平成 20 年度　修士論文

能動的なパケットフィルタリングを伴う制約最適化モデルによるDDoSの攻撃元推定

指導教員
松尾　啓志　教授

名古屋工業大学大学院　情報工学専攻
平成 19 年度入学　学生番号　19417594

行方　裕紀

目 次

第 1 章

はじめに

近年，不正なパケットを送ることでインターネット上のサービスを妨害する Denial of Service(DoS) 攻撃 [1] や，複数のホストから DoS 攻撃を行う Distributed DoS(DDoS) 攻撃 [2] が問題となっている．本研究では，DDoS 攻撃を攻撃に使用されるパケットの量で大きく 2 種類に分ける．

- SYN Flood 攻撃 [3] などが代表とされる，CPU やメモリなどのシステムリソースを過負荷状態・オーバーフロー状態にしたり，ネットワーク帯域をあふれさせる，大量の不正なパケットを送信するもの
- Ping of Death などが代表とされる，OS やアプリケーションの脆弱性をつく，少量の不正なパケットを送信するもの

特に脆弱性をつく攻撃に対しては，OS やアプリケーションに対してパッチの適用やバージョンアップなどによって，ほとんどの攻撃は回避できるようになるが，回避することが難しい攻撃や，新しい手法の登場によって完全な回避は困難とされている．DDoS 攻撃の根本的な対策の一つは，攻撃トラヒックの送信元を特定しトラヒックの送信を停止させることである．しかし，通常，攻撃によって送信されるパケットの送信アドレスは改竄されているため，送信元の特定は容易ではない．そこで，攻撃トラヒックの追跡により，攻撃元を特定する DDoS 攻撃追跡技術の研究が広く行われている [4] [5] [6]．DDoS 攻撃の攻撃元の検出方法としては，これまでに，発信源アドレスの偽造パケットの発信源追跡技術 (IP トレースバック技術) やトラヒックパターンを用いた手法などが提案されている．しかし，これらの手法は，大量の不正なパケットを送信する攻撃に対して効果を発揮するものである．そこで，大量のパケットを伴わない，少量のパケットによる DDoS 攻撃の検出手法が必要とされている．

本論文では，パケットの量に依存しない検出手法として，能動的なパケットフィルタリングによる攻撃元の検出手法を提案する．攻撃元の検出問題を制約最適化問題 [7] として形式化を行い，IP トレースバックで得られる一部の情報を使用することで解探索の効率化を行った．提案手法では，能動的なパケットフィルタリングを行った際に得られ

る情報を制約として保持することで，その後の解探索において探索範囲の削減を行っている．また，探索を行う単位の切替えによる探索の高速化手法を提案する．

第 2 章では，DDoS 攻撃の攻撃形態と既存手法について説明をする．第 3 章では，DDoS 攻撃の攻撃元の推定問題を定義し，問題の形式化を行う．第 4 章では，DDoS 攻撃の攻撃元の推定問題を制約最適化問題として形式化を行い，提案手法を示す．第 5 章では，シミュレーション実験を行い，提案手法の有効性を示す．第 6 章では，まとめと今後の課題を示す．

第 2 章

DDoS 攻撃に対する関連研究

本章では，DDoS 攻撃を攻撃パケットの量の観点から分類して説明する．そして，DDoS 攻撃の攻撃元の検出手法に関する研究を紹介し，それぞれの関係と有効性について述べる．また，DDoS 攻撃の攻撃形態について既存手法の問題点について述べる．

2.1 DDoS 攻撃の攻撃形態

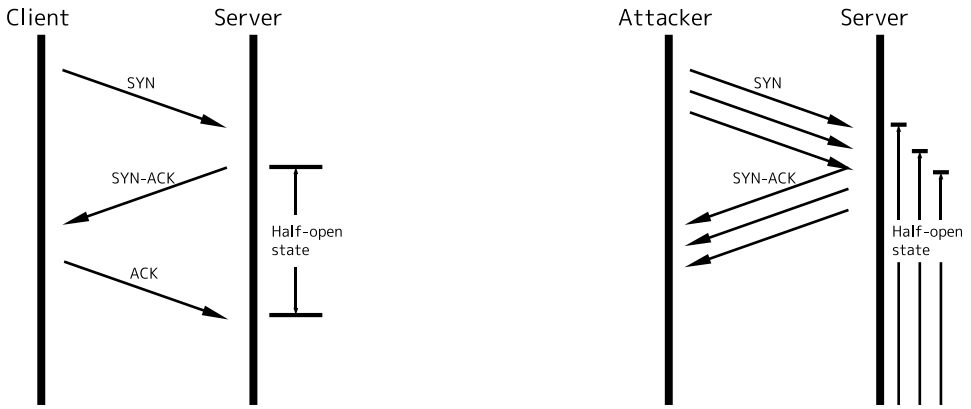
大量のパケットを送信する攻撃例として Flood 攻撃を，少量のパケットを送信する攻撃例としてセキュリティホールを衝く脆弱性を狙った攻撃を上げる．以下，個々の攻撃について説明する．

2.1.1 Flood 攻撃

Flood 攻撃は，大量のパケットを送信することにより攻撃先のネットワークデバイスやサーバに負荷をかける攻撃である．現在この種類の攻撃は，SYN Flood 攻撃，UDP Flood 攻撃，Ping Flood 攻撃，Smurf 攻撃，fraggle 攻撃，F5 攻撃などが確認されている．以下に，代表的な攻撃である SYN Flood 攻撃について示す．

SYN Flood 攻撃

TCP で接続を確立するには，図 2–1(a) に示す様に，クライアントがサーバに「SYN パケット」を送信し，サーバがクライアントに「ACK パケット」を返信し，最後にクライアントが ACK パケットを送り返すという手順を踏む．最後の ACK パケットが届くまでサーバ側は「応答待ち」状態のまま待機することになり，その接続のために用意されたメモリ領域などのリソースを開放できなくなる．SYN Flood 攻撃は，図 2–1(b) の様に，攻撃者が大量の SYN 要求を送信し，わざと ACK パケットを送らずに放置する．サーバ側は「応答待ち」の接続数が限界を超え，新たに接続を受け付けられない状態になる．この攻撃では，大量の SYN パケットが長時間にわたって被害者に送信される．



(a) スリーウェイハンドシェイク　　(b) SYNFlood

図 2–1　通常時と SYN Flood 攻撃時のサーバーの状態

2.1.2　脆弱性を狙った攻撃

脆弱性を狙った攻撃は，メモリリソースを消費するもの，プロトコルスタックのセキュリティホールに対して攻撃を行うものなど，プログラムの脆弱性を狙ったものである．この攻撃の特徴としては，システム資源を占拠する Flood 攻撃と違い，攻撃に使用するパケット数が非常に少ない．現在この種類の攻撃は，Ping of Death 攻撃，Land 攻撃，Teardrop 攻撃などが確認されている．以下に，代表的な攻撃である Ping of Death 攻撃，Land 攻撃について示す．

Ping of Death 攻撃

Ping of Death 攻撃は IP 実装に関するバグを衝く攻撃で，RFC 792 で定義された ICMP Echo パケットの最大サイズ (65536 バイト) より大きいサイズを送信する．OS やルータなどは，IP パケットの最大サイズよりも大きなパケットを受け取ると，バッファ・オーバーフローを引き起こしてしまう．一般的に 65,536 バイトの ICMP Echo パケットを送ることはネットワークプロトコルに違反するものであるため，分割して送信する．被害者はパケットを組み立てている時にバッファ・オーバーフローを起こす．このバグを抱えているホストは攻撃を受けるとフリーズする可能性がある．この攻撃では，少量の ICMP パケットが被害者に送信される．

Land 攻撃

Land 攻撃では，発信元の IP アドレスとポート番号を送信先のものと同じものに詐称した SYN パケットを送りつける．受信先がこの SYN パケットを受け取ると，送信元のアドレスに対して，ACK/SYN パケットを返信しようする．ところが送信元アドレスは，受信先 (つまり自分自身) のアドレスに細工されているため，自分自身に対して返信してしまうことになる．これにより，システムの負荷が高まり，最終的には応答不能になってしまう．この攻撃では，少量の SYN パケットが被害者に送信される．

2.2 DDoS 攻撃に対する既存手法

DDoS 攻撃に対する既存手法を紹介する．まず侵入検知システムとパケットフィルタリングを紹介し，本研究の目標である攻撃元の検出との関係について言及する．また攻撃元の検出については，IP トレースバック技術として ICMP 方式，パケットマーク方式，また，トラヒックパターンを用いた手法の 3 方式を挙げ，それぞれの特徴を述べる．

2.2.1 侵入検知システム

侵入検知システム (以下，IDS) は，ホストへの不正アクセス等の侵入を検知するシステムである [8]．一般に IDS は，既存の侵入のトラヒックパターンをまとめた，シグニチャと呼ばれるファイルと，実際のトラヒックパターンをバイナリマッチングさせて侵入を検知する．一方，IDS では DDoS 攻撃自体の検知を目的としており，攻撃元を特定して原因を突き止めることは困難である．本研究では，DDoS 攻撃自体の検知は IDS の利用を想定し，攻撃元の検出に焦点をおく．

2.2.2 パケットフィルタリング

パケットフィルタリングは，ルータやファイアウォールが持っている機能の一つで，送られてきたパケットを検査して通過させるかどうか判断する機能である．パケットのヘッダにはプロトコルや送信元アドレス，送信先アドレスやポート番号などの情報が含まれており，これを参照して通過するかどうかが決定される．どのような方針に基づいて判断するかは，そのネットワークの管理者が任意に設定することができる．最も一般的かつ簡便なセキュリティ技術として知られており，最近のルータは大半が持っている

機能だが，よく知られているだけにパケットフィルタリングを回避する手段も多く，他の技術と併用することが肝要である．

2.2.3 ICMP 方式

DDoS 攻撃の攻撃元の検出手法としては，ITRACE [9] がある．ITRACE では，ルータがパケットを中継した際に，低い確率で送信先アドレスに ICMP メッセージを送信する．攻撃パケットが多い DDoS 攻撃ならば，全ての中継したルータからメッセージが届くはずである．しかし，ITRACE の ICMP メッセージの中継は，ルータの性能に依存するため，一部のルータでリンクの負荷が高い場合正常に処理されない場合がある．また，攻撃パケットが少ない DDoS 攻撃では全ての中継したルータからメッセージが届く確立は低い．また，ITRACE では送信する ICMP メッセージ分のパケットを増やして攻撃元の検出をするにもかかわらず，ICMP パケットを拒否するサーバが多いため，正常に動作しない場合がある．

2.2.4 パケットマーク方式

別の検出手法として，中継したルータの IP アドレスをパケットにマークする方式が考えられる．IP には，IP Record Route オプションがあり，ルータのアドレスを順次オプションヘッダに追加できる．しかしオプションヘッダには 20 バイトという上限があり，9 個のルータアドレスをパケットに追加するとそれ以上アドレスを追加できなくなる．この問題を解決するパケットマーク方式の改良として，確率的パケットマーク方式 [10] が提案された．確率的パケットマーク方式では，被害者を根とするルータ有向グラフの各辺を確率的にサンプリングする．確率的パケットマーク方式では，全ての経路上の中継ルータのサンプルが必要であり，攻撃パケットが少ない DDoS 攻撃では全てのサンプルを集めることが難しくなる．

2.2.5 トラヒックパターンを用いた手法

ネットワークに観測点を設置し，異常なトラヒックの動向をとらえ，DDoS 攻撃の攻撃元を検出する手法 [11] [12] が提案されている．この手法では，ネットワークのある地点に複数の攻撃トラヒックが入力され，合流したトラヒックが別のリンクから出力される場合に，入力リンクと出力リンクのパターンの比較過程を二次計画問題として扱う．

そして，その二次計画問題を解くことで各入力トラヒックが出力トラヒックに含まれている割合を算出し，その割合に基づいて攻撃に関連する入力リンクを決定する．パケット数の観測は，各パケットの情報を抽出する必要がないため，負荷が軽く，高速なネットワークでも観測が容易である．これは，一般的には MIB（Management Information Base）II としても実装されており，SNMP（Simple Network Management Protocol）によって容易に得られる情報である．しかし，トラヒックパターンを用いる手法では，攻撃パケットが少ない DDoS 攻撃では，攻撃トラヒックが他のトラヒックに埋もれてしまい攻撃リンクの判定を誤りやすい．

2.2.6 既存手法の対象とする DDoS 攻撃の攻撃形態と問題点

以上をふまえ，関連研究の分類を表 2–1 に示す．(○…良い，△…悪い，×…不可)

表 2–1 関連研究の分類

評価項目	IDS	フィルタ	ICMP	マーク	トラヒック
攻撃元の検出	×	×	○	○	○
攻撃パケットの量 (多)	×	×	○	○	○
攻撃パケットの量 (少)	×	×	△	△	△
検出した情報の正当性	×	×	○	○	×

IDS とフィルタは攻撃元の検出を目的としていないので，むしろ攻撃元の検出とは共存する関係にある．他の 3 つの攻撃元の検出方式については，各攻撃形態による手法の有効性という観点で以下に述べる．ICMP 方式，パケットマーク方式，トラヒックパターンを用いた手法では，攻撃パケットが多い DDoS 攻撃では検出精度が高い．しかし，攻撃パケットが少ない DDoS 攻撃では，ICMP 方式，パケットマーク方式では，全てのサンプルを集めることができない可能性があり，トラヒックパターンを用いた手法では，攻撃リンクの判定を誤りやすい．

DDoS 攻撃発生時に検出した情報に関しては，トラヒックパターンを用いた手法では，攻撃リンクの判定の合否が確認できないことから正確な情報を得ることができない．

本研究では，攻撃パケットの量に依存しない DDoS 攻撃の検出手法を目標とし，攻撃元の検出を行う際に，パケットフィルタリングを用いる検出手法を提案する．また，攻撃元の IP ネットワーク上での位置を推定する手法として，検出した情報の正当性の面

からIPトレースバックを用いる．次章に，能動的なパケットフィルタリングによる攻撃元の検出方法とIPトレースバックによる攻撃元の位置の推定方法を示す.

第 3 章

DDoS攻撃の攻撃元検出問題の設定と形式化

本章では，DDoS 攻撃の攻撃元の検出問題を示し，その問題の形式化，および，攻撃元の推定に用いる情報の形式化を行う．そして，能動的なパケットフィルタリングを用いた検出手法を提案する．パケットフィルタリングでは，あるリンクにおいて被害者へ送信されるの全てのパケットを遮断するため，攻撃パケットの量に依存しない．

3.1 問題設定

IP ネットワーク上で発生する DDoS 攻撃を想定し，攻撃元を能動的なパケットフィルタリングによって推定する問題を考える．以下に問題設定を示す．

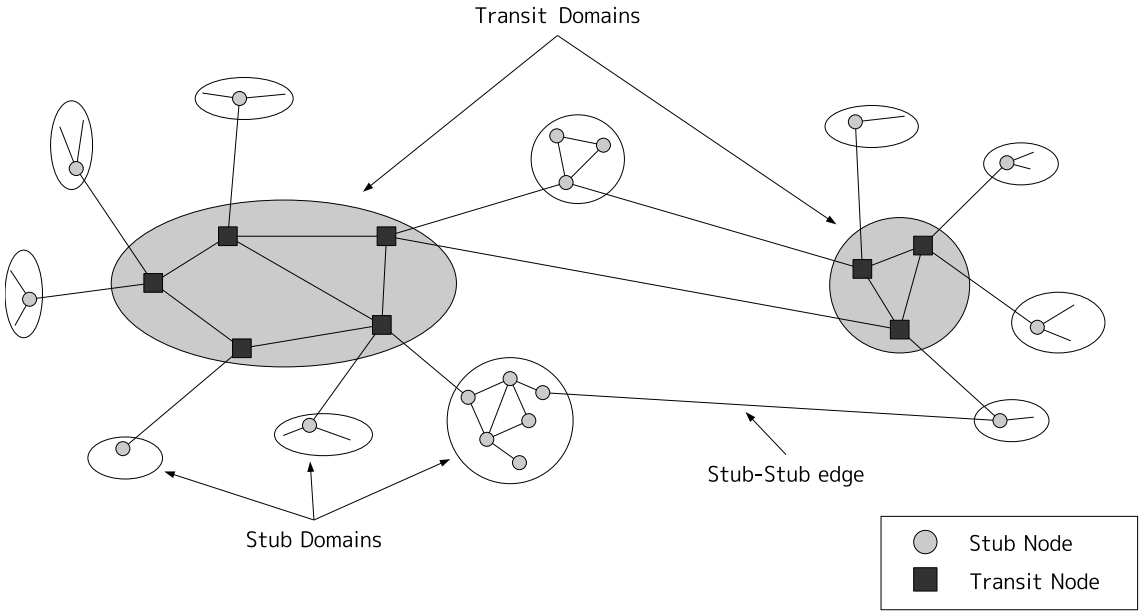


図 3–1　Transit-Stub 型の IP ネットワークのトポロジの例

- IP ネットワークのトポロジは，複数の Transit ノード (ルータを想定) とそれにつ

ながる複数の Stub ノード (端末を想定) によって構成されるトポロジを想定する (図 3–1 に Transit-Stub 型の例を示す)

- 攻撃元は複数の Stub ノードとする
- 被害者は単一の Stub ノードとする (被害者≠攻撃元)
- 攻撃元は特定の経路を使用して被害者に攻撃パケットを送信する
- 被害者が特定の攻撃パケットを受け取ると ”被害者は攻撃を受けている ”とする
- 2 種類の攻撃パターンを想定する

(a) 複合型　　(b) 単一型

図 3–2　攻撃パターン

  - 複合型:
    図 3–2(a) に示すように，複数の攻撃元のうち，全ての攻撃元の攻撃パケットを被害者が受け取ると被害者は攻撃を受ける
    (任意の攻撃元の攻撃パケットを遮断すると攻撃は止まる)
  - 単一型:
    図 3–2(b) に示すように，複数の攻撃元のうち，単一の攻撃元の攻撃パケットを被害者が受け取ると被害者は攻撃を受ける
    (全ての攻撃元の攻撃パケットを遮断すると攻撃は止まる)

- パケットヘッダを確認することで, あるリンク上における特定のデスティネーションノードへの全てのパケットを遮断することができる (パケットフィルタリング)
- 被害者が攻撃を受けているかどうかは全てのノードで観測できる
- 全ての Stub ノードは特定の経路を使用して被害者と通信を行う
- ルーティングはルーティングテーブルに従い，経路の迂回・ソースルーティングは考慮に入れない

上記の問題設定を用いて，全ての攻撃元である Stub ノードの推定を行う．このとき，攻撃元でない Stub ノードが選ばれてはいけない．次節に，能動的なパケットフィルタリングを用いた攻撃元の検出方法を示す．

3.2 能動的なパケットフィルタリングを用いた攻撃元の検出

攻撃元の検出を行うにあたって，”パケットフィルタリングを行う位置 ”と ”遮断するパケットの種類 ”を決定し，被害者が攻撃を受けているかどうかを観測する．本手法では，パケットフィルタリングによって得られる情報を使用することで攻撃元の推定を行う．得られる情報を表 3–1，表 3–2 に示す．

表 3–1　攻撃パターンが複合型の場合に得られる情報

条件	被害者への攻撃が止まる．	被害者への攻撃が止まらない．
得られる情報	パケットフィルタリングによって被害者との通信経路が失われている Stub ノードの集合に，攻撃元を含む	パケットフィルタリングによって被害者との通信経路が失われていない Stub ノードの集合に，攻撃元を全て含む

表 3–2　攻撃パターンが単一型の場合に得られる情報

条件	被害者への攻撃が止まる．	被害者への攻撃が止まらない．
得られる情報	パケットフィルタリングによって被害者との通信経路が失われている Stub ノードの集合に，攻撃元を全て含む	パケットフィルタリングによって被害者との通信経路が失われていない Stub ノードの集合に，攻撃元を含む

この時，上記の情報より，以下の条件を満たすパケットフィルタリングを求めることで，過不足のない攻撃元の推定が行える．

- 攻撃パターンが複合型の場合の条件

  目的　　パケットフィルタリングによって被害者との通信経路が失われている Stub ノードの数が最大となる．

  制約条件　被害者への攻撃が止まらない．

- 攻撃パターンが単一型の場合の条件

  目的　　パケットフィルタリングによって被害者との通信経路が失われていない Stub ノードの数が最大となる．

  制約条件　被害者への攻撃が止まる．

上記の条件を満たすとき，複合型では，パケットフィルタリングによって被害者との通信経路が失われていない Stub ノードの集合が攻撃元の集合となる．単一型では，パケットフィルタリングによって被害者との通信経路が失われている Stub ノードの集合が攻撃元の集合となる．

3.3 問題の形式化

攻撃元を能動的なパケットフィルタリングによって推定する問題に使用する記号を以下に示す.

- ノード集合　$N := \{n_i | i = 1, ..., M\}$　　M はノード数
  - Stub ノード集合　　$N^{Stub} \subset N$
  - Transit ノード集合　$N^{Transit} \subset N$
    $(N = N^{Stub} \cup N^{Transit}, \phi = N^{Stub} \cap N^{Transit})$
- リンク集合　　$L := \{l_{i,j} | n_i, n_j \in N, i \neq j\}$　$(l_{i,j} = l_{j,i})$

パケットフィルタリングでは，パケットの IP ヘッダを確認し，指定するデスティネーションノードへのパケットを遮断する. 使用する記号を以下に示す.

$N_{l_{i,j}}^{filter} (\subset N^{Stub})$：リンク $l_{i,j}$ において，パケットフィルタリングを行う際に指定するディスティネーションノードの集合

$N^{PossibleToCommunicate} (\subset N^{Stub})$：被害者と通信可能な Stub ノードの集合 (以下，N^{PTC})

$N^{ImpossibleToCommunicate} (\subset N^{Stub})$:被害者と通信不可能な Stub ノードの集合 (以下，N^{ITC})

リンク $l_{i,j}$ で, 何もパケットフィルタリングを行わないときは,$N_{l_{i,j}}^{filter} = \phi$ となる. Stub ノード n_a が攻撃を受けている時，各リンク $l \in L$ 上でパケットフィルタリング (N^{filter} を決定) をする．求めるべき攻撃元の集合を $N^{Attackers}$ とする．この時のパケットフィルタリングによって観測される情報を表 3–3，表 3–4 に示す.

表 3–3　攻撃パターンが複合型の場合に得られる情報

条件	被害者への攻撃が止まる.	被害者への攻撃が止まらない.
得られる情報	$N^{Attackers} \cup N^{PTC} \neq \phi$	$N^{Attackers} \subset N^{ITC}$

表 3–4　攻撃パターンが単一型の場合に得られる情報

条件	被害者への攻撃が止まる.	被害者への攻撃が止まらない.
得られる情報	$N^{Attackers} \subset N^{PTC}$	$N^{Attackers} \cup N^{ITC} \neq \phi$

上記の設定で，以下の式を最適化する．

- 攻撃パターンが複合型の場合の条件

  目的関数　$max|N^{ITC}|$

  制約条件　被害者への攻撃が止まらない．

- 攻撃パターンが単一型の場合の条件

  目的関数　$max|N^{PTC}|$

  制約条件　被害者への攻撃が止まる．

上記の条件を満たすとき，攻撃パターンが複合型では N^{PTC} が推定可能な攻撃元の集合となる．単一型では N^{ITC} が攻撃元の集合となる．

3.4　攻撃元の推定に使用できる情報の形式化

本論文では，攻撃元の推定に使用できる情報としてルーティングテーブルの情報，IP トレースバックの結果を想定した．以下に，想定したルーティングテーブル，IP トレースバックとその形式化について示す。

3.4.1　ルーティングテーブル

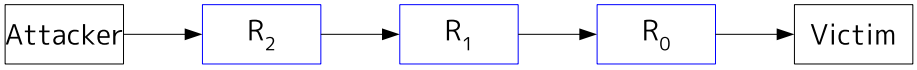
想定する IP ネットワークのトポロジに存在するルータにはルーティングテーブルがあり，デスティネーションノードに対して次の送信先が決まっている．

ルーティングテーブルの形式化

Transit 間，Transit-Stub(デスティネーションのみ) 間のリンク $l_{i,j}(n_i, n_j \in T)$ は，ルーティングテーブルに基づきリンクの向きが定められている．使用する記号を以下に示す．

$hop^{n_a}_{n_i,n_j}$：ノード n_i からのノード n_j までの hop 数　(デスティネーションが n_a の時のルーティングテーブルに従う)

3.4.2　IP トレースバック



				start	end	distance
P(R0)			p	R0		0
P(R1)		p	p(1-p)	R1	R0	1
P(R2)	p	p(1-p)	p(1-p)^2	R2	R1	2

図 3–3　辺サンプリング

IP トレースバックの手法として，確率的パケットマーキング方式を想定した．この手法では，IP Record Route オプションを用いて経路上のルータアドレスを確率的にオプションヘッダに追加していく．以下に，start,end,distance の固定フィールドを使った辺サンプリングの手順を示し，図 3–3 に概念図を示す．

(1) 各ルータは確率 p で自身のアドレスを start にマークし，distance を 0 に設定する

(2) 手順 (1) でマークせず且つ distance が 0 の場合，自身のアドレスを end にマークする

(3) 手順 (1) でマークしない場合，常に距離を 1 つ増やす

(4) 被害者は，サンプリングされた辺 (start,end) を多数集め，距離 distance の小さい順に被害者を根とする有向グラフに挿入し，攻撃経路を再構築する

図 3–3 が示すように，被害者からの距離が d(ホップ) におけるルータ R_d がマークしたパケットが被害者に届く確率 $P(R_d)$ は $p(1-p)^{d-1}$ である．特に，攻撃元からの攻撃パケットが N パケット通過している時の確率は $Np(1-p)^{d-1}$ である．これらの確率事象は互いに独立であるため，全ルータのマークしたパケットを受け取る確率は $Np(1-p)^{d-1}$ となる．このとき受け取ればよいパケット数は N 種類である．ここでクーポンコレクター問題より，ln(N) + O(1) 個のパケットを集めれば，N 種類のパケットが集まる．よって，経路を再構築するために必要な総パケット数の期待値 E(N) は，$E(N) < \frac{ln(N)}{p(1-p)^{d-1}}$ で得られる．

IP トレースバックによる攻撃元の推定

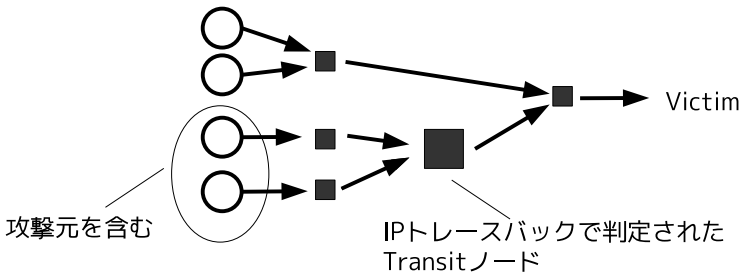


図 3–4　IP トレースバックによる攻撃元の検出

IP トレースバックは，攻撃者の攻撃経路に使用された Transit ノードで，ある確率で ”攻撃パケットが通過した ”と判定できるものとする．本手法では，IP トレースバックにより得られた判定により，攻撃元の IP ネットワーク上での位置を推定する．この判定により，図 3–4 に示すように，IP トレースバックの判定を受けた Tranit ノードに経路を持っている Stub ノードの集合に攻撃元を含んでいることが検出できる．探索手法への使用方法は，4.3 章で示す．IP トレースバックの判定で使用する記号を以下に示す．

$N^{AttackPacketPassed}(\subset N^{Transit})$：IP トレースバックにより，攻撃パケットが通過したと判定された Transit ノードの集合

第 4 章

提案手法

本章では，DDoS 攻撃の攻撃元の検出問題に対しての提案手法を示す．提案手法では，取り扱う情報別に 3 つの層を持たせた．検出問題を制約最適化問題として形式化を行い，IP トレースバックの情報を用いた 2 種類の解探索の戦略を提案する．IP トレースバックはパケットマーク方式を想定し，集めることができた一部のサンプルだけを使用する．これは，攻撃パケット数が少ない場合では，全てのサンプルを集める事は困難となるためである．また，探索単位の切替えによってサイクル数を削減する手法を示す．

4.1 提案手法の処理の流れと構造

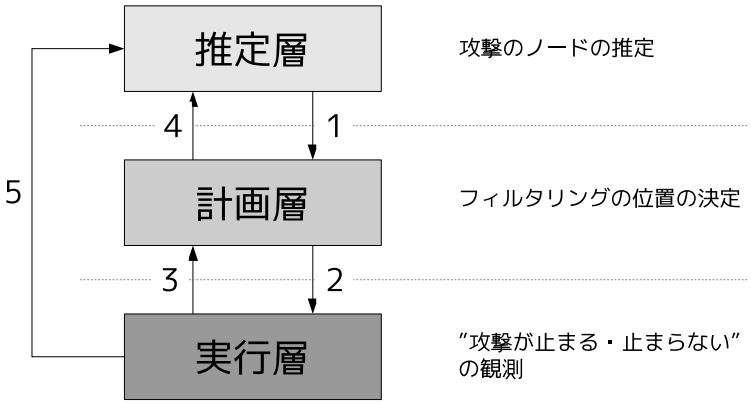


図 4–1 提案手法の階層構造

提案手法では，取り扱う情報別に処理をわけ，3 層の構造を持つようにした．図 4–1 に，提案手法の階層構造を示す．また提案手法では，攻撃元を推定する情報として IP トレースバックの判定，及びルーティングテーブルの情報があるときを想定した．

各層で取り扱う情報

各層で取り扱う情報を以下に示す．

(推定層) 攻撃のモデル，その攻撃を観測するモデルおよびIPトレースバックの判定など，結果として使える情報のモデル

(計画層) IPネットワーク上でとりうる動作とその結果により観測される情報のモデル

(実行層) 被害者への攻撃の有無を観測する情報のモデル

情報の流れ

各層で処理された情報は，他の層へ渡される．図4–1では，情報の流れる向きを矢印とした有向辺で示した．図4–1で使用した，情報1〜5を以下に示す．

1 推定層で決定された解

2 計画層で決定された解

3 被害者が攻撃を受けているか，受けていないか

4 計画層の解と情報3の結果によって観測される情報

5 被害者が攻撃を受けているか，受けていないか

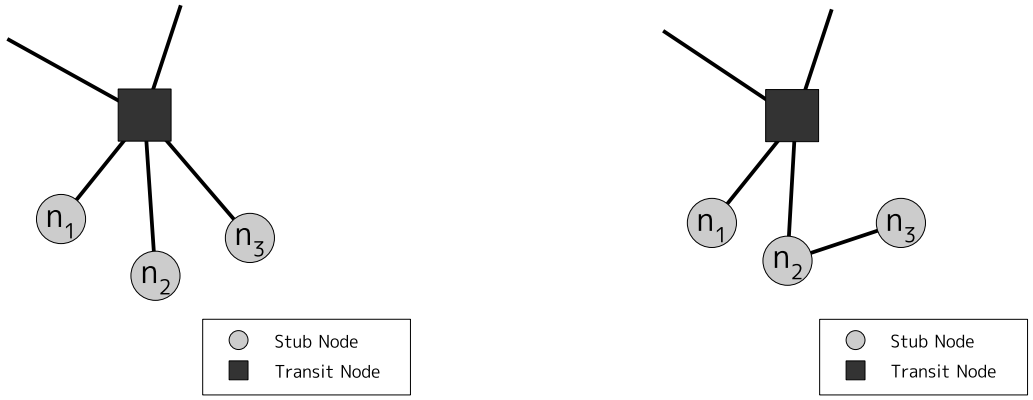
特に，情報**3，4**はIPネットワークのトポロジに依存する情報である．図4–2(b)のようなStub-Stub間のエッジをもつIPネットワークのトポロジが存在したとする．この時，Stubノードn_2のみをパケットフィルタリングの対象とすることはできず，Stubノードn_3のパケットも同時にパケットフィルタリングを行ってしまう．本論文では，Stubノードに接続するエッジは，図4–2(a)のTransit-Stubエッジのみとし，Stub-Stubエッジは存在せず，情報**3，4**は使用しないものとする．

提案手法の処理の流れ

全体の処理の流れを ”**処理順序** [**処理の行われる層**]:処理の内容 ”の形式で以下に示す．

I [**推定層**]: 問題・情報 (IPトレースバックの判定, ルーティング) を元に推定層で制約最適化問題 (以下，COP) を生成する．

II [**推定層**]: 推定層で作成されたCOPの解の決定を行う．



(a) Trasnit-Stub エッジ　　(b) Stub-Stub エッジ

図 4–2　トポロジに存在するエッジ

III　[**計画層**]: 推定層の解を元に，計画層で COP を生成する．

IV　[**計画層**]: 計画層で作成された COP の解の決定を行う．

V　[**実行層**]: 計画層の解を元に，IP ネットワークのトポロジ上でフィルタリングを実行する．

VI　[**推定層**]: 実行層の実行結果を元に，推定層で生成された COP に制約の追加を行う．

VII　[**推定層**]: 制約を満たす全ての解の探索が終了すれば解を決定する．まだ未探索の解が存在すれば，2 へ

この手法では，処理”I → II → III → IV → V → VI → VII → II → III →···→ VII” というサイクル動作を取る．以降，サイクルとは，処理”II → III → IV → V → VI → VII ”の事を表す．また，提案手法では制約最適化問題を用いて解の探索を行う．制約最適化問題の説明は 4.2 節で行う．なお，推定層・計画層での COP の生成方法，推定層の制約の追加処理など各層で行われる処理の詳細は後の章で示す．4.3 章で推定層，4.4 章で計画層，4.5 章で実行層を示す．

4.2 制約最適化問題 (COP) の形式化

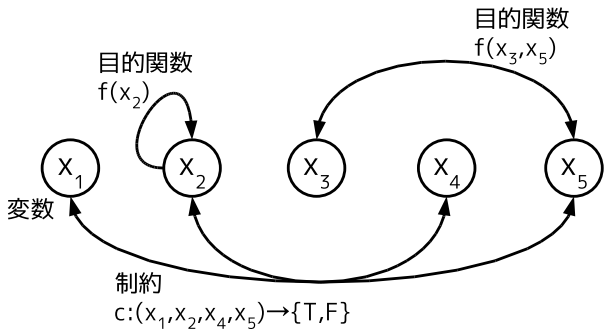


図 4–3　制約網の例

制約最適化問題（COP:Constraint Optimization Problem）は，制約充足問題 (CSP: Constraint Satisfaction Problem) を，制約違反時の罰則としての違反点数と, 解の良さを数値化した評価関数を与えることにより拡張したものである．制約を満足する解の中で評価関数を最大にするような変数への値の割当てを解とする組合せ探索問題である．COP は，組合せ$<$ X,D,C $>$で形式化される．

- 変数集合 $X = \{x_1, ..., x_m\}$
- 領域集合 $D = \{D_1, ..., D_m\}$
- 制約条件集合 $C = \{c, ...c_n\}$

解探索に使用する，制約網の例を図4.2に示す．各変数x_iは領域D_iより値d_iを選択する．C を制約条件の集合, F を制約条件に対する評価関数の集合とする．x_iは変数$\{x_j, ...\}$と制約条件$c \in C$により関係する．制約条件cに対応する評価関数$f \in F$により，変数値の割当$\{(x_i, d_i), (x_j, d_j), ...\}$についての利得が評価される．大域利得はすべての評価関数に対する利得の総和であり，大域利得を最大にする変数値の割当が最適解である．

4.3 推定層

推定層では，IP トレースバックの判定・ルーティングテーブルの情報，及び，実行層の結果，を元に攻撃元の推定が行われる．推定手法には COP が用いられ，”変数”，”変数の持つ値域”，”初期に生成される制約・評価関数”，”実行層の結果により追加される制約”が生成される．

”初期に生成される制約・評価関数”を以下に示す．

- 最小の攻撃元の集合を推定するための評価関数 $f^{NumberOfElements}$
- IP トレースバックを用いた攻撃元の特定に関する制約 $c^{PossibleToReach}$
- IP トレースバックの判定における距離に関する評価関数 $f^{IP\_NumOfHops}$
- IP トレースバックの判定における攻撃元の数に関する評価関数 $f^{IP\_NumOfAttakers}$

評価関数 $f^{NumberOfElements}$，$f^{IP\_NumOfHops}$，$f^{IP\_NumOfAttakers}$ はノードに対しての「攻撃元であるか」という評価であり，この評価関数によってサイクル動作における実行層での探索順序 (解探索の戦略) が決定される．推定層の解とは,「攻撃元である」と推定を行ったノードの集合であり，全ての制約を満たす範囲で探索が行われる．制約 $c^{PossibleToReach}$ は，3.3 章で示した制約条件に相当する．しかし，3.3 章の制約条件の全てを満たしておらず，一部分である．足りない制約条件は，”実行層の結果により追加される制約”で生成される．

”実行層の結果により追加される制約”を以下に示す．

- 実行層での結果が「攻撃が止まる」の場合に追加される制約 c^{Stop_1}・c^{Stop_2}
- 実行層での結果が「攻撃が止まらない」の場合に追加される制約 $c^{NotStop}$

制約 c^{Stop_1}，$c^{NotStop}$ は，3.3 章で示した制約条件の一部に相当する．制約 c^{Stop_2} は，3.3 章で示した目的関数に相当する．サイクル動作の終了は，全ての制約条件 (制約 c^{Stop_1}，$c^{NotStop}$) を集めたときである．

攻撃のパターン (複合型・単一型) が異なると，実行層での結果により得られる情報が異なる (3.3 章-表 3, 表 4)．これに伴い，”実行層の結果により追加される制約”も異なる．単一型か複合型かは，IDS が検知できるものとし，以降では，攻撃の種類が**単一型**のみを考察することとする．

4.3.1 ”変数”，”変数の持つ値域”の生成方法

Stub ノードごとに変数を生成し，そのとりうる値域 $D = \{0,1\}$(0…攻撃元でない, 1…攻撃元である) とする．ここでは，Stub ノード n_i に対応する変数を x_i とし, 値域 $D_i = \{0,1\}(D_i \subseteq D)$ を持つ．変数の集合は $X := \{x_i | i = 1, ...N^{Stub}\}$ と表記できる (N^{Stub} は Stub ノード数).

4.3.2 ”初期に生成される制約・評価関数”の生成方法

以下に，推定層で初期に生成される評価関数 $f^{NumberOfElements}$，$f^{IP\_NumOfHops}$，$f^{IP\_NumOfAttakers}$ と制約 $c^{PossibleToReach}$ を示す.

最小の攻撃元の集合を推定するための評価関数

3.1 章で示した DDoS 攻撃の攻撃の検出問題の解では，攻撃元でない Stub ノードを「攻撃元である」と判定してはいけない．この判定が起きないようにする為には,「攻撃元である」と判定する攻撃元の数をなるべく少なくすればよい．そこで，評価関数 $f^{NumberOfElements}$ は，全ての変数に対して生成し,「攻撃元である」と判定した攻撃元の数が少ないほど利得を上げる．また，優先すべきことは解に全ての攻撃元を含んでいることであり，評価関数 $f^{NumberOfElements}$ は全ての評価関数の中で優先度の最も低い評価関数である．

$x(\in X)$ に関する評価関数 $f^{NumberOfElements}$ を以下に示す．また，x_i における $f^{NumberOfElements}$ を図 4–4 に示す.

$$f^{NumberOfElements}(x) = \begin{cases} \alpha & if(x = 0) \\ 0 & otherwise \end{cases}$$

(α は定数．今回使用した α の値については，4.3.2 項で示す．)

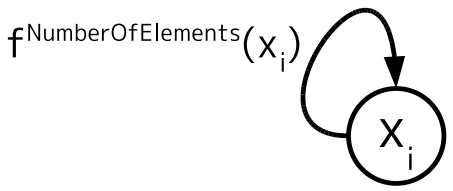


図 4–4　評価関数 $f^{NumberOfElements}$

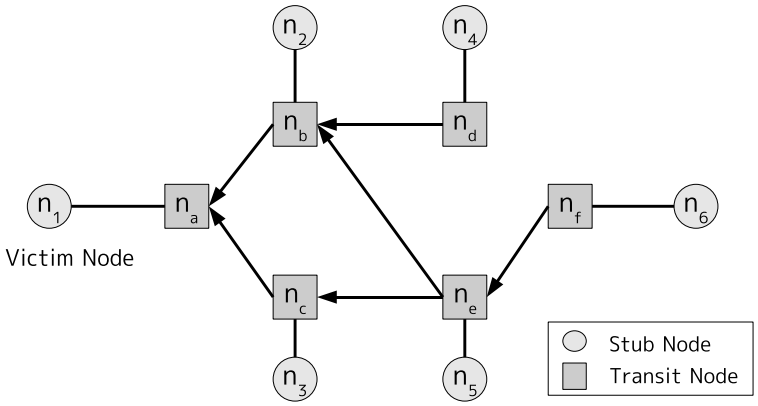


図 4–5　ネットワークモデル図

IP トレースバックを用いた攻撃元の特定に関する制約条件

ルーティングテーブル，IP トレースバックの結果を利用することにより，攻撃元の範囲を特定できる．そこで，攻撃パケットが通過したことが確認された Transit ノード $n_i(\in N^{AttackPacketPassed})$ について考える．Transit ノード n_i を通過した攻撃パケットを送信した攻撃元は，Transit ノード n_i へ，ルーティングテーブルに従った経路を持っている Stub ノードに限定される．そこで，Transit ノード n_i への経路を持っている Stub ノードの集合を $N_i^{PossibleToReach}$ とし，変数の集合を $X_i^{PossibleToReach}$ と定義する．

Stub ノードの集合 $N_i^{PossibleToReach}$ には，Transit ノード n_i を経路に使用した攻撃元が存在し，これは”変数の集合 $X_i^{PossibleToReach}$ は，　最低でも 1 つの変数が値”1”を選択する”という制約条件 $c_i^{PossibleToReach}$ が生成される．

$$c_i^{PossibleToReach} : \sum_{x \in X_i^{PossibleToReach}} x \rightarrow \{F,T\}, \{F \quad 0, T \quad otherwise,\}$$

この制約条件 $c^{PossibleToReach}$ は，$N^{AttackPacketPassed}$ の全ての要素について生成される．

例を示す．図 4–5 のような IP ネットワークのトポロジがあるとする．各 Transit ノードには，ルーティングテーブルがある．ルーティングテーブルを持った Transit ノードは，パケットの送信に関しての方向が定められている．この方向を有向辺として表す．図 4–5 は，デスティネーションノードが Stub ノード n_1 の場合である．ここで，$N^{AttackPacketPassed} = \{n_b, n_c\}$ ならば，$N_b^{PossibleToReach} = \{n_2, n_4, n_5, n_6\}$, $N_c^{PossibleToReach} = \{n_3, n_5, n_6\}$ となる．Stub ノード集合 $N_b^{PossibleToReach}$ に対応する変数集合は $X_b^{PossibleToReach} = \{x_2, x_4, x_5, x_6\}$ となる．$N_c^{PossibleToReach}$ も $N_b^{PossibleToReach}$ と同様であり，ここでは省略する．制約 $c_b^{PossibleToReach}$ と $c_c^{PossibleToReach}$ を，図 4–6 に示す．

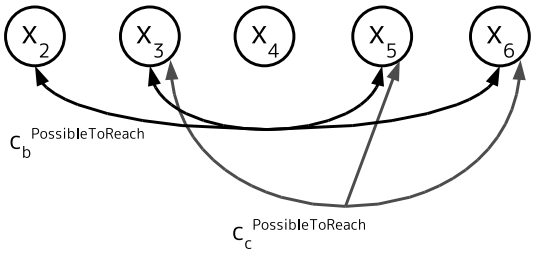


図 4–6　制約 $c^{PossibleToReach}$

IP トレースバックの判定における距離に関する評価関数

3.4.2 節で示した IP トレースバックの手法では，マークの上書きが生じるため，Transit ノードに距離 (ホップ数) が近い Stub ノードの方がマークされたパケットが被害者に届く確率が高い．そこで，評価関数 $f^{IP\_NumOfHops}$ は変数 $x(\in X^{PossibleToReach})$ ごとに生成し，Transit ノードに近い Stub ノード $(\in N^{PossibleToReach})$ ほど利得を高くする．以下に，$x(\in X^{PossibleToReach})$ に関する評価関数を示す．

$$f^{IP\_NumOfHops}(x) = \begin{cases} x.value & if(x = 1) \\ 0 & otherwise \end{cases}$$

(単調減少を行う関数を $\beta(x)$ とする．Transit ノード n_a，Stub ノード n_i 間のホップ数が d の時 $x.value = \beta(d)$ である．今回使用した $\beta(x)$ については，4.3.2 項で示す．)

IP トレースバックの判定における攻撃元数に関する評価関数

Stub ノードの集合 $N^{PossibleToReach}$ の中で多くのノードを「攻撃元である」と推定すると，推定した攻撃元の集合の中に攻撃元を含む割合が増える．一方で，推定した攻撃元が「攻撃元である」割合が減ってしまう．推定した攻撃元が「攻撃元である」割合を増やすため，なるべく少ないノードを攻撃元と推定することにする．そこで，評価関数 $f^{IP\_NumOfAttakers}$ は，変数の集合 $X^{PossibleToReach}$ に対して生成し，攻撃元であると判定されるノード数が少ない組合せほど利得を高くした．また，上記は逆となる，多くのノードを「攻撃元である」と推定する評価関数についても，5.3 節にて評価を行った．以下に，$X^{PossibleToReach}(\in \forall X^{PossibleToReach})$ に関する評価関数を示す．

$$f^{IP\_NumOfAttakers}(x_1, x_2, ...) = \begin{cases} \dfrac{1}{\sum\limits_{x \in X^{PossibleToReach}} x} & if(\sum\limits_{x \in X^{PossibleToReach}} x \neq 0) \\ 0 & otherwise \end{cases}$$

初期に生成される最終的な評価関数

初期に生成される最終的な評価関数について説明する．具体的には，$f^{NumberOfElements}$，$c^{PossibleToReach}$，$f^{IP\_NumOfHops}$，$f^{IP\_NumOfAttakers}$ を結合する．$c^{PossibleToReach}$ に対応する評価関数 $f^{PossibleToReach}$ を以下に示す．関数 $f^{PossibleToReach}$ の引数は $X^{PossibleToReach} = \{x_1, x_2, ...\}$ とする．

$$f^{PossibleToReach}(x_1, x_2, ...) = \begin{cases} 0 & if(\sum_{x \in X^{PossibleToReach}} x \neq 0) \\ -\infty & otherwise \end{cases}$$

"$f^{PossibleToReach}$，$f^{IP\_NumOfHops}$，$f^{IP\_NumOfAttakers}$ "は，変数の集合 $X^{PossibleToReach}$ に対する評価関数である．結合した評価関数を f^{IP} とし，以下に示す．また，関数 f^{IP} の引数は $X^{PossibleToReach} = \{x_1, x_2, ...\}$ とする．

$$f^{IP}(x_1, x_2, ...) = \begin{cases} \sum_{x \in X^{PossibleToReach}} f^{IP\_NumOfHops}(x) \\ \qquad \times f^{IP\_NumOfAttakers}(x_1, x_2, ...) & if(\sum_{x \in X^{PossibleToReach}} x \neq 0) \\ -\infty & otherwise \end{cases}$$

$f^{IP\_NumOfHops}$ と $f^{IP\_NumOfAttakers}$ では，$f^{IP\_NumOfHops}$ の評価を優先して行うように，4.3.2 項の β (x)，4.3.2 項の α を決定した．

β (x) は，以下の条件を満たす関数

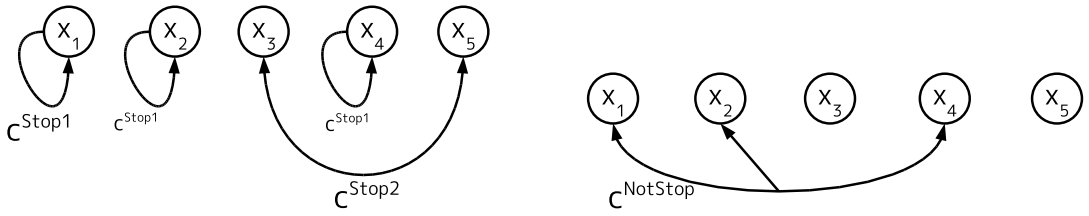
$$2^{(n-x)} \leq \beta\ (x) \qquad (\text{ただし}, n = \max_{n \in N_i^{PossibleToReach}} hop_{n,n_i}^{n_a})$$

α は以下の条件を満たす値

$$\alpha \leq \max_{N^{PossibleToReach} \in \forall N^{PossibleToReach}} (\frac{f^{IP}}{|N^{PossibleToReach}|})$$

以上より，最終的な評価関数を以下に示す．関数 f の引数は $X = \{x_1, x_2, ...\}$ とする．

$$f(x_1, x_2, ...) = \sum_{x \in X} f^{NumberOfElements}(x) + \sum_{X^{PossibleToReach} \in \forall X^{PossibleToReach}} f^{IP}(x_1, x_2, ...)$$



(a) 制約 c^{Stop}　　(b) 制約 $c^{NotStop}$

図 4–7　実行層の結果によって推定層で生成される制約

4.3.3　”実行層の結果から追加される制約 ”の生成方法

実行層の結果から追加される制約 c^{Stop_1}, c^{Stop_2}, $c^{NotStop}$ は，推定層で初期に生成された COP に追加される．この制約 c^{Stop_1}, c^{Stop_2}, $c^{NotStop}$ にかかわる変数は，推定層の解に関係する．あるサイクルにおける推定層での変数を以下の集合として定義する．

- 変数値”1”を選択した変数集合　　$X^{Presume}$
- 変数値”0”を選択した変数集合　　$X^{NotPresume}$

この時に追加する制約 c^{Stop_1}, c^{Stop_2}, $c^{NotStop}$ を以下に示す．

攻撃が止まる時に追加される制約

$$c^{Stop_1} : x \rightarrow \{T, F\}, \{T \quad 0, F \quad 1\}$$

$$c^{Stop_2} : \prod_{x \in X_{Presume}} x \rightarrow \{T, F\}, \{T \quad 0, F \quad otherwise\}$$

攻撃が止まらない時に追加される制約

$$c^{NotStop} : \sum_{x \in X_{NotPresume}} x \rightarrow \{F, T\}, \{F \quad 0, T \quad otherwise\}$$

攻撃が止まる時，3.3 節 表 3–4 より $X^{Presume}$ に対応する Stub ノードの中に全ての攻撃元を含んでいる事がわかる．これは，”攻撃元でない”と判定した Stub ノードには攻撃元を含まないことから，$X^{NotPresume}$ の各変数に対し，変数値”0 ”を選択するという制約 c^{Stop_1} が生成される．また，より少ない攻撃元の集合を探索するために，$X^{Presume}$ に対し，いずれかの変数が変数値”0 ”を選択するという制約 c^{Stop_2} が生成される．

攻撃が止まらない時，$X^{NotPresume}$ に対応する Stub ノードの中に攻撃元を含んでいる事がわかる．このことより，”攻撃元でない”と判定した Stub ノードに攻撃元を含む

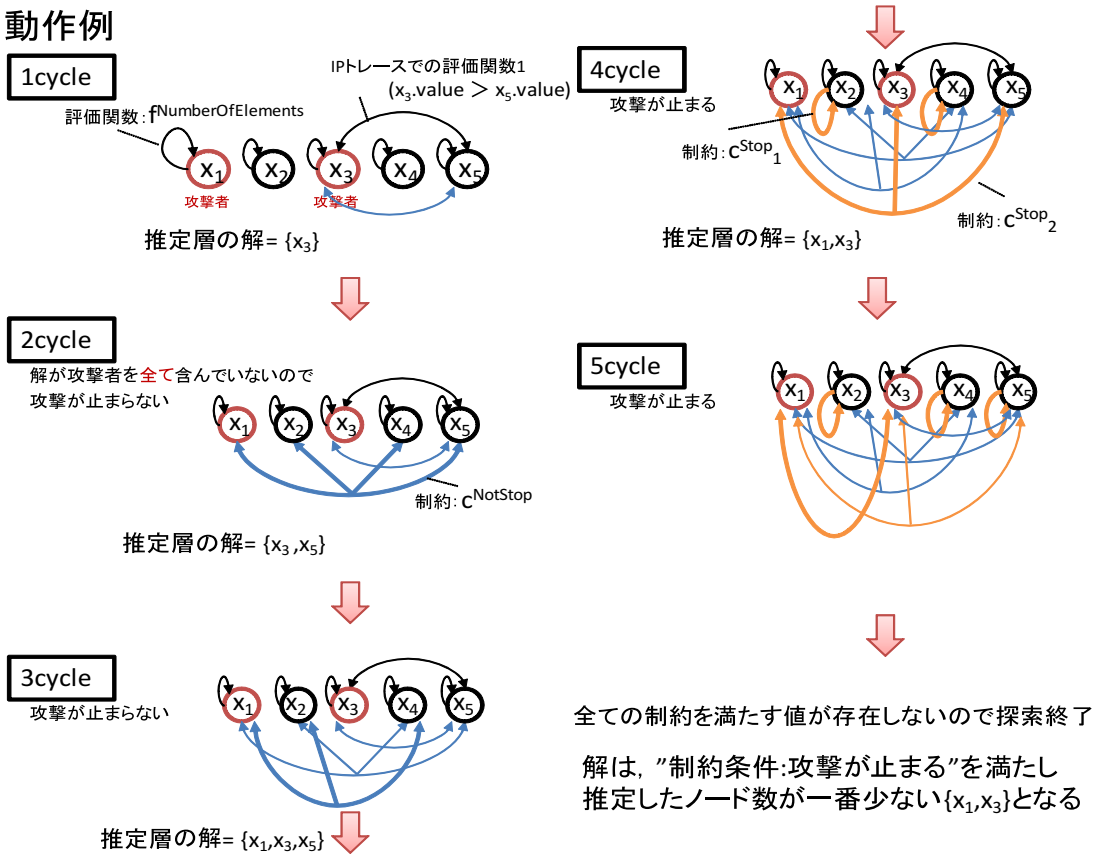


図 4–8　推定層におけるサイクル動作の例

ため，$X^{NotPresume}$ の各変数に対し，いずれかの変数が変数値”1”を選択するという制約 $c^{NotStop}$ が生成される．

例として，あるサイクルでの推定層の解が $X^{Presume} = \{x_1, x_2, x_4\}$，$X^{NotPresume} = \{x_3, x_5\}$ だとする．この時の，攻撃が止まる時に追加される制約 c^{Stop_1}，c^{Stop_2} を図 4–10(a) に，攻撃が止まらない時に追加される制約 $c^{NotStop}$ を図 4–10(b) に示す．

提案手法では，各サイクルにおいて，$c^{NotStop}$ か c^{Stop_2} を生成するため，これら制約により同じ変数値の組合せを探索することを防ぐ．これは，次のサイクル以降では，制約の追加により以前に探索した解は制約条件を満たす解とならないためである．また，サイクル動作における，解の探索の終了は制約を満たす全ての組合せの探索が終了したときである．

推定層における，サイクル動作の例を図 4–8 に示す．
x_1, x_2, x_3, x_4, x_5 の変数があるとし，攻撃元は，x_1, x_3 とする．以下にサイクル動作における処理を示す．

1 サイクル目： IP トレースの結果・ルーティングテーブルの情報を基に，初期の評価関数 $f^{NumberOfElement}$, $f^{IP}(= c^{PossibleToReach}, f^{IP\_NumOfHops}, f^{IP\_NumOfAttakers})$ が生成される．$N^{PossibleToReach} = \{x_3, x_5\}$，$x_3.value > x_5.value$ とする．この制約網を解き，解 $= \{x_3\}$ となる．

2 サイクル目： 実行層より「攻撃が止まらない」という結果がでる．$X^{NotPresume} = \{x_1, x_2, x_4, x_5\}$ に対して，制約 $c^{NotStop}$ が追加される．この制約網を解き，解 $= \{x_3, x_5\}$ となる．

3 サイクル目： 実行層より「攻撃が止まらない」という結果がでる．$X^{NotPresume} = \{x_1, x_2, x_4\}$ に対して，制約 $c^{NotStop}$ が追加される．この制約網を解き，解 $= \{x_1, x_3, x_5\}$ となる．

4 サイクル目： 実行層より「攻撃が止まる」という結果がでる．$X^{Presume} = \{x_2, x_4\}$ に対して，制約 c^{Stop_1} が追加される． $X^{Presume} = \{x_1, x_3, x_5\}$ に対して，制約 c^{Stop_2} が追加される．この制約網を解き，解 $= \{x_1, x_3\}$ となる．

5 サイクル目： 全ての制約を満たす値が存在しないので探索終了．

探索終了間までで，推定層において 4 サイクル目の解が制約条件「攻撃がとまる」を満たし，目的関数 $|N^{PTC}|$ を最大にする．従って，最適解は 4 サイクル目の解 $= \{x_1, x_3\}$ となる．

4.4 計画層

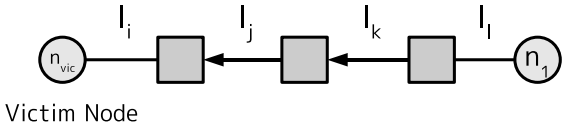
計画層では, 推定層で決定された解を元に,IP ネットワーク上でのパケットフィルタリングの位置と種類を決定する．計画層で行われる COP の生成方法について示す．計画層で生成される制約を以下に示す.

- パケットフィリタリングの位置を決定する制約 c^{Filter}
- 被害者との通信経路を残す制約 $c^{NotFilter}$

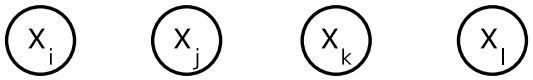
制約 c^{Filter}，$c^{NotFilter}$ は，推定層の解 $X^{Presume}$，$X^{NotPresume}$ を元に生成される．以下に詳細を示す.

4.4.1 ”変数”，”変数の持つ値域”の生成方法

リンクごとに変数を生成し，そのとりうる値域 $D = \{\phi, n_{victim}\}$ とする．”ϕ”は，”何もフィルタリングをしない”に相当し，”n_{victim}”は，”ディスティネーションノードが n_{victim} のパケットをフィルタリングする”に相当する．ここでは，リンク l_i に対応する変数を x_i とし，値域 $D_i = \{\phi, n_{victim}\}(D_i \supseteq D)$ を持つ．変数の集合は $X := \{x_i | i = 1, ..., L\}$ と表記できる (L はリンク数)．制約の生成方法の例を図 4–9 に示す.



(a) 被害者-Stub ノード間の経路

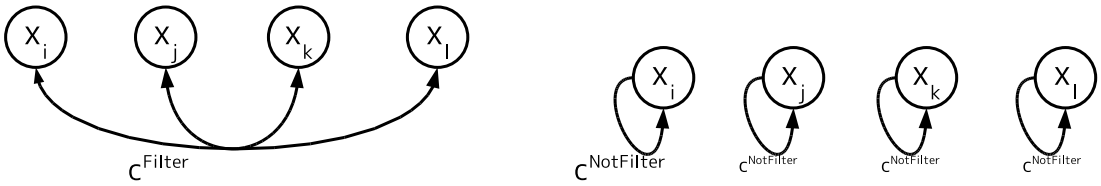


(b) 生成される変数

図 4–9　変数の生成方法

4.4.2 制約の生成方法

推定層で生成される制約 c^{Filter} と $c^{NotFilter}$ は，Stub ノード-被害者間の全ての経路に対して生成される．被害者を n_{victim} とし，推定層で”攻撃元である”と判定された Stub ノードの集合（$=X^{Presume}$ に対応するノード）を $N^{Presume}$ とする．図 4–9(a) のような,



(a) $n_i \in N^{Presume}$ の時に生成される制約　　(b) $n_i \notin N^{Presume}$ の時に生成される制約

図 4–10　生成される制約

被害者から Stub ノード n_1 までの経路があり，経路上のリンク l_i，l_j，l_i，l_l があるとする．この時，リンクの集合を R_{il} とし, それに対応する変数集合を X_{il} とする. 生成される制約 c^{Filter}，$c^{NotFilter}$ を以下に示す.

$n_i \in N^{Presume}$ の時に生成される制約

$$c^{Filter} : X_{il} \rightarrow \{F, T\}, \{F \quad \exists_1 x = n_{victim} | x \in X_{il}, T \quad otherwise\}$$

$n_i \notin N^{Presume}$ の時に生成される制約

$$c^{NotFilter} : x \rightarrow \{F, T\}, \{F \quad \phi, T \quad otherwise\}$$

$n_i \in N^{Presume}$ の時，経路上のどこかで攻撃元のパケットをフィルタリングする必要がある．そこで，X_{il} に対して，どれか一つの変数が変数値 ”n_{victim} を選択するという制約 ”c^{Filter} が生成される．

$n_i \notin N^{Presume}$ の時，攻撃元でないノードの経路上でパケットフィルタリングをしてはいけない．そこで，各変数に対して，変数値”ϕ”を選択するという制約 $c^{NotFilter}$ が生成される．図 4–9(a) の時に生成される制約を，図 4–10 に示す.

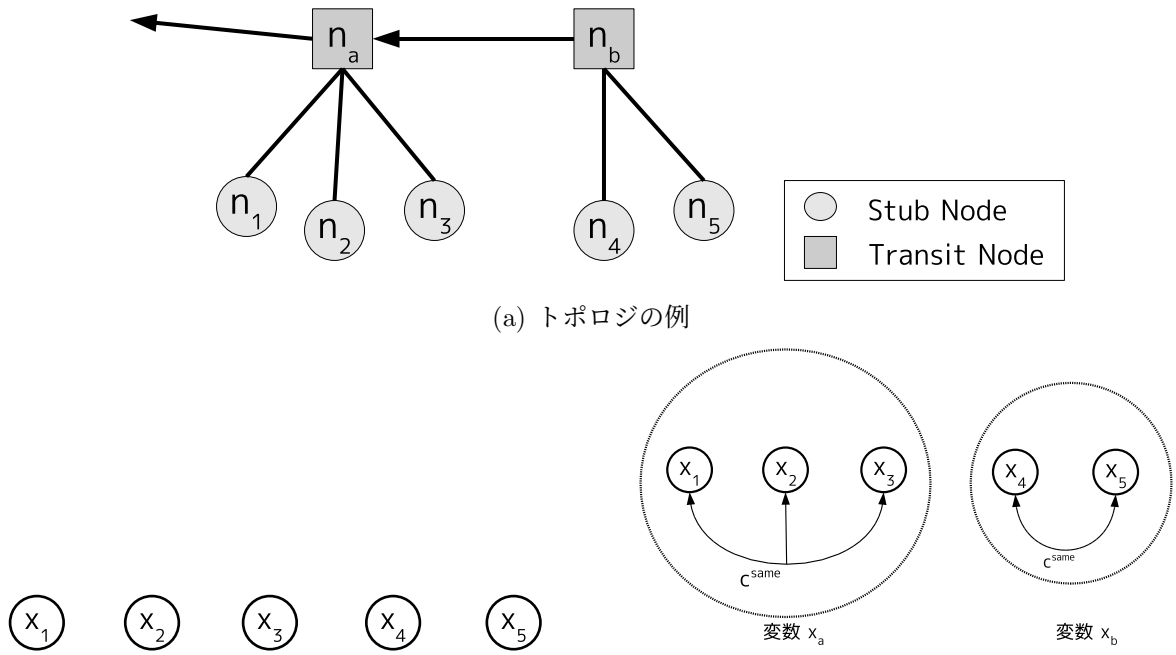
4.5　実行層

実行層では，計画層で求めた解を元に，IP ネットワークでパケットフィルタリングを行い，被害者への ”攻撃が止まる ”，”攻撃が止まらない ”の観測を行う．この観測方法には，2.2.1 節でしめした IDS(侵入検知システム) などが上げられる．本研究は，攻撃元の推定に焦点を置くため攻撃の有無は 100%わかるものとする.

4.6 探索を行う単位の切替えによる高速化手法

この手法では，Stub ノード単位と Transit ノード単位に探索を行う単位を切替える．Stub ノード単位の探索は，粒度が細かい探索 (より厳密な問題) と考えられ，また，Transit ノード単位の探索は，荒い探索 (緩和された問題) と考えられる．この 2 つの粒度の違う問題を，環境に応じて切替えることによりサイクル数の削減を行なう．以下に，単位を切替える際に追加する制約，探索を行う単位の切替えによる手法，制約の追加による制約の不整合の解決方法を示す．

4.6.1 Transit ノード単位での探索



(a) トポロジの例

(b) Stub 単位時の変数と制約

(c) Transit 単位時の変数と制約

図 4–11　緩和手法による変数の生成方法

従来の DDoS 攻撃の攻撃元の検出問題の目的は，”攻撃元である全ての Stub ノードの検出”であった．本手法では，目的を”攻撃元を接続する全ての Transit ノードの検出”とする．そのため，Transit ノードごとに変数を生成し，その変数の持つ値域 $D = \{0, 1\}$ とする．”0”は”接続する Stub ノードに攻撃元を含まない”に相当し，”1”は”接続する Stub ノードに攻撃元を含む”に相当する．本手法では，Transit ノードごとに生成した変数に対して，従来の探索単位である Stub ノードごとに生成した変数に制約 c^{Same} を追加する

ことで Transit ノード単位の探索を行えるようにする．Transit ノード i に接続する Stub ノードの集合を $X_i^{Connect}$ とした時，以下に制約 c^{Same} を示す．

$$c^{Same} : X_i^{Connect} \rightarrow \{F, T\}, \{F \quad x_i \neq x_j | x_i, x_j \in X, T \quad otherwise\}$$

例として，図 4–11(a) のようなトポロジがあるとする．Sutb ノード単位での変数の生成したときを図 4–11(b) に示す．図 4–11(b) では，図 4–11(a) 上の Stub ノード n_1，n_2，n_3，n_4，n_5 に対応した変数 x_1，x_2，x_3，x_4，x_5 が生成される．Transit ノード単位での変数の生成したときを図 4–11(b) に示す．図 4–11(a) で Transit ノード n_a に接続する Stub ノード n_1，n_2，n_3 について考える．Transit ノード単位の探索を行う場合，Transit ノード n_a に接続する Stub ノードに対応する変数 x_1，x_2，x_3 には同じ変数の値を選択するという制約 c^{Same} が生成される．図 4–11(c) で示すように，制約 c^{Same} は x_4，x_5 に対しても生成される．

4.6.2 探索を行う単位の切替えによる手法

探索を行う単位の切替えにあたって，第一に，各単位での探索における変数の数を考える．Trasnit ノード単位の探索では，変数の数が Stub ノード単位での探索より少なくなり，少ないサイクル数で最適解を求められると考えられる．この考えに基づいた手法を以下に示す．

切替え手法 1

Transit ノード単位で探索を行った後に，Stub ノード単位で探索を行う．

切替え手法 1 では，Transit ノード単位での探索で生成される制約 c^{Stop1} が，総サイクル数に影響をあたえると考えられる．

第二に，解探索にかかる総サイクル数は各サイクルで追加される制約 c^{Stop1}，c^{Stop2}，$c^{NotStop}$ に影響すると考えられる．特に，制約 c^{Stop1} は，追加される制約の中で最も，探索を限定する範囲が大きい．この制約 c^{Stop1} が，より効率的に作成される問題の単位に切替える必要があると考えられる．この考えに基づいた手法を以下に示す．

切替え手法 2

初期は，Transit ノード単位で探索を行う．探索中に，連続して N 回”攻撃が止まらない”という結果がでたら探索単位を Transit ノード単位から Stub ノード単位に

切替えを行う．同様にして，Stub ノード単位の場合も Trasni ノード単位へと切替えを行う．

切替え手法 2 では，より制約 c^{Stop1} が生成される単位へと探索単位の切替えが行われる．しかし，探索単位の切替えによって制約の不整合が生じる場合があり，Stub ノード単位の時に生成された制約を探索に生かせない場合が生じる．制約の不整合については次項で説明を行う．

4.6.3 探索単位の切替えによって生じる制約の不整合

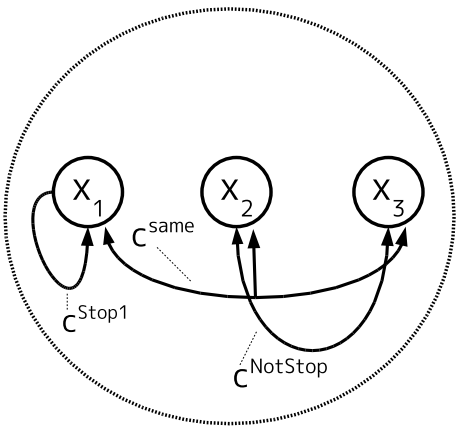


図 4–12 制約の不整合の例

制約 c^{Same} を生成した場合に，4.3.3 節の制約 c^{Stop1} と制約 $c^{NotStop}$ との間に不整合が生じる場合がある．不整合の例を図 4–12 に示す．この例の場合，c^{Same} により変数 x_1 , x_2, x_3 は同じ変数値を選択しなくてはならない．しかし，変数 x_1 は制約 c^{Stop1} により変数値 ”0 ”を選択し，また，制約 $c^{NotStop}$ により変数 x_2, x_3 のどちらかは変数値 ”1 ”を選択する．この時，全ての制約を満足する変数値の組合せは存在しない．そのため $X_i^{Connect}$ 内に限り，この 3 つの制約 c^{Stop1}，$c^{NotStop}$，c^{Same} をすべて満足する必要をなくし，$c^{Same} > c^{NotStop} > c^{Stop1}$ となる優先順位を付ける事とした．

第 5 章

実験

本章では，提案手法の有用性を示すために実験を行う．予備実験では，問題の規模に対して，ネットワークのトポロジと規模に対する，問題の性質・規模を実験的に評価した．実験 1，2 では，解の探索に異なった戦略を用いて評価を行った．実験 3 では，探索を行う単位の切替えによる高速化手法の評価を行った．

5.1 予備実験

推定層で初期に生成される”IP トレースバックを用いた攻撃元の特定に関する制約 $c^{PossibleToReach}$” についての評価を行うために予備実験を行った．この処理は，4.1 節で示した，

I [**推定層**]**:** 問題・情報 (IP トレースバック, ルーティング) を元に推定層で COP(初期に生成される制約) を生成する．

で行われる処理である．

COP で生成される ”IP トレースバックを用いた攻撃元の特定に関する制約 $c^{PossibleToReach}$ ”について調べた．”IP トレースバックを用いた攻撃元の特定に関する制約 $c^{PossibleToReach}$ ” は 4.3.2 で記述した通り，IP トレースバックの結果とルーティングテーブルの情報に影響を受ける．COP の問題の性質・規模で，制約はアルゴリズムの探索時間に大きく影響する．IP ネットワークトポロジのノード数を変化させたときに，”IP トレースバックを用いた攻撃元の特定に関する制約 $c^{PossibleToReach}$” の評価を行った．

評価基準

評価基準は，以下の 3 点とする．

- ”制約 $c^{PossibleToReach}$ の数 ”$(=|N^{AttackPacketPassed}|)$

- ”変数１つ辺りの制約の数 ”($=\sum_{n_i \in N^{AttackPacketPassed}} |X_i^{PossibleToReach}|/M$)

- ”制約１つにかかわる変数の数 ”
  ($=\sum_{n_i \in N^{AttackPacketPassed}} |X_i^{PossibleToReach}|/|N^{AttackPacketPassed}|$)

IP トレースバックによる攻撃元の検出方法

IP トレースバックには 3.4.2 節で示した，確率的パケットマーク方式を想定する．ルータがパケットをマークする確率 p は，0.05 とした．

ルーティングテーブルの生成方法

ルーティングテーブルの生成・管理方式には，あらかじめ固定されたルートを設定しておく「スタティックルーティング」を想定する．全ての Transit ノードがルーティングテーブルを持つ．ルーティングテーブルは，Stub ノード間の経路が最小ホップ数となるように設定した．また，最小ホップ数での経路が複数個ある場合には全ての経路を選択できるようにした．

測定方法

DDoS 攻撃を想定し，攻撃元 (複数)，被害者 (単数) は Stub ノードの中からランダムで決定した．被害者と攻撃元の重複はないものとした．ノードの数に対して，10 種類のトポロジを用意し，１種類につき 10 回の試行を行い，その平均値を取った．(値は 10 種類× 10 回=100 回の平均値)

5.1.1 予備実験１

総ノード数を変化させた場合の，”制約 $c^{PossibleToReach}$ の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”を調べた．予備実験 1 は以下の環境で行った．

実験環境

使用する IP ネットワークのトポロジは以下の条件を満たすものとする．

- IP ネットワークのトポロジ:Transit-Stub 型
- Transit ドメイン数:1
- Stub ドメイン数:1(Transit ノード当り)
- Stub ドメイン数:1(Stub ドメイン当り)

攻撃元の数が全ノードの 10%以下となるように攻撃元数を 10 にした．IP トーレスバックでは，攻撃パケット数を，経路の再構築に必要なパケット数の 1/4 である 10 パケットにした．

結果と考察

制約の評価を図 5–1 に示す．図 5–1 は，横軸が ”総ノードの数 (台) ”，縦軸が ”制約の数 (図中　Num_of_Const．縦左軸．以下同様) ”，”変数１つ辺りの制約の数 (図中 Num_of_Const(per One_Var)．縦右軸．以下同様) ”，”制約１つにかかわる変数の数 (図中　Num_of_Var(per One_Const)．縦左軸．以下同様) ”となっている．

実験に使用した，IP ネットワークのトポロジのパラメータを図 5–2 に示す．図 5–2 は，横軸が ”トポロジ中に存在する全ノードの数 (台) ”，縦左軸が ”トポロジ中に存在する全リンク数 (図中　Num_of_Links) ”，”各 Stub ノードから被害者までの全経路数 (図中.Num_of_Routes) ”，縦右軸が”各 Stub ノードから被害者までの平均 hop 数 (図中 Num_of_Average_Hops)”となっている．

また，ノード数 200 の時に 100 回の試行で生成された $X^{PossibleToReach}$ の要素数別に生成数を計測したものを図 5–3 に示す．図 5–3 は，横軸が $X^{PossibleToReach}$ の要素数，縦軸が 100 回の実験における生成数となっている．

図 5–1:Num_of_Const より，制約の数はノード数が変化しても変化していない．また，図 5–2:Num_of_Average_Hops より，今回生成した，ノード数 100〜500 のトポロジは各 Stub ノードから被害者までの平均 hop 数がほぼ一定であることがわかる．攻撃元をランダムで選択しているので，攻撃元から被害者までの平均 hop 数も一定であったと考えられる．このため，どのノード数のトポロジでも制約の数が同じになったと考えらる．Transit-Stub 型のトポロジでは，各 Stub ノードからの平均 hop 数がノード数を増加して大きく変化しないのが特徴である．今回実験に用いたトポロジは Transit ドメイン数が１であることも影響し，この特徴が顕著に現れたものだと考えられる．

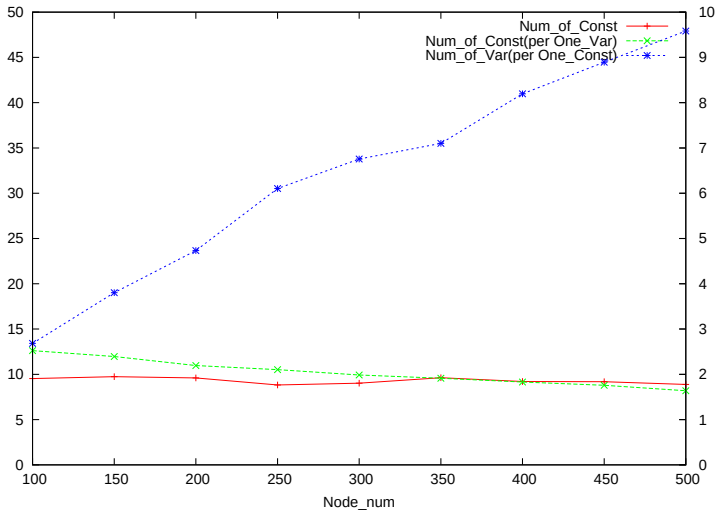


図 5–1　ノード数と ”制約の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”の関係図

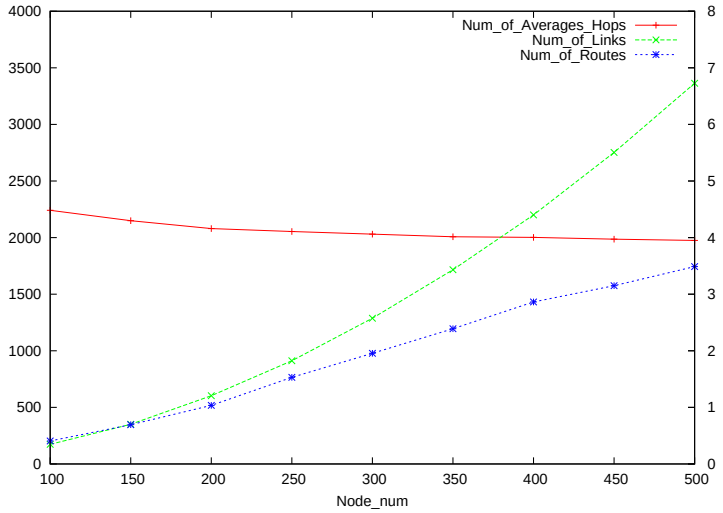


図 5–2　ノード数と平均ホップ数・リンク数・経路数の関係図

図 5–1:Num_of_Const(per One_Var) より，変数１つ辺りの制約の数はノード数が増えれば増えるほど線形に減少している．この結果から，各 Stub ノードから被害者までの平均 hop 数が一定のトポロジならノード数が小さい方が密な制約網が生成されることがわかる．これは，図 5–1:Num_of_Var(per One_Const) で制約１つにかかわる変数の数はノード数が増えれば増えるほど線形に増加している事からも同様の事が言える．

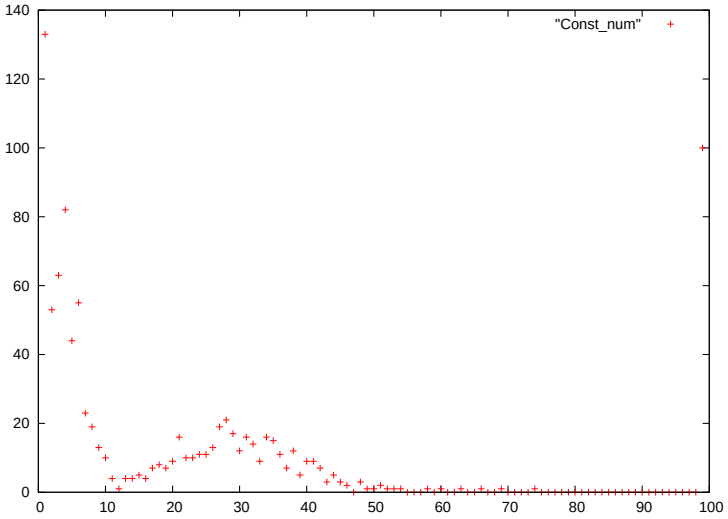


図 5–3　制約の要素数の分布図

図 5–3 より，ノード数 200 の場合では，要素数が 1〜10 までに多く分布している．また，要素数が 99 の値で 100 回となっている．これは，被害者からの距離が 1 ホップの Transit ノードでは攻撃パケットが集中し，今回の 100 回の試行中すべてにおいて，この Transit ノードで IP トレースバックのパケットが通過したという判定が起こったためである．

5.1.2　予備実験 2

攻撃元の平均 hop 数を変化させた場合の，”制約 $c^{PossibleToReach}$ の数 ”，”変数 1 つ辺りの制約の数 ”，”制約 1 つにかかわる変数の数 ”を調べた．

実験環境

使用する IP ネットワークのトポロジは 5.1.3 節 予備実験 1 の条件を満たすものとする．小規模の IP ネットワークを想定し総ノード数は 500 とした．また，攻撃元はその 1%である 5 台とした．IP トーレスバックでは，攻撃パケット数を，経路の再構築に必要なパケット数の 1/4 である 10 パケットにした．

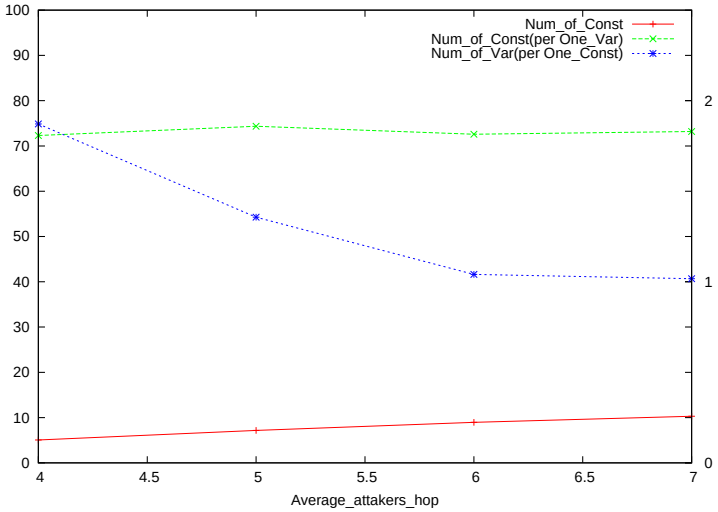


図 5–4　攻撃元から被害者までの平均 hop 数と ”制約の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”の関係図

結果と考察

攻撃元から被害者までの平均 hop 数を変化させたときの結果を図 5–4 に示す．図 5–4 の横軸は，”攻撃元から被害者までの平均 hop 数 ”，両縦軸は図 5–1 と同じ．図 5–1:Num_of_Const より，攻撃元から被害者までの平均 hop 数が上がるにつれて制約の数が増加している．これは，hop 数が増加したことにより多くの Transit ノードで IP トレースバックのパケットが通過したという判定が起こったためである．しかし，判定の確率は $10 \times 0.05 \times (1 - 0.05)^d$　(d はホップ数) であり，Num_of_Const のグラフの傾きは徐々に小さくなっていくと想定される．

図 5–1:Num_of_Const(per One_Var) より，変数１つ辺りの制約の数は一定であることがわかる．これは，図 5–1:Num_of_Var(per One_Const) で表される，制約１つにかかわる変数の数が減少しているためだと考えられる．制約１つにかかわる変数の数が減少した理由は，攻撃元から被害者までの平均 hop 数が上がり，$|N^{PossibleToReach}|$ が少ない所 (つまり，攻撃元に近い所) で IP トレースバックのパケットが通過したという判定が起こったためだと考えられる．

5.1.3 予備実験 3

攻撃元一台辺りの攻撃パケット数を変化させた場合と攻撃元の数を変化させた場合の，”制約 $c^{PossibleToReach}$ の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”を調べた．

実験環境

使用する IP ネットワークのトポロジは 5.1.3 節 予備実験 1 の条件を満たすものとする．小規模の IP ネットワークを想定して総ノード数は 200 とした．また，攻撃元は総ノード数の 5%である 10 台とした．小規模の IP ネットワークを想定して総ノード数は 200 とした．IP トーレスバックでは，攻撃パケット数を，経路の再構築に必要なパケット数の 1/4 である 10 パケットにした．

結果と考察

攻撃元一台辺りの攻撃パケット数を変化させた場合と，攻撃元の数を変化させた場合の結果をそれぞれ，図 5–5，図 5–6 に示す．図 5–5 の横軸は，”攻撃元一台の攻撃パケット数 ”，両縦軸は図 5–1 と同じである．図 5–6 の横軸は，”攻撃元の数 ”, 両縦軸は図 5–1 と同じである．図 5–5 より，攻撃元一台辺りの攻撃パケット数を増やせば増やす程，制約の数が線形に増えている．これは，多くの Transit ノードで攻撃元の ”攻撃経路に使用された ”という判定が起こったためである．また，それに伴い変数１つ辺りの制約の数が増えている．制約１つにかかわる変数の数が減少傾向にあるため，攻撃元一台の攻撃パケット数を増やせば増やす程，要素数の少ない制約が生成されていることになる．攻撃元一台の攻撃パケット数を増やしたときの $N^{AttackPacketPassed}$ の図を，図 5–7(a) に示す．図中には左上に 1，2，3 と番号付けされた 3 つの Transit-Stub 型のトポロジを示した．図中の矢印はルーティングテーブルの向きを示している．1，2，3 の順に攻撃パケット数が増えたものとする．このように，攻撃パケット数が増えると攻撃元に近い Transit ノードで ”攻撃経路に使用された ”という判定が起こり，要素数の少ない制約が生成されることが予測される．

図 5–6 も，図 5–5 同様の傾向がみられる．しかし，図 5–6 の制約の増加は攻撃元の増加に伴ったものである．”攻撃元一台の攻撃パケット数を増加させたとき ”と ”攻撃元数が増加したとき ”の $N^{AttackPacketPassed}$ の違いを図 5–7(b) に示す．図中の Patern1 の

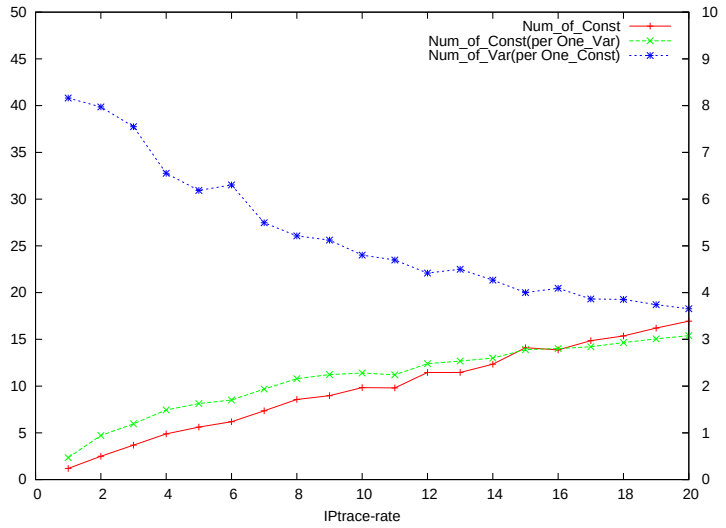


図 5–5　攻撃元一台の攻撃パケット数と ”制約の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”の関係図

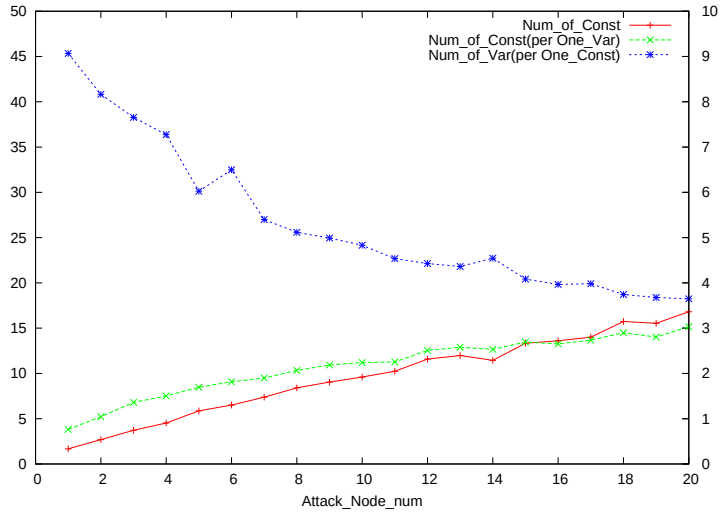
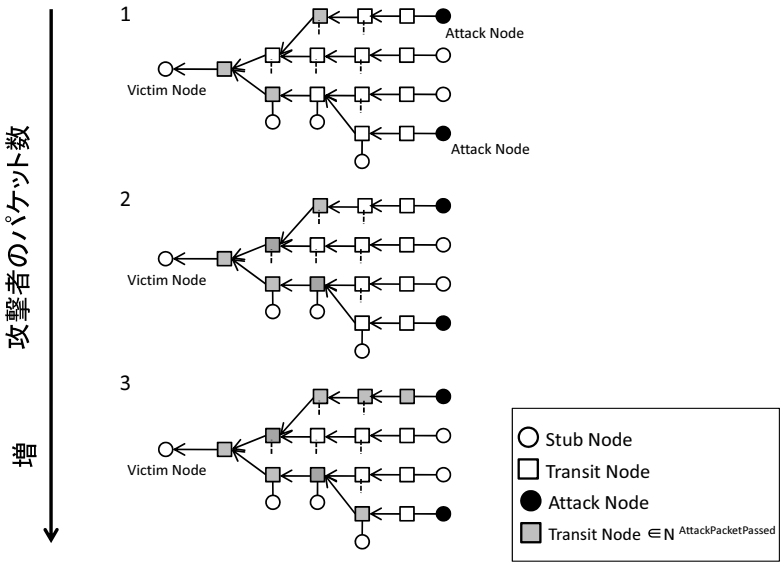
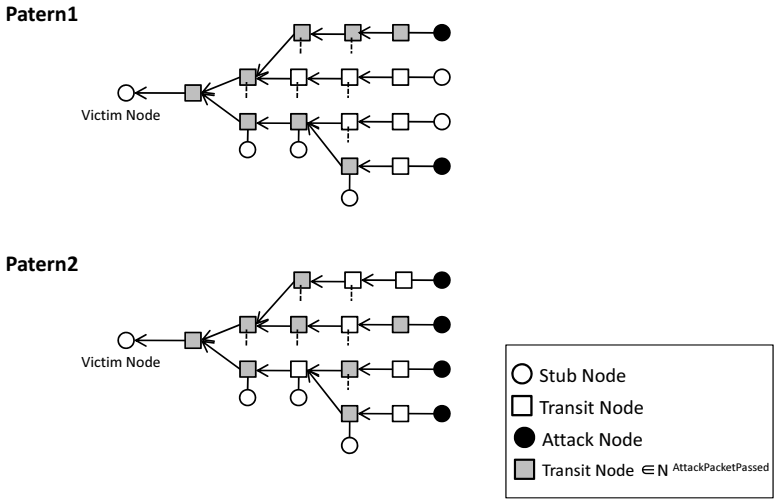


図 5–6　攻撃元の数と ”制約の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”の関係図



(a) 攻撃元一台の攻撃パケット数の変化と $N^{AttackPacketPassed}$ の関係



(b) "攻撃元一台の攻撃パケット数の増加"と"攻撃元数が増加"のときの $N^{AttackPacketPassed}$

図 5–7　$N^{AttackPacketPassed}$ の分布

トポロジが，"攻撃元一台の攻撃パケット数を増加させたとき "に相当し，Patern2 のトポロジが，"攻撃元数が増加したとき "に相当する．Patern1 のでは，被害者と攻撃元の 1 つの経路上に $N^{AttackPacketPassed}$ が多く存在しているのに対して，Patern2 では，1 つの経路上には存在していない．図 5–5 は制約の増加によって解の精度がよくなると考え

られる．また，図 5–6 では攻撃元の増加により制約は増加したが，解の精度は変わらないか，もしくは悪くなると考えられる．解の精度の検証は，次の課題である．

5.1.4 予備実験のまとめ

トポロジのノード数，被害者から攻撃元までの平均 hop 数，攻撃元数，攻撃元一台辺りの攻撃パケット数を変化させたときに，生成された制約 $c^{PossibleToReach}$ の COP に対する問題の性質・規模を検証した．”制約の数 ”，”変数１つ辺りの制約の数 ”，”制約１つにかかわる変数の数 ”の増減は線形であり，指数関数的な増加は見受けられなかった．

5.2 実験１

4 章 提案手法 の解の探索にかかるサイクル数の評価を行うために実験を行った．IP トレースバックの結果が異なると，初期に生成される制約と評価関数が異なる．そのため，解の探索にかかるサイクル数に変化が生じると考えられる．”DDoS 攻撃の攻撃元の数を変化した場合 ”，”攻撃元からの攻撃パケット数を変化させた場合 ”のそれぞれについてサイクル数を測定した．”IP トレースバックによる攻撃元の検出方法”，”ルーティングテーブルの生成方法”，”測定方法 ”は 5.1 節 予備実験と同様とする．

5.2.1 実験環境

実験 1，2，3 の DDoS 攻撃の攻撃パターンは単一型とする．実験 1 は以下の環境で行った．

想定する IP ネットワークのトポロジ

想定する IP ネットワークのトポロジは以下の条件を満たすものとする．

- IP ネットワークのトポロジ:Transit-Stub 型
- Transit ドメイン数:3
- Transit ノード数:10(Transit ドメイン当り)
- Stub ドメイン数:2(Transit ノード当り)

- Stub ドメイン数:1(Stub ドメイン当り)
- 総ノード数:90(うち Stub ノード数:60)

今回のトポロジでは，IP トレースバックによって一台の攻撃元からの攻撃パケットの経路を特定するには約 40 パケット必要である．本提案手法では，IP トレースバックの結果が不十分であることを想定しているため，”DDoS 攻撃の攻撃元の数を変化した場合 ”では，一台の攻撃元からの攻撃パケット数は IP トレースバックで経路の特定にかかるパケット数の 1/4 である 10 パケットとした．”攻撃元からの攻撃パケット数を変化させた場合 ”では，攻撃元の数は 5 台の場合とした．

5.2.2 結果と考察

結果をそれぞれ，図 5–8，図 5–9 に示す．図 5–8 の横軸は DDoS 攻撃の攻撃元の数である．縦軸は 100 回の試行において，”0〜60 サイクル ”，”61〜100 サイクル ”，”101〜サイクル ”で終了した回数である．図 5–9 の横軸は攻撃元からの攻撃パケット数であり，縦軸は，図 5–8 と同様である．

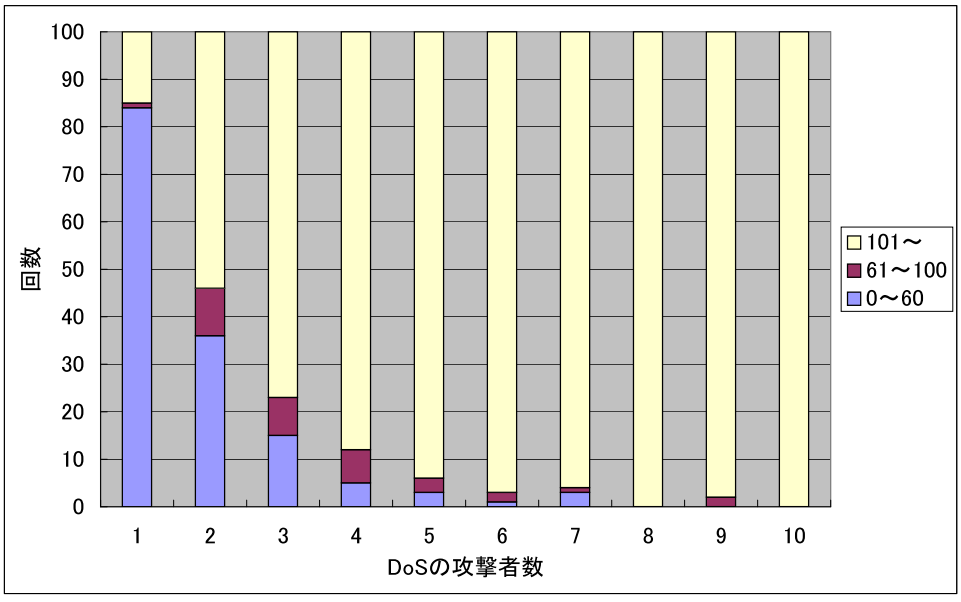


図 5–8　DDoS 攻撃の攻撃元の数を変化させた場合のサイクル数

図 5–8 より，DDoS 攻撃の攻撃元の数を増やすにつれて解の探索にかかるサイクル数が増加している．これは攻撃元が増えると，実行層での「攻撃が止まる」という結果が

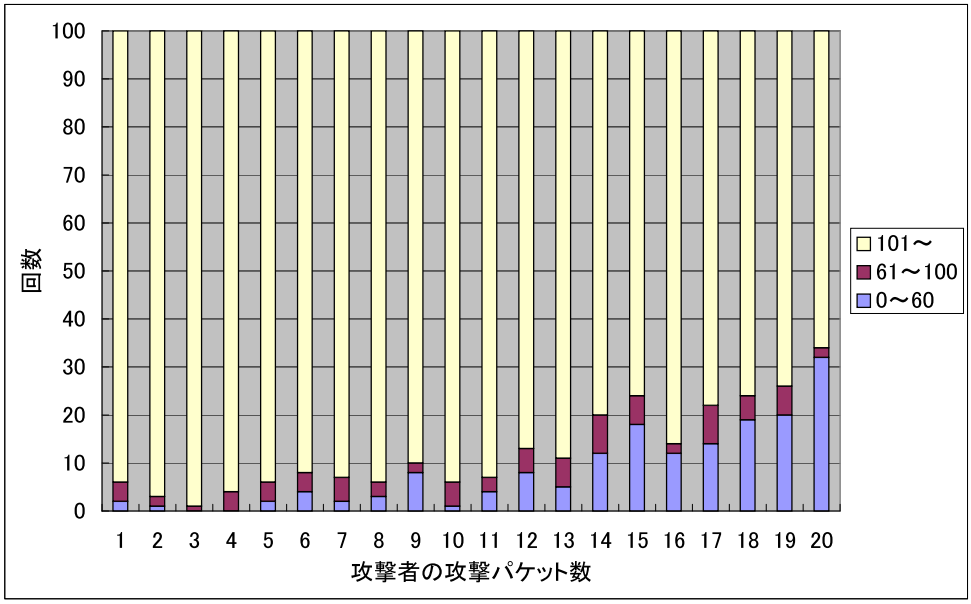


図 5–9　攻撃元からの攻撃パケット数を変化させた場合のサイクル数

少なくなるためだと考えられる.「攻撃が止まる」という結果を得るためには，推定層で，全ての攻撃元を含んだ解を選出しなければならない．これは，攻撃元の数が増えるにつれて困難になると考えられ，そのためサイクル数が増加したと考えられる．

図 5–9 より，攻撃元からの攻撃パケット数が増加するにつれて解の探索にかかるサイクル数が減少している．これは，IP トレースバックのサンプリング数が図 5–7(b)-Patern1(5.1 章) の様に増えたため，初期に多くの制約が生成されたためだと考えられる．この制約により，推定層でより精度のよい推定が行えるようになったためサイクル数が減少したと考えられる．

5.3　実験 2

評価関数 $f^{NumberOfElements}$，$f^{IP\_NumOfHops}$，$f^{IP\_NumOfAttakers}$ は，サイクル動作における実行層での探索順序を決定する．5.2 節 実験 1 では $f^{IP\_NumOfAttakers}$ により，変数 $x(\in X^{PossibleToReach})$ はなるべく変数値 ”0 ”(攻撃元でない) を選択するようになっていた．実験 2 では $f^{IP\_NumOfAttakers}$ を，変数 $x(\in X^{PossibleToReach})$ はなるべく変数値 ”1 ”(攻撃元である) を選択するようにした．評価関数を変更し，解の探索の戦略を変化させた時のサイクル数を調べた．以下に変更した評価関数を示す．関数の引数は $X^{PossibleToReach} =$

$\{x_1, x_2, ...\}$ とする．

$$f^{IP\_NumOfAttakers}(x_1, x_2, ...) = \sum_{x \in X^{PossibleToReach}} x$$

サイクル数の変化を調べるために，”DDoS 攻撃の攻撃元の数を変化した場合 ”，”攻撃元からの攻撃パケット数を変化させた場合 ”のそれぞれについてサイクル数を測定した．”DDoS 攻撃の攻撃元の数を変化した場合 ”の攻撃元からの攻撃パケット数は IP トレースバックで経路の特定にかかるパケット数の 1/4 である 10 パケットとした．”攻撃元からの攻撃パケット数を変化させた場合 ”の攻撃元の数は 5 台とした．

5.3.1 実験環境

実験環境は，5.2.1 節　実験 1 と同様の環境で行った．

5.3.2 結果と考察

結果をそれぞれ，図 5–10，図 5–11 に示す．図 5–10 の横軸は DDoS 攻撃の攻撃元の数である．縦軸は，図 5–8 と同様である．図 5–11 の横軸は攻撃元からの攻撃パケット数であり，縦軸は，図 5–8 と同様である．

図 5–10，図 5–11 では，それぞれ図 5–8，図 5–9 と同じ傾向が見られる．図 5–11，図 5–9 を比較すると，100 サイクル以内で探索が終了してる回数は図 5–10 が全ての場合で多い．しかし，60 サイクル以内で探索が終了している回数は，攻撃元からの攻撃パケット数が 11 パケット以上の場合では図 5–9 の方が多い．特に，攻撃元からの攻撃パケット数が 20 パケットの場合ではこの差が顕著に表れている．実験 1 の評価関数では，なるべく変数値 ”0 ”(攻撃元でない) を選択することから，実行層での「攻撃が止まる」という結果が出た場合に，1 サイクルにおける制約 c^{Stop_1} が多くの変数にたいして生成され，非常に少ないサイクル数 (60 サイクル以下) で探索が終了する場合がある．しかし,「攻撃が止まる」という結果がでる回数は少ないと考えられる．一方，実験 2 の評価関数では，非常に少ないサイクル数で探索が終了する場合は実験 1 の評価関数に比べて少ないが，100 サイクル以下で探索が終了する場合が多い．これは，なるべく変数値 ”1 ”(攻撃元である) を選択することから，実験 1 にくらべると 1 サイクルにおける制約 c^{Stop_1} は少しの変数に対してしか生成されないが,「攻撃が止まる」という結果が多く出るためだと考えられる．以下に実験 1 と実験 2 の各サイクルにおける解探索の傾向を示す．

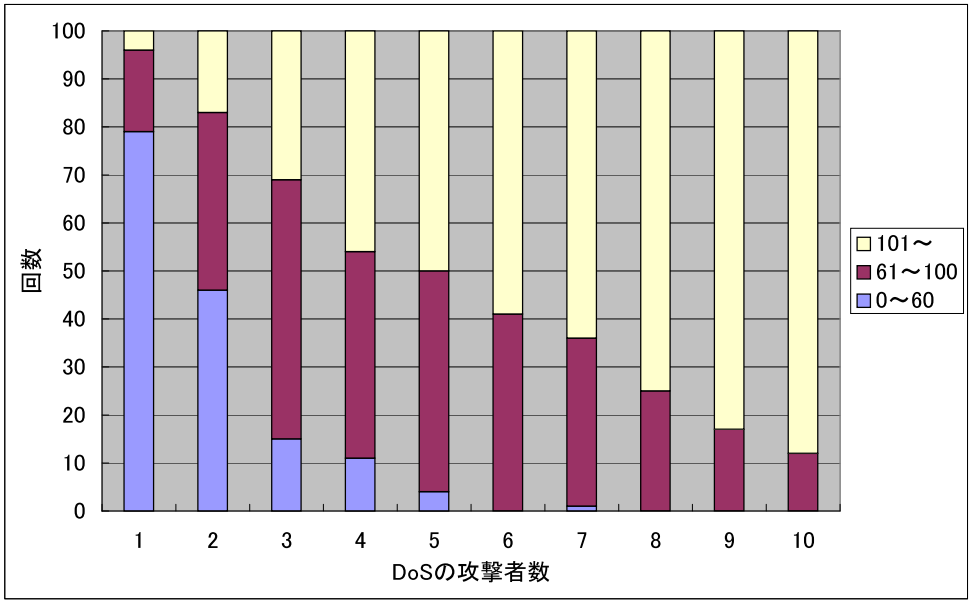


図 5–10　DDoS 攻撃の攻撃元の数を変化させた場合のサイクル数

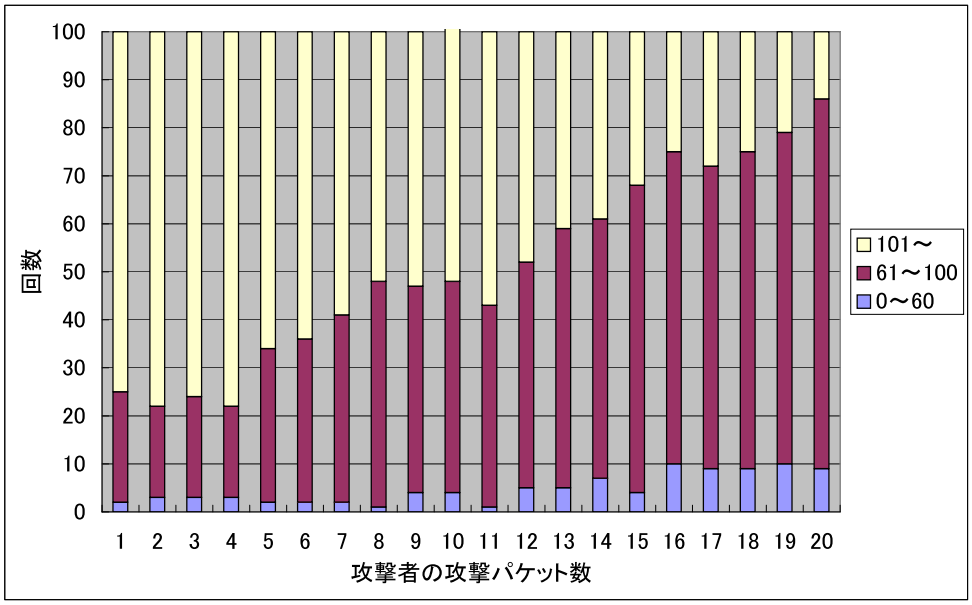


図 5–11　攻撃元からの攻撃パケット数を変化させた場合のサイクル数

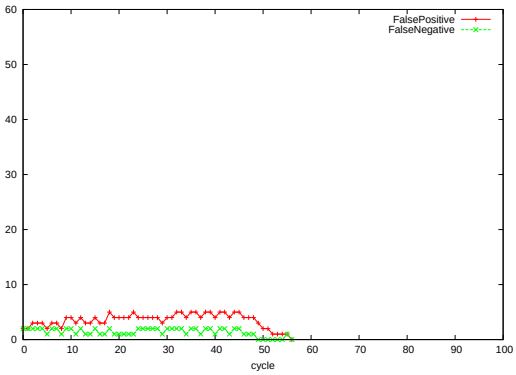
実験 1，2 の探索における FP・FN の収束のしかた

実験１，２のある試行に対して各サイクルの，攻撃元誤判定 ($FalsePositive : FP$)，攻撃元未判定 ($FalseNegative : FN$) を測定した．FP,FN は，以下とする．また，100

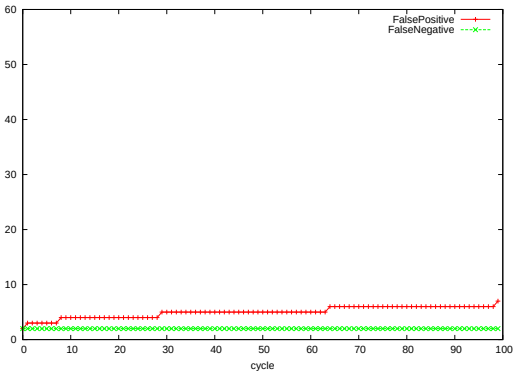
サイクル以内で解を発見できたものを”成功”とし，発見できなかったものを”失敗”とする．結果を実験 1 の評価関数のものを図 5–14　実験 2 の評価関数のものを図 5–13 にしめす.

FP = ”攻撃元である”と判定された通常ノード

FN = ”攻撃元でない”と判定された攻撃元

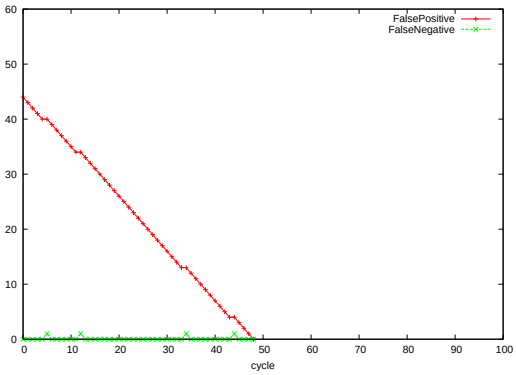


(a) 成功時の FP と FN

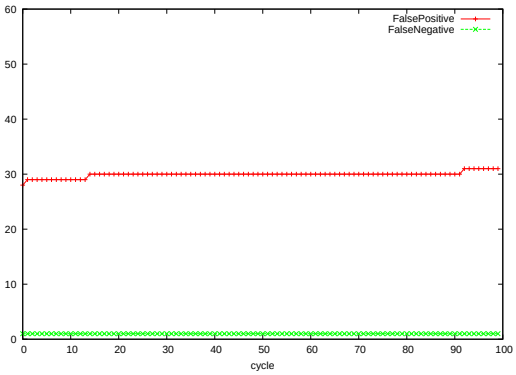


(b) 失敗時の FP と FN

図 5–12　実験１の評価関数における成功時，失敗時における FP と FN



(a) 成功時の FP と FN



(b) 失敗時の FP と FN

図 5–13　実験２の評価関数における成功時，失敗時における FP と FN

図 5–14，図 5–13 の FP より，実験 1 で使用した評価関数では，実験 2 に使用した評価関数にくらべて非常に少ない数の Stub ノードを”攻撃元である”と推定している．また，FN を比べると実験 1 で使用した評価関数では，ほぼに 1 以上になっており，一方，実

験 2 で使用した評価関数は，ほぼ 0 になっている．これは，実験 1 で使用した評価関数では，探索途中で実行層の結果がほぼ「攻撃が止まらない」になっているのに対して，実験 2 で使用した評価関数では，結果がほぼ「攻撃がとまる」になっているのを表している．

5.4 実験 1，2 のまとめ

実験 1 より，攻撃元が増えると全ての攻撃元の特定までに，探索に必要とするサイクル数が増える事がわかった．実験 2 より，初期に生成する評価関数を変更することにより，解の探索順序が変化し，異なった戦略をとる事がわかった．トポロジや攻撃元の数に応じた戦略が必要となる．

5.5 実験 3

Transit ノードにつく Stub ノードの数を変化 (トポロジを変化) させると，探索を行う単位を変化させることによってサイクル数が変化することを確認する．4.6 節 切替え手法 2 では，N を調整することでよりサイクル数の少ない手法で探索を行える事を検証する．また，攻撃元の数を変化させた場合についても検証を行う．この手法には，実行層の結果が「攻撃が止まる」の多い，実験 2 で使用した評価関数を用いた．

5.5.1 実験環境

想定する IP ネットワークのトポロジ

想定する IP ネットワークのトポロジは以下の条件を満たすものとする．

実験環境 1

- IP ネットワークのトポロジ:Transit-Stub 型
- Transit ドメイン数:3
- Transit ノード数:5(Transit ドメイン当り)
- Stub ドメイン数:2(Transit ノード当り)

- Stub ノード数:1(Stub ドメイン当り)
- 総ノード数:45(うち Stub ノード数:30)

実験環境 2

- IP ネットワークのトポロジ:Transit-Stub 型
- Transit ドメイン数:3
- Transit ノード数:5(Transit ドメイン当り)
- Stub ドメイン数:5(Transit ノード当り)
- Stub ノード数:1(Stub ドメイン当り)
- 総ノード数:90(うち Stub ノード数:75)

今回のトポロジでは，IP トレースバックによって一台の攻撃元からの攻撃パケットの経路を特定するには約 40 パケット必要である．攻撃元からの攻撃パケット数は経路の特定に必要なパケットの半分以下である 15 パケットとする．切替え手法 2 における切替え時のパラメータは実験環境 1 では N=1，実験環境 2 では N=5 とする．

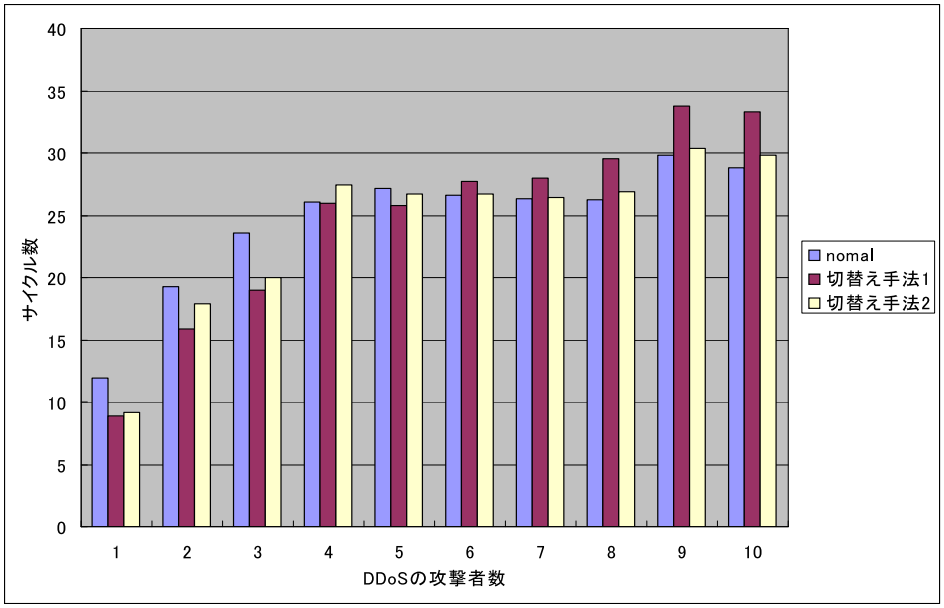
測定方法

攻撃元は Transit ノードに接続される Stub ノードの中に 1 台だけ存在するものとした．被害者はランダムで選択した．被害者と攻撃元の重複はないものとした．10 種類のトポロジを用意し，１種類につき 10 回の試行を行った．(10 種類× 10 回=100 回)

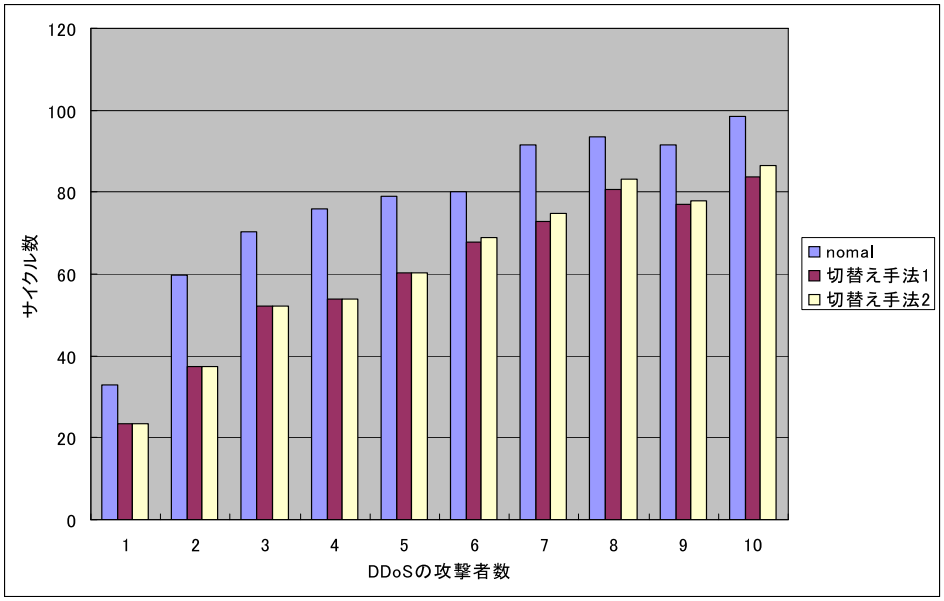
5.5.2 結果と考察

実験環境 1 での結果を図 5–14(a) に，実験環境 2 での結果を図 5–14(b) に示す．図中の”nomal”は，問題の単位の切替えずに Stub ノード単位で探索を行ったものである．

攻撃元の数についての考察を行う．実験環境 1 において，図 5–14(a) の攻撃元の数が 1〜5 の場合では，nomal と切替え手法 1 のサイクル数を比較すると，切替え手法 1 の方がサイクル数が少ない．また，攻撃元の数が 6〜10 の場合では，nomal と切替え手法 1 のサイクル数を比較すると，nomal の方がサイクル数が少ないこのことより，攻撃元の



(a) 実験環境 1 の時



(b) 実験環境 2 の時

図 5–14　探索を行う単位の変更による手法

数が 1〜5 の場合では，Trasnit ノード単位での探索が有効であり，攻撃元の数が 6〜10 の場合では，有効でないといえる．

切替え手法 2 は，攻撃元の数が 1〜5 の場合では nomal と比較すると少ないサイクル数で解を発見できている．また，攻撃元の数が 6〜10 の場合では切替え手法 1 と比較す

ると少ないサイクル数で解を発見できている．切替え手法 2 は，探索の単位が有効な方に変更するので，nomal と切替え手法 1 の両方の特性を持っていると考えられる．

実験環境につていの考察を行う．実験環境 1 図 5–14(a) と実験環境 2 図 5–14(b) の，攻撃元の数が 6〜10 のところに注目する．この範囲で，nomal と切替え手法 1 のサイクル数を比較すると，実験環境 1 では nomal が少なく，実験環境 2 では切替え手法 1 の方が少ない．切替え手法 2 のサイクル数はサイクル数の少ない手法に近い値となっている．実験環境が異なると，手法の有効範囲が異なってくるが，切替え手法 2 ではパラメータ N を調整することで (環境 1 では N=1，環境 2 では N=5)，よりサイクル数の少なくなる手法の特性を持たせることが可能であると考えられる．

5.6 実験 3 のまとめ

攻撃元の数が変化した場合にたいして，探索を行う単位を変更する事によって，攻撃元の数が少ない場合に関してサイクル数の削減が行えた．また，探索の単位を状況に応じて変化をさせることで，その問題の単位にあった探索を行える手法を提案した．切替え手法 2 における，切替え時のパラメータ N の決定方法は今後の課題である．パラメータ N は，攻撃元の数，Transit ノード一台辺りの Stub ノードの数，に影響を受ける．

第 6 章

処理の分散化

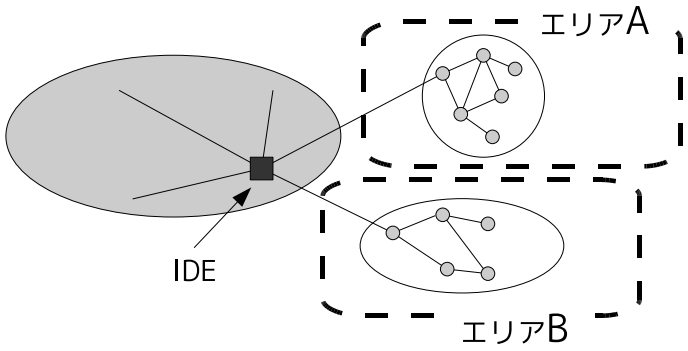


図 6–1　処理の分散化

本章では，推定層の処理の分散化について考察を行う．推定層の処理を並列に行うことによって，処理の高速化が行える．推定層で並列に処理を行うためには，実行層でのパケットフィルタリングにより得られる結果が複数のエリア別に，取得できればよいと考えられる．複数の結果の取得方法として，複数の IDS による観測地点がある場合があげられる．IDS によって，攻撃トラヒックの通過を調べることが可能であれば，図 6–1 のように，ルータへ IDE を設置することにより，実行層でエリア A，エリア B に対して攻撃パケットの通過の有無が観測できる．これにより，エリア A とエリア B の並列探索が行えると考えられる．しかし，攻撃パケット数が少ないときでは攻撃トラヒックを完全に検知できない場合などが考えられる．パケットフィルタリングの結果が不明確である場合の推定手法は今後の課題である．

第 7 章

まとめ

DDoS 攻撃の攻撃元の推定方法として制約最適化問題を用いた解法を提案した．この手法は，DDoS 攻撃の攻撃形態としてパケット数の少ない攻撃を対象としており，攻撃パケットの量に依存せず，攻撃元が少ないときに有効であることを示した．また，2 つの種類の評価関数を用いて実験を行い，評価関数によって異なる戦略を取ることを示した．問題の単位を Transit ノード単位と Stub ノード単位とにわけ，探索の単位を探索効率のよい単位に切替えることで，環境に依存しない高速化手法を提案した．

今後の課題として，DDoS 攻撃の攻撃元の検出問題のモデルに対しては，分散制約最適化問題への拡張，木構造をするトポロジなど特定のトポロジに特化した探索手法の考察などがあげられる．

謝辞

本研究のために多大な御尽力を頂き、日頃から熱心な御指導を賜った名古屋工業大学の松尾啓志教授、津邑公暁准教授、齋藤彰一准教授、松井俊浩助教に深く感謝致します。また、本研究の際に多くの助言、協力をして頂いた松尾・津邑研究室ならびに齋藤研究室の皆様に深く感謝致します。

参 考 文 献

[1] “Denial of service attacks”, *available at* http://www.cert.org/tech_tips/denial_of_service.html.

[2] http://www.cert.org/homeusers/ddos.html.

[3] “Cert advisory ca-1996-21 tcp syn flooding and ip spoofing attacks.”, *available at* http://www.cert.org/advisories/CA-1996-21.html.

[4] 原嶋俊介：“Dos 攻撃に対する逆探知機構の設計と実装”, 慶應義塾大学大学院修士論文, 2001.

[5] 高田有則, 中山雅哉：“改良型 iTrace 手法 (iTrace-pt 手法) の反射型 DDoS 攻撃への適用とその効果”, 情報処理学会 研究報告, 10 2007.

[6] 竹本芳樹, 双紙正和, 宮地充子：“DoS 攻撃に対する偽造耐性をもつ改良パケットマーキング法の提案と評価”, 情報処理学会 研究報告, 03 2007.

[7] 横尾真, エドモンド H. ダーフィ, 石田亨, 桑原和宏：“分散制約充足による分散協調問題解決の決定化とその解法”, 信学論 Vol.J75-D-I No.8 pp.704-713, 08 1992.

[8] Packet Monster: http://www.sfc.keio.ac.jp/ keiji/backup/ids/pakemon/.

[9] S. M. Bellovin: “Icmp traceback messages”, *Internet Draft: draft-bellovin-itrace-00.txt*, 03 2000.

[10] A. Karlin S. Savage, D. Wetherall and T. Anderson: “Practical network support for ip traceback”, *In ACM SIGCOMM 2000 pp. 295-306*, 00 2000.

[11] 坂口薫, 太田耕平, 和泉勇治, 加藤寧, 根元義章：“2 次計画法に基づいたトラヒックパターンの比較による DDoS の追跡”, 信学論 Vol.J85-B No.8 pp.1295-1303, 08 2002.

[12] 角田裕, 内海宏律, 和泉勇治, 根元義章：“トラヒックの最小流量に着目したトラヒックパターンのフィルタリングによる DDoS 攻撃追跡の精度向上方式”, 信学論 Vol.J90-B No.10 pp.969-979, 2007.