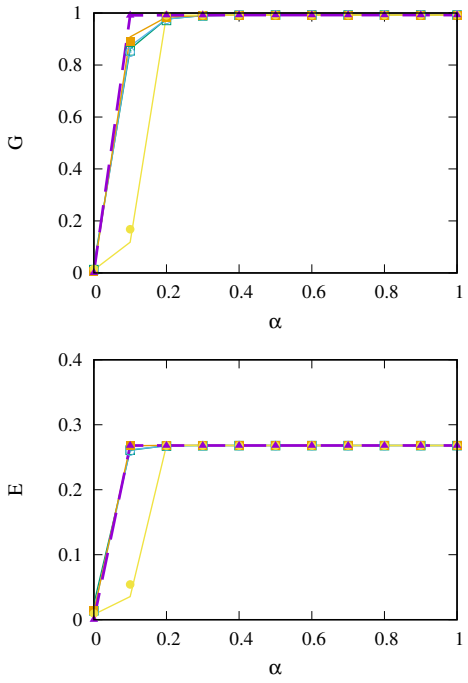


図 5.13: $\mu = 1$, 初期故障ノード:8 個, 最大コスト経路優先選択

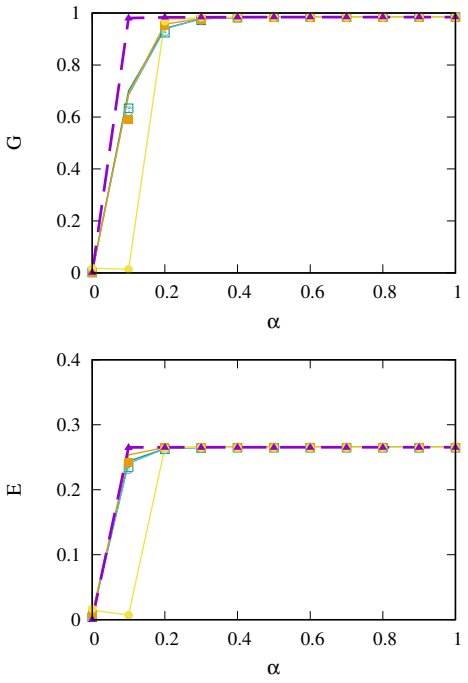


図 5.14: $\mu = 1$, 初期故障ノード:16 個, 最大コスト経路優先選択

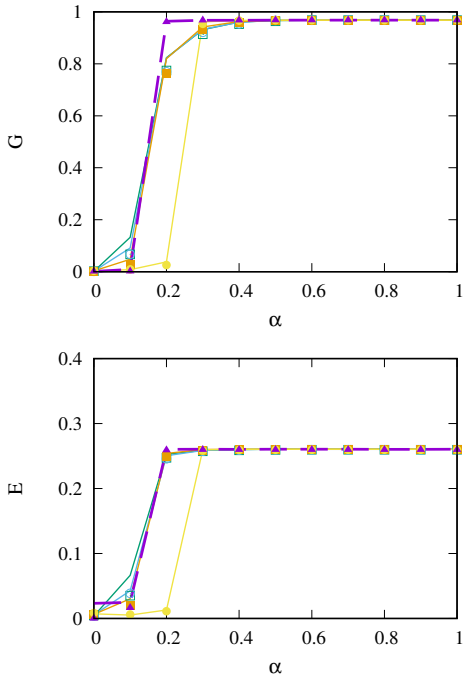


図 5.15: $\mu = 1$, 初期故障ノード:32 個, 最大コスト経路優先選択

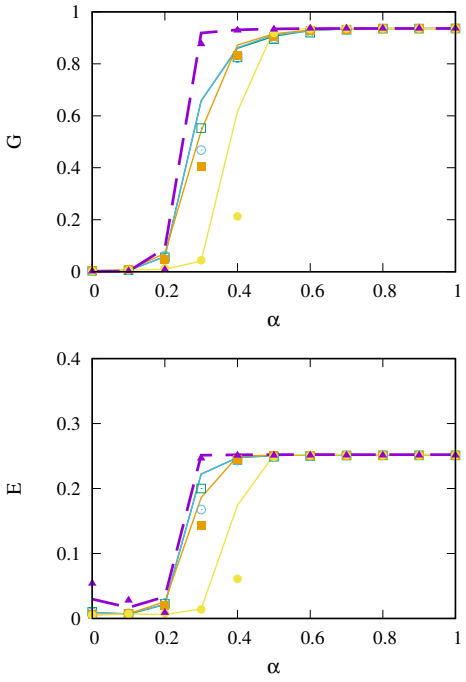


図 5.16: $\mu = 1$, 初期故障ノード:64 個, 最大コスト経路優先選択

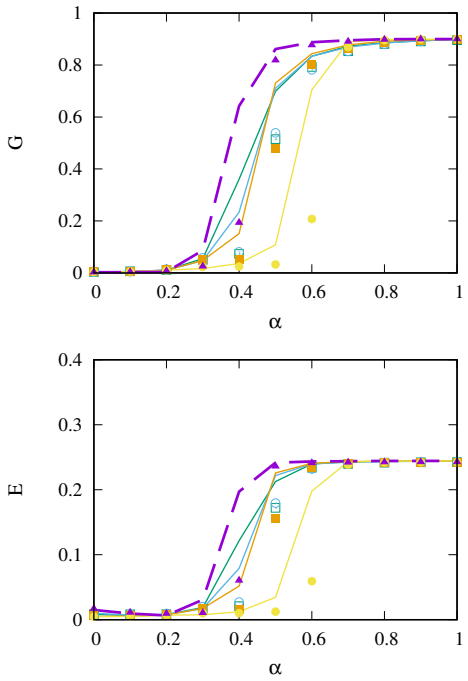
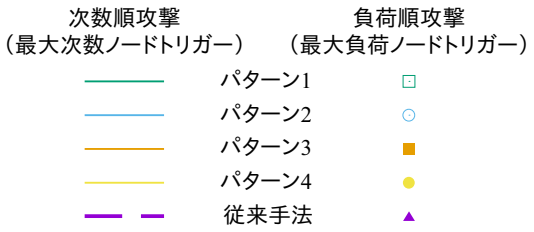


図 5.17: $\mu = 1$, 初期故障ノード:100 個, 最大コスト経路優先選択



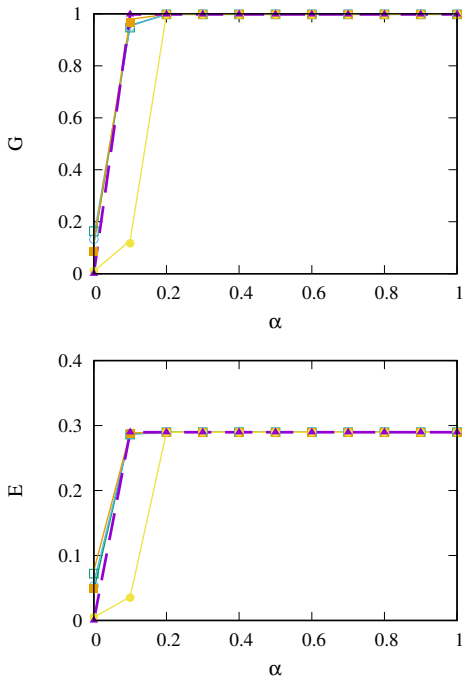


図 5.18: $\mu = 3$, 初期故障ノード:2 個, 最大コスト経路優先選択

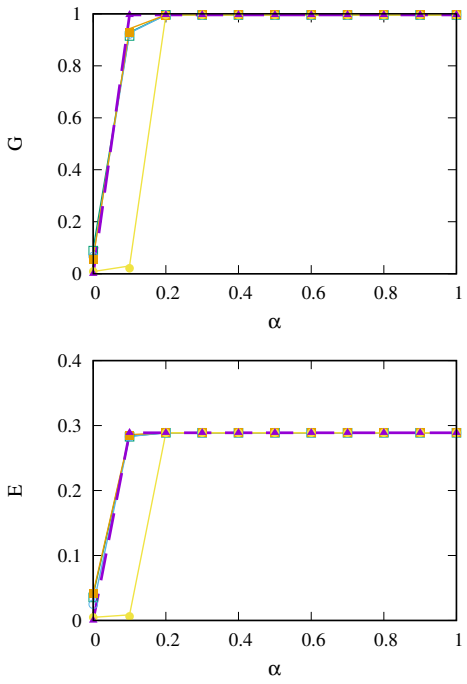


図 5.19: $\mu = 3$, 初期故障ノード:4 個, 最大コスト経路優先選択

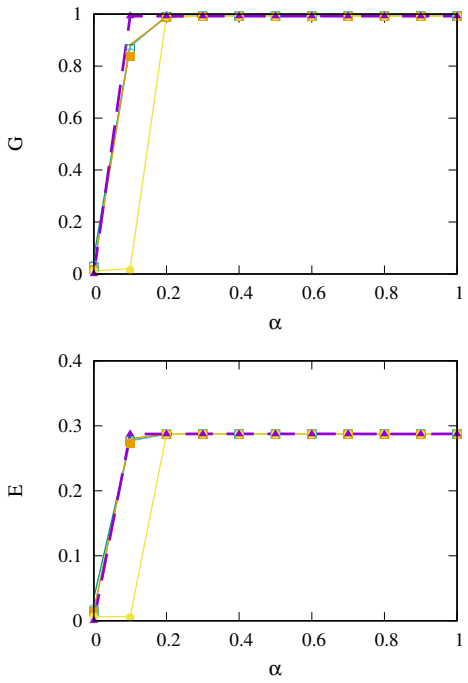


図 5.20: $\mu = 3$, 初期故障ノード:8 個, 最大コスト経路優先選択

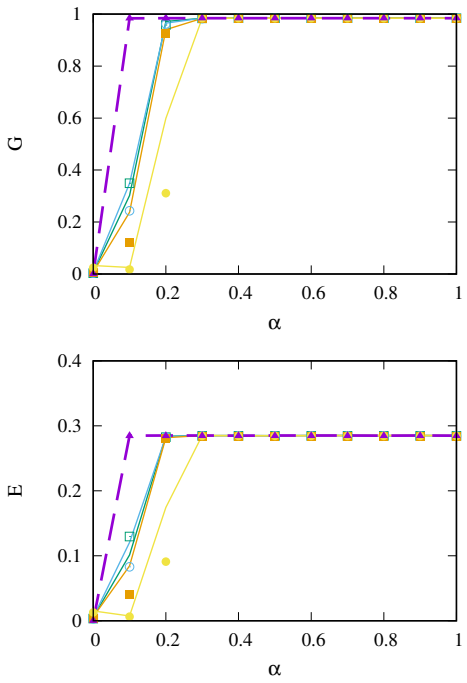


図 5.21: $\mu = 3$, 初期故障ノード:16 個, 最大コスト経路優先選択

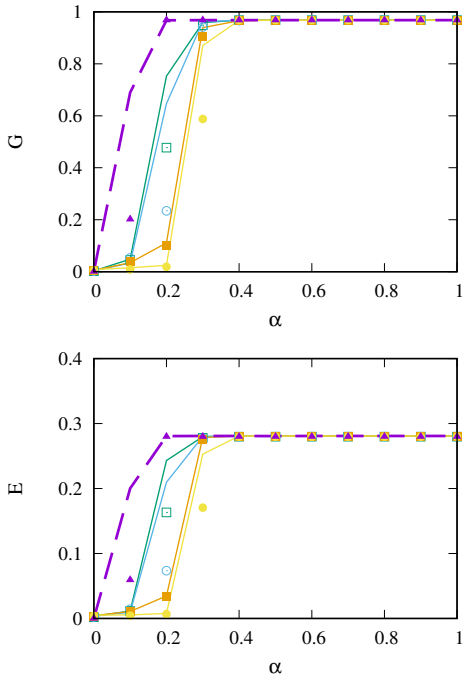


図 5.22: $\mu = 3$, 初期故障ノード:32 個, 最大コスト経路優先選択

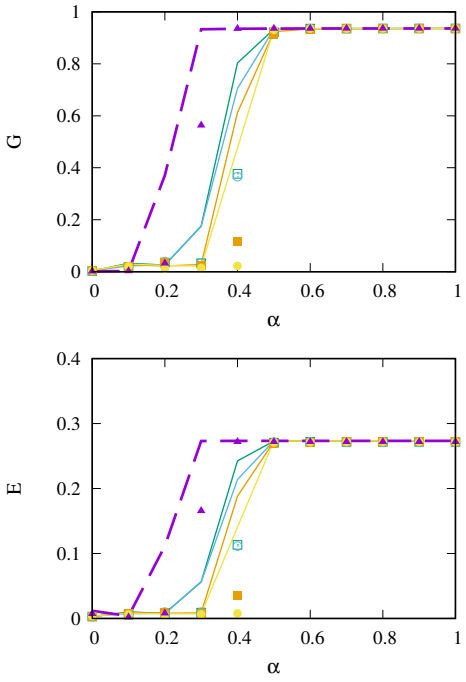


図 5.23: $\mu = 3$, 初期故障ノード:64 個, 最大コスト経路優先選択

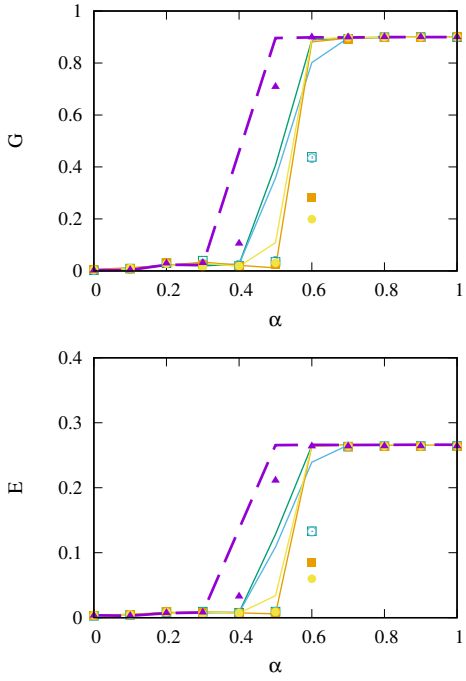
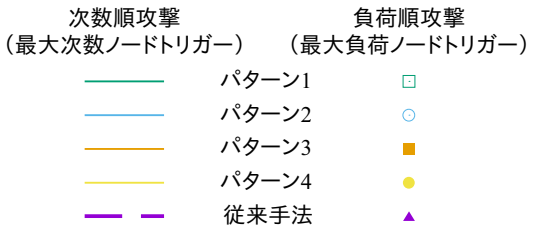


図 5.24: $\mu = 3$, 初期故障ノード:100 個, 最大コスト経路優先選択



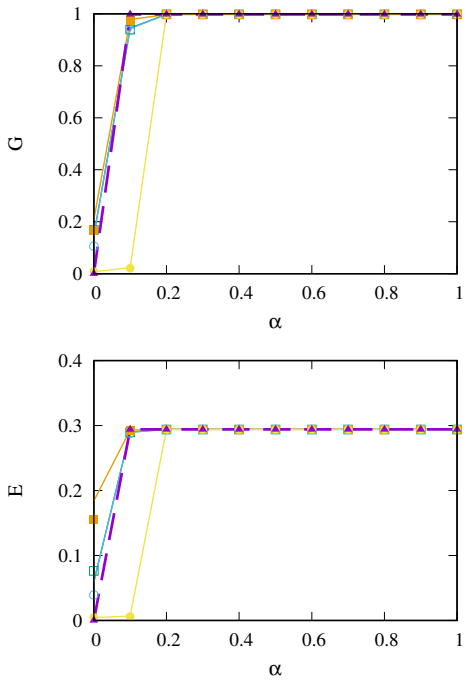


図 5.25: $\mu = 5$, 初期故障ノード:2 個, 最大コスト経路優先選択

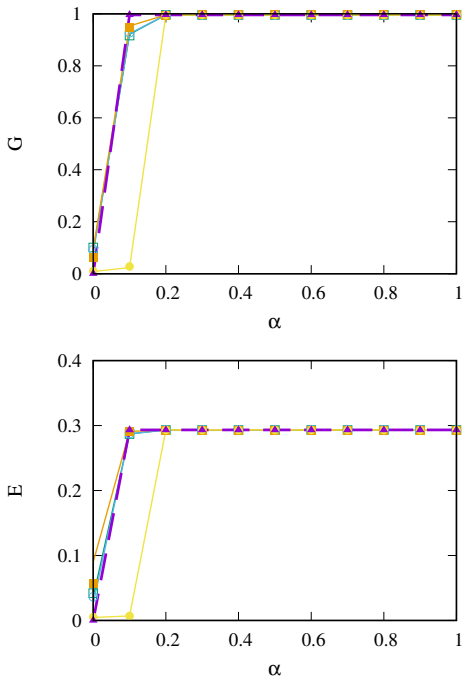


図 5.26: $\mu = 5$, 初期故障ノード:4 個, 最大コスト経路優先選択

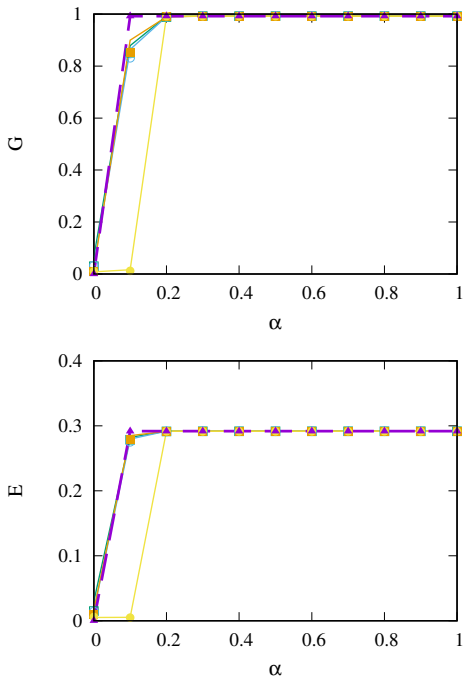


図 5.27: $\mu = 5$, 初期故障ノード:8 個, 最大コスト経路優先選択

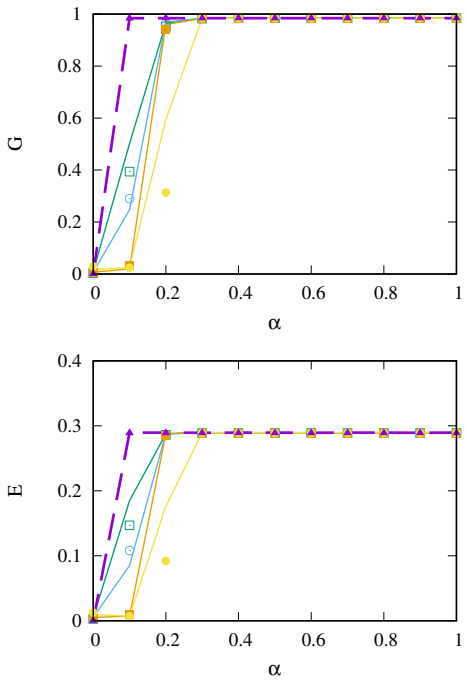


図 5.28: $\mu = 5$, 初期故障ノード:16 個, 最大コスト経路優先選択

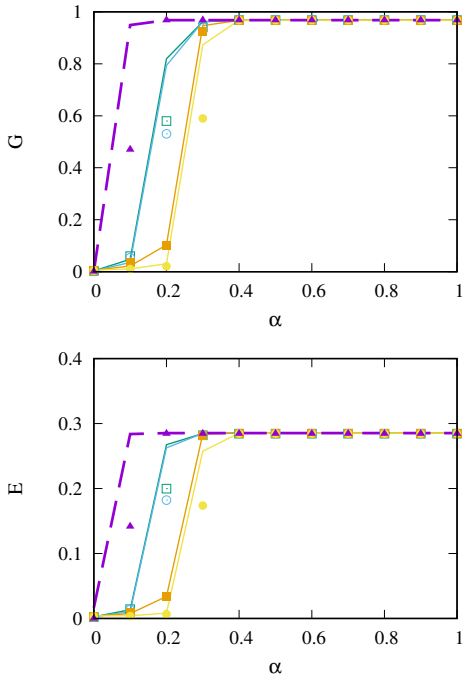


図 5.29: $\mu = 5$, 初期故障ノード:32 個, 最大コスト経路優先選択

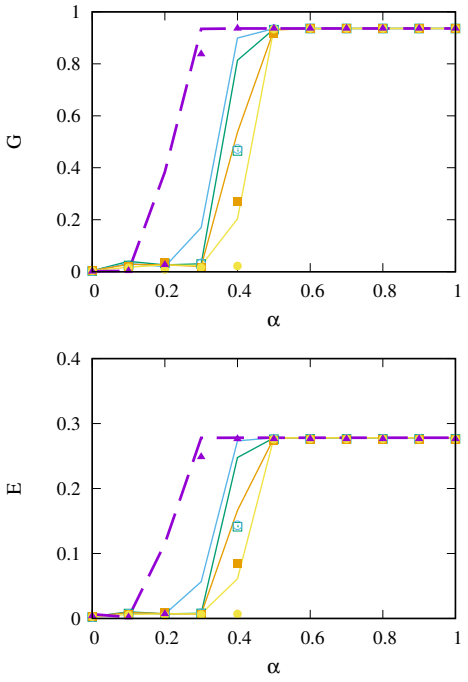


図 5.30: $\mu = 5$, 初期故障ノード:64 個, 最大コスト経路優先選択

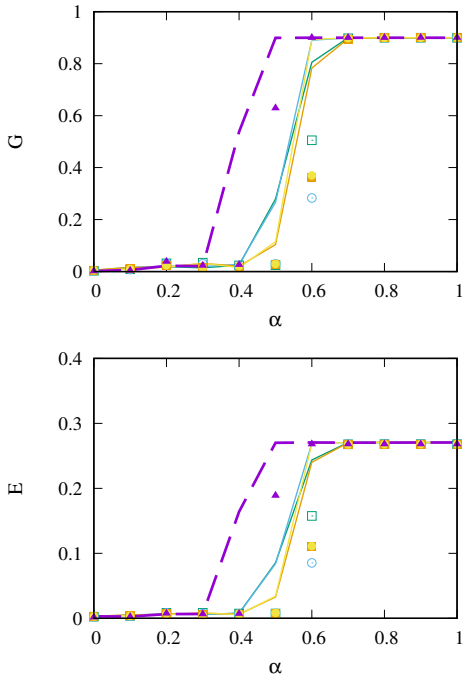
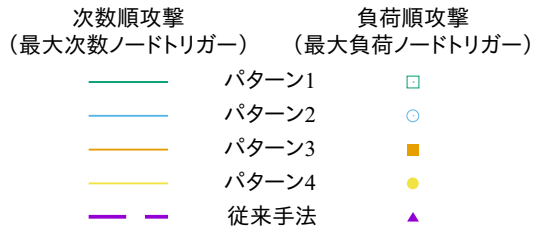


図 5.31: $\mu = 5$, 初期故障ノード:100 個, 最大コスト経路優先選択



5.2.2　最小コスト経路優先選択

図 5.32 から 5.52 は最大コスト経路優先選択で行ったシミュレーション結果である. $\mu = 1$, $\mu = 3$, $\mu = 5$ すべてにおいて最大連結成分比 G が維持できており, カスケード故障を抑制できていることが確認できる. また, 5.2.1 では初期故障ノード数が 16 個以上となると本提案ではカスケード故障の抑制が難しかったが, 最小コスト経路優先選択では図 5.38, 5.45, 5.52 の初期故障ノードが 100 個の場合でもパターン 7 においてカスケード故障の抑制効果が確認できる.

特に, 図 5.43, 5.44, 5.45 の $\mu = 3$ における初期故障ノードが 32, 64, 100 個では, パターン 7 は耐久性パラメータ $\alpha = 0.05$ で最大連結成分比 G を 9 割程度維持できており, 従来手法 [6] を上回るカスケード故障の抑制効果が確認できる.

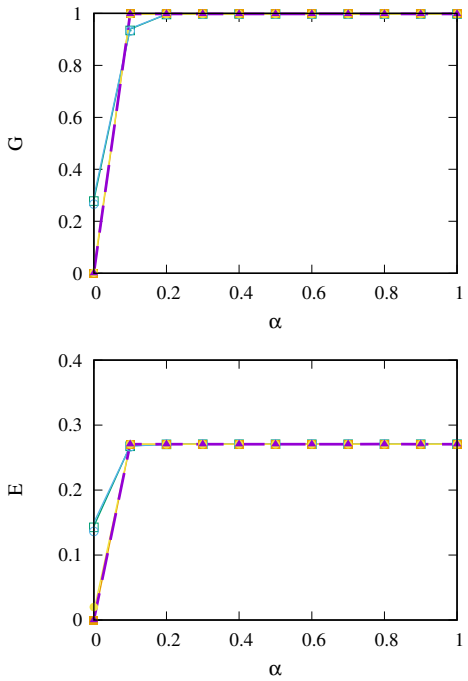


図 5.32: $\mu = 1$, 初期故障ノード:2 個, 最小コスト経路優先選択

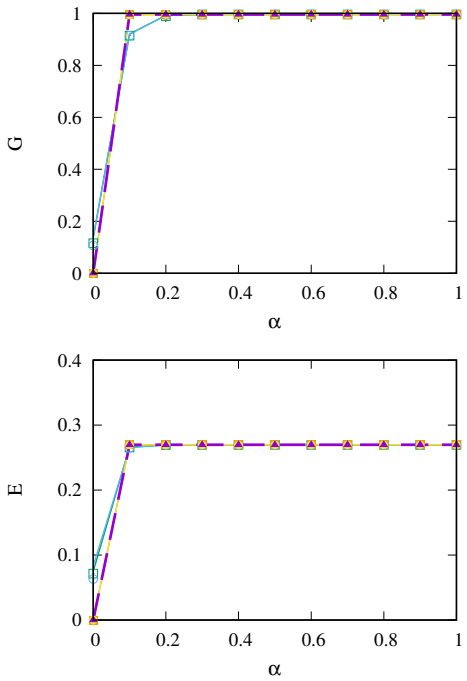


図 5.33: $\mu = 1$, 初期故障ノード:4 個, 最小コスト経路優先選択

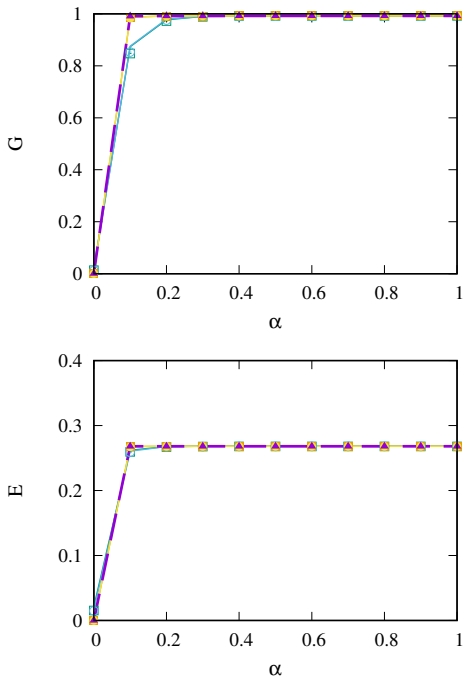


図 5.34: $\mu = 1$, 初期故障ノード:8 個, 最小コスト経路優先選択

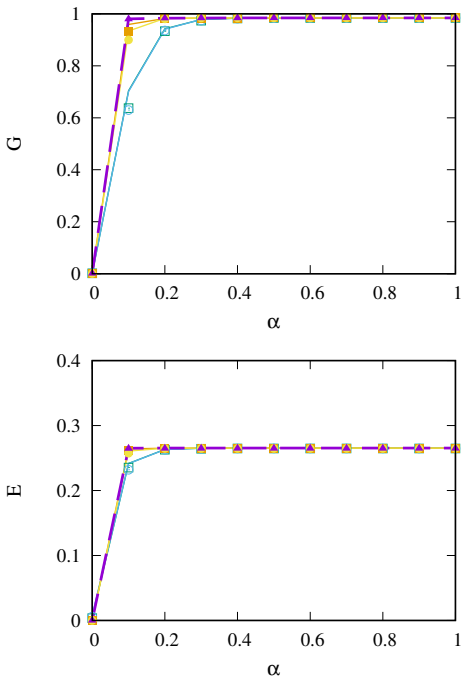


図 5.35: $\mu = 1$, 初期故障ノード:16 個, 最小コスト経路優先選択

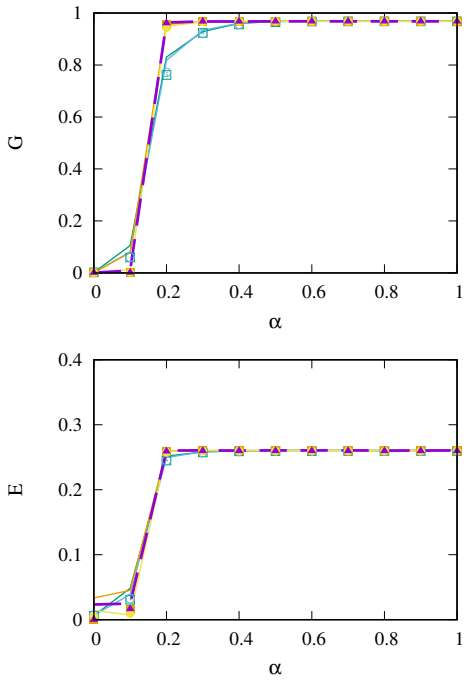


図 5.36: $\mu = 1$, 初期故障ノード:32 個, 最小コスト経路優先選択

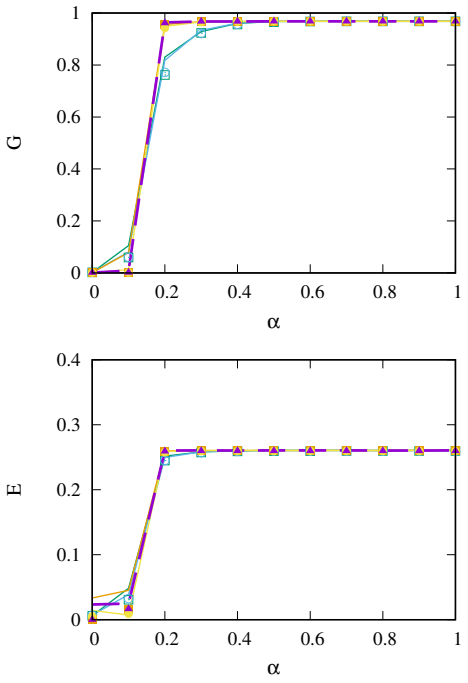


図 5.37: $\mu = 1$, 初期故障ノード:64 個, 最小コスト経路優先選択

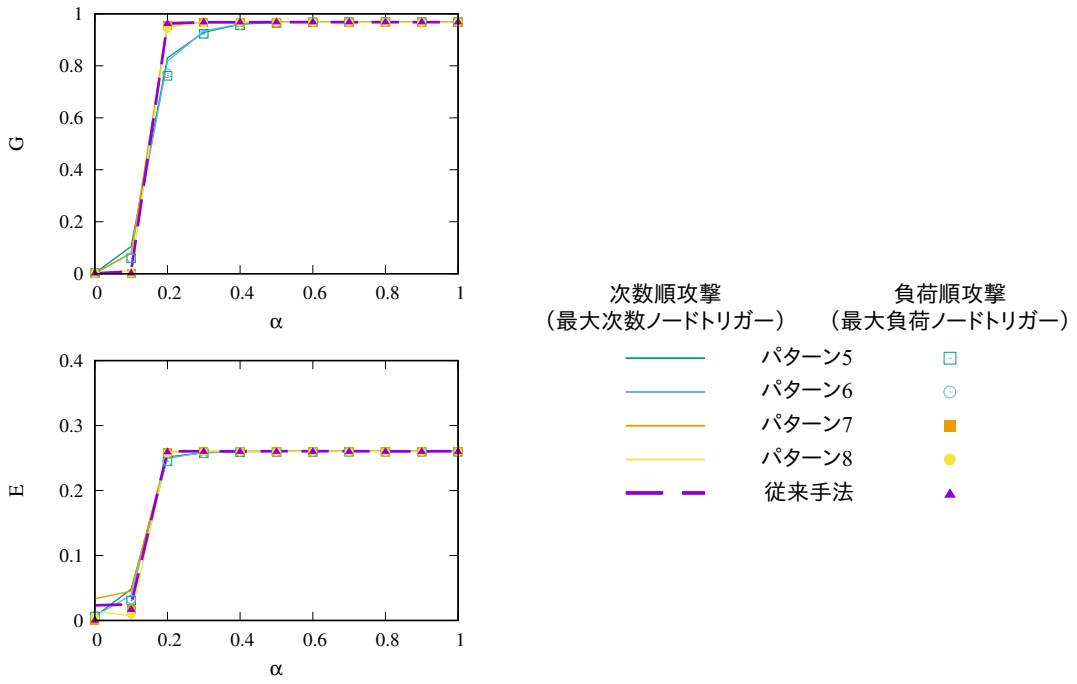


図 5.38: $\mu = 1$, 初期故障ノード:100個, 最小コスト経路優先選択

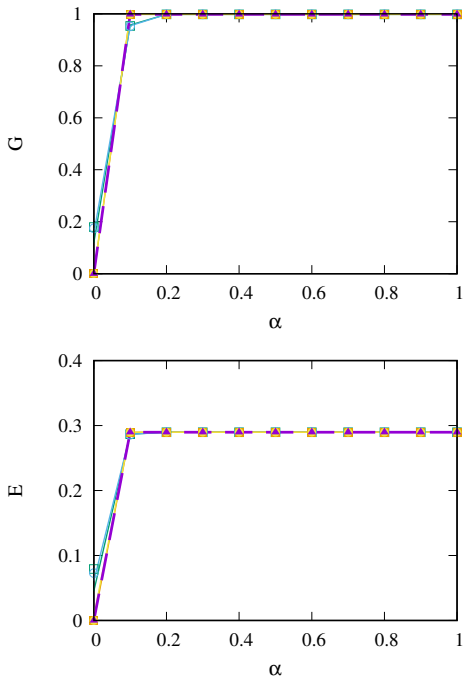


図 5.39: $\mu = 3$, 初期故障ノード:2 個, 最小コスト経路優先選択

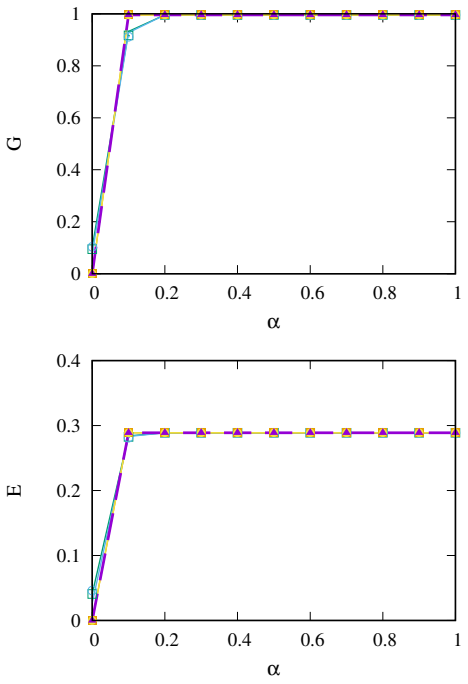


図 5.40: $\mu = 3$, 初期故障ノード:4 個, 最小コスト経路優先選択

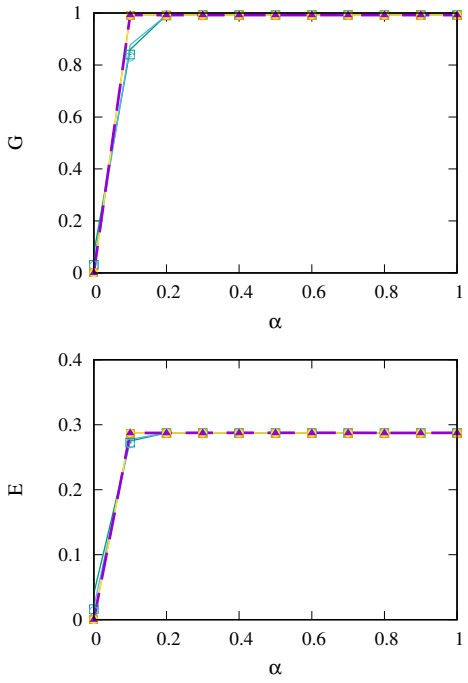


図 5.41: $\mu = 3$, 初期故障ノード:8 個, 最小コスト経路優先選択

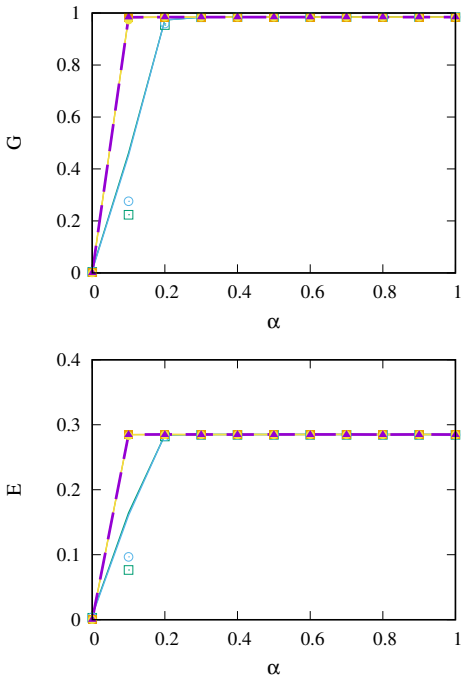


図 5.42: $\mu = 3$, 初期故障ノード:16 個, 最小コスト経路優先選択

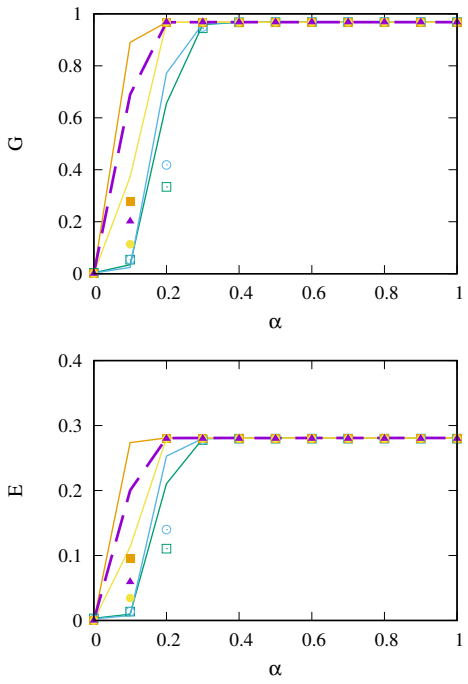


図 5.43: $\mu = 3$, 初期故障ノード:32 個, 最小コスト経路優先選択

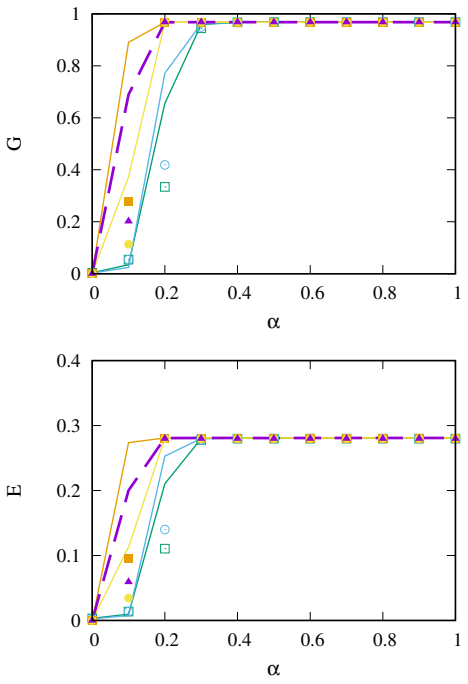


図 5.44: $\mu = 3$, 初期故障ノード:64 個, 最小コスト経路優先選択

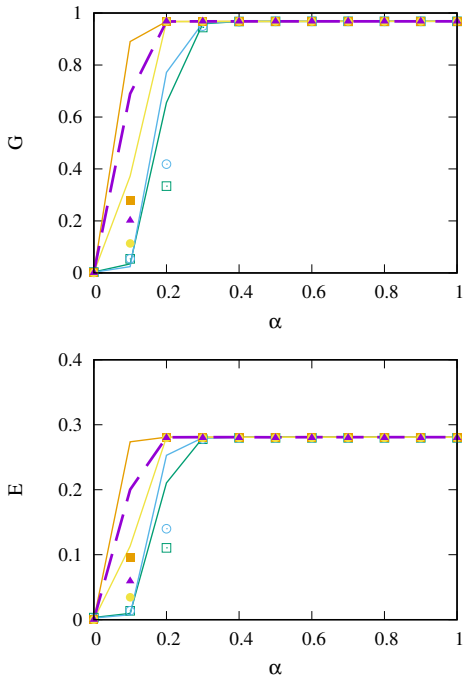
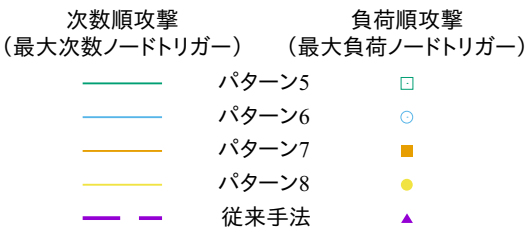


図 5.45: $\mu = 3$, 初期故障ノード:100 個, 最小コスト経路優先選択



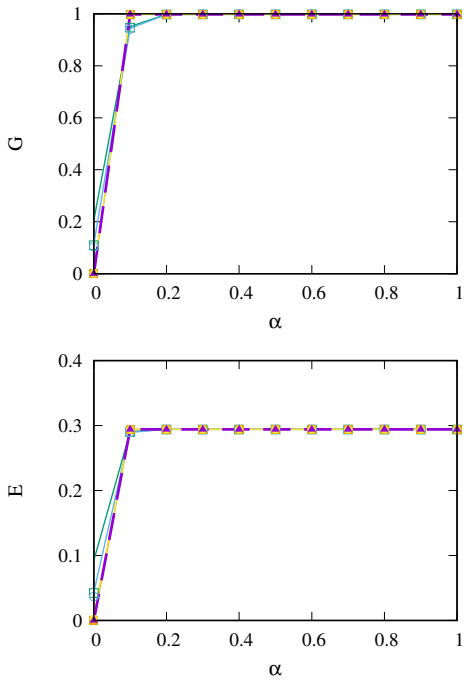


図 5.46: $\mu = 5$, 初期故障ノード:2 個, 最小コスト経路優先選択

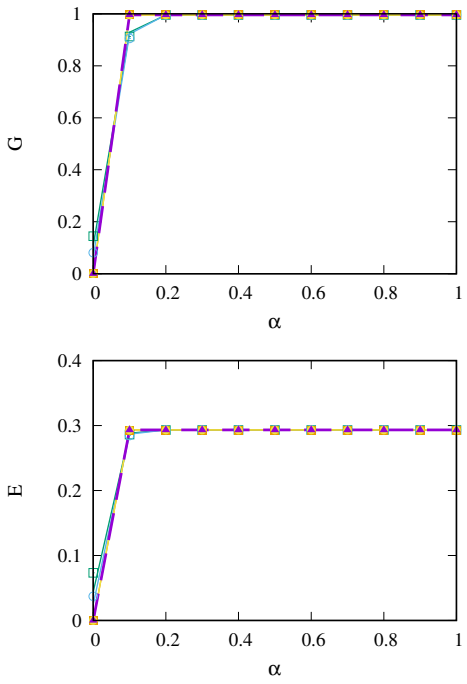


図 5.47: $\mu = 5$, 初期故障ノード:4 個, 最小コスト経路優先選択

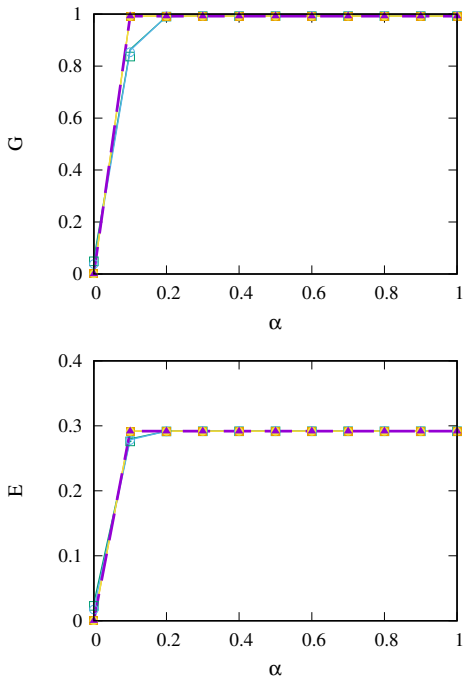


図 5.48: $\mu = 5$, 初期故障ノード:8 個, 最小コスト経路優先選択

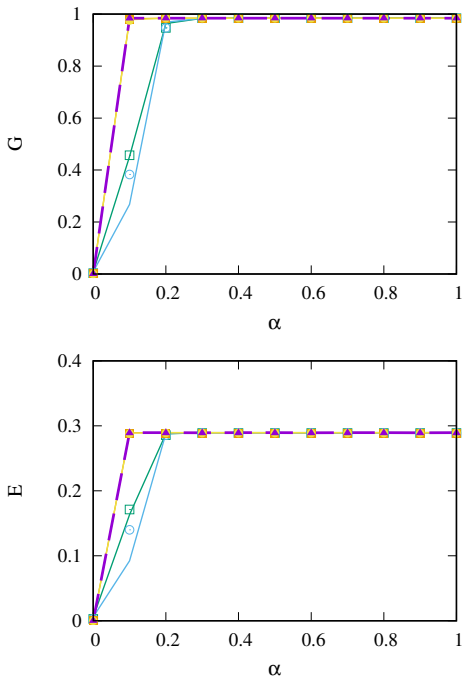


図 5.49: $\mu = 5$, 初期故障ノード:16 個, 最小コスト経路優先選択

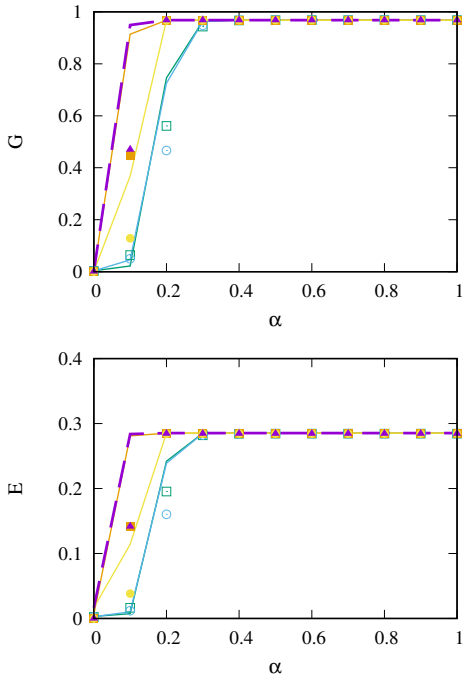


図 5.50: $\mu = 5$, 初期故障ノード:32 個, 最小コスト経路優先選択

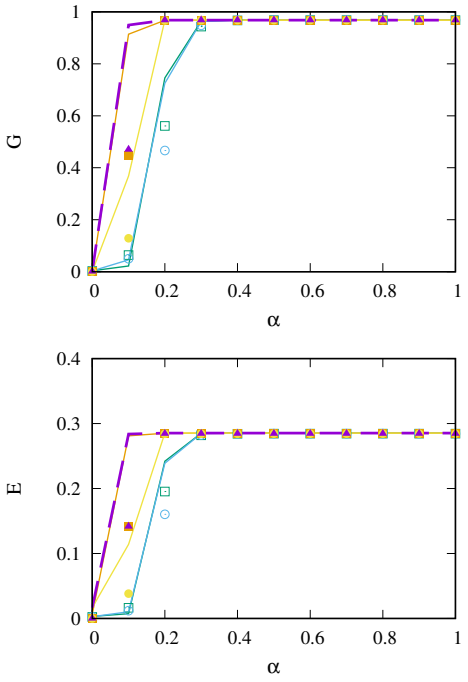


図 5.51: $\mu = 5$, 初期故障ノード:64 個, 最小コスト経路優先選択

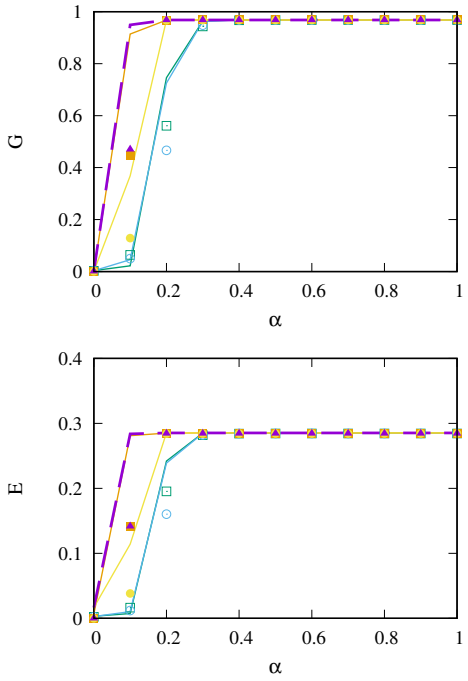
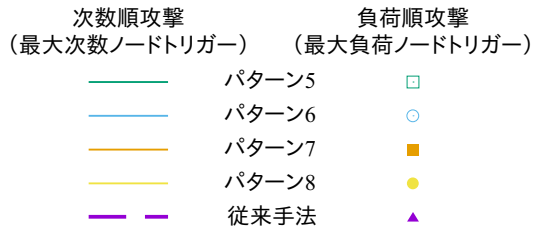


図 5.52: $\mu = 5$, 初期故障ノード:100 個, 最小コスト経路優先選択



第6章　おわりに

本論文では, 3.2.3 節の Step 3-1 における s, t の選び方の順序を考え, シミュレーションによりカスケード故障の抑制効果を調査した.

シミュレーションを行うにあたり対象としたネットワークは, 2.1.1 節の BA モデルと 2.1.2 節の玉葱状ネットワークであり, ノード数は $N = 10^3$, 追加リンク数は $m = 4$ で生成した.

以下に結果を整理する.

- 玉葱状ネットワークについては従来手法より耐久性パラメータ α が 0 に近い部分でも最大連結成分比 G を最大 4 割程度維持することができ, カスケード故障を抑制できていることが確認できた. これは, ネットワーク許容負荷の余力があまり無いという点で, 現実のネットワークを考えたときに費用等をかけずにカスケード故障の抑制が可能になることは注目すべきである.

- SF ネットワークについては玉葱状ネットワークのように α が 0 に近い部分において最大連結成分を維持することはできなかったが, パターン 1 と 2, パターン 7 と 8 において従来手法 [6] と同等の最大連結成分比 G, ネットワークの効率 E を維持できていることを確認できた. また, s, t の選び方の順序を変えるだけで 4.1 節におけるパターン 4 のような特性を示すことが判明した.

- 4.3.1 節で述べた経路探索本数の評価において, カスケード故障を抑制する効果を維持しながら経路の探索回数が最大約 80%削減できることを示した.

- 複数の初期故障（トリガー）ノードによるカスケード故障については, 初期故障ノードの数が増加しても玉葱状ネットワークにおける最小コスト経路優先選択ではカスケード故障の抑制効果が確認できた. 特に, $\mu = 3$ でのパターン 7 については初期故障（トリガー）ノードが 100 個でも $\alpha = 0.05$ で最大連結成分比 G を 9 割程度維持できている.

今後検討すべき課題として, 式 (2.3) の許容負荷では, 回線 1 本当りの最大伝送量を仮定しているが, より現実的なネットワークを考えるには 1 本 1 本それぞれ異なる伝送量とすることが考えられる. また, 3.3 節の **II** と **IV** に対しては, 順序付けではなく完全に分散した場合のアルゴリズムを考えるとすると, その同時実行制御や相互に悪影響を与え合った大規模なロールバックが生じるなど, 深刻な課題が残る.

謝辞

本研究の遂行及び論文作成に当たり, ご多忙の中多大なご指導と助言をしていただいた林 幸雄教授に深く感謝申し上げます.

最後に, 井手 勇介助教並びにお互い切磋琢磨し助け合った林研究室生の皆様方に深く感謝いたします.

平成 31 年　見雪 雄哉

参考文献

[1] Barabási, A.-L., Albert, R. Emergence of scaling in random networks, Science, 286(5439), 509-512, 1999.

[2] Hayashi, Y., Spatially self-organized resilient networks by a distributed cooperative mechanism, Physica A: Statistical Mechanics and its Applications, 457, 255-269, 2016.

[3] Hayashi, Y., A new design principle of robust onion-like networks self-organized in growth, Network Science, 6(1), 54-70, 2018.

[4] Motter, A. E., Lai, Y. C., Cascade-based attacks on complex networks. Physical Review E, 66(6), 065102, 2002.

[5] Motter, A. E. Cascade Control and Defense in Complex Networks, Physical Review Letters 93, 098701, 2004.

[6] Hayashi, Y., Uchiyama, N. Onion-like networks are both robust and resilient, Scientific Reports 8, 11241, 2018.

[7] Barabási, A.-L., Albert, R., Jeong, H. Mean-filed theory for scale-free random networks, Physica A 272, 173187, 1999.

[8] 北米大停電　ニューヨーク大停電　ニューヨーク停電　newyork blackout アメリカ・カナダ大停電の教訓.
http://www.bo-sai.co.jp/newyorkteiden.htm

[9] 2003 年 8 月 14 日 北米北東部停電事故に関する調査報告書, 北米北東部停電調査団, 2004.
http://www.enecho.meti.go.jp/category/electricity_and_gas/electric/pdf/nayousar2.pdf

[10] 平成 30 年北海道胆振東部地震に伴う大規模停電に関する検証委員会最終報告, 2018.
https://www.occto.or.jp/iinkai/hokkaido_kensho/hokkaidokensho_saishuhoukoku.html

[11] Freeman, L.C., A set of measures of centrality based on betweenness, *Sociometry*, Vol. 40, pp. 35-41, 1977.

[12] Latora, V., Marchiori, M., Efficient behavior of small-world networks, Physical Review Letters, 87(19), 198701, 2001.

[13] 落合 秀也, 最短経路問題.
http://www.hongo.wide.ad.jp/~jo2lxq/dm/lecture/08.pdf