修士学位論文

題目

ソフトウェア部品グラフのべき乗則の調査と部品検索への応用

指導教員
井上 克郎 教授

報告者
市井 誠

平成 18 年 2 月 13 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

平成 17 年度 修士学位論文

ソフトウェア部品グラフのべき乗則の調査と部品検索への応用

市井 誠

内容梗概

ソフトウェア部品の再利用を支援し，ソフトウェア開発のコストを削減する為にソフトウェアの部品検索が広く用いられている．ソフトウェア部品とはソフトウェアの構成単位であり，参照などの形で互いに利用関係をもつ．ソフトウェア部品を頂点，利用関係を有向辺としたグラフはソフトウェア部品グラフと呼ばれ，ソフトウェア部品検索においても用いられている．ソフトウェア部品グラフを特長づける要素として各頂点の次数分布があり，単一のソフトウェアに対するグラフにおいてべき乗則が成り立つことが報告されている．べき乗則に従うグラフの性質については，WWW 上のページのリンク関係やソーシャルネットワーク等の分野などで近年活発に研究が進められているが，ソフトウェア部品検索において対象となる大規模なソフトウェア集合を対象としたソフトウェア部品グラフを対象とした研究はこれまで行われていなかった．

そこで本研究では，まず大規模なソフトウェア集合のソフトウェア部品グラフがべき乗則に従うかどうかについて調査を行った．さらに，ソフトウェア部品検索への応用を考慮し，キーワード検索などにより取得した部品群からなる部分グラフの性質についても調査した．その結果，多数のソフトウェアを含む集合も単体ソフトウェアと同様べき乗則に従うこと，キーワード検索によって得られた部分グラフに対しても，比較的少数の部品数で構成されるにもかかわらずべき乗則に従うことがわかった．次に，この結果を踏まえて，ソフトウェア部品検索への応用について考察し，検索結果の順位付け手法の改良を提案した．提案手法では，部品グラフから個々の部品の相対的重要度をもとめる手法 Component Rank を検索結果における部分グラフに対して適用している．さらに，適用実験を通じてその有効性を評価した．

主な用語

ソフトウェア部品グラフ (Software Component Graph)
次数分布 (Degree Distribution)
べき乗則 (Power Law)
部品検索 (Component Retrieval)

目 次

1　まえがき

近年，短期間で大規模かつ高品質なソフトウェアを開発するための技術に対する需要が高まっており，その要素技術の一つとしてソフトウェア部品の再利用がある [7, 16]．ソフトウェア部品とはソースコードやドキュメント，実行ファイルやライブラリなどのソフトウェアの構成単位のことを指す．過去に開発されたソフトウェア部品を再利用することにより開発やテストにかかるコストを削減することができる．ソフトウェア部品の再利用はその有用性が示されているにもかかわらず，実際の開発現場への導入は困難であると言われている．その原因として，開発者が必要とするソフトウェア部品を取得することが困難であることが指摘されている．その問題への取り組みとしてソフトウェア部品検索システムが構築されてきた [39]．ソフトウェア部品検索システムとは，多数のソフトウェア部品を収集・蓄積することで部品リポジトリを構築し，キーワードなどの検索クエリによりリポジトリ中から部品を検索し，取得するシステムである．ソフトウェア部品検索システムを用いることにより，開発者は効率的に必要な部品を取得することが出来る．

ソフトウェア部品間には互いに利用する・利用されるといった利用関係が存在し，ソフトウェア部品を頂点・利用関係を有向辺としたグラフはソフトウェア部品グラフ，もしくは単に部品グラフと呼ばれる．部品グラフは依存関係の解析やモジュール化などのソフトウェア理解および保守を目的とした手法によく用いられる [10, 12]．ソフトウェア部品検索では主として部品に含まれる単語などの部品単体から得られる情報を基にして検索がおこなわれるが，部品グラフを考慮することで，より開発者にとって有益な情報を提供できる．ソフトウェア部品検索システム SPARS-J で用いられるランキング手法 Component Rank 法では，部品グラフを解析し再利用性の観点から部品の重要度を評価することで，単に検索語に適合しているだけでなく，実際の再利用に適した部品を開発者に提示できる [15, 38]．

近年の研究により，頂点の次数分布がべき乗則に従う部品グラフが報告されている．頂点の次数分布はグラフを特徴付ける重要な要素であり，それがべき乗則に従うグラフは様々な分野において注目されている [2, 4, 24]．ソフトウェア部品グラフに関しては，Java および C++ソフトウェアから得られた部品グラフにおいてべき乗則が成り立つことが知られている [22, 34]．これらの研究では単体のオブジェクト指向ソフトウェアに対して調査されているが，その他の部品グラフ，例えば部品検索システムで扱われる大規模なソフトウェアの集合に対してもべき乗則は成り立つと考えられる．また，グラフの数学的な性質を中心に述べられているが，部品グラフのもととなるソフトウェアや含まれるそれぞれの部品の性質との関連を調査することで，次数やその分布があらわす意味を得られると考えられる．

本研究では，様々な部品グラフに対し，次数，特に入力次数のべき乗則性の観点から調査をおこなう．具体的には，ソフトウェア部品検索への応用を考慮し，単体のオブジェクト指

向ソフトウェアに限らず，部品検索時にデータベースとして用いられる大規模なソフトウェア集合や，部品検索により得られる部品群，すなわち大規模なソフトウェア集合のサブセットといった様々な対象に対し調査する．また，データセット間の比較や次数とソフトウェアメトリクスとの比較により，データセットおよび個々の部品に関する，次数から得られる特徴を考察する．

また応用として調査結果に基づくソフトウェア部品検索手法の改良について考察する．一般の検索システムがそうであるように，ソフトウェア部品検索においても検索結果として多数の部品が得られるため，利便性の為にその順位付けが重要である．調査により得られた，大規模ソフトウェア集合の性質や部品検索の結果であるソフトウェア集合のサブセットの性質を基に，部品検索結果の部品グラフの性質を利用した部品検索結果の順位付け手法の改良を提案する．また，実際の適用結果について報告する．

以降，2 節で部品グラフおよびべき乗則について述べ，3 節で実験内容およびその結果について述べる．4 節でソフトウェア部品検索への応用について述べ，最後に 5 節で本研究のまとめと今後の課題について述べる．

2　ソフトウェア部品グラフとべき乗則

2.1　ソフトウェア部品グラフ

一般にソフトウェア部品とは再利用を前提に設計されたソフトウェアの実体 [6] とされるが，本研究ではより一般的にソフトウェアの構成要素をソフトウェア部品として扱う．以降，ソフトウェア部品を単に部品と呼ぶ．

ソフトウェアを構成する部品は互いに独立であり，部品間でメッセージをやりとりし，協調作業をおこなうことで一つの機能を提供する．例えば Java のクラスを部品としてとらえると，他クラスのフィールド参照やメソッド呼び出しがメッセージに相当する．このようなメッセージのやりとりはそれぞれの部品の属性や振る舞いを利用するために行われる．このように，ある部品が別のある部品を利用するとき，部品間に利用関係が存在するという．利用関係の定義は部品の定義によって異なり，UML であれば依存や汎化などの関連が利用関係にあたり，動作するオブジェクト指向プログラムであればフィールド参照やメソッド呼び出しなどが利用関係にあたる．

部品と利用関係を定義することにより，部品グラフが構築できる．部品グラフは部品間の利用関係をグラフ上に表現したもので，各部品を頂点とし，部品間の利用関係を利用する側から利用される側への有向辺で表す．図 1 は二つのソフトウェアシステム X と Y を部品グラフとして表現したものである．X は a から e の 5 つ，Y は f から i の 4 つの部品で構成されており，部品 c は 部品 a および b を利用し，部品 d および e は 部品 c を利用している．同様に部品 h と i は 部品 g を利用し, 部品 g と f はお互いに利用しあっている．

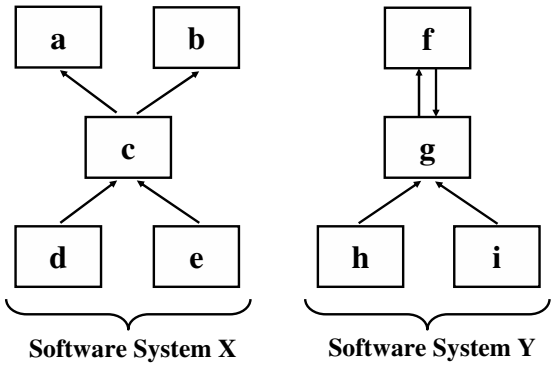


図 1: 部品グラフの例

関連研究　部品グラフはソフトウェアの評価や理解支援，保守支援を目的とした手法において多く利用される．青木メトリクス [3] では HF (Hierarchy Factor), RF (Reference Factor), PF (Polymorphsm Factor), CP (Class Popularity), MP (Message Popularity) の 5 メトリクスが定義されているが，このうち HF はクラスを部品，継承関係を利用関係とした部品グラフから，RF はクラスを部品，参照関係を利用関係とした部品グラフから求められる．また，CP はクラスの利用数，MP はメッセージの利用数であり，部品グラフの次数をクラスの人気度を測るメトリクスとして定義している．Chiricota らは，部品グラフの辺の強さを定義してグラフ分割を行い，ソフトウェア部品のクラスタリングを実現している [12]. Chatzigeorgiou らはは WWW 上の重要なページ群を抽出する HITS アルゴリズム [18] をオブジェクト指向ソフトウェアの設計評価に応用し，責任の集中するクラスを特定してその度合いを定量化している [10]．また，Ma らはグラフの複雑度として用いられる，次数のエントロピーを基にし，ソフトウェアの複雑度を定量化する手法を提案している [20].

2.2　べき乗則

べき乗則とは，度数分布がべき分布に従う性質のことであり，Pareto の法則，Zipf の法則とも呼ばれる [1, 25]. べき乗則は論文の共著関係 [24]，WWW 上のページのリンク関係 [2] など様々な分野において見られる．特に接続次数がべき乗則に従うグラフ (ネットワーク) のことはスケールフリーネットワークと呼ばれる．

べき乗則は以下の確率分布関数で表される．

$$P[X = x] \sim x^{-a} \tag{1}$$

$$\log P[X = x] \sim -a \log x \tag{2}$$

式 (1) は，ある要素 X が値 x をもつ確率 P は，x の $-a$ 乗に比例することをあらわしている．この式の両辺の対数を取ると式 (2) のようになり，ある分布がべき乗則に従うときは，横軸を値，縦軸を度数として両対数軸でプロットした際，点が傾き $-a$ の直線上に並ぶことがわかる．

しかし，現実のデータを両対数軸上にプロットした場合，値の大きな部分でばらつきが生じ，確認やパラメータを求めることが困難であるため，累積度数によるプロットをおこなう．このときに点が直線上に並べば，べき乗則に従っていると言える．これは，式 (1) を累積分布関数あらわすと式 (3) のようになることから分かる．

$$P[X > x] \sim x^{-(a-1)} \tag{3}$$

ソフトウェアにおけるべき乗則　Valverde らは，JDK や Java アプリケーションのクラス図を，クラスおよびインターフェースを部品，汎化や依存といった関連辺を利用関係とする部品グラフとみなした時，接続次数に関してべき乗則が成り立つことを示している [34, 35]．また，べき乗則を示すだけでなく，グラフがスモールワールド性 [36] をもつことも示し，これらの性質は再利用性や理解容易性などを考慮し，最適なソフトウェア設計を行った結果であると考察している．Myers は C++アプリケーションのクラス図がべき乗則に従うこと，および同時にスモールワールド性が成り立つことを示している [22]．Myers の実験では，接続辺を入力と出力に分けて調査し，非対称性をもつことが示されている．また，部品グラフと同様の性質をもつグラフの生成モデルを提案している． Challet らは Linux のソフトウェアパッケージの依存関係グラフおよび関数の呼び出し関係について調査している [9]．以上の研究はすべてソースコードなどをもとにした静的な利用関係から部品グラフを構成しているが，Potanin らは動作する Java プログラムにおけるオブジェクト間のメッセージのやりとりの関係がべき乗則に従うことを示している [26].

3　実験

本節では，まず実験の目的について述べた上で，準備として調査対象とする部品グラフおよび用いる特性量やメトリクスについて述べる．続いて実験内容および部品グラフの解析に用いた手法について述べ，実験結果の説明および考察を行う．

3.1　目的

本研究では，[34, 35, 22] と同様，静的な利用関係に関して調査する．対象言語としてオブジェクト指向言語である Java に加え，手続き指向言語である C 言語も扱う．これまでの研究では，クラス図から得られるような粗い利用関係をもとに部品グラフを構築しているが，本研究ではより詳細な関係を解析し，全ての静的な利用関係に基づく部品グラフを構築して解析する．また，これまでの研究では単体のソフトウェアを個別に調査し，それらの間で共通する数学的性質を発見することに主眼がおかれている．本研究では，ソフトウェア工学の手法，特にソフトウェア部品検索への応用を視野に入れ，既存研究により調査および考察されていない，大規模なソフトウェア集合やその部分集合に対する調査，また個々のソフトウェアの設計との関連に注目した調査などを行う．加えて，次数の値とメトリクス値との比較を行うことにより，次数が個々の部品のどのような性質をあらわしているか調査する．

3.2　準備

3.2.1　ソフトウェア部品グラフの定義

本研究では，Java および C のソフトウェアの部品グラフを対象とした実験をおこなう．それぞれについて，部品および利用関係を定義し，部品を頂点，利用関係を有向辺とした部品グラフを構築する．以下にそれぞれの詳細を示す．

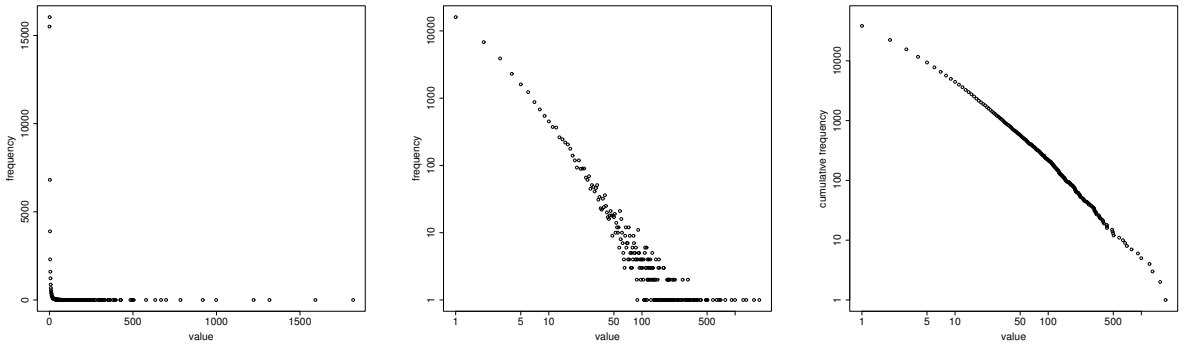
Java

部品 クラスおよびインターフェース．

利用関係 ソースコードの静的解析により得られる全ての利用関係．
詳細については 3.4 節にて述べる．[35, 22] では，利用関係をフィールド宣言や継承関係など一部に限定しメソッド呼び出しなどを無視しているが，本研究では区別せず全て用いる．

C

部品 ソースファイルのコンパイルにより生成されるオブジェクトファイル．
ソースファイルを用いずにオブジェクトファイルを用いるのは，一般に C 言語で



(a) (b) (c)

図 2: べき乗則のプロット

はマクロ記述による関数のラップや，コンパイル時のオプションによる利用関数の場合分けが行われ，正確な利用関係の解析が困難なためである．

オブジェクトファイルとソースファイルは必ずしも一対一対応しないが，多くの場合は 1 つのソースファイルから 1 つのオブジェクトファイルが生成されるため，オブジェクトファイル間の利用関係は，ソースファイル間の利用関係と見なすことができる．

利用関係 シンボルの参照関係．シンボルとは，オブジェクトファイル中に定義もしくは参照される変数および関数を指す．

3.2.2 べき乗則に関する特性量

本実験において用いる部品グラフの特性量について説明する．

次数分布 まず，次数の度数分布を実際にプロットすることにより，視覚的にべき乗則に従うかどうかを確認する．しかし，べき乗則に従う分布を通常の軸上にプロットすると，図 2 (a) のようになり非常に特徴をつかみづらい．そこで，両対数軸を用いてプロットする．このとき，べき乗則に従う分布は 2 節で述べた様に，傾きが負の直線となる．しかし，実際のデータを対数軸上にプロットすると図 2 (b) の様に値の大きい部分にはばらつきが見られる．そこで，ばらつきを吸収するため，図 2 (c) の様に累積度数によるプロットを行う．

また，べき分布に従う度合い，およびべき分布としてのパラメータである 式 (1)〜(3) における a の値を，累積度数プロットの結果を直線へ線形回帰することにより求める．べき分布に従う度合いとしては，R^{*2} を用いる [19]．R^{*2} は自由度調整済み寄与率とも呼ばれ，モデルにより対象データを説明できている割合を示す．

クラスタリング係数　クラスタリング係数とは，グラフ中のあるノードに関し，そのノードに接続しているノード間に辺が存在する割合である．あるノード i のクラスタリング係数は以下の式により求められる．

$$C_i = \frac{2n}{k_i(k_i - 1)} \tag{4}$$

ここで k_i はノード i の次数であり，n は，ノード i に接続する辺をもち，かつ互いの間に辺の存在する 2 ノードの組み合わせの数である．この時は辺の向きは考慮せず，無向辺として扱う．

度数 k ごとの平均クラスタリング係数 $C(k)$ を求めたときに，$C(k) \sim k^{-1}$ となればグラフに階層性があると言われている [27]．[22] では，単体の C++ソフトウェアに対して $C(k)$ をもとめ，階層性を示している．

3.2.3　ソフトウェアメトリクス

次数との比較に用いるソフトウェアメトリクスについて説明する．本実験では以下の 6 種類を用いる

LOC クラスのコメントを除いた行数．一般に LOC と表記した際はコメント込みの行数を指すが，ここではコメントを除いた行数とする．

NOM クラス中で定義されたメソッド数．親クラスから継承したメソッドは含めない．

NOM(PUB) クラス中で定義された public メソッド数．NOM と同様，親クラスから継承したメソッドは含めない．

WMC 重みにサイクロマチック数を用いた，重み付きメソッド和 [11]．クラスの複雑度を表すメトリクスであり， WMC の値が大きいほど複雑なクラスであるといえる．

LCOM5 クラス中のメソッドが属性を利用する割合を表す凝集度メトリクス [8]．あるクラス中の全てのメソッドがそれぞれ全ての属性を利用するとき 0, 全てのメソッドがそれぞれただ 1 個の属性のみを利用する時 1 となる値であり，この値が小さいほど凝集度が高い．LCOM5 は以下の式により求められる．

$$LCOM5 = \frac{\text{実装メソッド数} - \frac{\text{各メソッドで参照される属性数の和}}{\text{属性数}}}{\text{実装メソッド数} - 1} \tag{5}$$

TCC クラス中のメソッドが属性を共有する割合を示す凝集度メトリクス [5, 8]．属性を共有するペア数を，全ての public メソッドのペア数で割った値であり，この値が大きいほど凝集度が高い．

実験では，次数とこれらのメトリクスとの相関を求める．ただし，LCOM5 は実装を持つメソッドが 1 個以下，TCC は実装をもつ public メソッドが 1 個以下のときは定義されないため，LCOM5 および TCC との相関は値が定義される部品のみを用いて相関を求める．

3.3 実験内容

3.3.1 対象データセット

解析対象として，6 種類の Java のデータセットと，3 種類の C のデータセットを用いる．その内容を以下に，またそれぞれの頂点数 (部品数) 辺数，および総行数を表 1 に示す．なお，C に関しては部品とソースコードが正確に一対一対応しないため正確な値では無いが，オブジェクトファイルと同一ディレクトリ中のソースコードの総行数を参考値として示す．

Java

APACHE Apache[30] からのソフトウェア群.

JDK Java の基本ライブラリである JDK1.4

ECLIPSE オープンソースの Java 開発環境である Eclipse[13]

ECLIPSE+JDK Eclipse に，利用するライブラリである JDK を加えたもの．ただし，Eclipse から利用されない部品については除いてある．

NETBEANS Eclipse と同様にオープンソースの Java 開発環境である NetBeans[23].

ALL Apache, SourceForge[28], NetBeans, Eclipse など，様々なオープンソースソフトウェアのリポジトリからのソフトウェア群.

C

FREEBSD FreeBSD 5.4 kernel [31]

LINUX Linux kernel 2.6 [32]

HTTPD Apache HTTP Server

3.3.2 実験 1: 様々なデータセットの調査

ここでは，様々なデータセットに対し，その部品グラフの次数分布や階層性について調査する．まず，全てのデータセットに関して，3.2.2 節で述べた次数分布およびクラスタリング係数を求める．また，入力次数・出力次数それぞれについてソートした時に上位の部品の特徴を確認する．また，以上から得られる共通点などの性質について考察する．

また，以下のようなデータセット間の比較により，分布に影響を及ぼす要因を考察する．

Data set		Number of nodes	Number of links	LOC
Java	APACHE	59486	303775	4.2M
	JDK	11556	107198	1.1M
	ECLIPSE	13941	140678	1.3M
	ECLIPSE+JDK	16666	212284	1.6M
	NETBEANS	13466	69187	1M
	ALL	180637	1808982	14M
C	FREEBSD	11270	120259	(6.1M)
	LINUX	1626	13675	(3M)
	HTTPD	493	3428	(270K)

表 1: データセット

ソフトウェア集合同士 含まれるライブラリなどによる違いを考察する．関連して，利用ライブラリを含まない単体ソフトウェアと，利用ライブラリを含む単体ソフトウェアのデータセットに関しても調査する

具体的には，ALL と APACHE の比較および ECLIPSE と ECLIPSE+JDK の比較をおこなう．

単体ソフトウェア同士 ソフトウェアの設計の違いは部品グラフの違いとして反映され，次数の分布にも影響を及ぼすと考えられる．これより，次数の分布からソフトウェアの設計の特徴を得ることができると考えられる．ソフトウェアの設計の特徴を得ることで，ソフトウェアの設計評価や理解支援を行うことができる．ここでは規模および機能の類似したソフトウェアについて分布を比較し，異なる点に関してその要因を考察する．

具体的には，どちらも Java の開発環境であり，似た規模をもつ ECLIPSE と NETBEANS を比較する．

3.3.3 実験 2: メトリクス値との比較

一般にメトリクス値は，直接的な意味以外にも部品の特性を示すことが多い．例えば行数は部品の規模を表すと同時に複雑度も表す．次数と，3.2.3 節で説明した 6 つのメトリクス値との相関を求めることにより，次数のメトリクスとしての意味を調査する. 相関はスピアマンの順位相関係数を用いて求める．

3.3.4 実験 3: サブセットの調査

ソフトウェア部品検索システムでの検索結果による部品グラフの性質を得るための実験を行う．データセット ALL からいくつかの条件で部品を抽出し，実験 1 と同様に入力次数の分布，出力次数の分布，クラスタリング係数をもとめる．この時の次数は，もとのデータセットでの次数ではなく抽出した部品群で閉じた利用関係，すなわち部分グラフにおける次数である．本実験で用いる抽出条件は以下の 3 つである．

ランダム ランダムに抽出した部品群

9 個のデータセット A1〜A9 を用意する．それぞれの部品数は，A1〜A3 は 1000 部品, A4〜A6 は 3000 部品, A7〜A9 は 10000 部品 とする．

利用関係 ランダムに選んだ部品と利用関係にある部品群

結果の部品数が 100-500 程度, 1000 前後, 5000 前後, 10000 前後となるような部品をそれぞれ 3 個ずつ選び，その部品を利用する部品群をデータセットとして用いる．それらを B1〜B12 とする．以下に，それぞれのデータセットの基点となる部品を，利用される部品数とともに示す.

B1 java.awt.GraphicsEnvironment(163)

B2 org.eclipse.jdt.internal.compiler.problem.ProblemReporter(245)

B3 javax.swing.JPopupMenu(488)

B4 java.io.OutputStreamWriter(972)

B5 org.apache.tools.ant.Project(1036)

B6 org.eclipse.swt.widgets.Control(2678)

B7 java.awt.event.ActionEvent(4941)

B8 junit.framework.TestCase(5430)

B9 junit.framework.Test(5492)

B10 java.util.HashMap(9286)

B11 java.io.IOException(10474)

B12 java.io.File(10545)

キーワード検索 ランダムに選んだキーワードで検索された部品群

結果の部品数が 100-500 程度, 1000 前後, 5000 前後, 10000 前後となるようなキーワードをそれぞれ 3 個ずつランダムに選んで検索した結果の部品群をデータセットとして

用いる．それらを C1〜C12 とする．以下に，用いたキーワードを検索結果の部品数とともに示す.

C1 getsummary (129)

C2 probe(149)

C3 invalid_argument(297)

C4 completely(829)

C5 labels(1002)

C6 maybe(1191)

C7 tool(4385)

C8 binding(4982)

C9 possible(5597)

C10 getstring(8938)

C11 voluntary(9150)

C12 throwable(10537)

また，以上のデータセットに対する実験に加え，SPARS-J の公開デモシステム [29] において実際に検索されたキーワードをもとに，キーワード検索による部品群の，部品数に応じた入力次数の分布のパラメータの変化を調査する．キーワードの一部を以下に示す.

```
accept account action ActionForm ActionForward ActionServlet
graphics groovy GUI Hash HashMap Hibernate
```

ただし，明らかに意味を持たないキーワードや，検索結果が 0 となるキーワードは除いた.

3.4 解析に用いる手法

Java Java ソースコードの解析，および利用関係の解析にはソフトウェア部品検索システムである SPARS-J の解析部を用いる．SPARS-J は Java ソースコードを解析し，クラスおよびインターフェースを部品として登録する．また，部品間の静的な利用関係を解析する．SPARS-J で解析される利用関係は以下の 7 通りである.

1. 継承：子クラスが親クラスを利用.
2. 抽象クラス実装：実装クラスが抽象クラスを利用.

3. インタフェース実装：実装クラスがインタフェースを利用.

4. 変数宣言の型：変数宣言したクラスが変数型を定義するクラスを利用.

5. インスタンス生成：インスタンス化したクラスが生成したクラスを利用.

6. フィールド参照：参照元クラスがフィールドを定義するクラスを利用.

7. メソッド呼出し：呼出し元クラスがメソッドを定義するクラスを利用.

本研究では，これらを区別せずに利用関係として用いる.

解析の流れ

1. 対象 Java ファイルの解析および登録

   SPARS-J 登録部が対象 Java ソースコードをデータベースに登録する．同時に，利用関係の解析および各種メトリクス値の計算が行われる.

2. 解析内容の出力

   SPARS-J 自体は検索システムであり，データベースの内容を直接参照する機能を持たないため，本研究では SPARS-J のデータベースおよび検索部と接続し，部品情報を得るツールを SPARS-J 検索部の一部として設計および実装した．このツールでは，クエリを受け付け，入出力次数やメトリクスの内容をデータベースから出力する．また，サブセットの抽出および部分グラフの構築，後述するクラスタリング係数の計算を行う.

   また，SPARS-J では検索時にキーワードに一致する索引語の種類および大文字/小文字の区別の有無，形態素解析の有無を指定できる．実験 3 にて部品群をキーワード検索により得る時は，全ての種類の索引語から，大文字小文字の区別および形態素解析をせずに検索を行った.

3. 出力情報の分析

   出力された情報を, 統計解析ソフト R [33] に入力し，度数分布図の出力や相関の計算などをおこない分析する.

C　オブジェクトファイル内にて定義・参照されているシンボル名は UNIX コマンド `nm` を用いて取得することができる [14]．`nm` は UNIX 上でコンパイルされたオブジェクトファイルを入力とし，そのオブジェクトファイル内で宣言されたシンボルを，その種類や定義の有

無に関する情報とともに出力する．nm により，個々のオブジェクトファイルに関し，そのオブジェクトファイル内に実装をもつ関数および変数は「定義が存在するシンボル」として出力され，他のオブジェクトファイルの関数および変数の参照は「定義の存在しないシンボル」として出力される．

以上のように nm により，それぞれのオブジェクトファイルにて定義されるシンボルおよび利用されるシンボルを得ることができるが，利用されるシンボルがどのオブジェクトファイルで定義されているのか，ということは解析されない．そのため，本研究では nm の出力結果を入力，解析されたオブジェクトファイル間の利用関係を出力とするツールを設計および実装して解析に用いた．

解析の流れ

1. 対象オブジェクトファイルの解析

   nm により対象となるオブジェクトファイル群のシンボル情報を抽出し，テキストファイルに出力する

2. 利用関係の解析

   1. の結果を入力とし，実装したツールによりオブジェクトファイル間の利用関係を解析し，出力する．

3. 出力情報の分析

   R を用い，Java の場合と同様に分析する．

3.5 結果

3.5.1 実験 1: 様々なデータセットの調査

全てのデータセットに対する，入力次数の累積度数分布を図 3 に，出力次数の累積度数分布を図 4 に, クラスタリング係数のプロットを図 5 に示す．また，それぞれの分布の特性値を表 2 および 3 に示す.

なお，図 4 からわかるように，出力次数の分布は値の大きな部分でのみ，べき乗則に従っていることがわかる．このため，特性値は出力次数 10 以上にて求めた.

入力次数，出力次数上位の部品について，特徴的なものをデータセットごとに以下に示す.

APACHE 入力次数の上位: ログ出力クラス他，共通して利用されるユーティリティクラス.

出力次数の上位: ALL でも上位であったプールクラスおよびバイトコード解析ツールの解析部．ALL などと比較し，上位部分に不自然に部品数の多い部分があらわれている．これは，その出力次数をもつ，ほぼ同内容のクラスが複数存在するためである．

JDK 入力次数の上位: ALL と同様，String クラス，Object クラス．

出力次数の上位: Java GUI フレームワークである AWT の実装クラス．

ECLIPSE 入力次数の上位: 独自の GUI 実装である SWT のクラスや例外クラス，ソフトウェア内のリソースにアクセスするためのインターフェースクラス．

出力次数の上位: Java エディタの実装クラスや，AST(抽象構文木) の実装クラス

ECLIPSE+JDK 入力次数の上位: JDK と同様，String や Object をはじめ JDK のクラス．

出力次数の上位: ECLIPSE と同様のクラス群により占められる．

NETBEANS 入力次数の上位: ロケールごとの表示メッセージを扱うクラスや，IDE 内でのデータ構造を扱うクラス．

出力次数の上位: 仮想ファイルシステムやエディタの実装クラス．

ALL 入力次数の上位: 分布の図から，極端に次数の大きなクラスが存在することが確認できるが，java.lang.String および java.lang.Object がそれらに該当する．それぞれ文字列を扱うクラス，全てのクラスの親クラスであり，Java プログラム中でほぼ必ず利用されるクラスである．また，上位はほとんど JDK のクラスにより占められる．

出力次数の上位: オブジェクトのインスタンスを プールするクラスおよびグラフ描画ツールののサンプルアプリケーション．

FREEBSD 入力次数の上位: メモリ領域の管理をおこなう関数を含むオブジェクトファイルや，ファイルを扱う関数を含むオブジェクトファイル．

出力次数の上位: 認証を扱うモジュールのオブジェクトファイルやファイルシステムの修復を行うコマンドのオブジェクトファイル．

LINUX 入力次数の上位: メモリ管理をおこなう関数を含むオブジェクトファイルや，リソースの管理をおこなうオブジェクトファイル，ログメッセージを扱うオブジェクトファイル．

出力次数の上位:ファイルシステムの実装のオブジェクトファイルや，様々なカーネルモジュールのオブジェクトファイル．

Data set		Incoming			Outgoing		
		a	$p-value$	R^{*2}	a	$p-value$	R^{*2}
Java	APACHE	2.37 ± 0.0109	1.01E-259	0.981	3.91 ± 0.0455	3.20E-69	0.981
	JDK	2.11 ± 0.00866	1.02E-201	0.987	3.90 ± 0.0663	1.49E-59	0.957
	ECLIPSE	2.20 ± 0.0163	1.76E-172	0.955	3.76 ± 0.0664	1.92E-67	0.942
	ECLIPSE+JDK	2.11 ± 0.0100	4.49E-249	0.976	3.83 ± 0.0695	7.50E-71	0.933
	NETBEANS	2.20 ± 0.00788	2.69E-164	0.994	4.36 ± 0.0645	7.07E-46	0.981
	ALL	2.02 ± 0.00145	0	0.999	4.38 ± 0.0426	1.11E-123	0.977
C	FREEBSD	2.01 ± 0.0114	1.18E-182	0.971	4.34 ± 0.0892	7.10E-50	0.949
	LINUX	1.96 ± 0.0244	1.93E-59	0.944	3.49 ± 0.0485	6.06E-43	0.982
	HTTPD	1.95 ± 0.0335	7.12E-30	0.947	2.95 ± 0.149	1.26E-10	0.899

表 2: 実験 1: 次数分布の特性量

Data set		slope	$p-value$	R^{*2}
Java	APACHE	1.03 ± 0.0361	1.12E-87	0.725
	JDK	0.793 ± 0.0206	1.15E-102	0.865
	ECLIPSE	0.789 ± 0.0229	1.99E-103	0.807
	ECLIPSE+JDK	0.810 ± 0.0166	2.83E-152	0.878
	NETBEANS	0.804 ± 0.0249	4.59E-72	0.866
	ALL	0.927 ± 0.0135	0	0.867
C	FREEBSD	0.967 ± 0.0416	3.85E-64	0.685
	LINUX	0.603 ± 0.0289	6.26E-39	0.805
	HTTPD	0.525 ± 0.0323	1.24E-22	0.824

表 3: 実験 1: クラスタリング係数

HTTPD 入力次数の上位: メモリ管理や文字列ユーティリティ，ログを扱うオブジェクトファイル

出力次数の上位: HTTP の実装をもつオブジェクトファイルや CGI を扱うオブジェクトファイル.

共通点に関する考察　入力次数の分布はどれもべき乗則にしたかっていると言える．ALL, APACHE では入力次数の対数分布がほぼ直線に並ぶのに対して，単体ソフトウェアである JDK, ECLIPSE, NETBEANS では，値の大きい部分でやや傾きに変化が見られる．複数のソフトウェアの集合の場合，JDK などの共通して利用されるクラスへ辺が集中し，極端に大きな値をもちやすいからであると考えられる．JDK を含むデータセット ALL, JDK, ECLIPSE+JDK では，JDK の中でも特に利用される Object および String が極端に大きな入力次数をもつことがわかる．また，このようなデータセットでは R^{*2} の値も高く，理想的

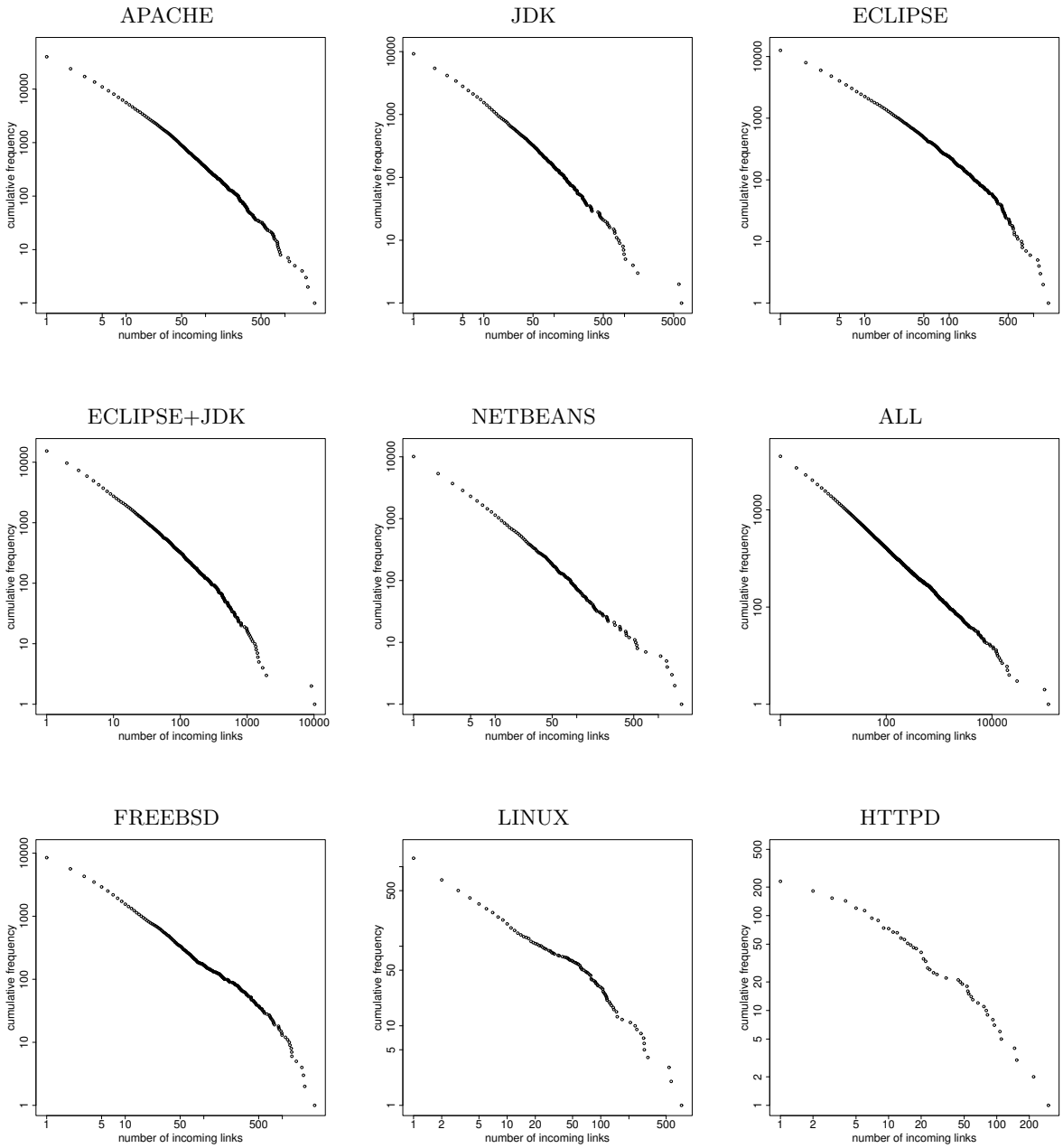


図 3: 実験 1: 入力次数の累積度数分布

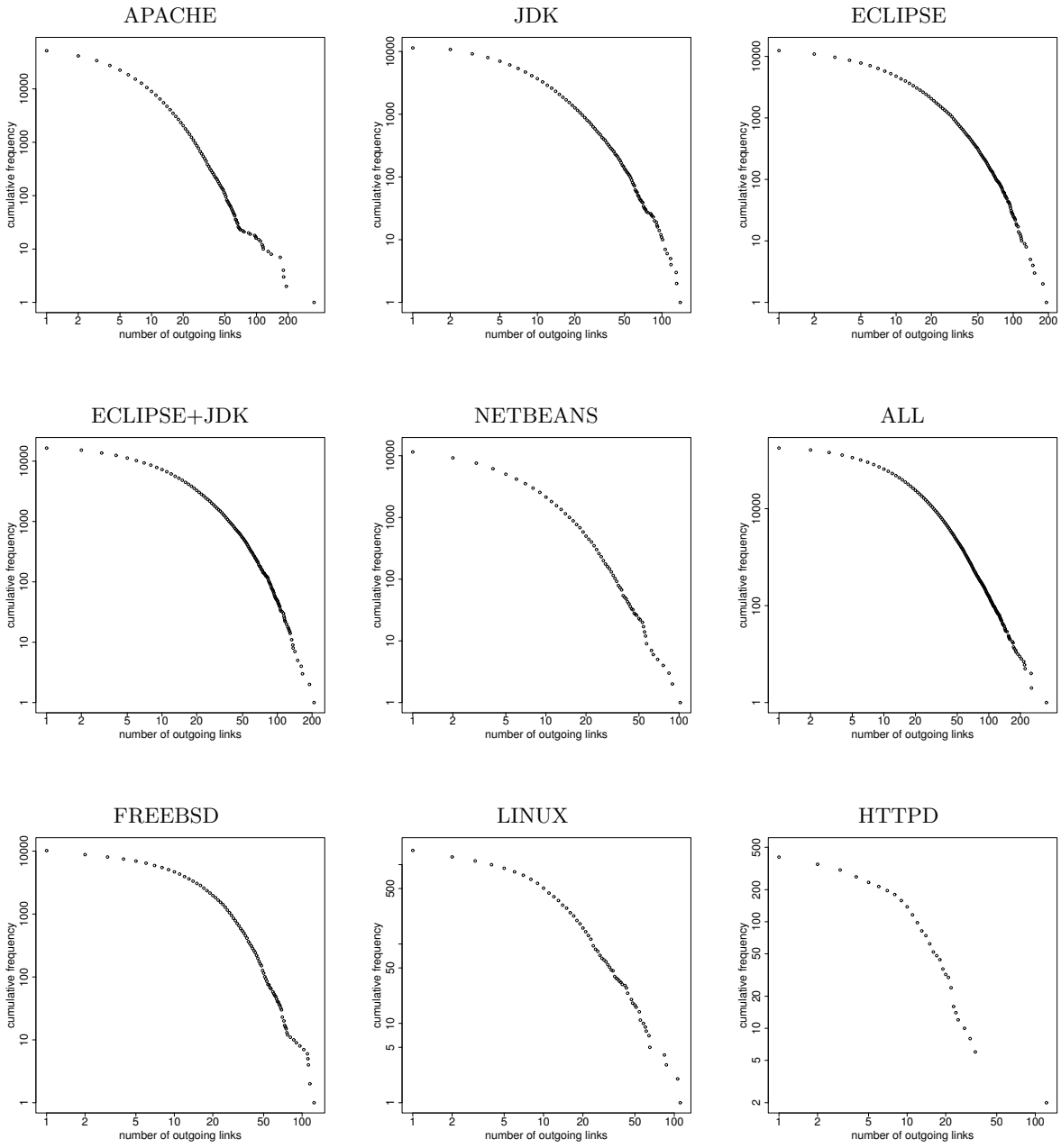


図 4: 実験 1: 出力次数の累積度数分布

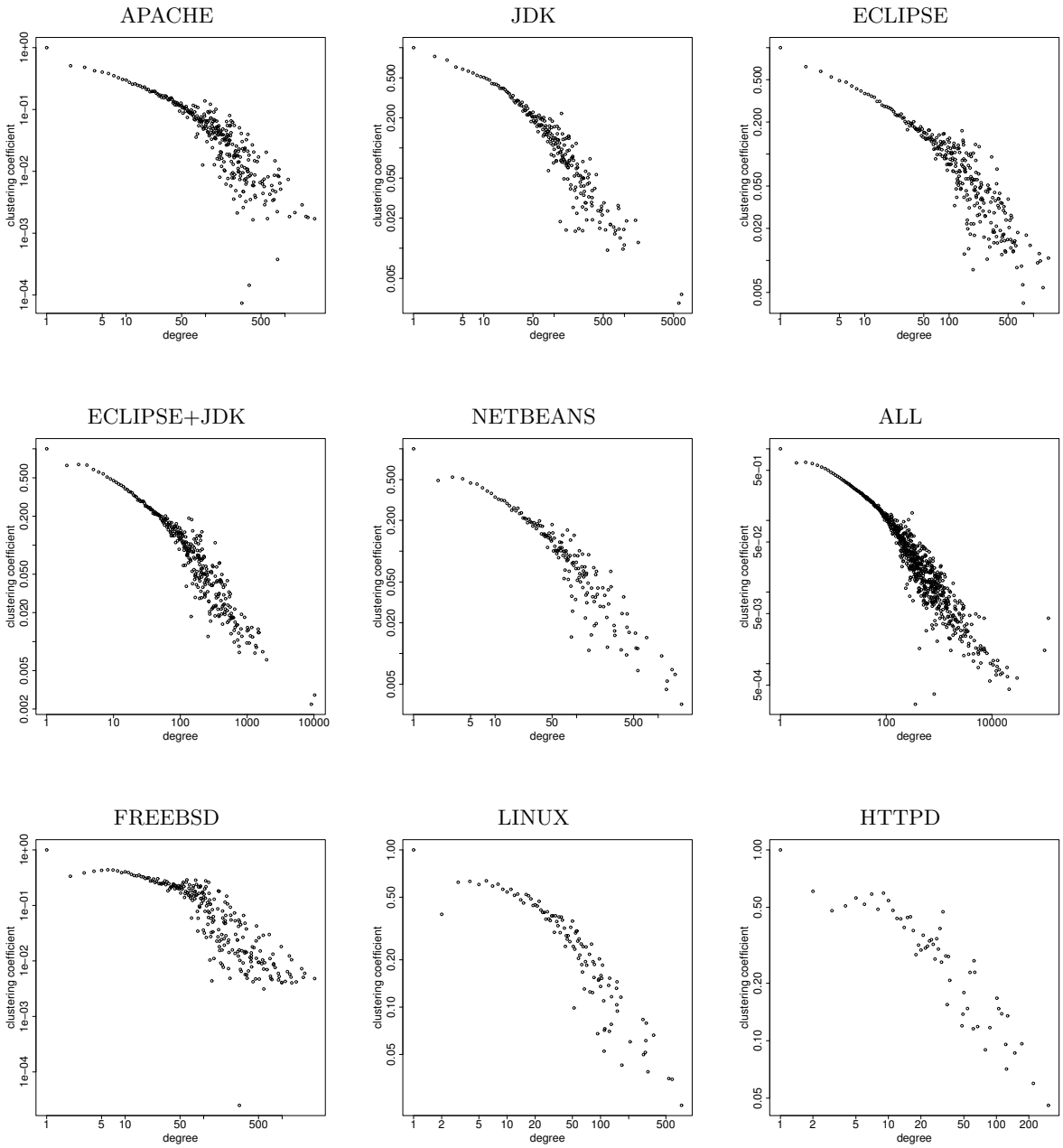


図 5: 実験 1: 次数ごとのクラスタリング係数

なべき分布に近いと言える．また，C のオブジェクトファイルもまた Java のクラスと同様のべき分布に従うことがわかった．その上位は Java の部品が抽象的なクラスであったのに対し，メモリ管理やリソース管理など，低レベルな処理を行う部品が多く見られたが，これは，Java はメモリ管理等の低レベル処理がバーチャルマシンにより隠蔽され，利用関係として出現しないことによる違いであると考えられる．傾きについてはデータセットに関わらずほぼ 2 となり，データセットの比較の際に一般的な値として利用できると考えられる．

出力次数については，値の大きい部分に関してはべき乗則に従っていると言えるが，次数 0 の付近で傾きに変化がみられる．この傾向は [22] では観測されていない．利用関係の取得をより詳細化した結果であると考えられる．この分布は，0 付近では対数正規分布に従い，値の大きな部分ではべき乗則に従う分布である Double-Pareto 分布 [21] であると考えられる．この分布はファイルサイズに見られる分布であり，部品内の記述内容から得た出力次数がこの分布に従うことは妥当であると考えられる．また，回帰直線の傾きは 3 〜4 となり，多くのクラスを含む ALL, APACHE ではより大きな値となっている．傾きが大きいことは全体の部品数と比較して次数の大きな部品が少ないことを示しており，データセットを大きくしても値域に大きな変化はないことを示していると考えられる．

次数ごとのクラスタリング係数のプロットの傾きは，APACHE および ALL にてほぼ 1 であり，多数のソフトウェアを含むことから構造の階層性が顕著に表れていると考えられる．ソフトウェア単体に関してはやや値が小さいが，階層性が見られると言える．C 言語のデータセットに関しては，FREEBSD に関しては極めて 1 に近い値が得られたが，LINUX および HTTPD では低い値となった．LINUX や HTTPD はそれぞれ単体のカーネルおよびアプリケーションなのに対し，FREEBSD ではカーネルに加えユーザランドのコマンド群が含まれることが影響していると考えられる．また，LINUX のオブジェクトファイルを見たとき，ソースコードと一対一対応せず，複数のソースコードから 1 オブジェクトファイルが得られる場合が多く見られた．そのため，小さい単位での利用関係が形成されず，階層性が現れなかったと考えられる．

データセット同士の比較

ALL と APACHE 入力次数に関しては，どちらも高い R^{*2} をもち分布形状もほぼ直線となるが，ALL では横軸，つまり入力次数の最大値が 10000 以上であるのに対して，APACHE では最大 1000 程度と含まれるクラス数の差以上の差が出ている．それに対して，出力辺数に関しては，それぞれのデータセット単体では分布の形状から同様の傾向ををもつようにみえるが，互いに比較した時には横軸のスケールに大きな差は見られない．このことから，出力次数はべき分布に近い性質をもちながらも何らかの上限をもつことがわかる．これは，出力次数の多いクラス，すなわち多くのクラスへの

利用関係を持つクラスは必然的にそれ自身が巨大となり，再利用性や保守性などで問題がおきるために現実には限度があることを反映していると考えられる．

ECLIPSE と ECLIPSE+JDK 入力次数については，ECLIPSE では上位のクラスが，ECLIPSE+JDK では JDK のクラスに押し下げられている．JDK のクラスは様々なソフトウェアで利用されるだけでなく，単体ソフトウェアの中でも多く利用されることがわかる．逆に，出力次数の上位には ECLIPSE と ECLIPSE+JDK で差が少ない．出力次数は，部品中で利用されるクラスがどの程度データセット中に含まれるかということに依存するため，データセット中で多くを占める Eclipse のクラスが上位に来たと考えられる．また，JDK 単体では上位に来る AWT などのクラスを Eclipse が必要としていないことも要因の一つであると考えられる．

ECLIPSE と NETBEANS 入力次数に関する特徴として，どちらも上位付近で傾きが変化し，大きくなるという点が見られる．変化しはじめる点は，NETBEANS が約 1000 付近なのに対し，ECLIPSE は小さく約 500 である．このことは R^{*2} の値の差としてもあらわれている．上位で傾きが大きくなることは，上位のクラスも極端に大きな値は持たないことを示す．また，最大値は NETBEANS の約 1900 に対して ECLIPSE は約 1500 であり，こちらも ECLIPSE の方が小さい．逆に，出力次数では分布の形状はほぼ同じであるのに対して，最大値は NETBEANS の約 100 に対して ECLIPSE は約 200 と大きな値をもつことから，全体として ECLIPSE の方が利用部品数が大きな傾向にあるといえる．

以上より，ECLIPSE の方が入力辺の最大値が小さく，比較的大きな値をもたない分布であることから，特定のクラスへの利用関係の集中度は低いと言える，また，それぞれ総行数や行数の分布にそれほど差がないのに対し，出力次数では最大値で 2 倍程度という大きな差が見られることから，同じ行数のコードであっても ECLIPSE はより多くの部品を利用していることがわかる．これらのことから，NETBEANS と比較した ECLIPSE のクラス設計の特徴としては，それぞれのクラスに与えられている責任が細分化され，多くのクラスの相互作用で処理をおこなう点があると考えられる．実際，ECLIPSE の入力次数の上位には抽象クラスやインターフェースが多く，中心的なクラスは抽象度が高い役割をもっていることが分かる．また，入力次数上位の部品の責任の度合いの比較するため，それぞれの public メソッド数の比較をおこなった．クラスの public メソッドは他のクラスに提供する機能であり，個数が多ければそのクラスが持つ役割や責任が大きいと考えられる．入力次数上位 10 件に関する public メソッド数を表 4，Boxplot により比較した結果を図 6 に示す．これより，ECLIPSE の方が public メソッド数が少ない傾向がわかる．このことからも，ECLIPSE の入力次

Order by incoming links	Number of public methods	
	ECLIPSE	NETBEANS
1	6	21
2	5	39
3	8	49
4	53	14
5	9	11
6	33	42
7	0	32
8	43	5
9	29	50
10	5	14
MEAN	19.1	27.7
MEDIAN	8.5	26.5

表 4: 実験 1: 入力次数上位部品 public メソッド数の比較

数上位のクラスは NETBEANS と比較して細分化された役割や責任をもっていると言える

3.5.2 実験 2: メトリクス値との比較

データセット ALL および ECLIPSE に対し，次数とメトリクス値の順位相関係数を求めたものを表 5 に示す.

これより，入力次数は他のメトリクスとほとんど相関が無いことがわかる．これに対し，出力次数は，行数およびメソッド数，複雑度と相関が高いことが分かる．実験 1 で出力次数の上位の部品は規模の大きな実装や複雑な実装をおこなっている傾向があったことを考慮すると，出力次数により，部品の規模や複雑度をあらわすことが出来ると考えられる．また，出力次数は凝集度 LCOM5 とやや相関が見られる． LCOM5 は凝集度の欠如をあらわしており，その相関の高さは出力次数が高い部品ほど凝集度が低いことを示す．これは，規模の大きな部品は凝集度が低くなりがちであることを反映していると考えられる．

直感的には，入力次数と出力次数は負の相関を持つと考えられるが，調査結果ではそのような傾向は見られなかった．実際に入力次数を横軸・出力次数を縦軸にプロットすると図 7 のようになった．これより，入力次数と出力次数のいずれかが高い部品は，もう一方が低い傾向があるが，その逆，すなわちいずれかが低い部品はもう一方が高いとは限らないことが分かる．またいずれも低い部品が大多数を占めることが分かる．

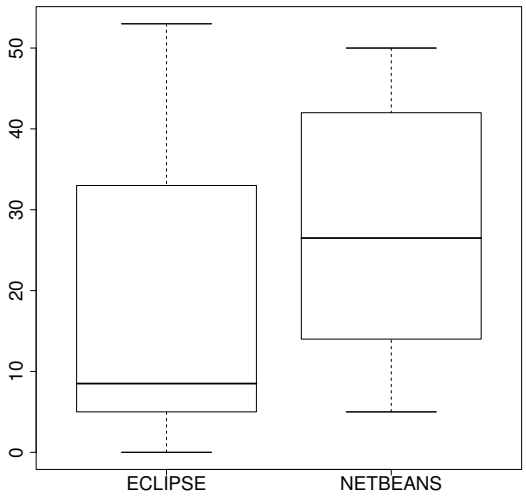


図 6: 実験 1: 入力次数上位部品 public メソッド数の比較

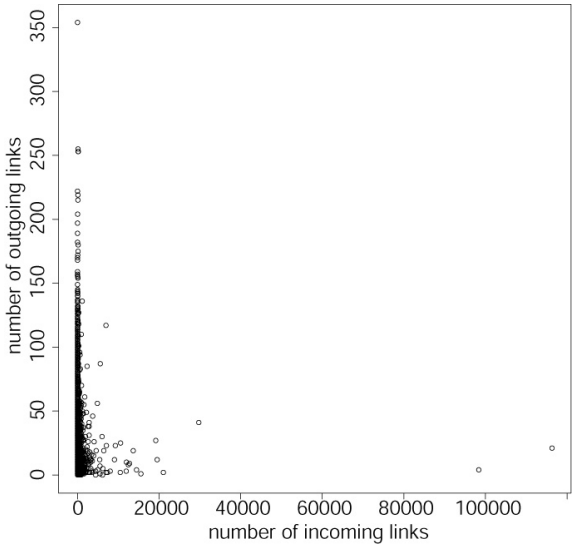


図 7: 実験 1: 入力次数と出力次数の比較

ALL

	Outgoing links	NCLOC	NOM	NOM(PUB)	WMC	LCOM5	TCC
Incoming links	0.004	0.071	0.239	0.178	0.075	0.120	0.133
Outgoing links	-	0.826	0.641	0.565	0.750	0.399	0.011

ECLIPSE

	Outgoing links	NCLOC	NOM	NOM(PUB)	WMC	LCOM5	TCC
Incoming links	-0.081	0.055	0.217	0.185	0.010	0.232	0.075
Outgoing links	-	0.726	0.635	0.514	0.679	0.409	-0.101

表 5: 実験 2: 次数とメトリクス値の相関

3.5.3 実験 3: サブセットの調査

実験 3 のそれぞれのデータセットの部品数および辺数を表 6 に示す．また，入力次数の分布を図 8 に，出力次数の分布を図 9 に，クラスタリング係数を図 10 に示す．なお図はデータセット A1,A4,A7,B1,B4,B7,B10,C1,C4,C7,C10 のみについて示している．また，特性量の一覧を表 7 に示す．なお特性量は，出力次数を省略し，より特徴を表す入力次数のみ示している．

ランダム 無作為に選んだ部品群は全体的に辺数が少なく，また 1000 部品では，べき分布とは言えないことがわかる．部品数が 5000 部品, 10000 部品ではべき分布の特徴を表す．しかし，クラスタリング係数を見ると，傾きの値のばらつきが大きく，R^{*2} の値も低い．このことから，もとの性質をあまり受け継がず，ランダムに近い性質をもつことがわかる．

利用関係 全体的に辺数が多く，頂点辺りの次数では全体に一番近い．1000 部品未満でもべき分布であると言える．ただし，全体的に a の値がもとのデータセットである ALL と比較して小さい．これは利用関係が一部の部品に集中する度合いが高いことを示している．もとのデータセットでは利用関係にある部品同士が利用関係にあるという性質があり，それが現れていると考えられる．しかし，部品数が小さいデータセットでは，クラスタリング係数の傾きの値が小さい．これは階層性が見られないことを示しており，もとのデータセットとやや性質が異なることを示している．

キーワード検索 利用関係のデータセットと同様，1000 を切る部品数でも入力次数にべき乗則が成り立つことが分かる．それに加え，やや R^{*2} の値が低いが，クラスタリング係数もよく似た特性を持っている．これは，関連のある部品同士は同じ語を含むことが多く，それらがまとめて部品グラフに含まれると利用関係が存在する確率が高いためであると考えられ

る．また，検索キーワードをクラス名やメソッド名として含む部品がヒットするとともに，その利用クラスも同時にヒットするために利用関係が形成されていると考えられる．

ただし，詳しくクラスタリング係数をみていくと，データセットにより異なる特性をもつことがわかる．C8 では傾きが小さく，C11 では R^{*2} が低い．キーワードが一般的な単語で，特定のドメインやライブラリに限定されない場合には利用関係がばらつき，階層性も低くなると考えられる．逆に C5 は部品数は 1000 付近で小規模であるが全体と近い性質をもつ．これはキーワードが labels という GUI 部品に用いられる語であり GUI 関係のライブラリおよびそのアプリケーションが多く含まれるからであると考えられる．

以上より，どの場合も，ある程度以上の部品数であれば入力次数がべき乗則に従うことがわかった．ただしランダムで取得した場合は，部品数が多い場合でも構造的な特性は全体集合と異なり，階層性が見られないことが分かった．部品検索により取得した場合は，部品数が 100 部品程度の少ない場合も比較的全体と近い性質をもつ．部品数が多い場合でも性質にはばらつきがあるが，これは検索語に依存したものであると考えられる．検索語が一般的な単語で様々なドメインの部品が検索される場合には利用関係のばらつきの多い，全体と性質がやや異なる部品グラフとなるが，検索語によりライブラリが検索される場合には，全体と性質の似た部品グラフが構成される．

実際のキーワードでの調査　実際に検索されたキーワードを用いて部品群に対する結果として，入力次数のべき乗則のパラメータ a の値および R^{*2} の値のヒストグラムを図 11, に，次数ごとのクラスタリング係数の傾きの値および R^{*2} のヒストグラムを図 12 に示す．

また，べき分布のパラメータ a, クラスタリング係数それぞれに関し，部品数ごとにプロットしたものを図 13 および図 14 に示す．なお横軸は対数軸とした．

これ，実際の検索語では多くの場合に a の値は 2.0 付近，クラスタリング係数の傾きは 1.0 付近であり，全体の集合と似た特性を表すことが分かる．しかし，検索語に依存したばらつきが見られ，部品数が 100 以下となるデータセットに関しては大きくばらついている．

3.6　考察

3.6.1　応用に関する考察

ソフトウェア部品検索への応用　実験により，大規模なソフトウェア集合は，単体のソフトウェアと同様に，部品グラフの次数がべき乗則に従うことが分かった．また，クラスタリング係数の特性から，階層性についても同様にもつことがわかった．また，そのサブセットに関し，キーワード検索により得られた部品群もまた同様の性質をもつことがわかった．

Data set		Number of nodes	Number of links
Random	A1	1000	52
	A2	1000	30
	A3	1000	73
	A4	5000	1085
	A5	5000	839
	A6	5000	1451
	A7	10000	6184
	A8	10000	4481
	A9	10000	4783
Use-relation	B1	163	1086
	B2	245	2386
	B3	488	705
	B4	972	1218
	B5	1036	3540
	B6	2678	22744

Data set		Number of nodes	Number of links
Use-relation	B7	4941	15424
	B8	5430	4179
	B9	5492	13311
	B10	9286	17201
	B11	10474	41079
	B12	10545	31296
Keyword Search	C1	129	124
	C2	149	76
	C3	297	1769
	C4	829	1123
	C5	1002	1564
	C6	1191	958
	C7	4385	16508
	C8	4982	21232
	C9	5597	31212
	C10	8938	24317
	C11	9150	26896
	C12	10537	49737

表 6: 実験 3: データセット

Data set		Incoming				Clustring coefficient				
		a			$p-value$	R^{*2}	slope			$p-value$	R^{*2}
Random	A1	2.27	±	0.177	5.61E-03	0.926	-2.000	±	NA	NA	NA
	A2	3.98	±	0.107	2.28E-02	0.997	-2.000	±	NA	NA	NA
	A3	1.91	±	0.261	3.94E-02	0.738	NA	±	NA	NA	0.000
	A4	2.13	±	0.0448	1.62E-15	0.971	-1.23	±	0.142	5.57E-07	0.831
	A5	2.53	±	0.0514	4.55E-14	0.983	-1.62	±	0.168	2.32E-06	0.892
	A6	1.99	±	0.0471	1.41E-15	0.952	-1.33	±	0.172	1.26E-06	0.787
	A7	1.94	±	0.0210	8.89E-38	0.979	-1.08	±	0.0990	1.14E-13	0.737
	A8	2.09	±	0.0284	7.58E-31	0.976	-1.20	±	0.115	4.07E-12	0.754
	A9	2.06	±	0.0251	1.32E-33	0.979	-1.35	±	0.127	2.22E-12	0.763
Use-relation	B1	1.83	±	0.0646	3.25E-12	0.867	-0.423	±	0.0715	6.18E-07	0.453
	B2	1.94	±	0.0402	6.45E-25	0.930	-0.403	±	0.0440	7.06E-13	0.585
	B3	2.17	±	0.0462	1.05E-13	0.975	-0.986	±	0.0995	1.76E-08	0.844
	B4	1.63	±	0.0815	3.42E-06	0.806	-1.21	±	0.122	1.10E-07	0.865
	B5	1.91	±	0.0193	6.46E-32	0.985	-0.921	±	0.105	6.61E-11	0.651
	B6	1.90	±	0.0159	2.68E-82	0.967	-0.824	±	0.0357	9.34E-45	0.822
	B7	1.76	±	0.0219	1.08E-38	0.957	-0.903	±	0.0435	2.07E-30	0.867
	B8	1.81	±	0.0503	2.45E-14	0.911	-1.17	±	0.0843	8.09E-15	0.856
	B9	1.71	±	0.0314	1.36E-25	0.921	-1.05	±	0.0590	2.57E-24	0.852
	B10	2.30	±	0.0203	2.98E-56	0.986	-1.07	±	0.0714	1.52E-23	0.759
	B11	1.85	±	0.0121	1.57E-93	0.978	-1.03	±	0.0447	4.25E-44	0.828
	B12	2.06	±	0.00985	9.50E-104	0.992	-1.07	±	0.0601	7.75E-33	0.755
Keyword Search	C1	2.62	±	0.0805	9.59E-07	0.983	-0.898	±	0.203	3.04E-03	0.700
	C2	3.04	±	0.263	1.49E-03	0.922	-0.756	±	0.230	3.01E-02	0.663
	C3	2.06	±	0.0534	2.28E-21	0.912	-0.530	±	0.0345	3.86E-18	0.855
	C4	2.12	±	0.0465	9.07E-18	0.960	-0.682	±	0.132	1.57E-05	0.462
	C5	2.21	±	0.0332	6.81E-24	0.980	-0.964	±	0.0918	6.78E-12	0.768
	C6	2.14	±	0.0326	5.09E-18	0.985	-0.751	±	0.161	1.05E-04	0.465
	C7	2.01	±	0.0158	2.82E-77	0.978	-0.893	±	0.0474	5.14E-35	0.775
	C8	2.35	±	0.0319	1.08E-62	0.950	-0.403	±	0.0453	2.11E-14	0.432
	C9	1.95	±	0.00828	3.51E-106	0.993	-0.830	±	0.0438	2.94E-36	0.765
	C10	2.12	±	0.00927	7.07E-107	0.993	-0.957	±	0.0472	8.34E-37	0.805
	C11	2.46	±	0.0248	1.91E-72	0.975	-0.910	±	0.0730	1.09E-21	0.616
	C12	1.94	±	0.00671	1.93E-130	0.994	-0.949	±	0.0223	1.60E-76	0.934

表 7: 実験 3: 入力次数の特性量とクラスタリング係数

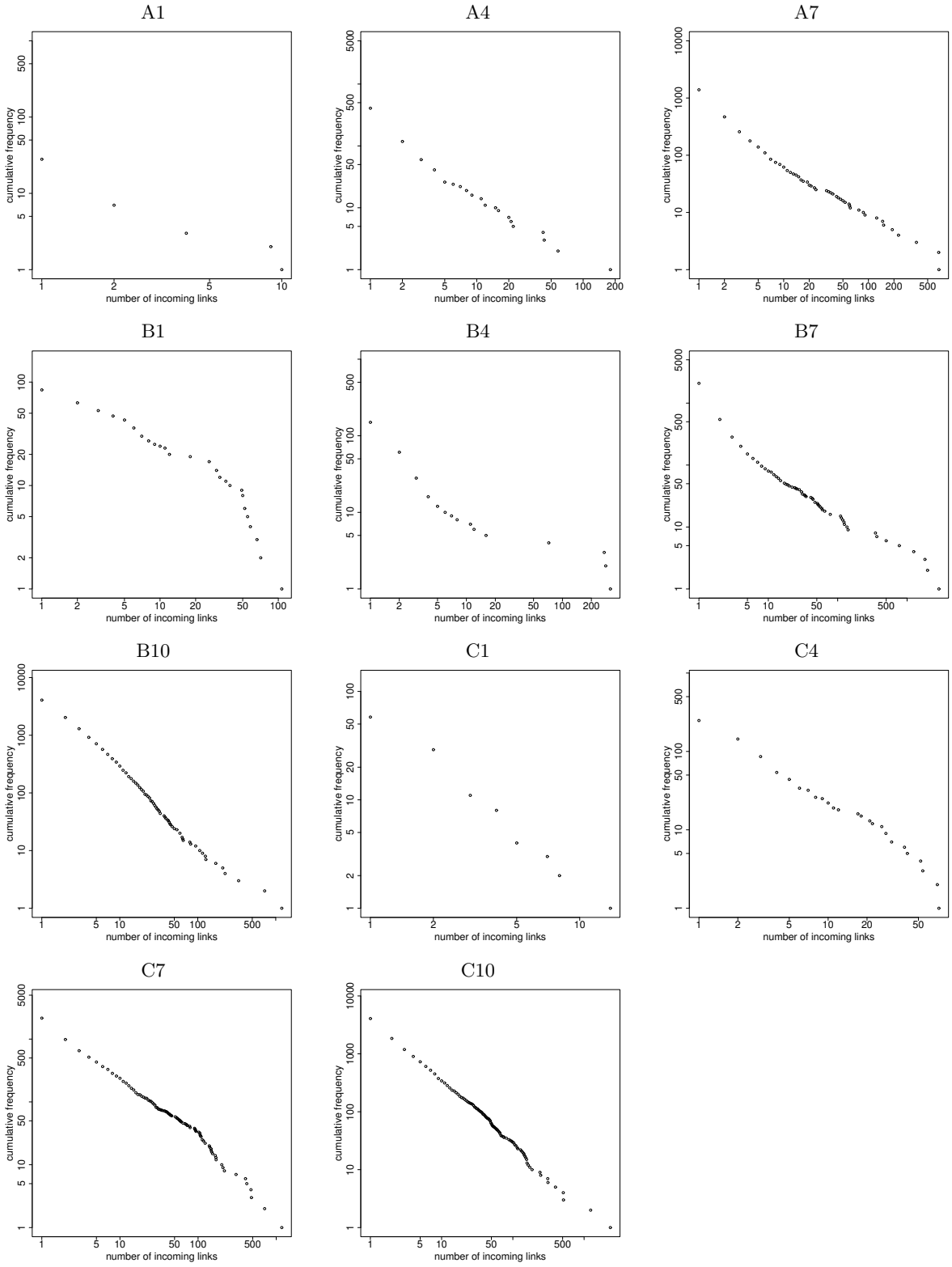


図 8: 実験 3: 入力次数の累積度数分布

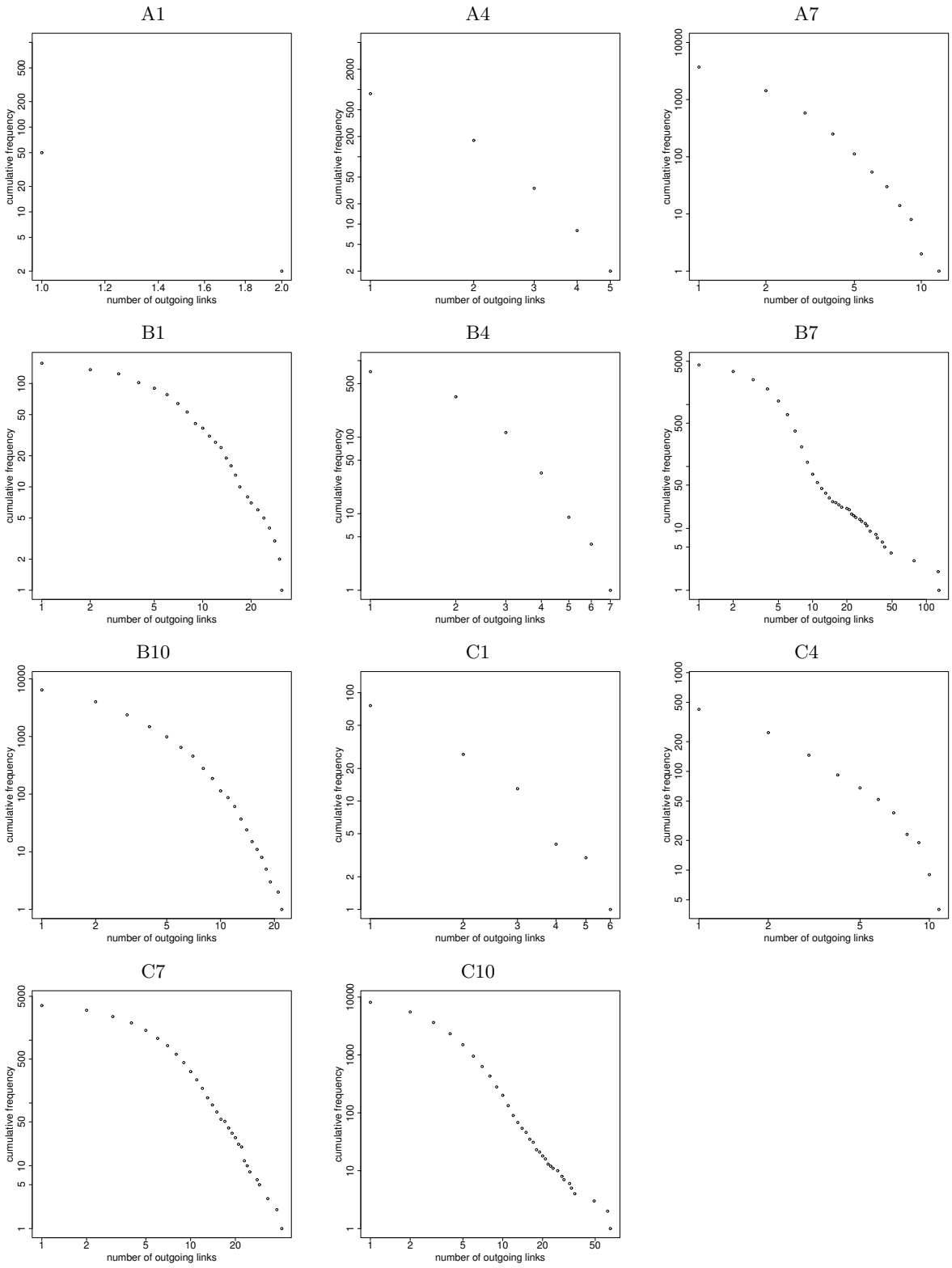


図 9: 実験 3: 出力次数の累積度数分布

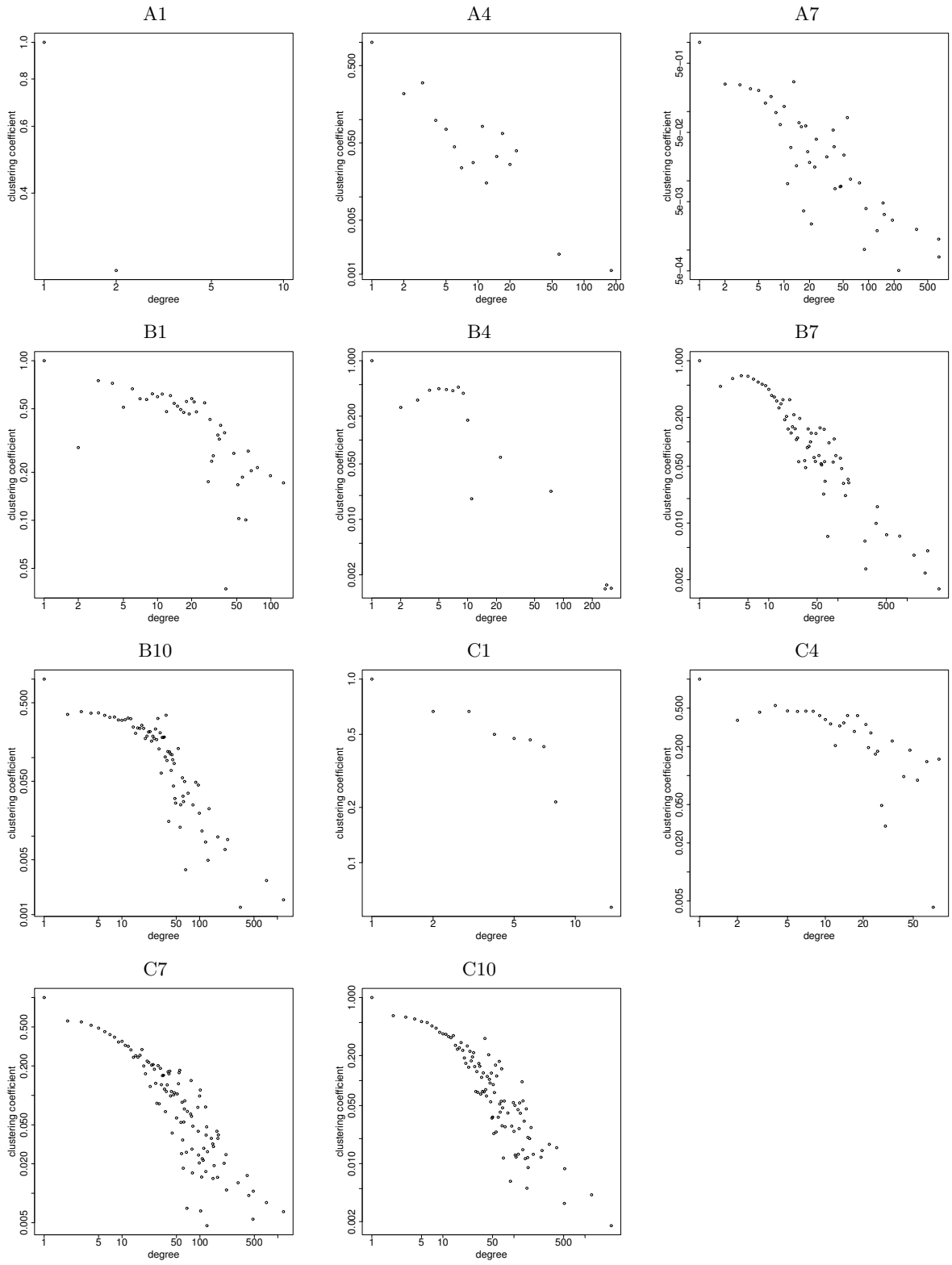


図 10: 実験 3: 次数ごとのクラスタリング係数

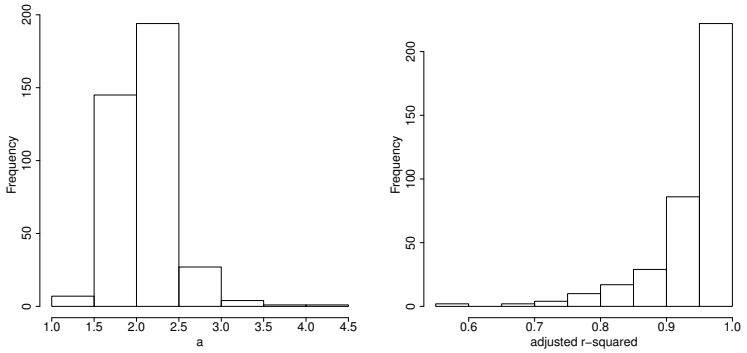


図 11: 実験 3: 入力次数分布のパラメータのヒストグラム

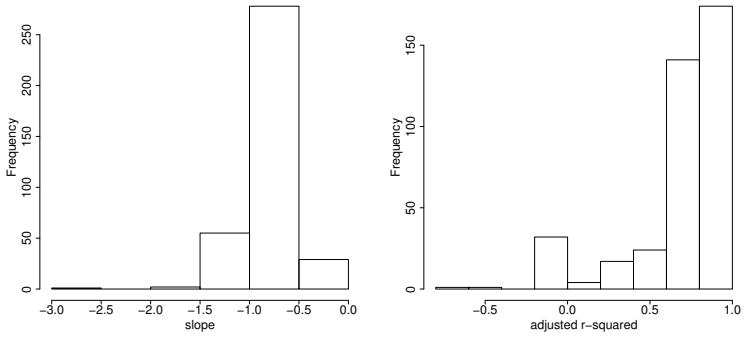


図 12: 実験 3: クラスタリング係数のプロットの傾きのヒストグラム

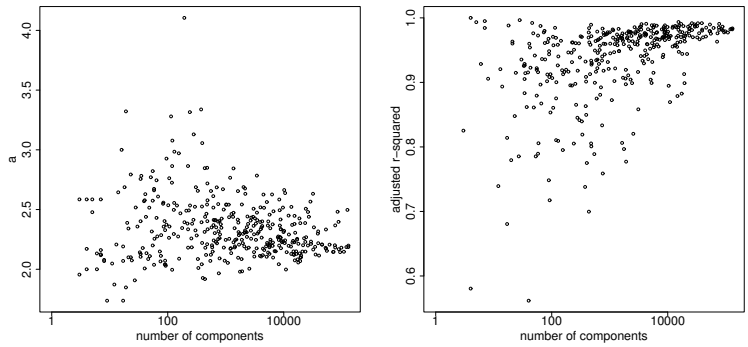


図 13: 実験 3: 入力次数分布のパラメータ

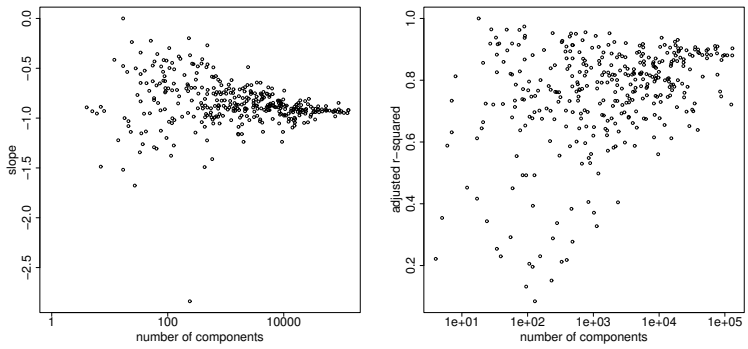


図 14: 実験 3: クラスタリング係数のプロットの傾き

これより，これまでソフトウェア単体や，ソフトウェア集合の部品グラフに対してのみ適用されてきた手法を検索結果の部品グラフに対して適用できると考えられる．4 節にてこの応用に関し詳しく述べる．

ソフトウェア設計評価への応用　ソフトウェア同士の比較実験により，次数の分布はソフトウェアの中心的なクラス設計を反映することがわかった．中心的なクラスがそれぞれもつ責任の大きさや，その個数に関して，クラス設計を俯瞰的に評価できるのではないかと考えられる．また，アンチパターンでは，一部のクラスに極度に依存するクラス設計や，逆に利用関係が極度に分散しているクラス設計は見直しの必要がある，と指摘されている．次数分布から，そういった利用関係に関する設計の問題点を検出できると考えられる．

また，ソフトウェア設計の評価はソフトウェア部品検索においても利用できる．ソフトウェア部品検索においては，部品データベースを良質な再利用性の高い部品で構成することが望ましい．ソフトウェア自体の設計が再利用性を意識した設計であれば，個々の部品，とくに部品検索において検索され再利用される部品の再利用性は高いと考えられる．ソフトウェアの設計を，再利用の観点から評価することが出来れば，部品データベース構築時に極端に設計の悪いアプリケーションは除くなどの処理を行うことで部品データベースの質の向上が望める．

4　ソフトウェア部品検索への応用

本研究での調査結果の応用として，ソフトウェア部品検索の順位付けの改良を提案する．本節では，まず準備としてソフトウェア部品検索における順位付けについて説明した後，提案手法とその適用結果について述べる．

4.1　ソフトウェア部品検索における順位付け

一般の情報検索システムにおいては，検索結果の適合性に加え結果を利用者に対して一覧表示する際の各情報の順位が重要である．検索問い合わせに対して適合した情報は必ずしも利用者の検索目的に適合せず，利用者の目的に適合する情報が下位であれば，利用者はその情報にたどり着くこと無く検索を諦めてしまう可能性があるためである．一般に利用者は検索結果の最初の一覧 (多くは 10 件) に求める情報が無ければ，検索問い合わせを変えるか，諦めてしまうと言われる．

同様のことがソフトウェア部品検索システムにおいても言え，Java ソフトウェア部品検索システム SPARS-J では ソフトウェア部品の特性を考慮した順位付け手法を用いている．以下ではその詳細を説明する．

4.1.1　SPARS-J での順位付け手法

SPARS-J は検索語により部品のソースコードの全部検索をおこなうシステムである．部品検索では，検索語と部品との適合性と同時に，その部品が実際にどの程度利用されているかということが重要である．ある部品がよく利用されているということは，その部品に実績があり，また利用例が多く再利用しやすいということを意味する． SPARS-J では，このような利用関係に基づく順位付け手法 Component Rank と, 検索語との適合性による順位付け手法 Keyword Rank を併用している．

Component Rank　まず類似部品群について説明する．実際の部品集合中には，コピーされて利用されているなどの理由により，全く，もしくはほとんど同一の部品が複数存在することがある．Component Rank ではこのような部品を類似部品群としてまとめ，解析の単位としている．

図 15(a) において，部品 c_1 と c_1'，部品 c_2 と c_2' はそれぞれ類似した部品とする．このとき，図 15(b) が部品群グラフとなり，部品群 C_1 には部品 c_1 および c_1'，C_2 には c_2 および c_2', C_3 には c_3, C_4 には c_4，そして C_5 には c_5 がそれぞれ属する．

Component Rank は部品群グラフに対して適用され，部品群に対する評価値が求められる．それぞれの部品の評価値は属する部品群の評価値である．評価値の求め方を以下に簡単

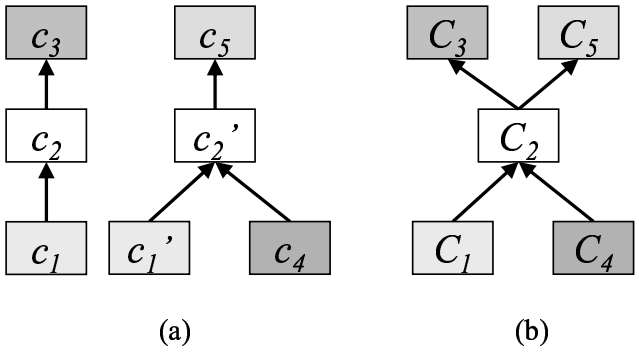


図 15: 部品グラフと部品群グラフ

に示す.

1. 各頂点に対し，総和が 1 となるような重みを与える
2. 各頂点の重みをその頂点を始点とする有向辺に配分するかたちで，各辺の重みを決定する.
3. 各頂点の重みを，その頂点を終点とする辺の重みの合計値として再定義する.
4. 以上 1.∼3. を，各頂点の重みが収束するまで繰り返す．最終的な値が各頂点に対応する部品群の評価値となる.

実際には以下のように求められる．W を各頂点の重みのベクトル, 行列 D を配分率，すなわち各有向辺の重みのその始点の重みに対する割合の行列とし，D^t は D の転置行列とする.

$$W = D^t W \tag{6}$$

この W を求めることで，各頂点に対応する部品群の評価値を計算することができる．式 (6) の性質より，D^t の固有値 1 の固有ベクトルを求める事で W を求める事ができる.

Keyword Rank　Component Rank により，部品の部品群中での相対的な重要度を得ることができるが，Component Rank では検索語と部品との適合性については考慮しない．そのため，明らかに適合しない部品であっても検索語にヒットすれば高く順位付けしてしまう可能性がある．そこで SPARS-J では検索語と部品との適合性を Keyword Rank により求める.

部品に含まれる語の中には，部品の内容と関連が深い語もあれば，ほとんど関係の無い語もある．検索語を部品の内容と関連の深い語として含む部品は利用者の検索目的にも適合する可能性が高いと考えられ，より上位に順位付けすることが必要であると考えられる．

Keyword Rank は，自然言語の文書検索システムで一般に用いられる TF-IDF 法 [17] に，対象がソフトウェア部品であることを考慮した変更を加えた手法である．TF-IDF 法では，特定の部品に偏って出現する索引語や，繰り返し出現する索引語が部品の内容と関連の高い索引語として扱われる．さらに Keyword Rank では，索引語に対してそのトークン種類に応じた重み付けを行っている．例えば索引語がクラス定義名やメソッド定義名として出現する時は高い重みを与え，変数名や文字列リテラルとして現れるときは低い重みを与える．

Keyword Rank による部品の評価値は検索時に以下の手順により求められる．まず，検索語にヒットした索引語ごとにトークン種類の重みと出現回数を乗算し，正規化する．続いてその値にヒットした部品数を総部品数で割った値の逆数を乗算する．この総和が部品の評価値となる．

SPARS-J では，以上で述べた Component Rank および Keyword Rank の順位を統合することにより，部品検索における有用な順位付けを実現している．

4.1.2 現状の課題

ライブラリとしての部品を再利用することを目的として検索する場合を考える．調査結果により示された様に，大規模な部品集合ではごく一部の部品に利用関係が集中する傾向がある．それに伴い，Component Rank の値もごく一部の部品が極端に高くなり，利用される頻度の高い部品と低い部品の間で Component Rank の差が大きくなる．このため，JDK などの評価値の極端に高い部品は，不適合であってもヒットすれば高く順位付けされ，他の適合する部品の順位を押し下げてしまう．このような不適合かつ評価値の高い部品が多く存在する場合，Keyword Rank を併用した場合でも補正しきれず，上位の適合率が悪くなる可能性が考えられる．

この問題は，順位の統合の際に Component Rank の比率を下げることで部分的に解決できるが，以下の理由から望ましくないと考えられる．ひとつは Keyword Rank による適合性ではライブラリ部品そのものより，ライブラリの利用側の部品がより上位とされることである．これは，ライブラリの利用側の部品では，処理の記述部分に利用するクラス名やメソッド名が繰り返し現れることで評価値が上がり易い為である．もう一つは，利用者は，検索語と部品との適合性は部品を見ることで容易に判断できるが，その信頼性などの再利用性については判断できないため，再利用を目的とした検索においては Component Rank を重視すべきであるからである．

このように，有効な再利用の為には部品群を大きくして様々な部品が含まれるようにする必要があるが，大規模な部品群では Component Rank がうまく機能しない問題が考えられる．Component Rank と Keyword Rank は相補的な順位付け手法である．順位の統合により，片方のみで順位の高い部品の順位を下げ，両方で共に上位の部品を最終的な上位に順位付けすることができる．しかし，不適合かつ Component Rank による評価値の高い部品が多くヒットした場合には，利用者の目的とする部品は Component Rank および Keyword Rank の両方でやや低い評価となり，利用者はある程度検索結果の一覧を下位まで辿らないと利用できる部品にたどり着けない．

4.2 提案手法

4.2.1 基本的なアイデア

前述した状況，すなわち利用頻度がそれほど多くないライブラリ部品を探す場合を考える．検索により得られた部品群の中では，利用者の目的に適合しない部品は比較的孤立していると考えられる．検索結果の部品群内で閉じた利用関係をもとにした部品グラフから Component Rank を計算したとき，孤立している適合しない部品の Component Rank は低くなり，逆に，必要とするライブラリ部品は，関連する部品や利用例などが同時に検索結果に含まれ，Component Rank 値が高くなると考えられる．これより，全体の部品グラフをもとに求めた Component Rank の代わりに検索結果の部品グラフで求めた Component Rank を用いることで，検索結果上位における適合率を改善できるのではないかと考えられる．

ここで，Component Rank 法は検索結果の部品群に適用した時に有効な結果が得られるのかどうかという点について考える．Component Rank は任意の部品グラフに適用できる手法ではない．例えば，部品間にほとんど利用関係が無い場合や，どの部品もほぼ同数の利用関係および被利用関係を持つ場合には重要度を判定できず適用できない．逆に，次数，特に入力次数がべき乗則に従うような性質をもつグラフでは，よく利用される一部のライブラリ部品に利用関係が集中し，さらに階層性が存在するグラフでは利用関係もまた階層的に集中するため，Component Rank 法により適切に部品の評価を行うことができると考えられる．

Component Rank の有効性は単体のソフトウェアおよびソフトウェア集合に対して示されている．これらの対象は明らかに関連性を持った集合であり，極端な例を持ち出さない限りは前述した様な状況には陥らず，直感的にも有効であると考えられる．しかし，検索結果の部品集合は単に索引語として検索語を含むという共通点しか持たず，実際に利用関係が存在するかどうかは分からない．

しかし，本研究による実験では，検索結果の部品による部分グラフも，数百程度の部品により全体と同様にべき乗則が成り立つことが示された．このことから，検索結果の部品群に

対して Component Rank を適用した場合にも，適正な結果を得られることが期待できる．

4.2.2 手法詳細

提案手法は，利用者の検索キーワードを入力とし，順位付けされた部品群を返す．以下にその詳細を示す．

1. キーワード検索により，キーワードに一致する索引語をもつ部品を取得する
2. 全体の部品グラフから得られる利用関係をもとに，部分グラフを作成する
3. 部分グラフを 類似部品群 にグループ化し，部品群グラフを作成する

   類似部品とはコピーなどにより生成された本来同一である部品のことで，Component Rank では，利用関係が分散して不当に値が低くなることを防ぐためにグループ化している．提案手法でもこのグループ化を用いる．
4. 部品群グラフをもとに Component Rank 値を求める
5. Component Rank 値順にソートし，ユーザへ提示

4.3 適用実験

4.3.1 実験内容

提案手法の有効性を評価するため，ライブラリ部品の再利用を目的とした検索語を用い，上位 10 件の適合率および実行時間を評価した． キーワードと，その目的を表 8 に，実行環境を表 9 に示す．

部品は，以下の (A),(B) の条件のいずれかをみたすとき適合とする．条件 (A) はクラスがライブラリとして設計されている場合，条件 (B) はライブラリとしては設計されていないが，コードを引用することで機能を利用可能である場合である．

(A) クラスがパッケージ外から利用可能であり，目的の機能を実現する役割をもつ public なメソッドを含む

なお，一般に複数のクラスの連携により一つの機能を実装することが多いため，対象クラスの役割が，目的の一部のみをみたす場合でも適合とみなす．ただし，メソッドの役割が別にあり，副次的に機能が実現されている場合は不適合とする．

(B) 条件 (A) は満たさないが，目的の機能を実現する，完結した実装をもつ

ただし実装に JDK 以外のクラスを用いている場合には不適合とみなす．

キーワード	目的
hash map	ハッシュマップを実現・操作
perl regular expression	Perl 互換の正規表現
file read string	ファイルから文字列を読み込む処理を実現
zip	ZIP アーカイブファイルを扱う
pdf	PDF フォーマットを扱う
xml parse	XML ファイルの解析
http connection	HTTP 接続をおこなう
image icon swing	Swing GUI 上で画像アイコンを表示
cvs	バージョン管理システム CVS を扱う
graph	グラフの描画

表 8: キーワード

OS	FreeBSD 4.11-STABLE
CPU	Intel(R) Pentium 4 3.2GHz
Memory	2.0 GB

表 9: 実行環境

また，部品の適合性は類似部品群単位で評価する．同じ類似部品群に含まれる部品同士で適合性が異なることは稀であるが，類似部品群中の半数以上の部品の適合により，部品群の適合とする．

4.3.2 実験結果

実験結果を表 10 に示す．また，各キーワードでのヒット数および検索に要した時間を表 11 に示す．表 10 より，ほとんどの場合において従来手法と比較して提案手法が適合率で上回ることが分かった．特に，キーワード “zip”, “pdf”, “xml parse”,“graph” において顕著な上昇が見られた．“zip” および “pdf” に関しては，コメント中にファイル名の拡張子などとして現れることが多く，検索時には多くの無関係な部品がヒットする．同様に “graph” はアルゴリズムでの意味のグラフを扱う部品が多くヒットする．また “xml parse” に関しては，XML は様々なアプリケーションで用いられ，XML に関係する部品が多数存在する．これらの場合，もとの Component Rank では無関係な部品であっても上位にランク付けしてしまうが，提案手法では適切な順位付けが出来た．

しかし，“zip” での実際の順位付けをみると，提案手法でも java.lang.Sytem という無関係な部品が最上位に順位付けされていた．この部品は Java の基本ライブラリである JDK の中でも基礎的な部品であり，検索結果内でも多くの部品に用いられていた．他の検索結果で

キーワード	提案手法	従来手法
hash map	0.2	0
perl regular expression	0.7	0.7
file read string	0.2	0.1
zip	0.6	0.2
pdf	0.7	0.2
xml parse	0.6	0.2
http connection	0.4	0.3
image icon swing	0.2	0.2
cvs	0.3	0.3
graph	0.5	0

表 10: 適用実験結果

キーワード	ヒット数	実行時間 (秒)		差 (秒)
		提案手法	従来手法	
hash map	11441	2.72	1.50	1.22
perl regular expression	94	0.15	0.14	0.01
file read string	10388	10.94	9.71	1.22
zip	1524	0.19	0.08	0.11
pdf	484	0.07	0.03	0.05
xml parse	4825	1.26	0.85	0.41
http connection	4364	1.95	1.60	0.35
image icon swing	1083	0.63	0.54	0.09
cvs	8561	0.96	0.34	0.62
graph	2112	0.27	0.12	0.15

表 11: 実行時間の比較

も，このような部品が含まれる時は上位に順位付けしてしまう傾向があった．

また，実行時間の増加量に関しては，最大でもヒット数が 1 万を超える “hash map” および “file read string” での 1.2 秒程度であり，それらの検索語では元の検索時間もやや長いことを考慮すると実用上問題の無い範囲であると考えられる．

4.4 考察

提案手法は，ライブラリ部品を目的として検索した際に，上位の適合率の改善という形で順位付けの改善が出来ることが分かった．特に，コメント中に一般に現れやすい単語により検索する場合や，様々な分野で使われる概念を扱うライブラリ部品を検索する場合において効果が見られた．

しかし，関係の無い部品であっても，検索結果の部品グラフの中でよく使われる場合には上位に来てしまう問題が明らかになった．しかし，これは JDK の基礎的なパッケージのごく一部の部品のみで見られる現象であるため，例えば当該クラスもしくはパッケージを検索対象から除外する等の手段で解決することが可能である．

今後，提案手法に関してはさらなる評価が必要であると考えられる．Component Rank 法は Keyword Rank 法と組み合わせて利用される．提案手法に関しても Keyword Rank 法と組み合わせた上での評価が必要である．

また，再利用性の観点からの評価が必要であると考えられる．提案手法では，検索結果に偏りがあり，本来再利用されるべき部品が少数の利用関係しか持たない場合には，その順位が下がり，本来はそれほど評価の高くない，別の部品が上位に来る可能性がある．単純な適合性という面からは問題無いが，再利用すべき部品が再利用出来ていないという観点から望ましく無い．今回の適用実験においては，そのような結果は見られなかったが，もとの Component Rank 法と比較する形での順位付けの評価実験が必要である．

5　まとめと今後の課題

本研究では，ソフトウェア部品グラフにおいてべき乗則が成り立つか否かということに関して調査を行った．その結果，単体ソフトウェアだけでなく，大規模なソフトウェア集合や，その一部を取り出した部品群に対しても，部品グラフの頂点の次数がべき乗則に従うことが明らかになった．クラスタリング係数に関しても調査したところ，階層性についても成り立っていることが明らかになった．また，データセット同士の比較により，次数の分布はソフトウェアの中心的な設計を反映することが明らかになった．

また，調査結果を踏まえ，ソフトウェア部品検索およびソフトウェア評価への応用に関して考察した．ソフトウェア部品検索への応用については，検索結果の順位付け手法の改良を提案した．提案手法では，部品グラフから個々の部品の相対的重要度をもとめる手法 Component Rank を検索結果における部分グラフに対して適用した．また，適用実験によりその有効性を評価し，検索結果上位の適合率を改善できることを示した．

今後の課題としては，異なる観点からの部品グラフの調査が挙げられる．グラフの接続次数がべき乗則に従うとき，同時にスモールワールド性 [36] が成り立つことが多いと言われている．本研究における調査では直接的な調査は行わなかったが，クラスタリングなどの手法を適用する際に重要な性質であるため，今後調査する必要がある．また，部品検索手法への応用として提案した手法に関し，再利用性の面からの順位付けの評価や，キーワードとの適合性を考慮した順位付け手法との統合，およびその評価をおこなうことが必要である．

謝辞

本研究の全課程を通して，常に適切な御指導および御助言を賜りました 大阪大学大学院情報科学研究科コンピュータサイエンス専攻　井上克郎教授に心より深く感謝致します.

本研究を通して，常に適切な御指導および御助言を賜りました 同 松下誠助教授に深く感謝致します.

本研究を通して，逐次適切なご指導およびご助言を頂きました 立命館大学情報理工学部情報システム学科　山本哲男講師に深く感謝いたします.

本研究を通して，逐次適切なご指導およびご助言を頂きました南山大学数理情報学部情報通信学科 横森励士講師に深く感謝いたします.

本研究を通して，逐次適切なご指導およびご助言を頂きました 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 川口真司氏に深く感謝いたします.

最後に，その他様々な御指導，御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 ソフトウェア工学講座 井上研究室の皆様に深く感謝いたします.

参考文献

[1] L. A. Adamic: "Zipf, Power-laws, and Pareto - a ranking tutorial", `http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html`, 2000

[2] R. Albert, H. Jeong and A. Barabási: "Diameter of the World-Wide Web", *Nature*, vol. 401, pp. 130-131, 1999.

[3] 青木 淳: " オブジェクト指向システム分析設計入門", ソフトリサーチセンター, 1993

[4] A. Barabási: "Linked: The New Science of Networks", Perseus Books Group, 2002

[5] J. M. Bieman and B. Kang: "Cohesion and reuse in an object-oriented system", In *Proceedings of the 1995 Symposium on Software reusability (SSR'95)*, pp. 259-262, 1995

[6] C. Braun: "NATO Standard for the Development of Reusable Software Components", *NATO Communications and Information Systems Agency*, 1992

[7] C. Braun: "Reuse, in John J. Marciniak, editor", *Encyclopedia of Software Engineering*, vol. 2, John Wiley & Sons, pp. 1055-1069, 1994

[8] L. C. Briand, J. W. Daly, and J. Wust: "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", Springer, *Empirical Software Engineering*, vol. 3, no. 1, pp. 65-117, 1998

[9] D. Challet and A. Lombardoni: "Bug propagation and debugging in asymmetric software structures", *Physical Review E*, vol. 70, 046109, 2004

[10] A. Chatzigeorgiou, S. Xanthos and G. Stephanides: "Evaluating Object-Oriented Designs with Link Analysis", In *Proceedings of the 26th International Conference on Software Engineering* (ICSE'04), pp. 533-542, 2004

[11] S. R. Chidamber and C. F. Kemerer: "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, June 1994.

[12] Y. Chiricota, F. Jourdan and G. Melançon: "Software components capture using graph clustering", In *Proceedings of the 11th International Workshop on Program Comprehension (IWPC'03)*, pp. 217-227, 2003

[13] "Eclipse", `http://www.eclipse.org/`

[14] “GNU Binutils”, http://www.gnu.org/software/binutils/

[15] K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, S. Kusumoto: “Ranking Significance of Software Components Based on Use Relations”, *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 213-225, 2005

[16] I. Jacobson, M. Griss and P. Jonsson: “Software Reuse”, *Addison Wesley*, 1997

[17] 北, 津田, 獅々堀: “情報検索アルゴリズム”, 共立出版, 2002

[18] J. Kleinberg: “Authoritative Sources in a Hyperlinked Environment”, *Journal of the ACM*, vol. 46, issue 5, pp. 604-632, 1999

[19] 久米, 飯塚: “回帰分析”, シリーズ 入門 統計的方法, 岩波書店, 1987

[20] Y. Ma, K. He, and D. Du: “A Qualitative Method for Measuring the Structural Complexity of Software Systems Based on Complex Networks”, In *Proceedings of 12th Asia-Pacific Software Engineering Conference (APSEC'05)*, pp. 257-263, 2005

[21] M. Mitzenmacher: ”Dynamic Models for File Sizes and Double Pareto Distributions”, *Internet Mathematics*, vol. 1, no. 3, pp. 305-333, 2003

[22] C. R. Myers: “Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs”, *Physical Review E*, vol. 68, 046116, 2003

[23] “NetBeans”, http://www.netbeans.org/

[24] M. E. J. Newman: “The Structure of Scientific Collaboration Networks”, In *Proceedings of the National Academy of Sciences of USA 98*, pp. 409-415, 2001.

[25] M. E. J. Newman: “Power laws, Pareto distributions and Zipf's law”, *Contemporary Physics*, vol. 46, pp. 323-351, 2005

[26] A. Potanin, J. Noble and M. Frean: “Scale-free Geometry in OO Programs”, *Communications of the ACM*, vol. 48, issue 5, pp. 99-103, 2005

[27] E. Ravasz, A.L. Barabási: “Hierarchical organization in complex networks”, *Physical Review E*, vol 67, 261121, 2003

[28] “SourceForge.net”, http://sourceforge.net/

[29] “SPARS-J”, http://demo.spars.info/

[30] “The Apache Software Foundation”, http://www.apache.org/

[31] “The FreeBSD Project”, http://www.freebsd.org/

[32] “The Linux Kernel Archives”, http://www.kernel.org/

[33] “The R Project for Statistical Computing”, http://www.r-project.org/

[34] S. Valverde, R. Ferrer-Cancho and R. V. Solé: “Scale-free Networks from Optimal Design” *Europhysics Letters*, vol. 60, no. 4, pp. 512-517, 2002

[35] S. Valverde and R. V. Solé: “Hierarchical Small Worlds in Software Architecture”, *Working paper of Santa Fe Institute*, SFI/03-07-44, 2003

[36] D. J. Watts and S. H. Strogatz: “Collective dynamics of ’small-world’ networks”, *Nature*, vol. 393, pp. 440-442, 1998

[37] R. Wheeldon and S. Counsell: “Power Law Distributions in Class Relationships”, *In Proceedings of Third IEEE International Workshop on Source Code Analysis and Manipulation (SCAM2003)*, pp. 45-57, 2003

[38] 横森, 藤原, 山本, 松下, 楠本, 井上: “利用実績に基づくソフトウェア部品重要度評価システム”, 電子情報通信学会論文誌 D-1, vol. J86-D-I, no.9, pp. 671-681, 2003

[39] 横森, 梅森, 西, 山本, 松下, 楠本, 井上: “Java ソフトウェア部品検索システム SPARS-J”, 電子情報通信学会論文誌 D-I, volJ87-D-I, no.12, pp. 1060-1068, 2004

付録

ここでは，3 節の実験 1 および実験 3 で取り扱ったデータセットの，入力次数および出力次数のプロットを示す．プロットは，3 節では累積度数プロットのみを示したが，ここでは通常のプロットも示す．なお，3 節で取り扱っていない以下のデータセットについても示す．

SF SourceForge.net から取得したオープンソフトウェアの集合

SF+JDK SF に JDK を加えた物．なお JDK のクラスは全て含まれる．

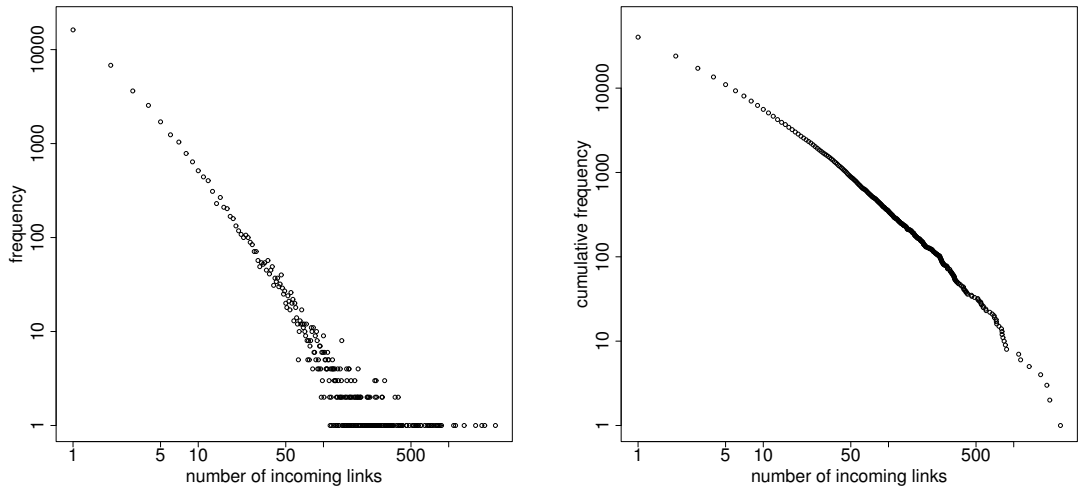
APACHE+JDK APAHCE に JDK を加えた物．同様に JDK のクラスは全て含まれる．

また，これらのデータセットの頂点数, 辺数, 行数を表 12 に示す．

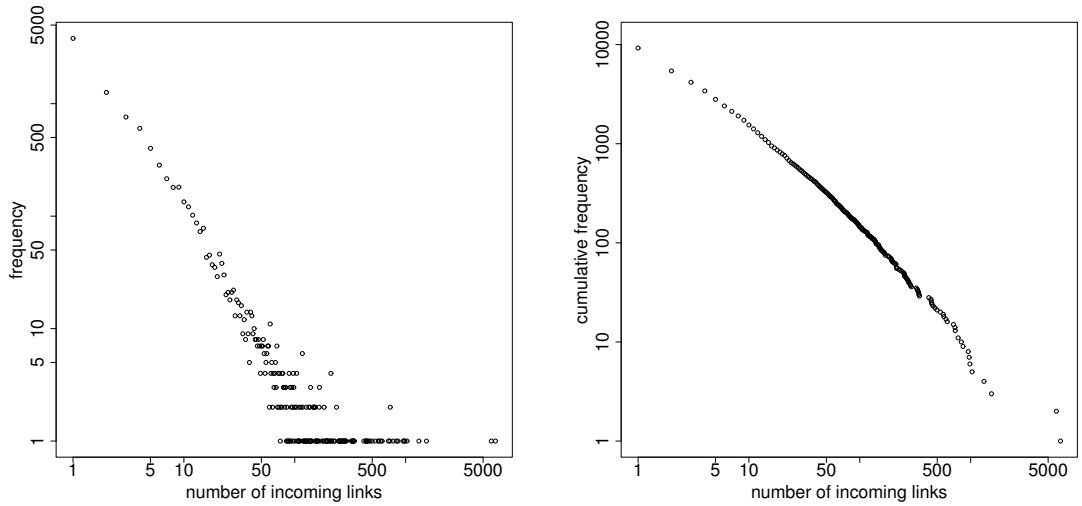
Data set	Number of nodes	Number of links	LOC
SF	53937	227835	4.1M
SF+JDK	65493	615375	5.2M
APACHE+JDK	71042	655175	5.7M

表 12: 追加 データセット

APACHE



JDK



ECLIPSE

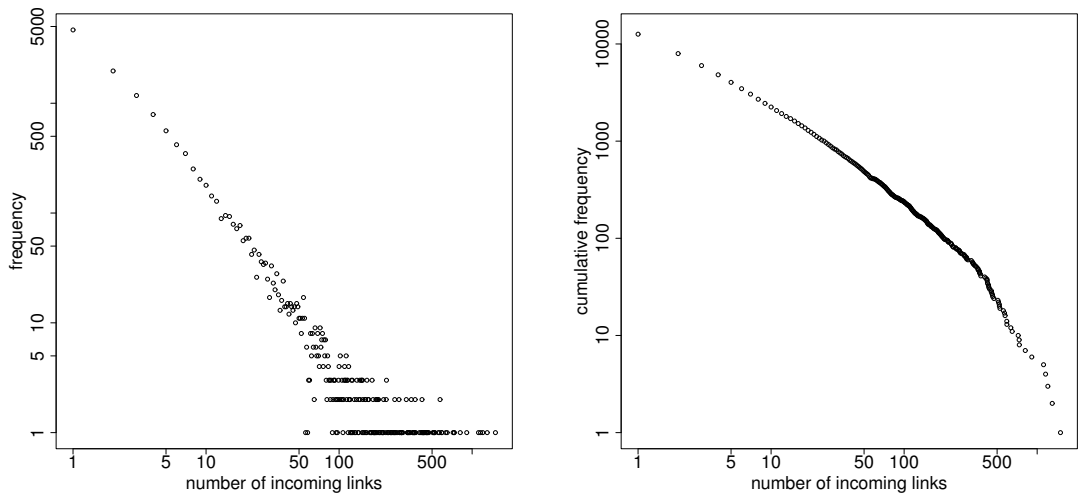
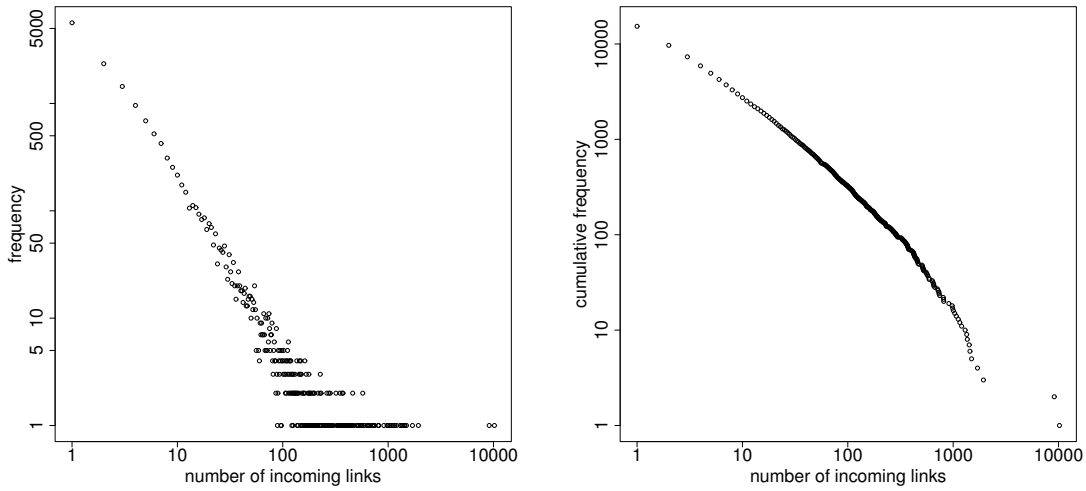
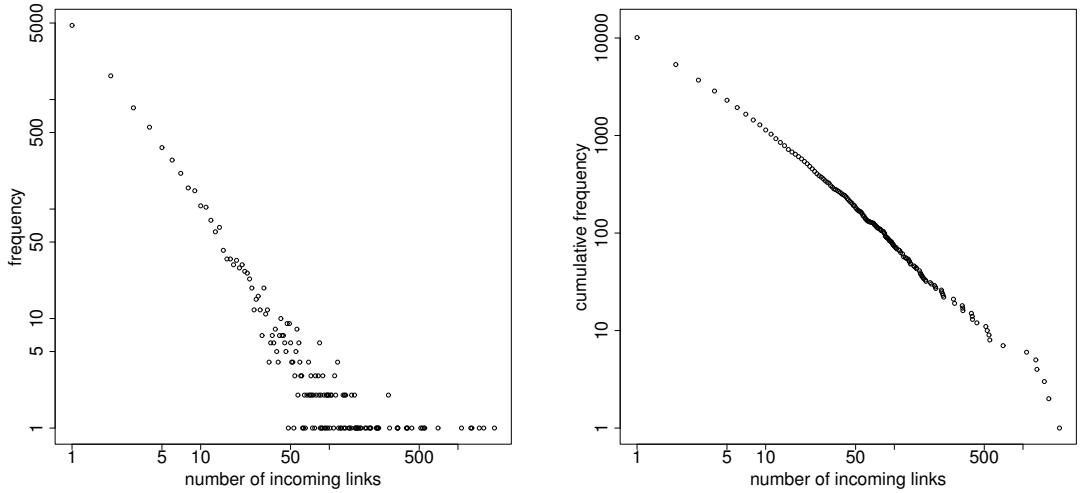


図 16: 実験 1 入力次数の分布 (1)

ECLIPSE+JDK



NETBEANS



ALL

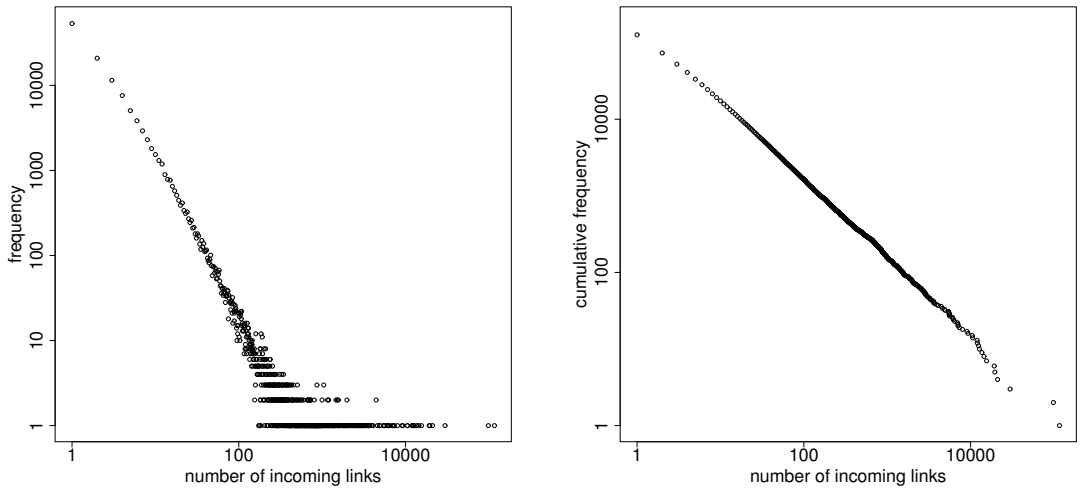
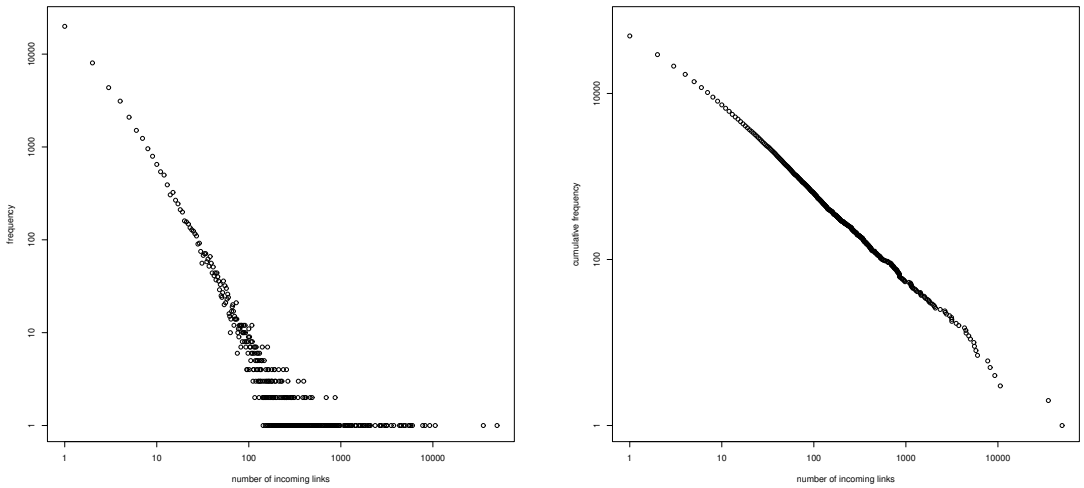
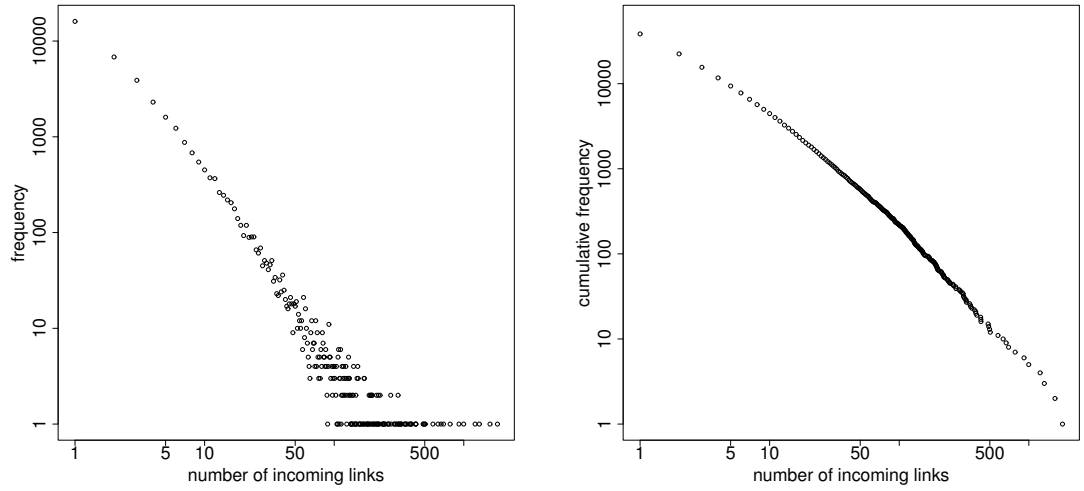


図 17: 実験 1 入力次数の分布 (2)

APACHE+JDK



SF



JDK+SF

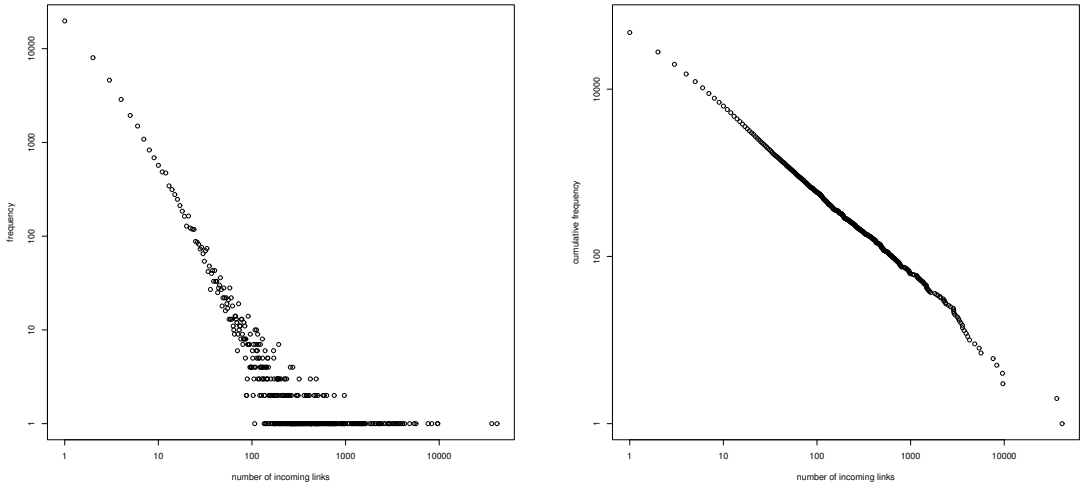
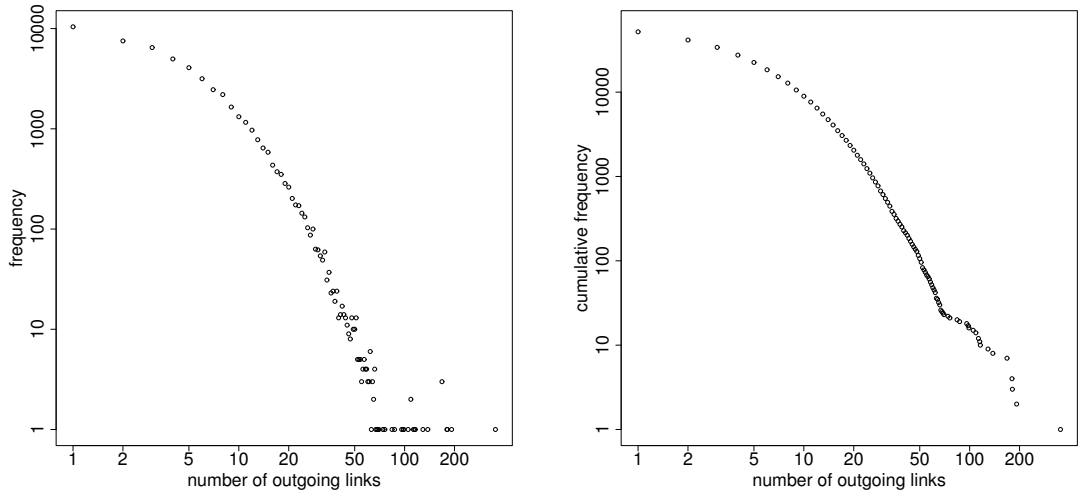
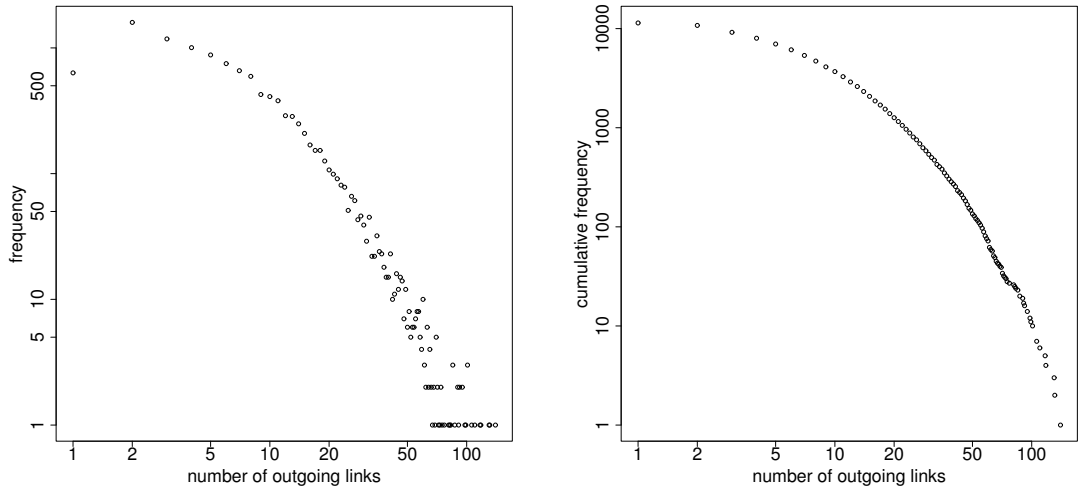


図 18: 実験 1 入力次数の分布 (3)

APACHE



JDK



ECLIPSE

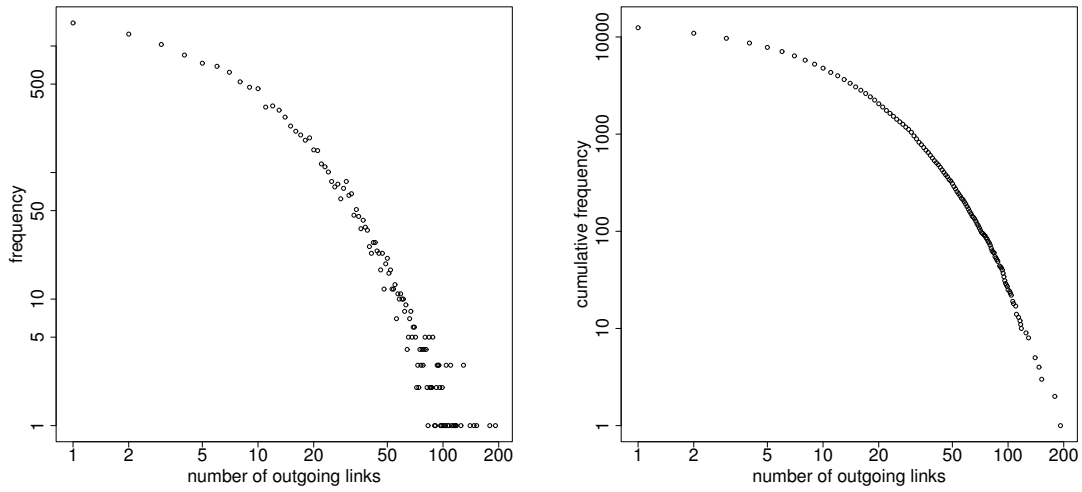
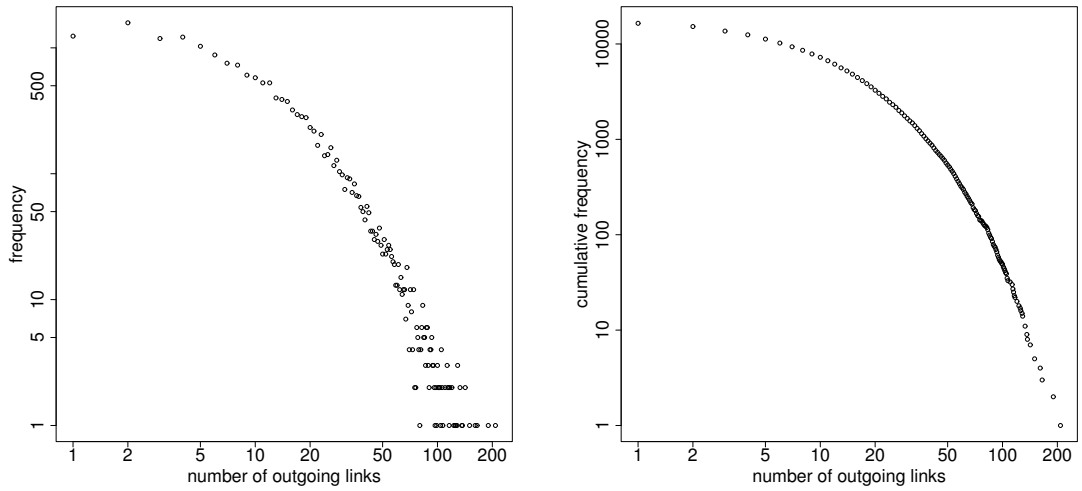
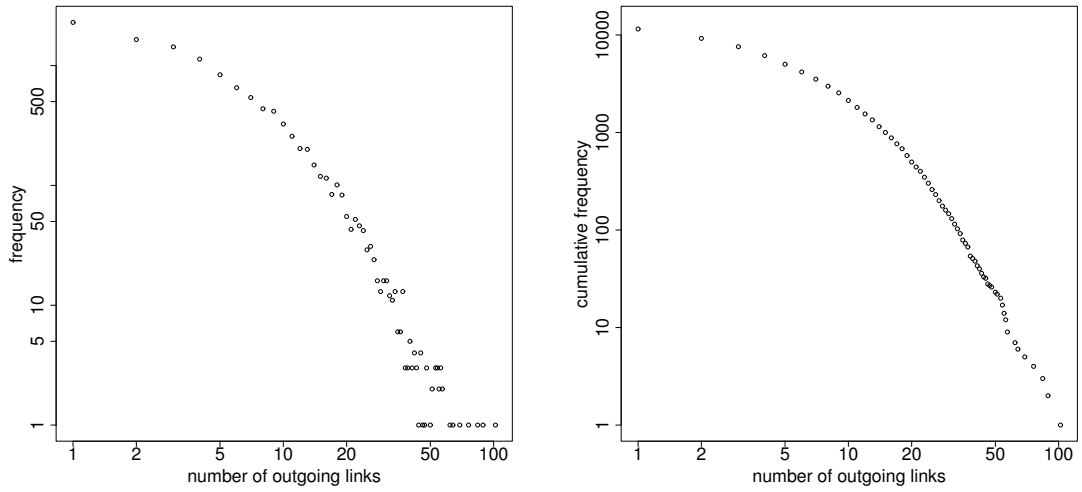


図 19: 実験 1 出力次数の分布 (1)

ECLIPSE+JDK



NETBEANS



ALL

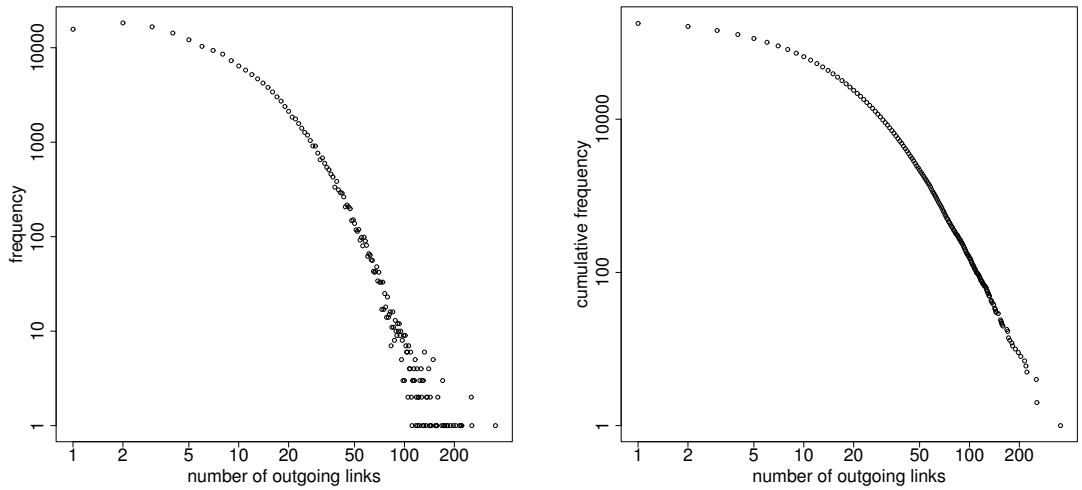
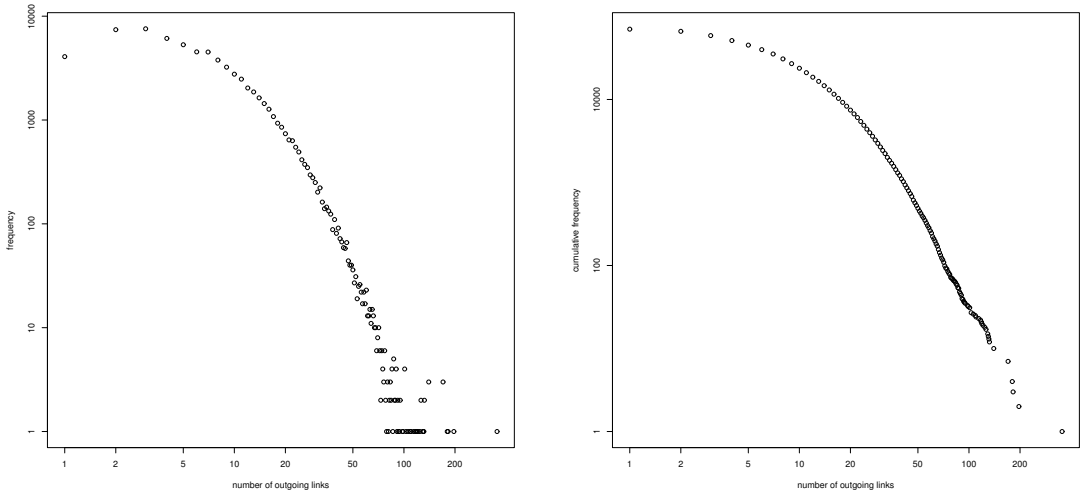
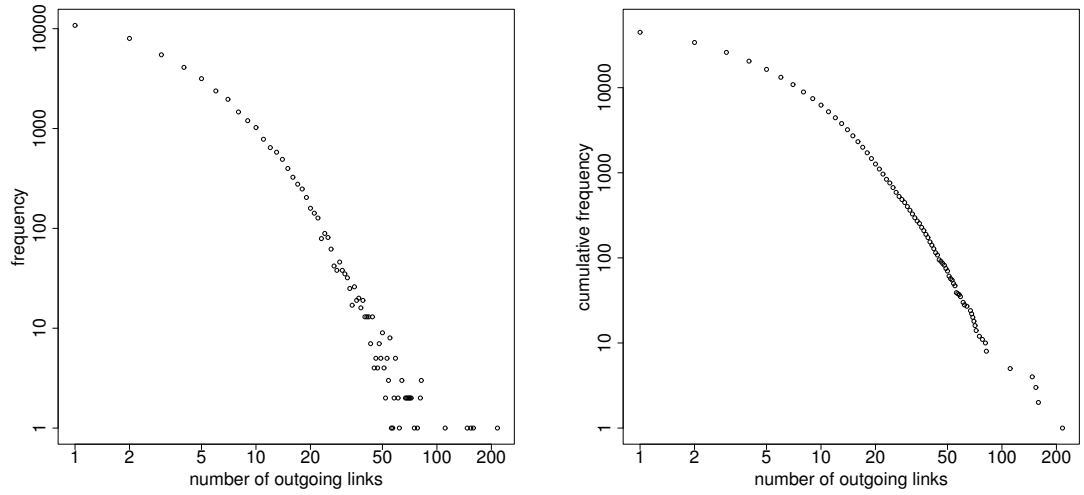


図 20: 実験 1 出力次数の分布 (2)

APACHE+JDK



SF



JDK+SF

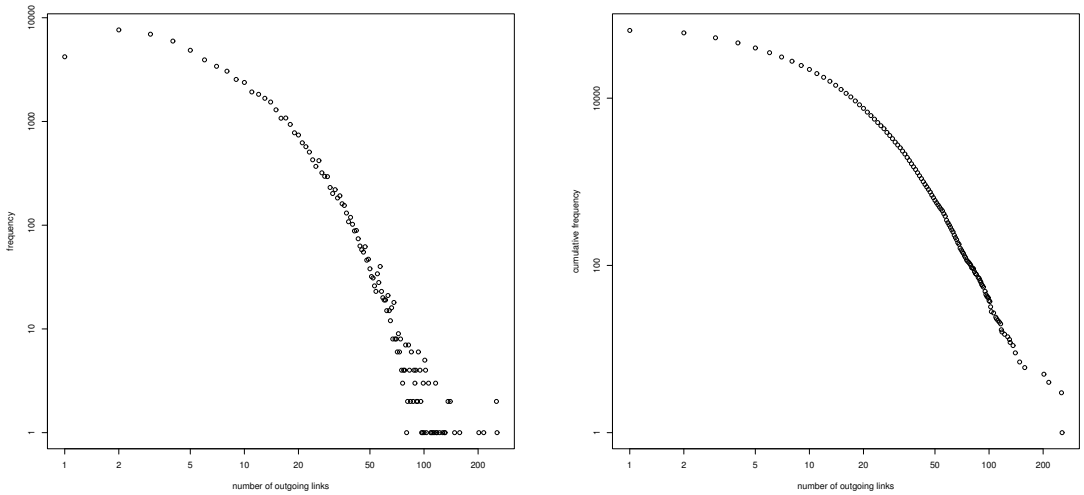
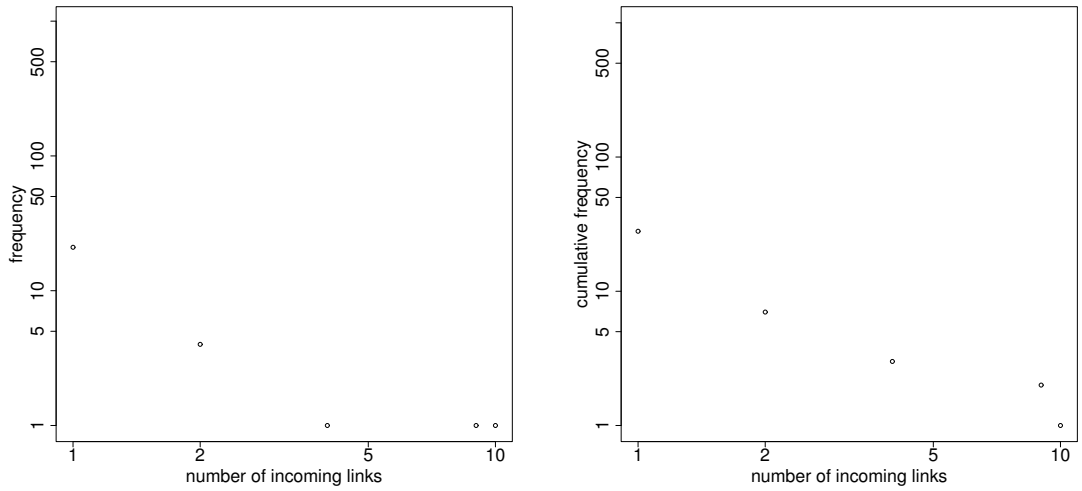
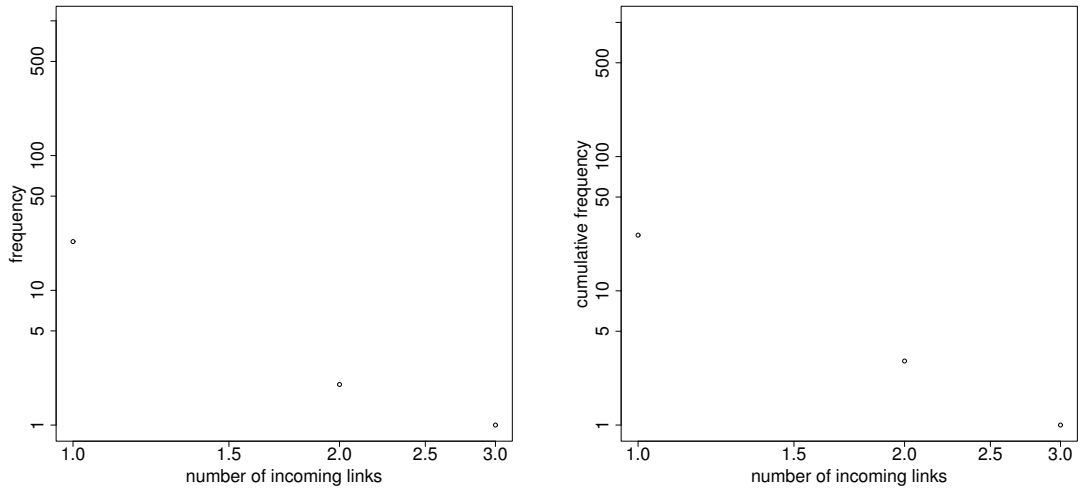


図 21: 実験 1 出力次数の分布 (3)

A1



A2



A3

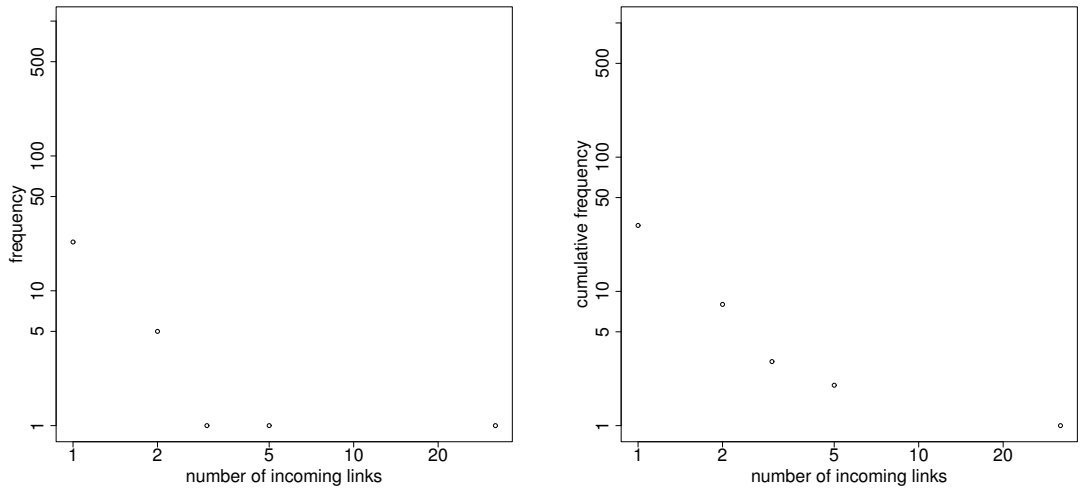
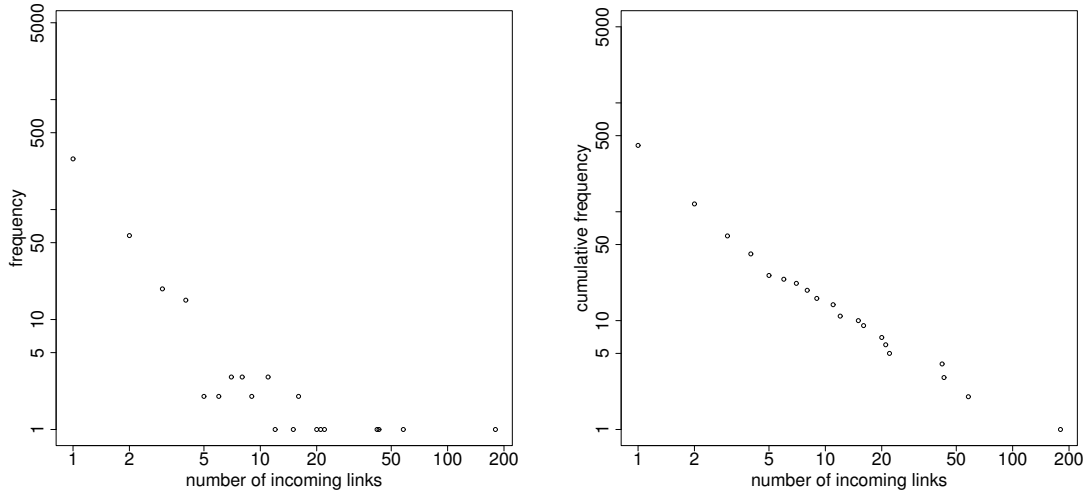
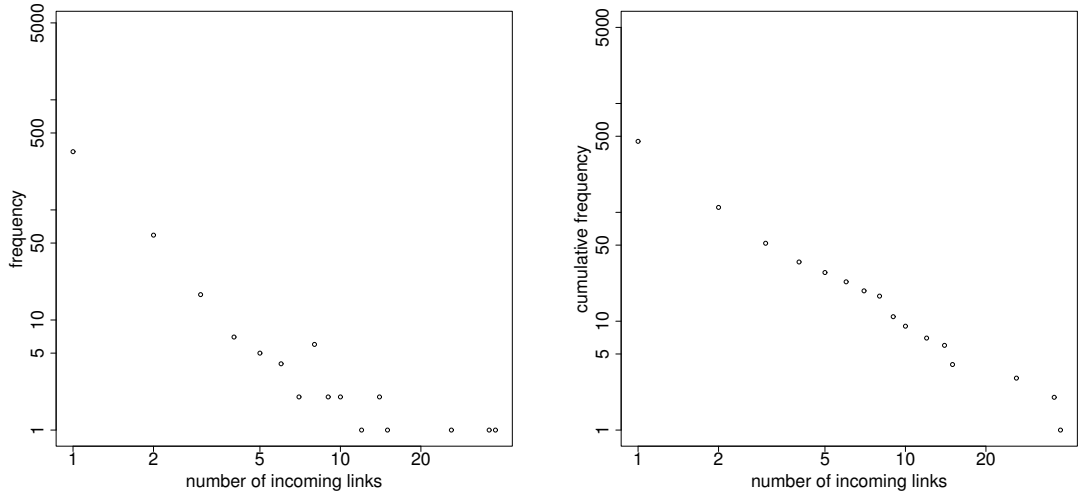


図 22: 実験 3 入力次数の分布 (1)

A4



A5



A6

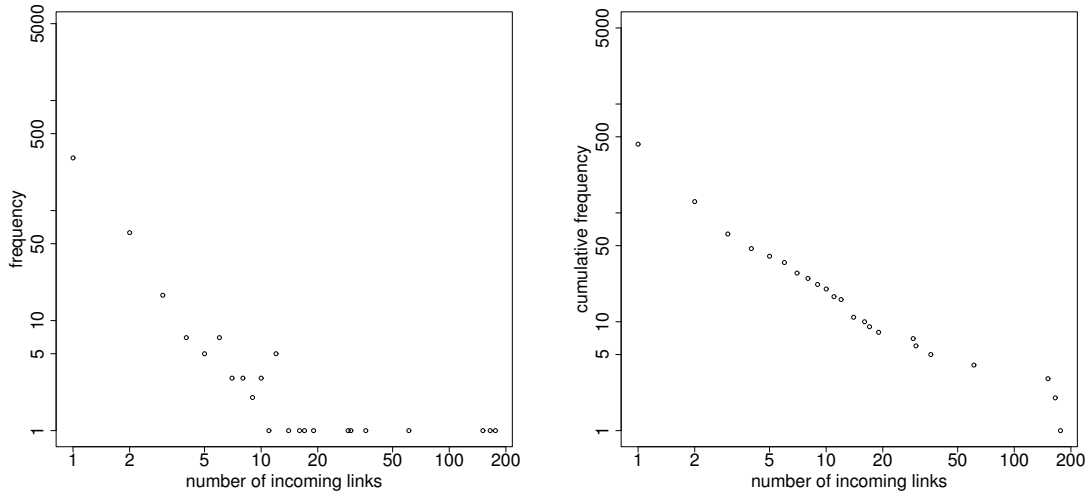
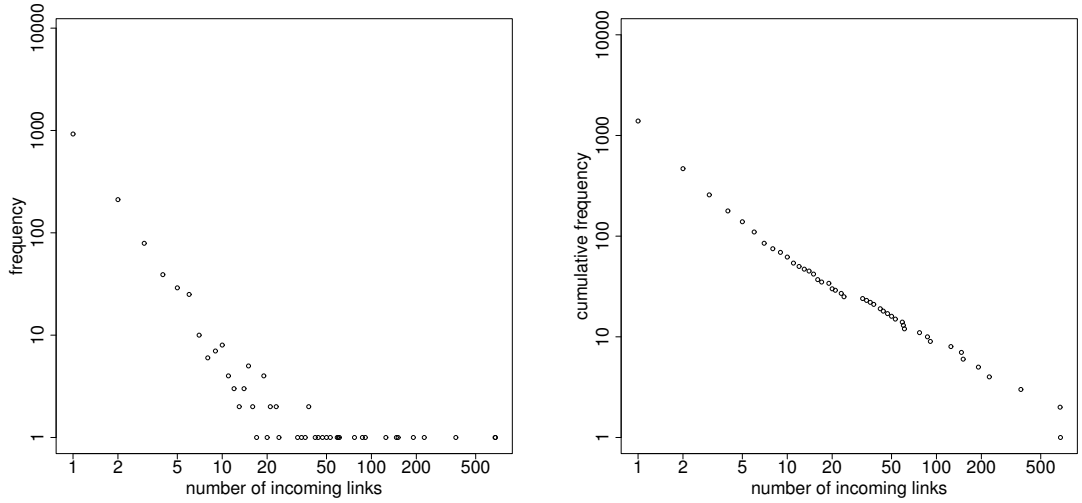
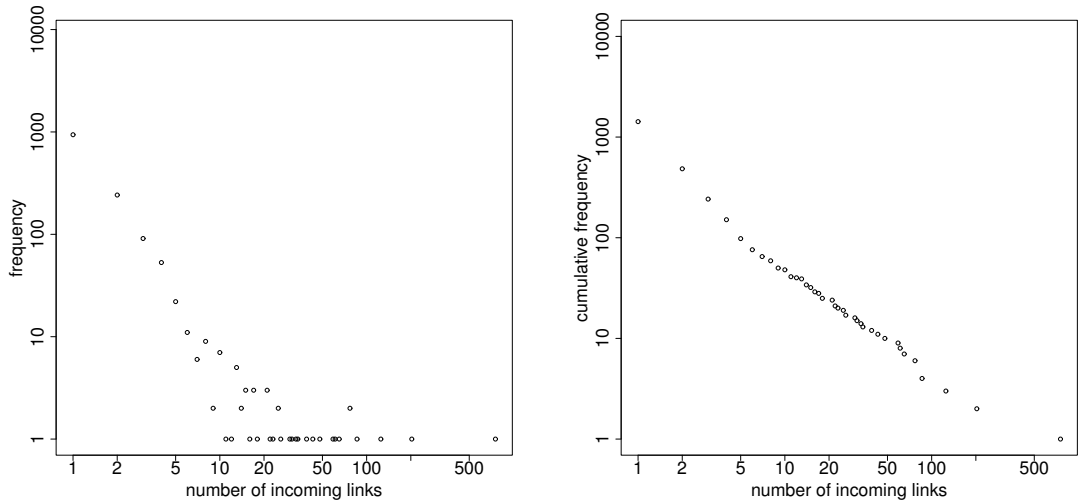


図 23: 実験 3 入力次数の分布 (2)

A7



A8



A9

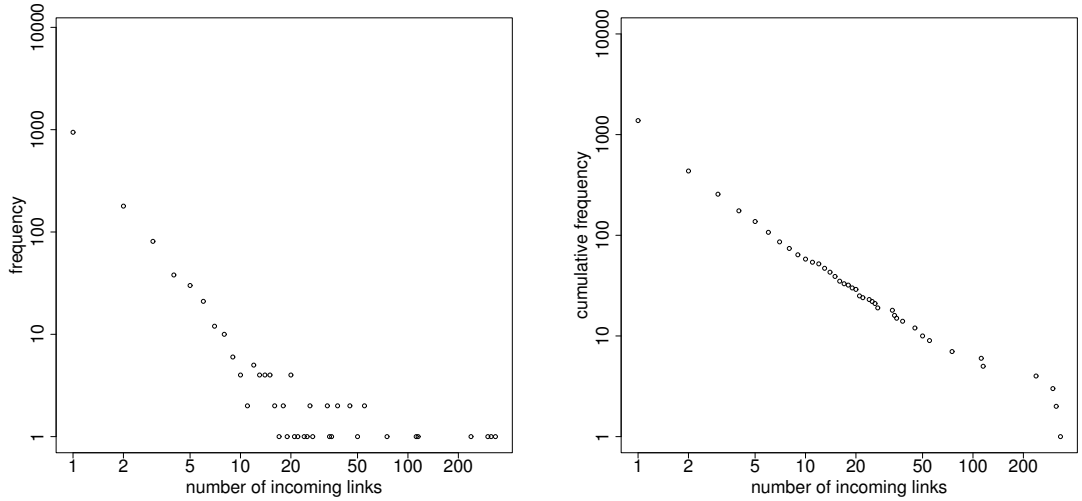
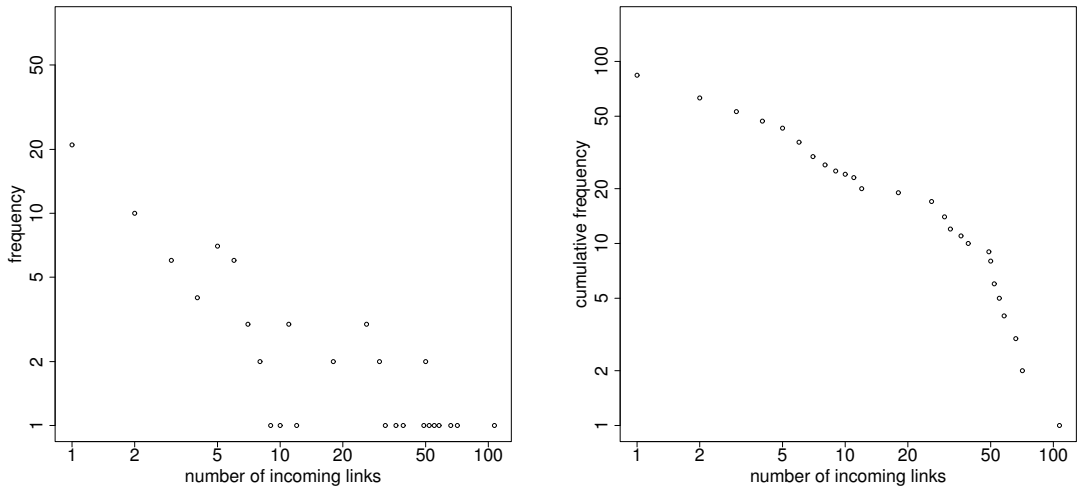
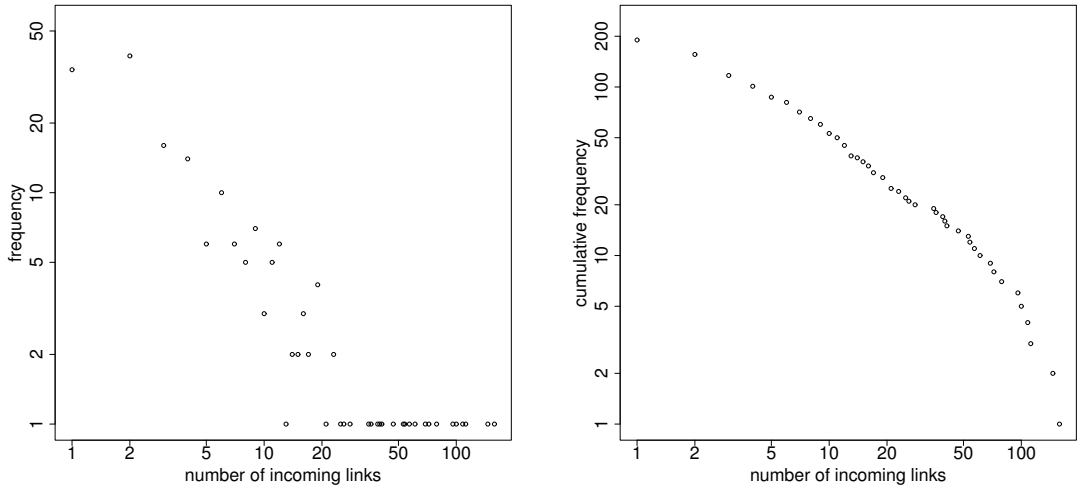


図 24: 実験 3 入力次数の分布 (3)

B1



B2



B3

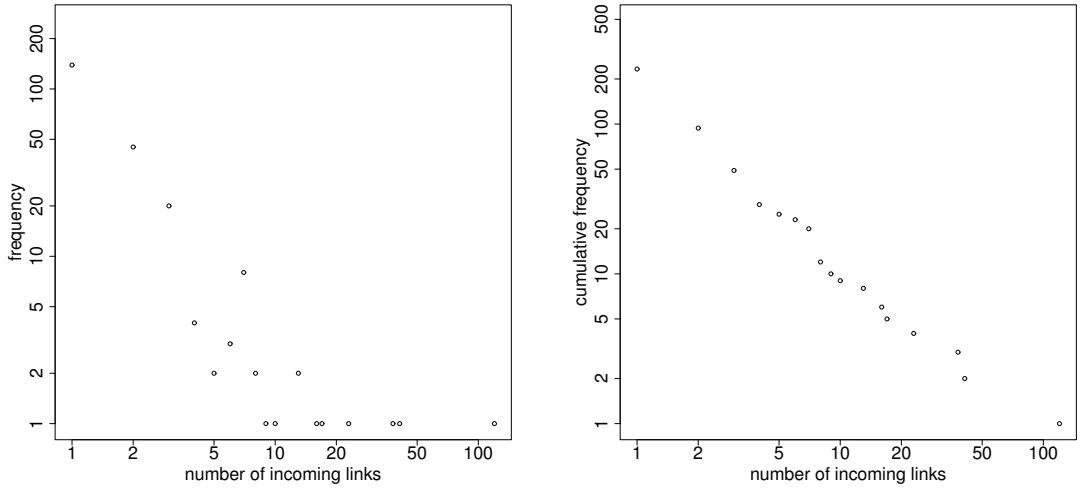
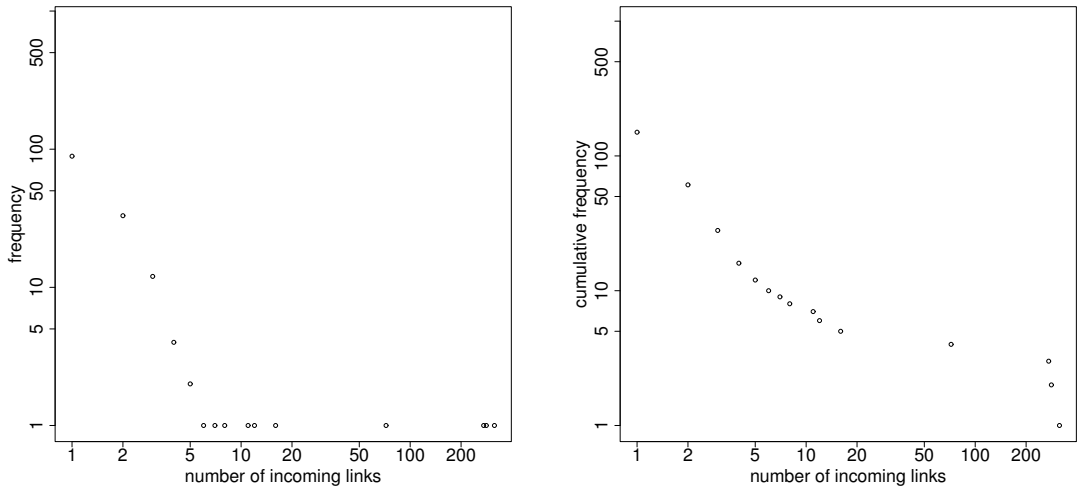
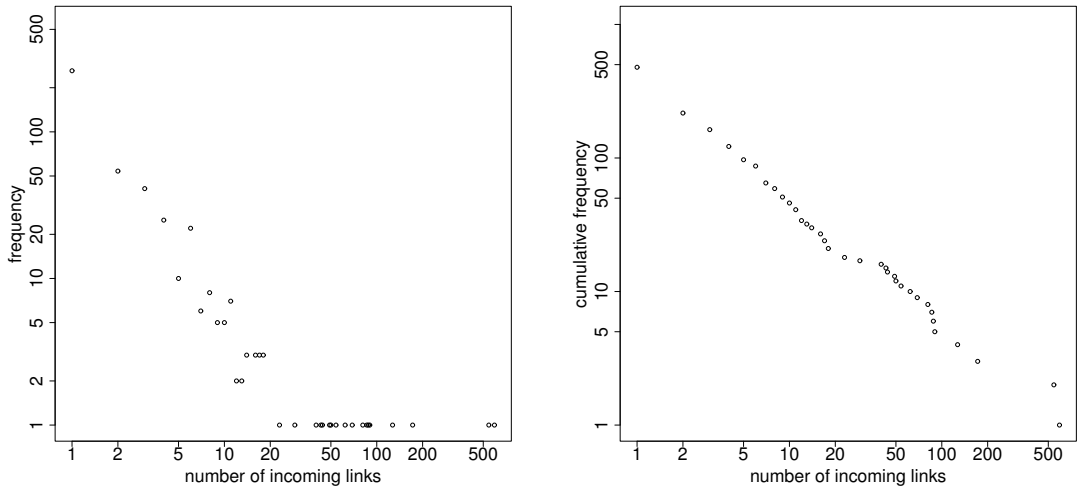


図 25: 実験 3 入力次数の分布 (4)

B4



B5



B6

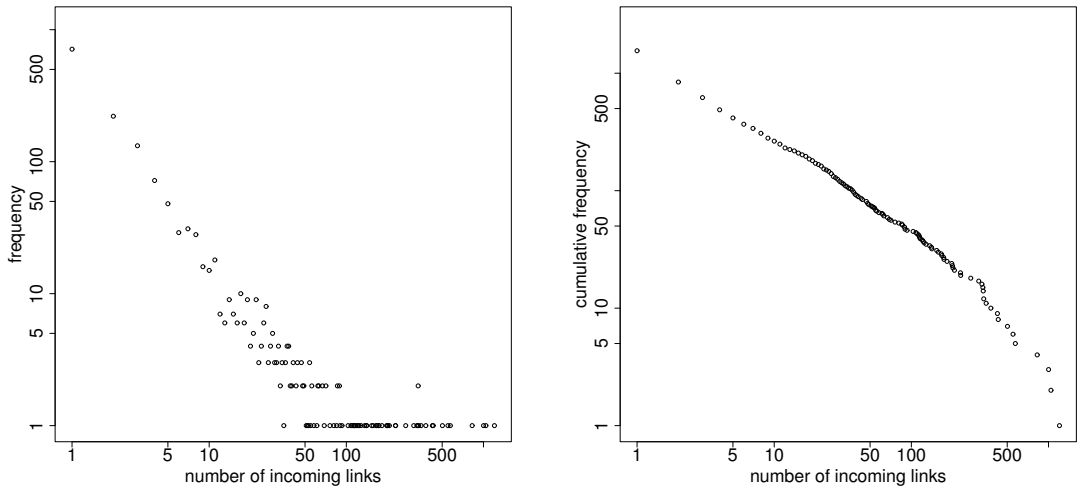
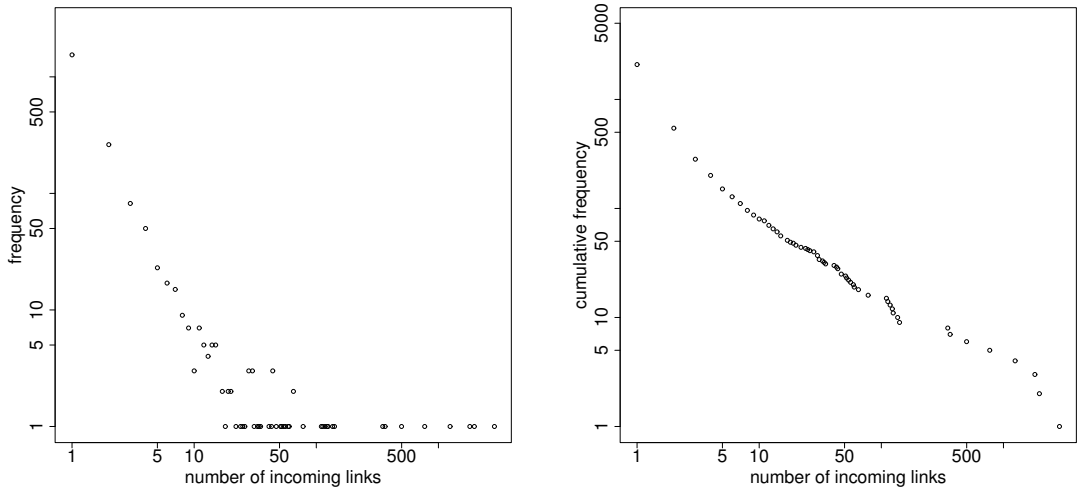
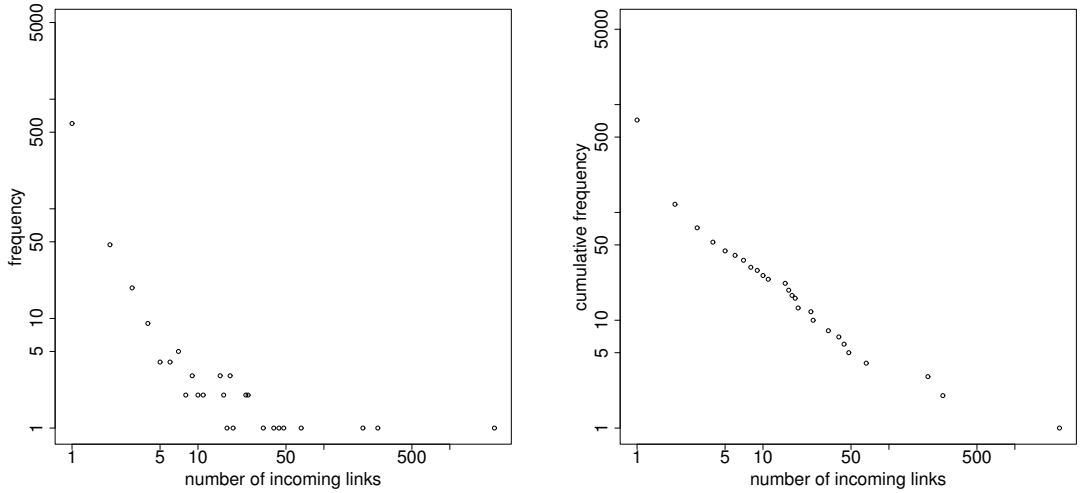


図 26: 実験 3 入力次数の分布 (5)

B7



B8



B9

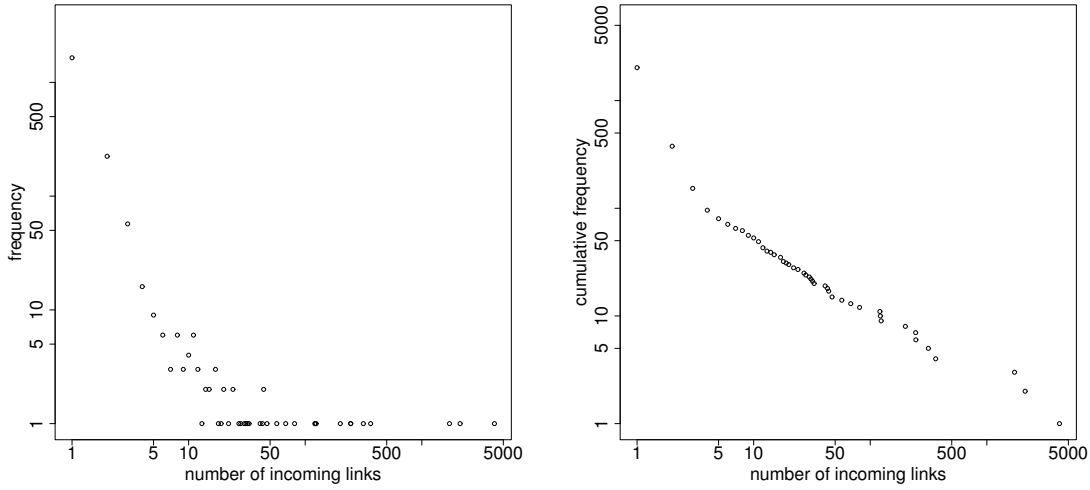
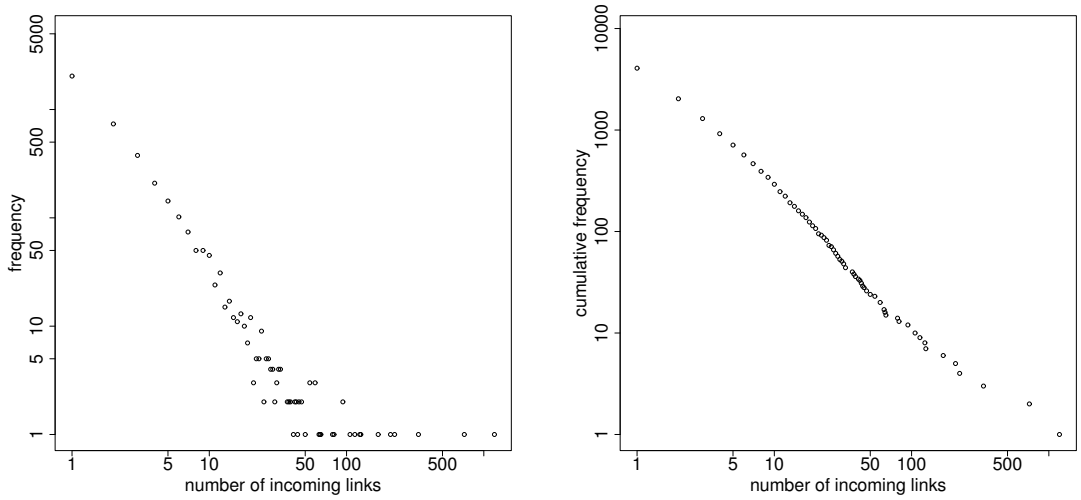
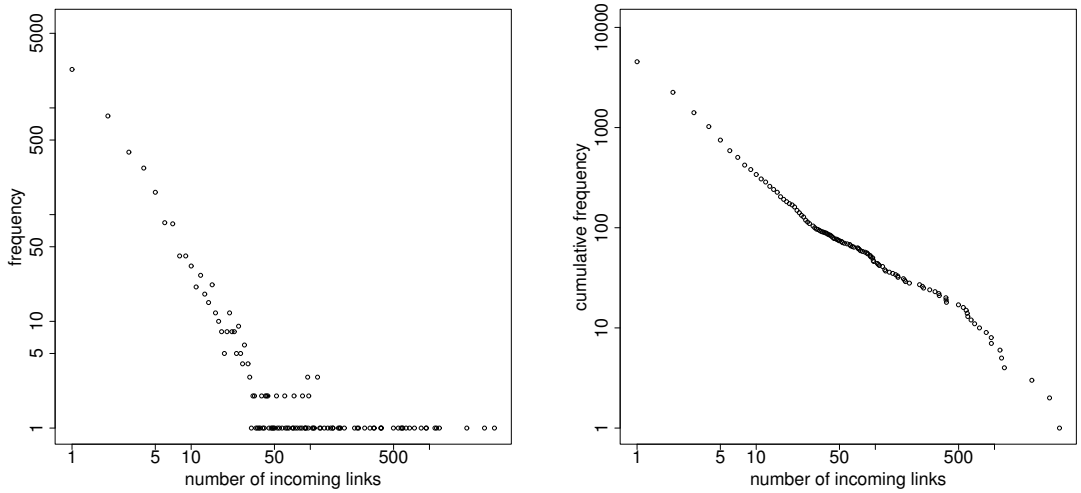


図 27: 実験 3 入力次数の分布 (6)

B10



B11



B12

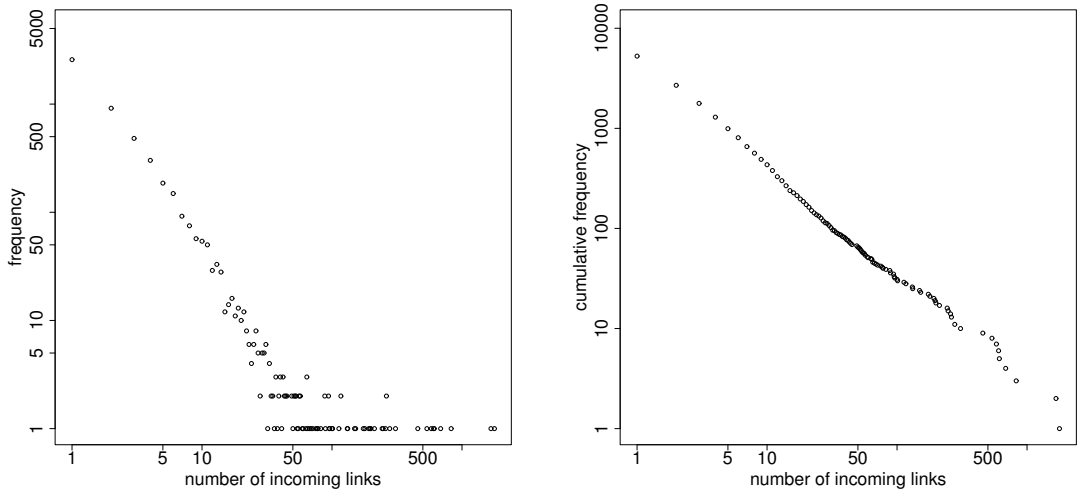
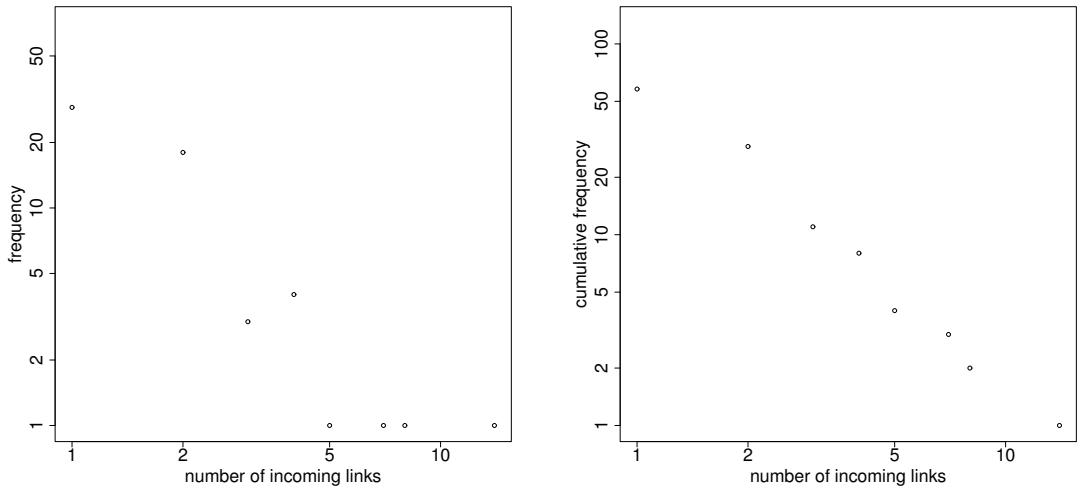
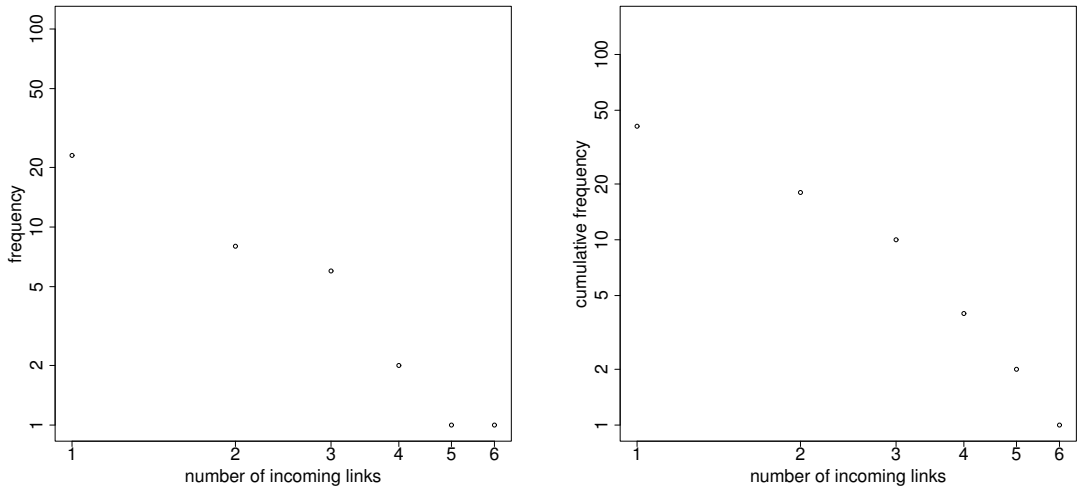


図 28: 実験 3 入力次数の分布 (7)

C1



C2



C3

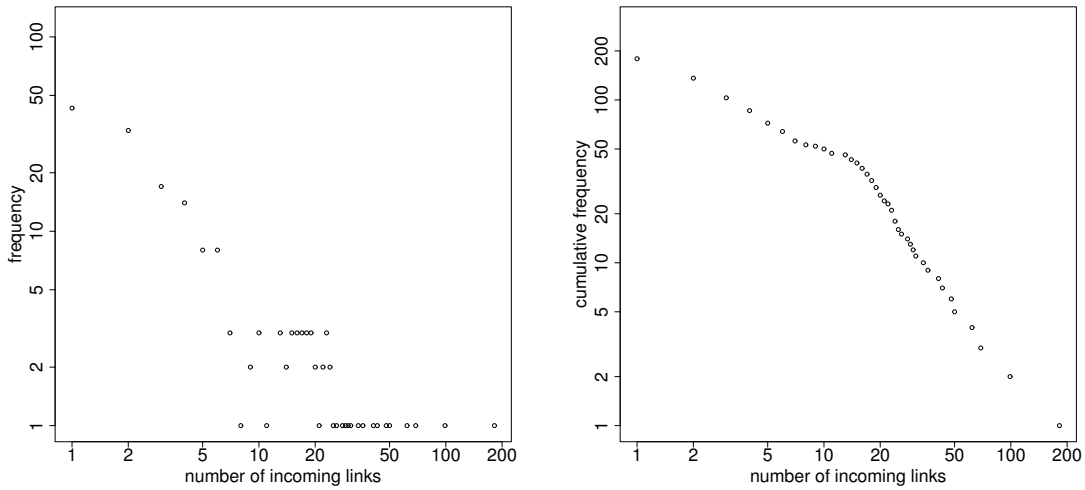
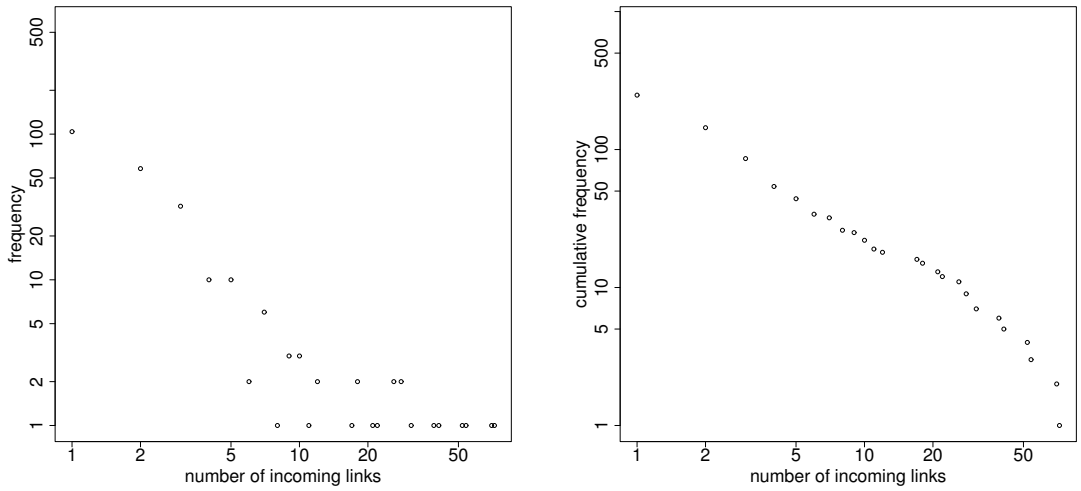
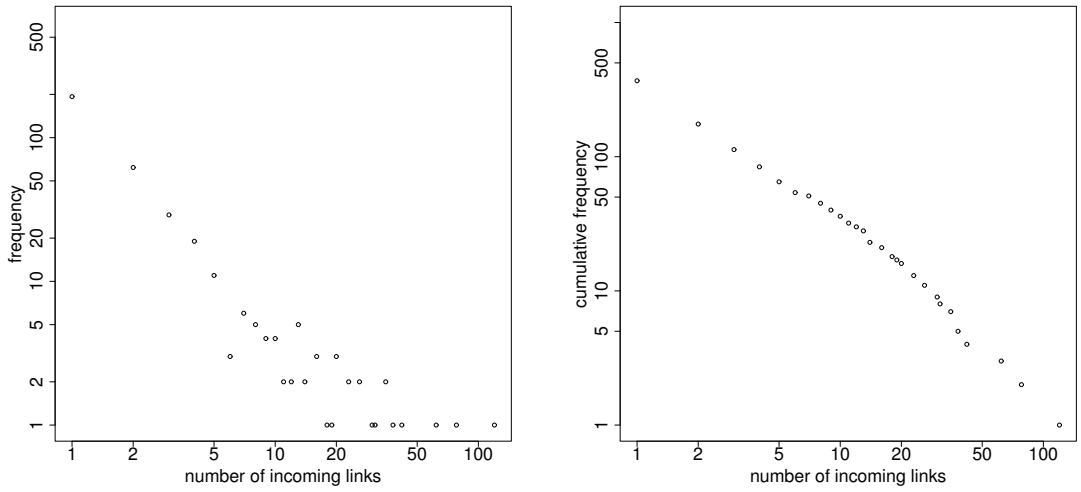


図 29: 実験 3 入力次数の分布 (8)

C4



C5



C6

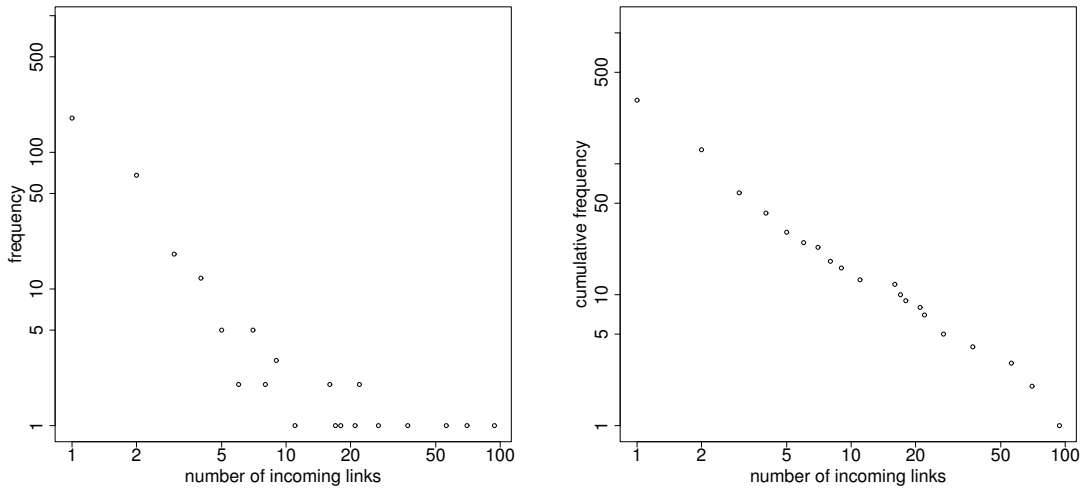
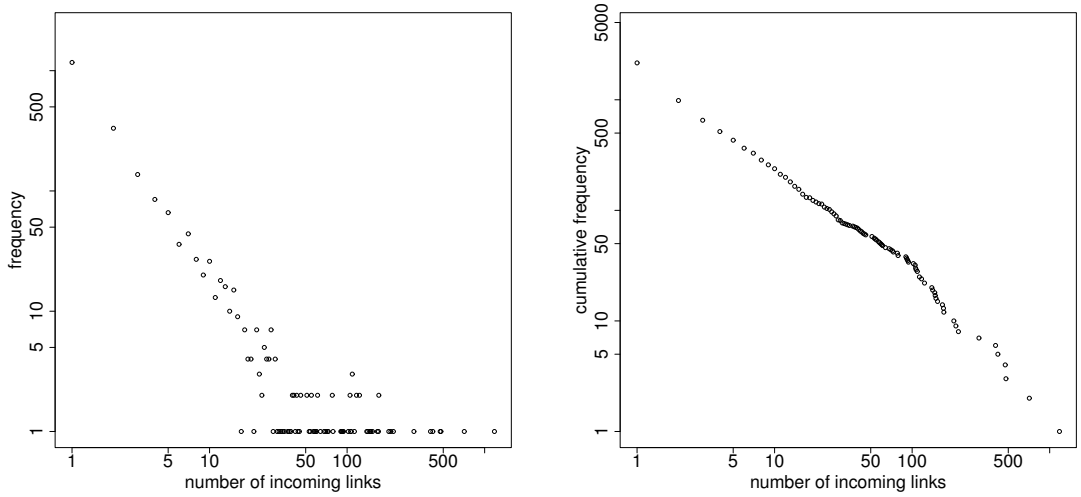
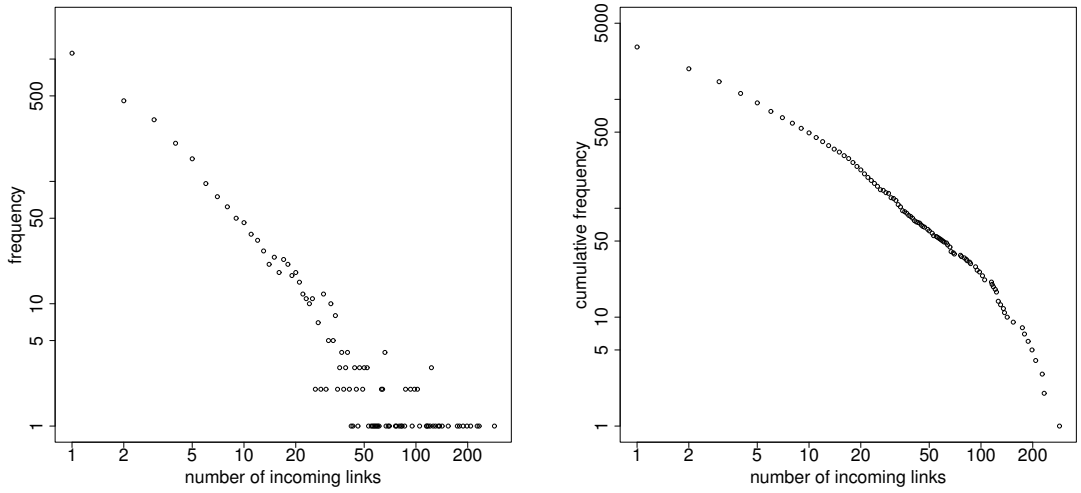


図 30: 実験 3 入力次数の分布 (9)

C7



C8



C9

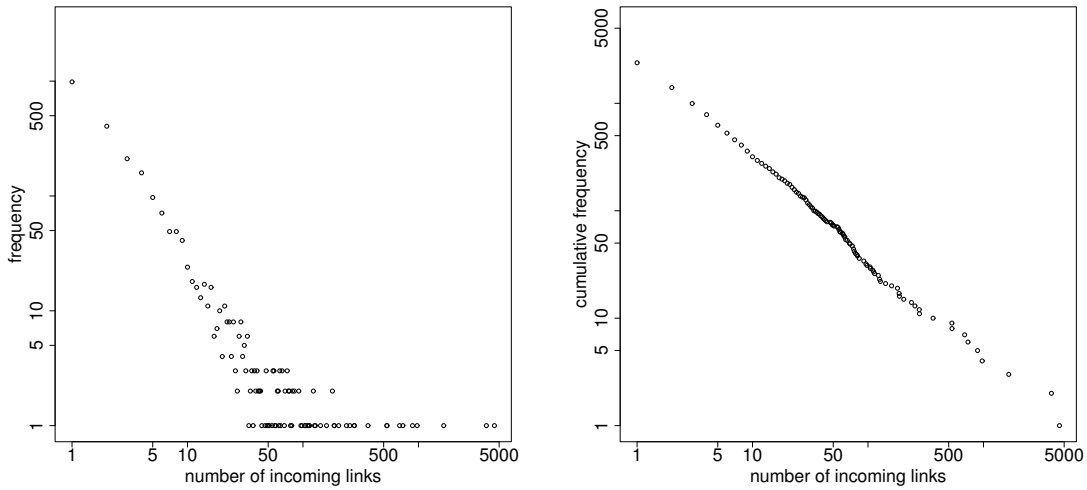
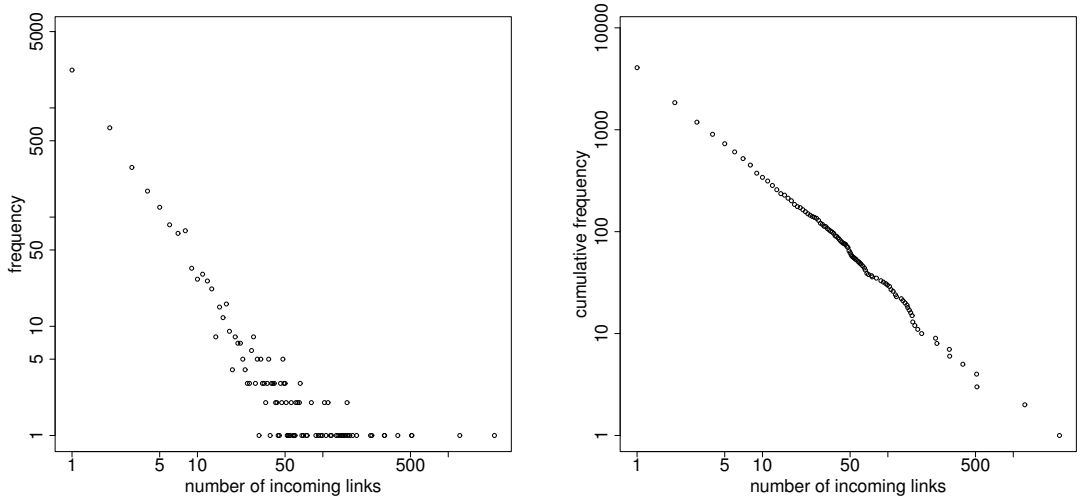
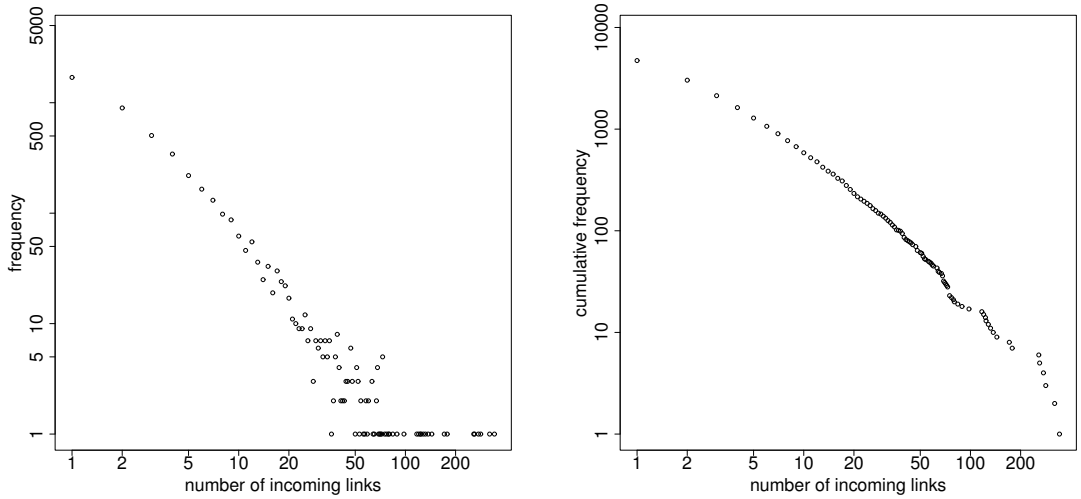


図 31: 実験 3 入力次数の分布 (10)

C10



C11



C12

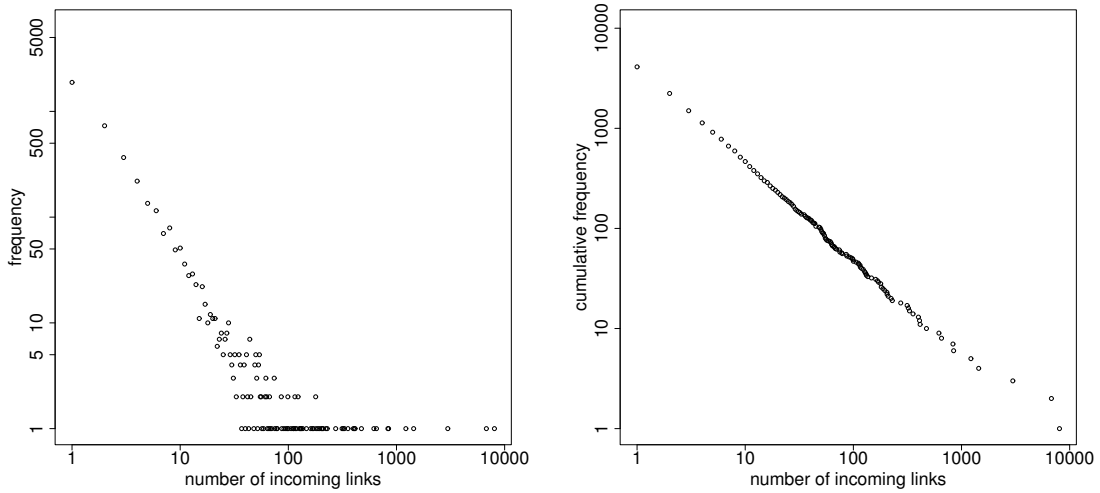
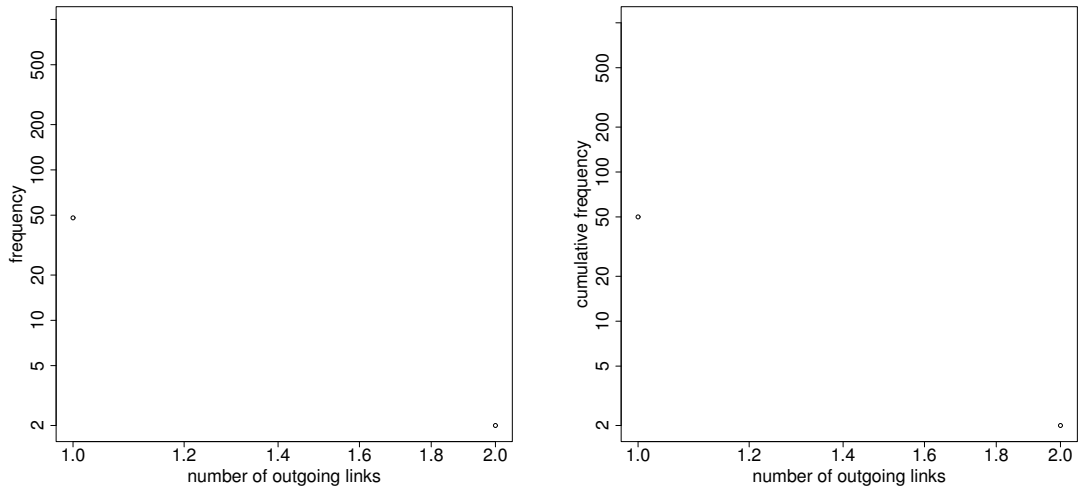
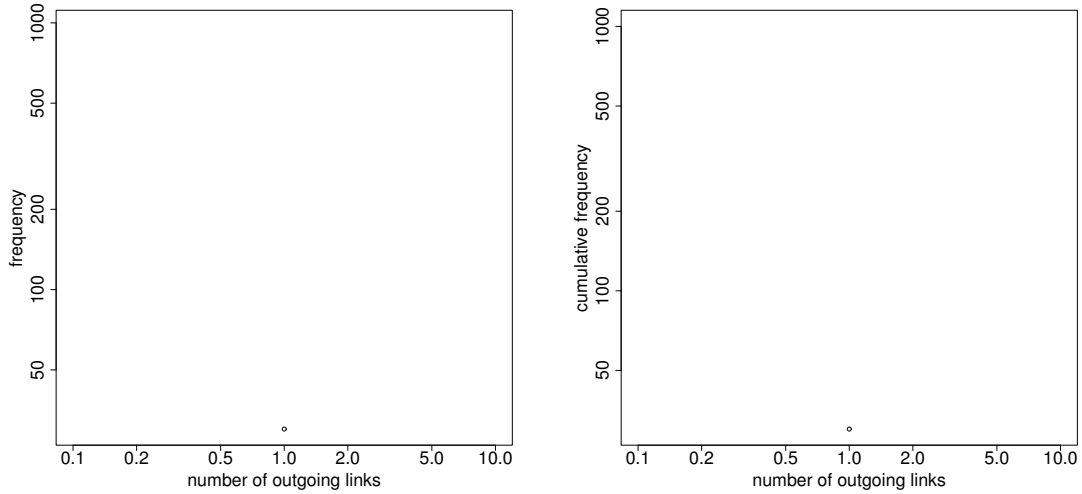


図 32: 実験 3 入力次数の分布 (11)

A1



A2



A3

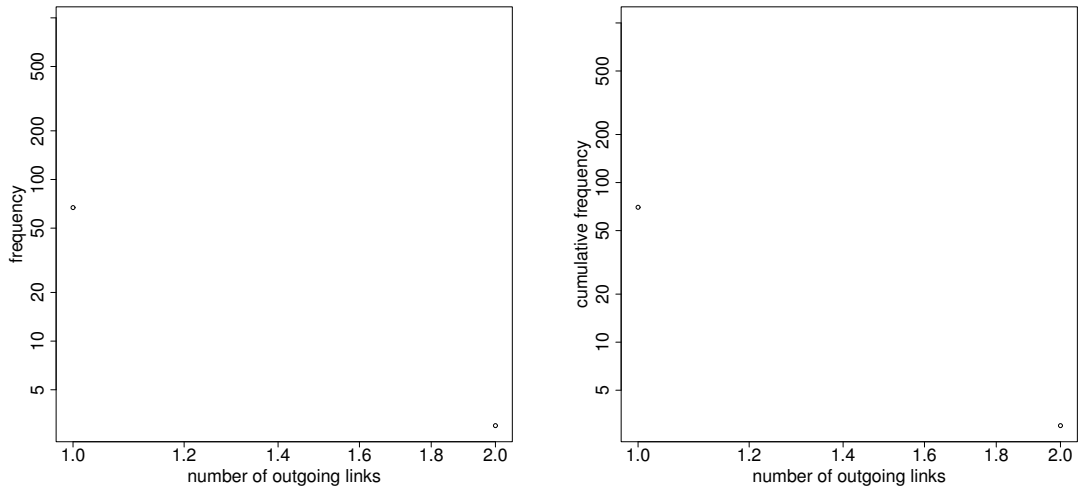
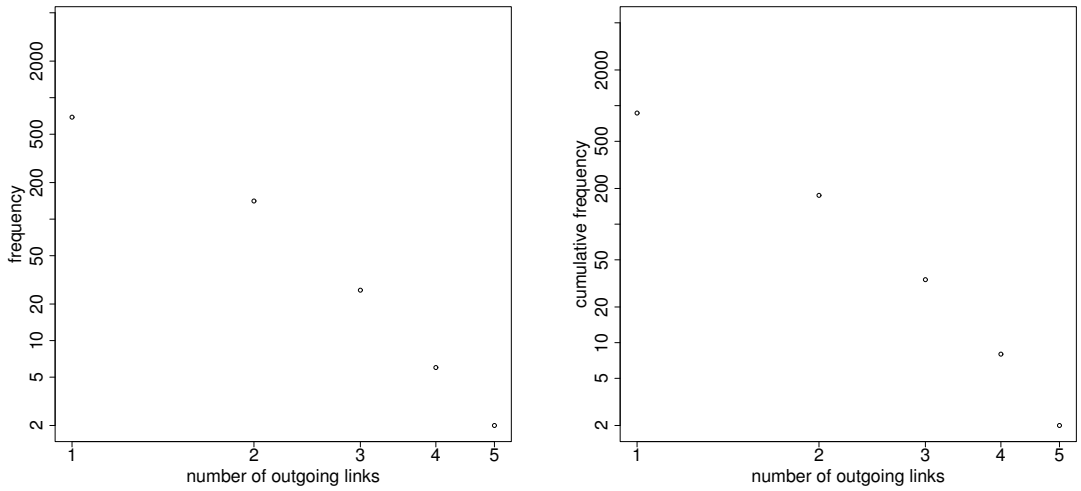
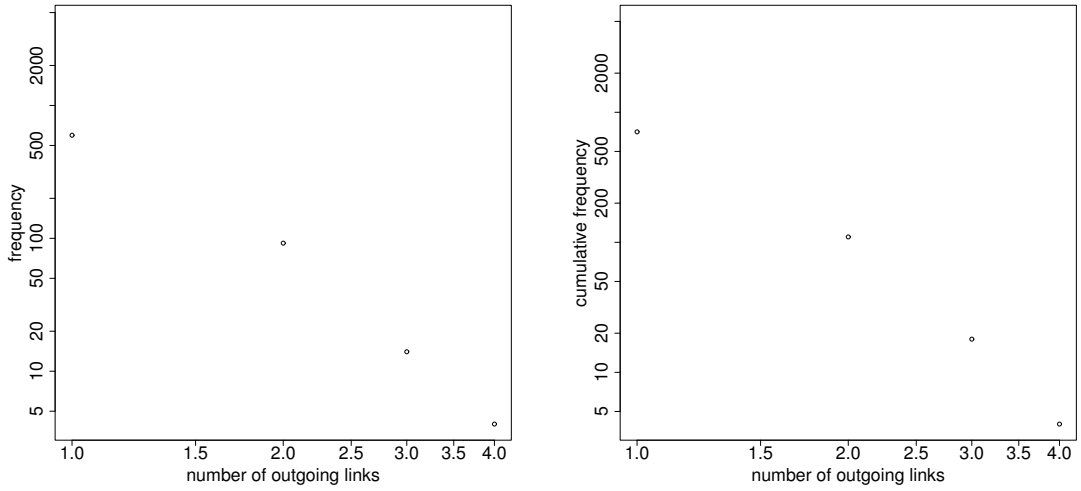


図 33: 実験 3 出力次数の分布 (1)

A4



A5



A6

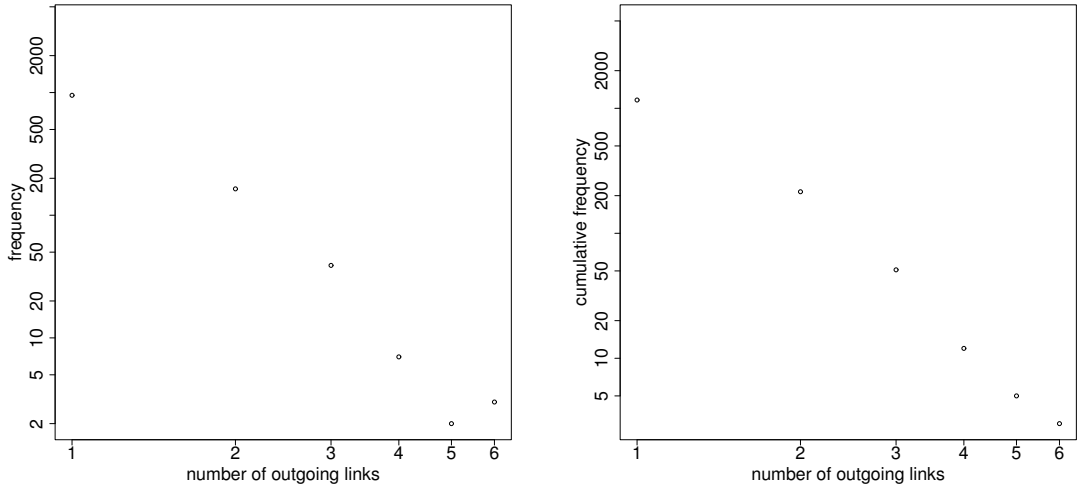
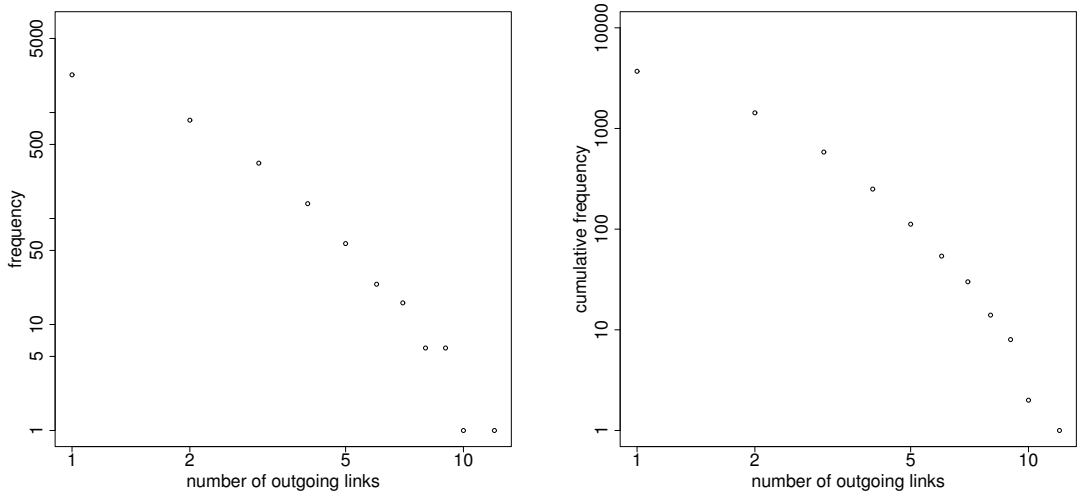
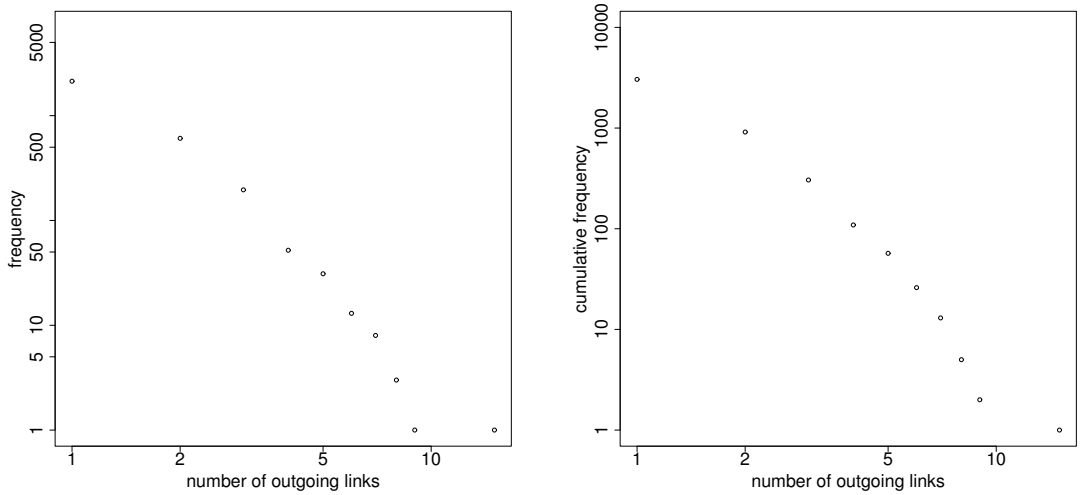


図 34: 実験 3 出力次数の分布 (2)

A7



A8



A9

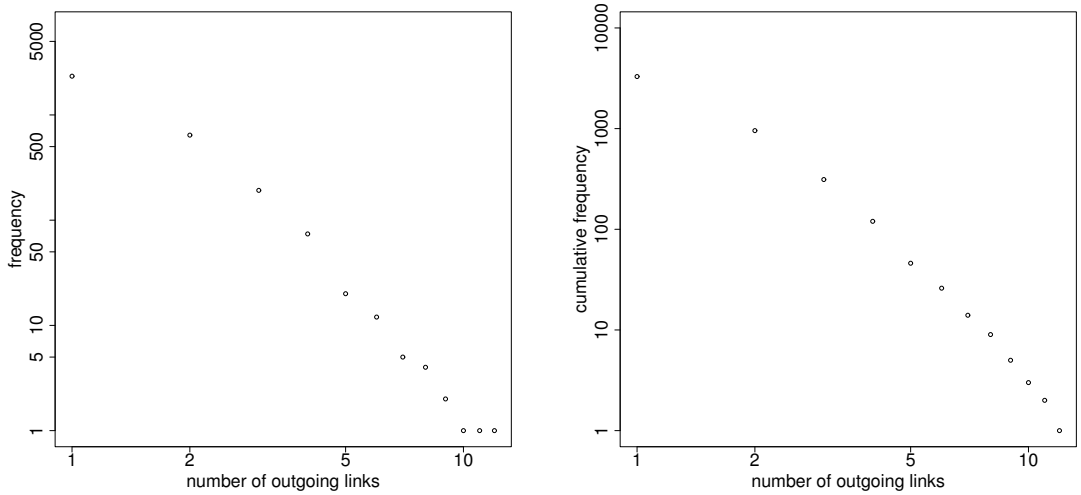
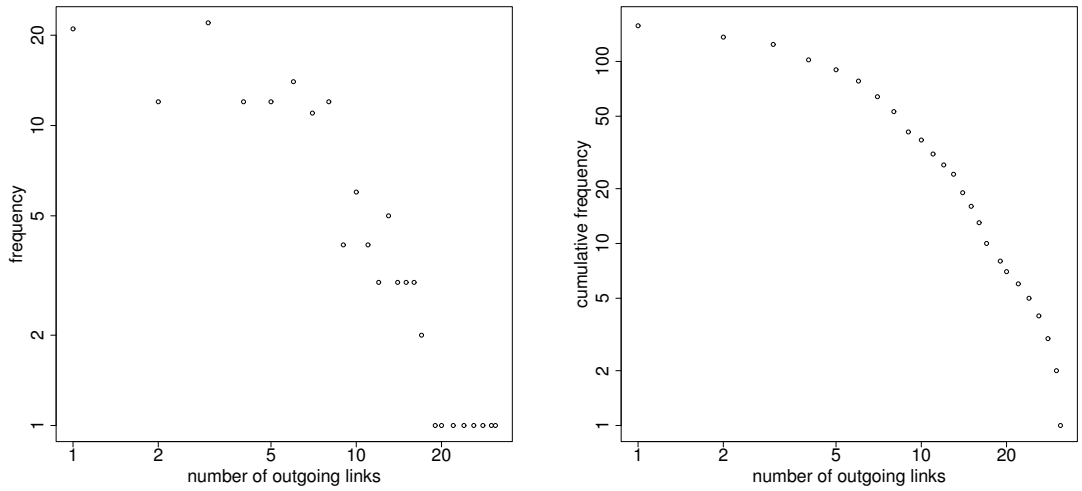
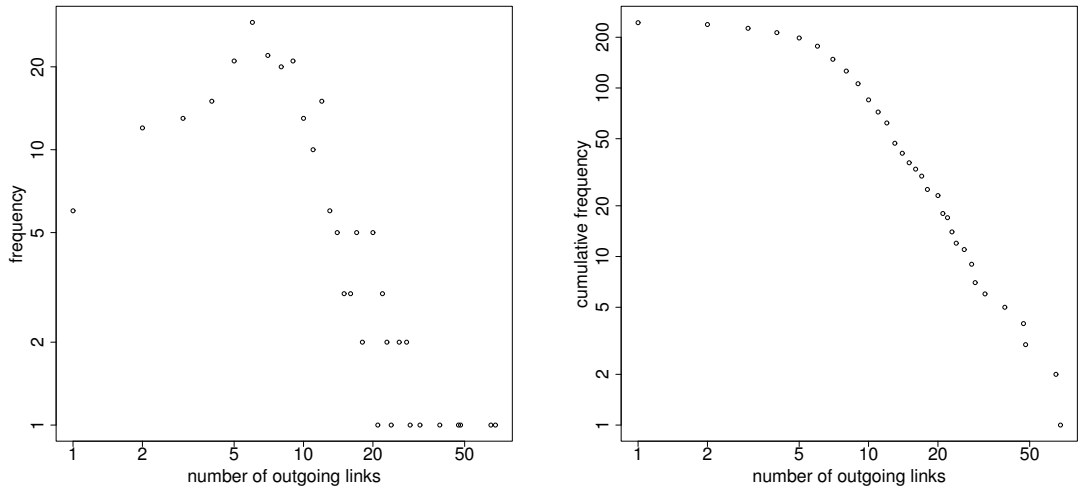


図 35: 実験 3 出力次数の分布 (3)

B1



B2



B3

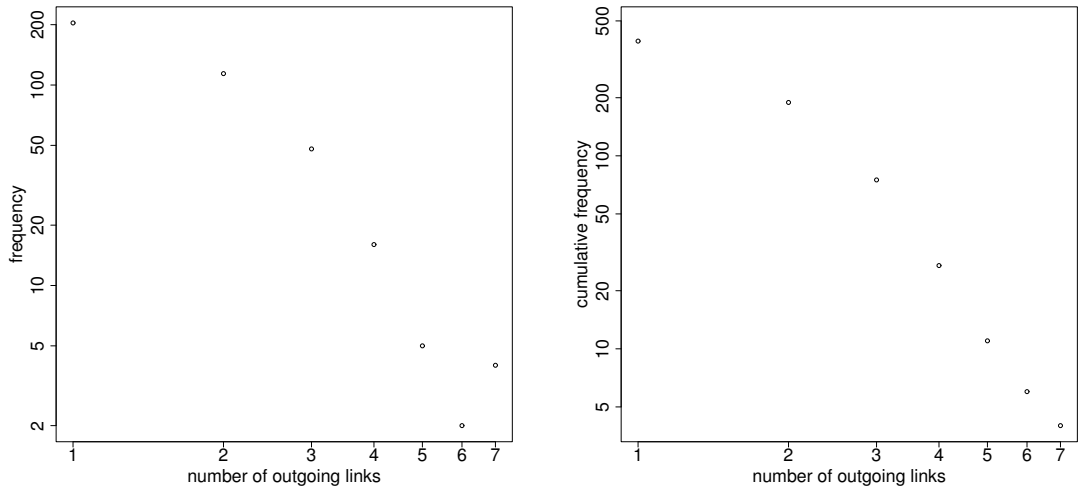
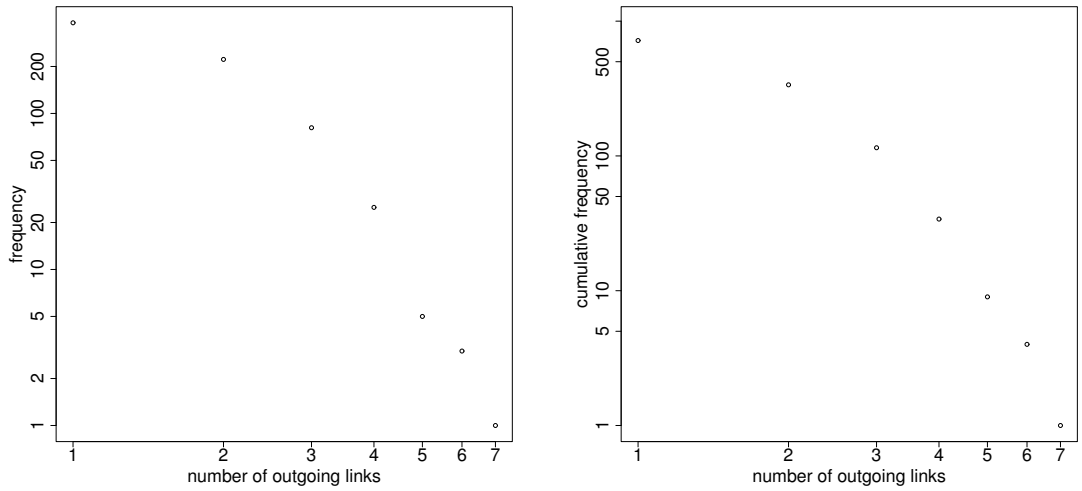
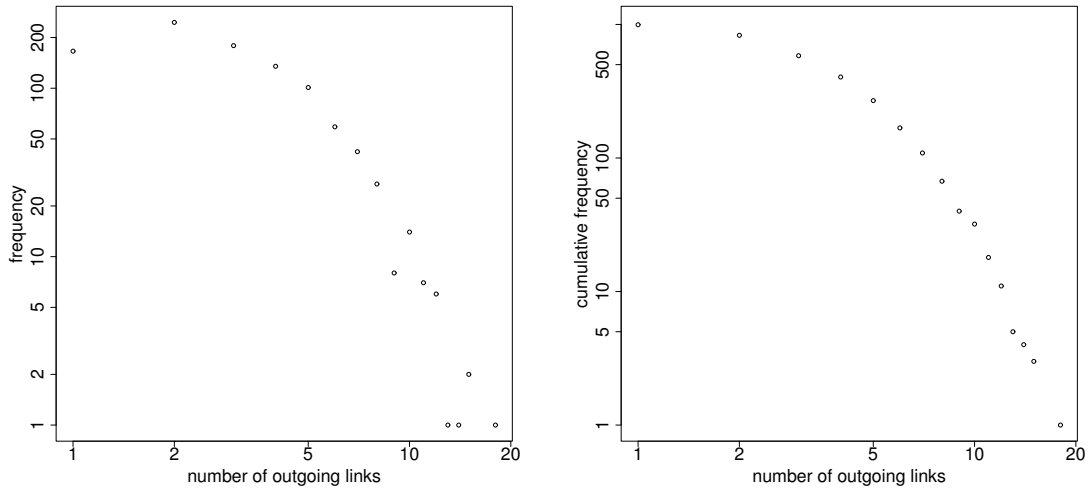


図 36: 実験 3 出力次数の分布 (4)

B4



B5



B6

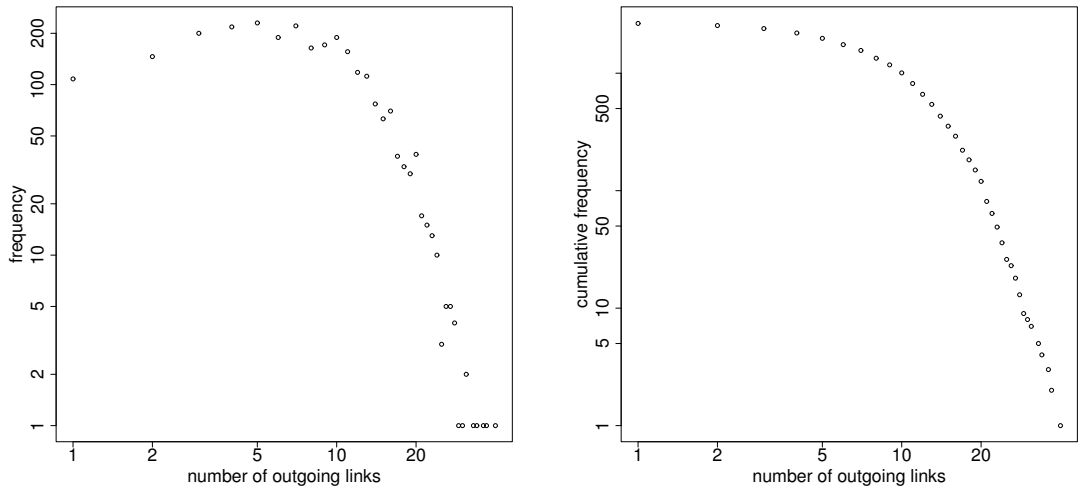
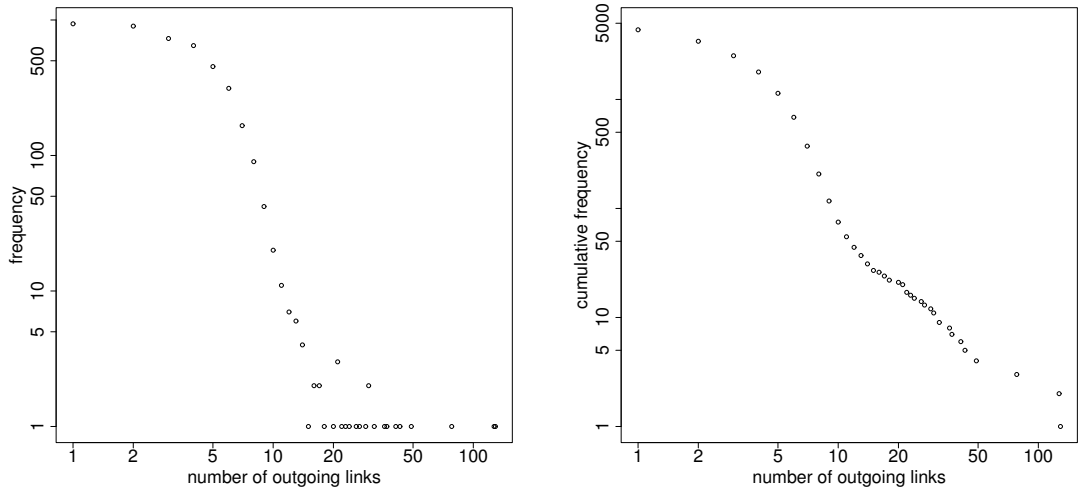
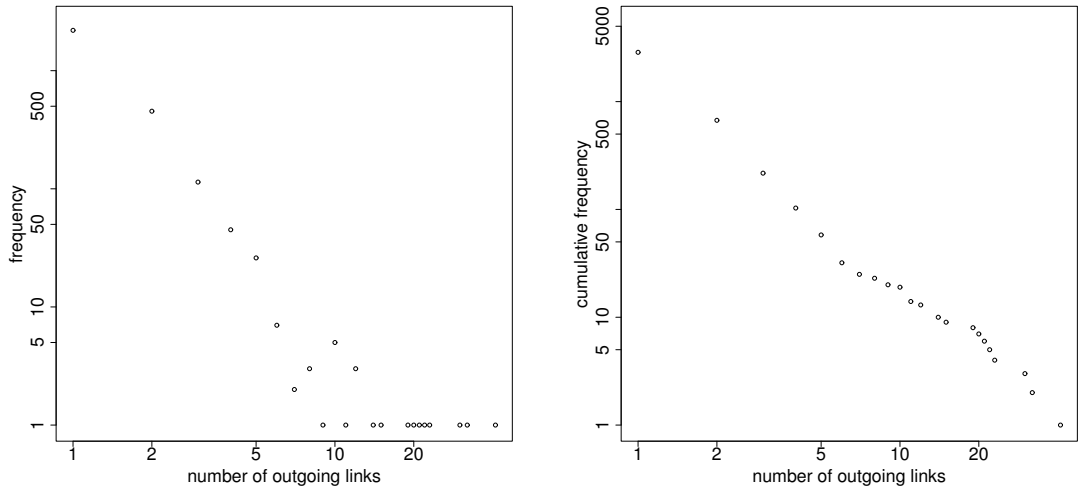


図 37: 実験 3 出力次数の分布 (5)

B7



B8



B9

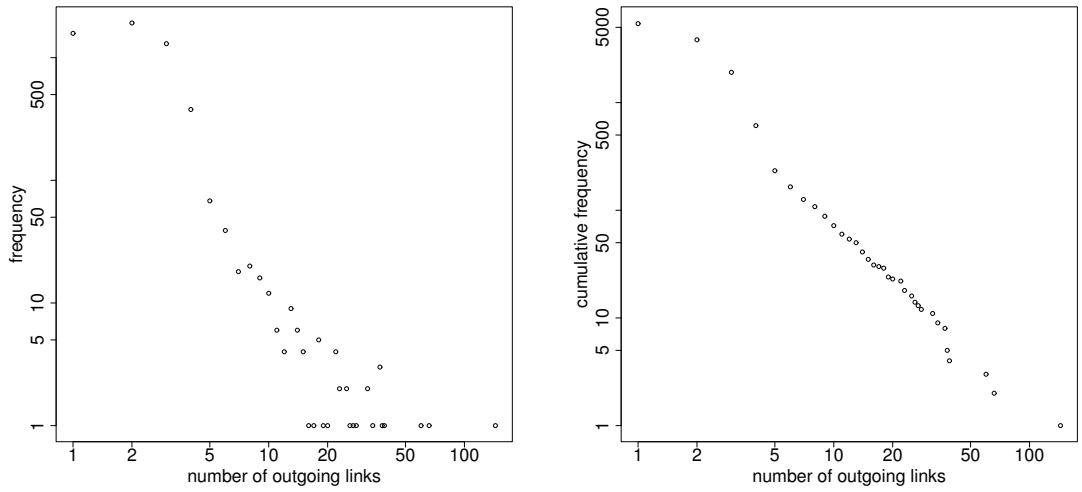
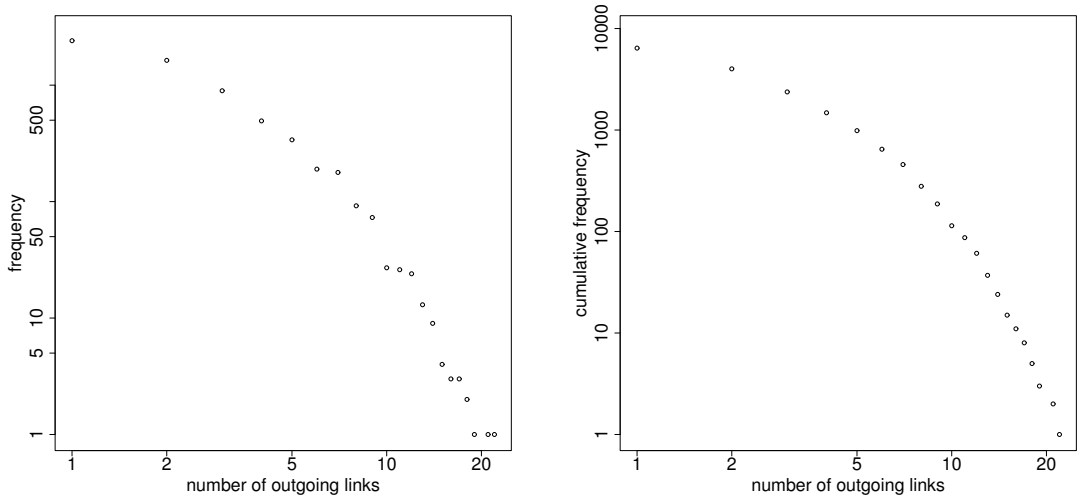
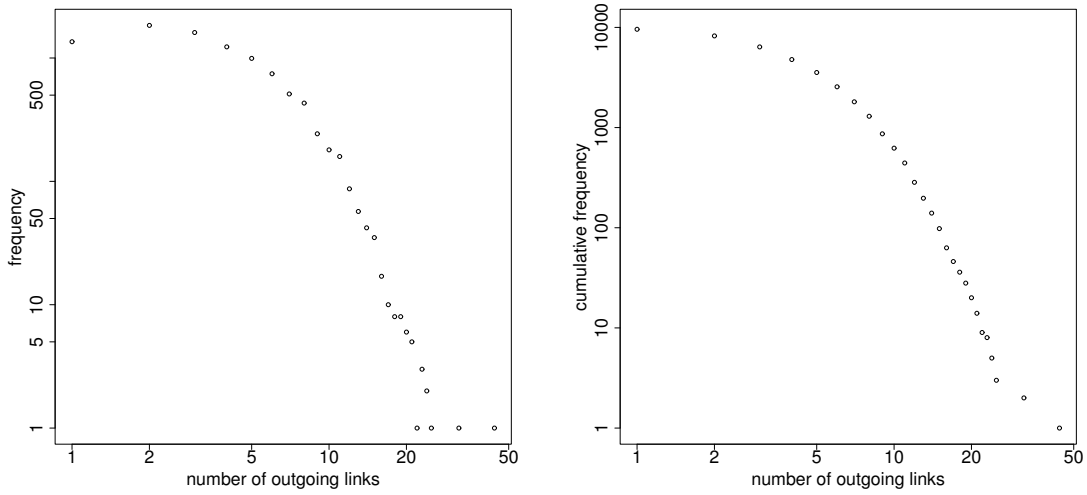


図 38: 実験 3 出力次数の分布 (6)

B10



B11



B12

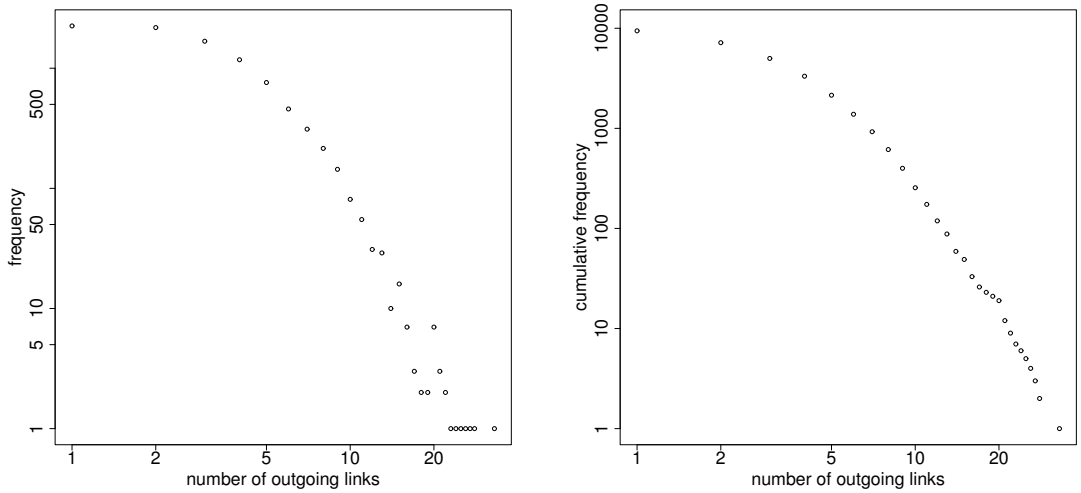
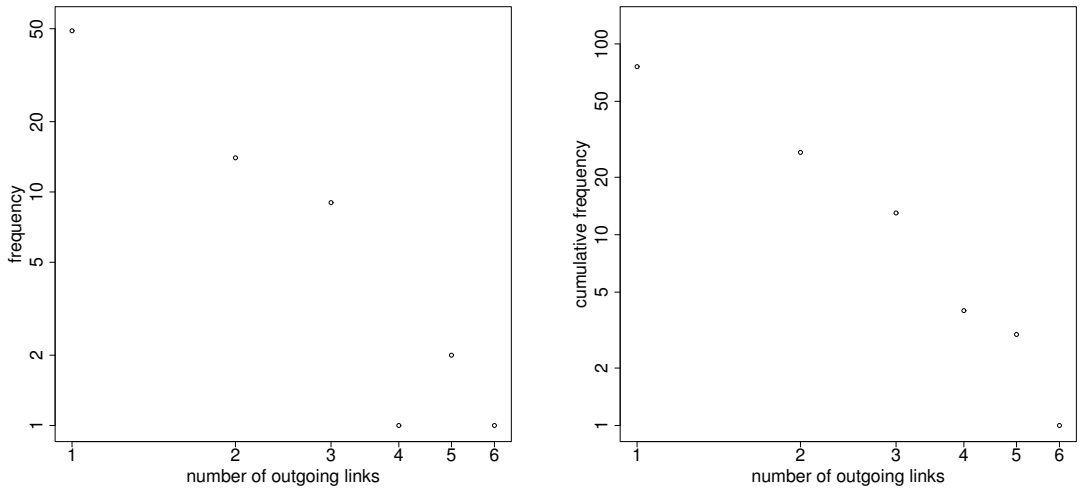
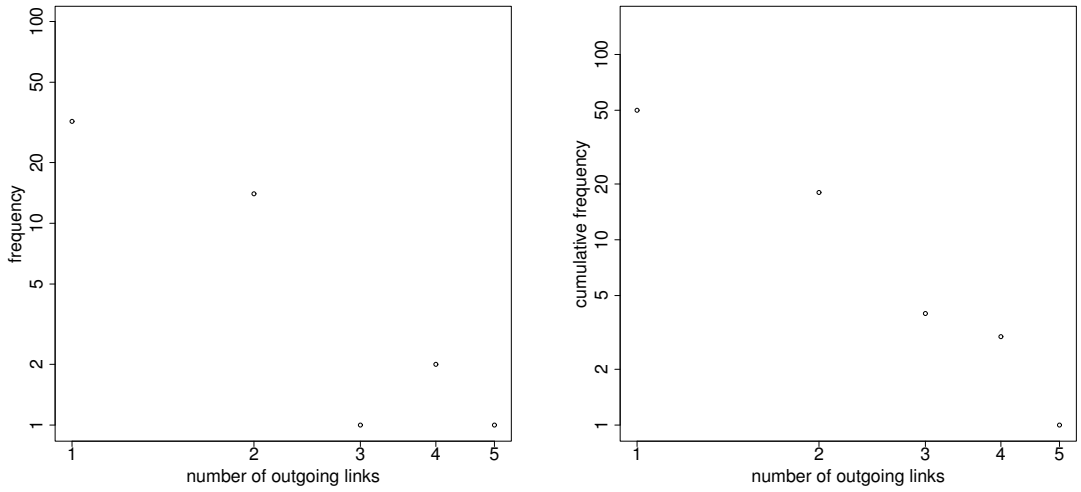


図 39: 実験 3 出力次数の分布 (7)

C1



C2



C3

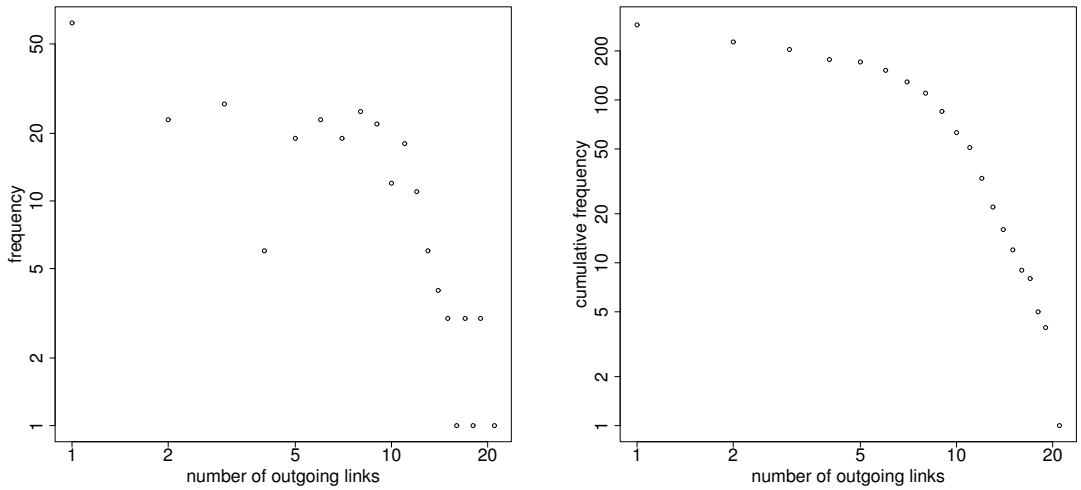
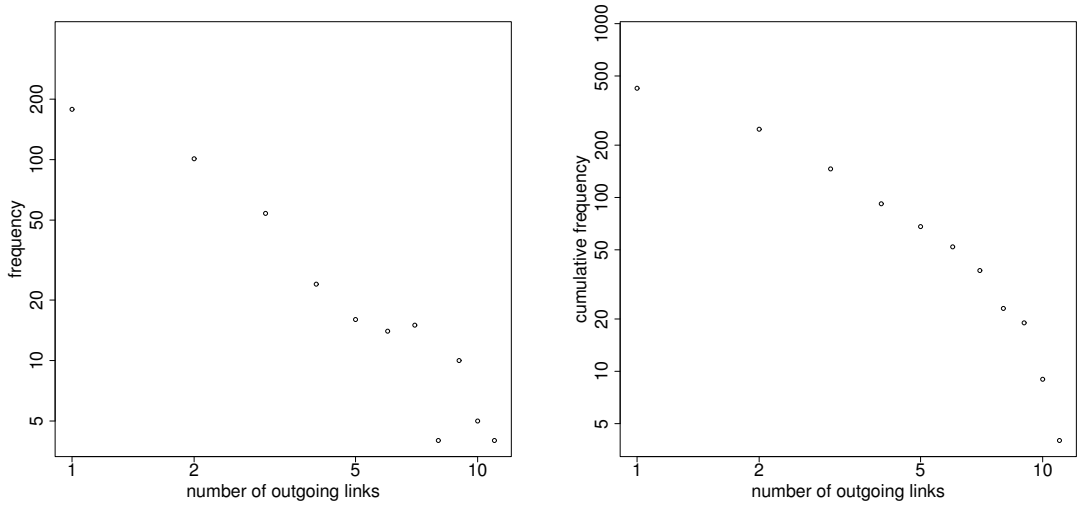
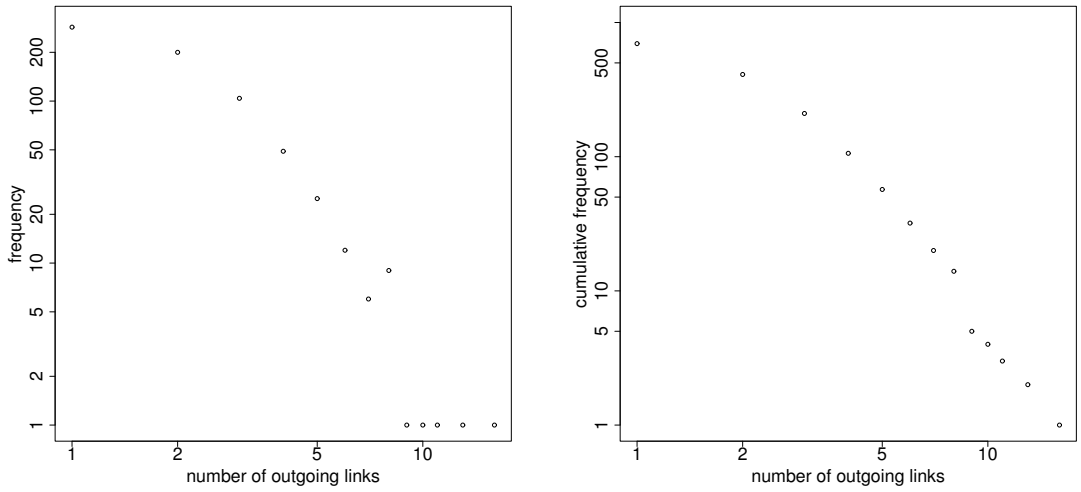


図 40: 実験 3 出力次数の分布 (8)

C4



C5



C6

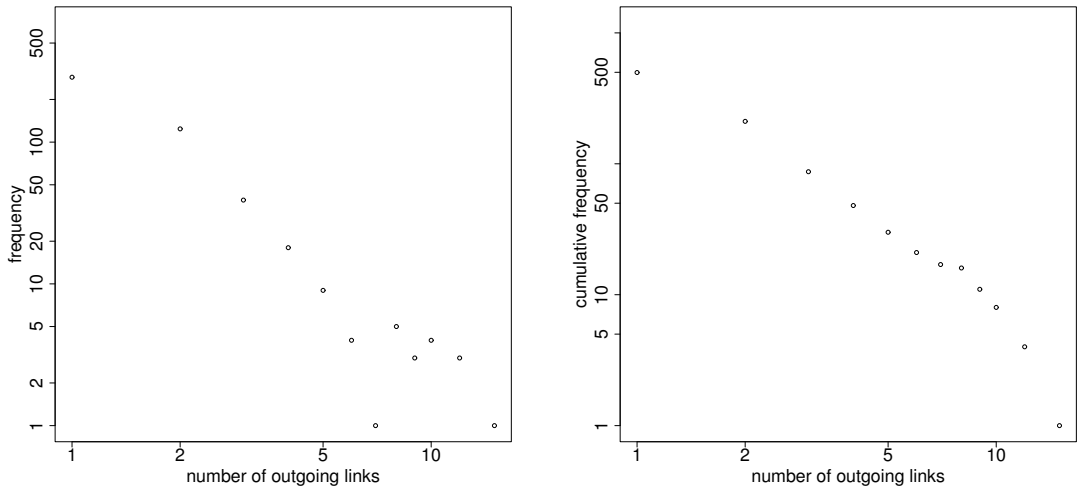
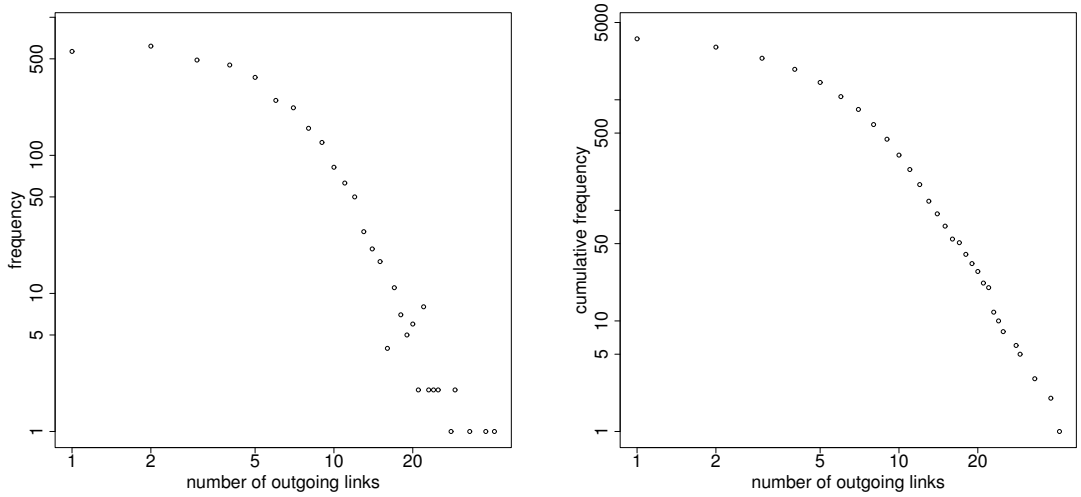
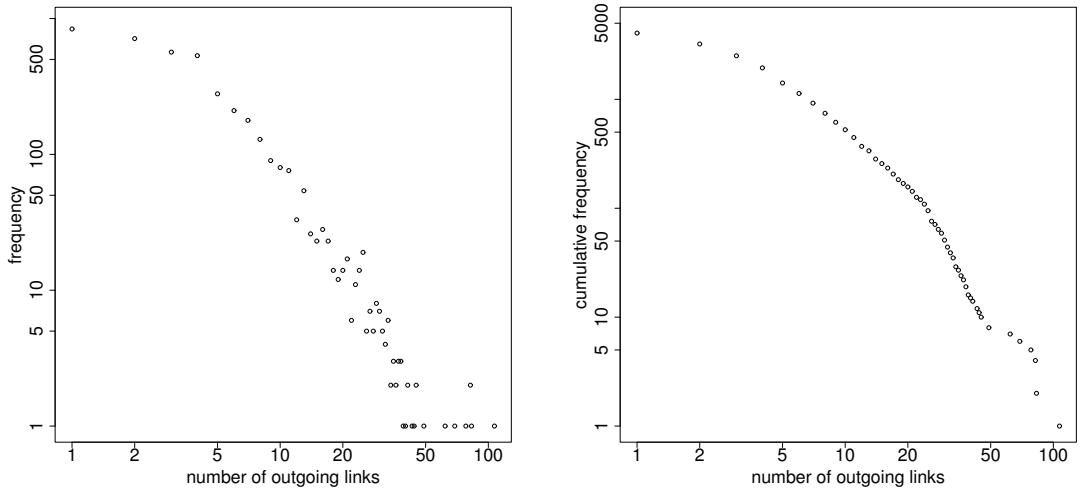


図 41: 実験 3 出力次数の分布 (9)

C7



C8



C9

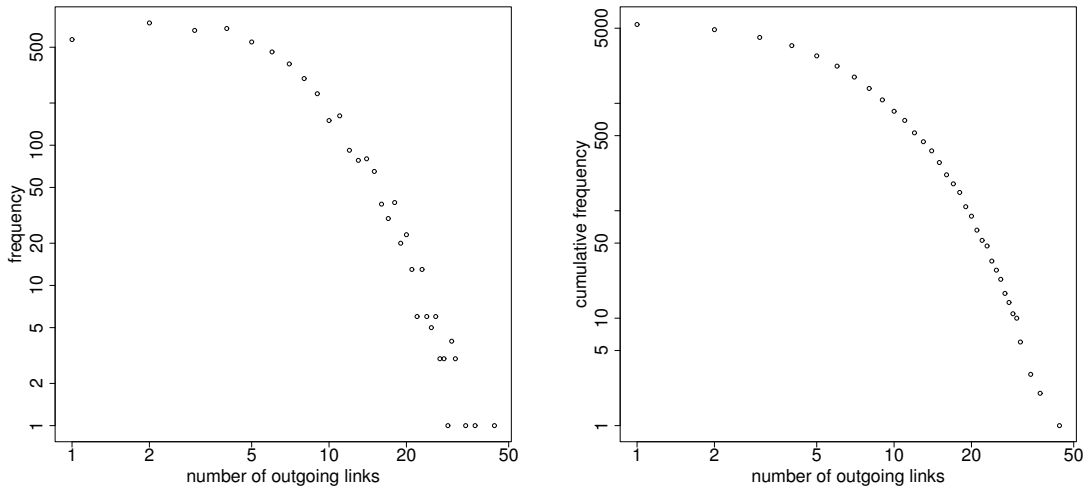
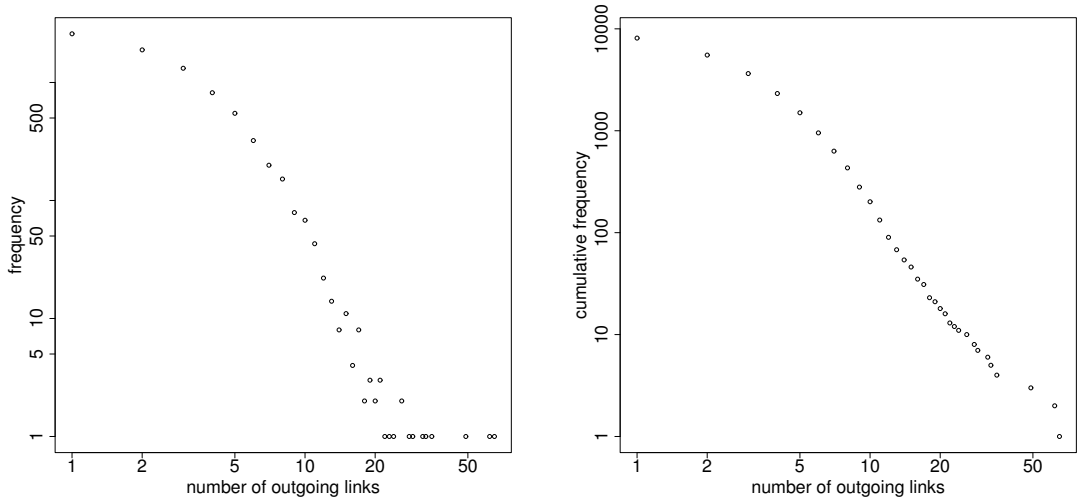
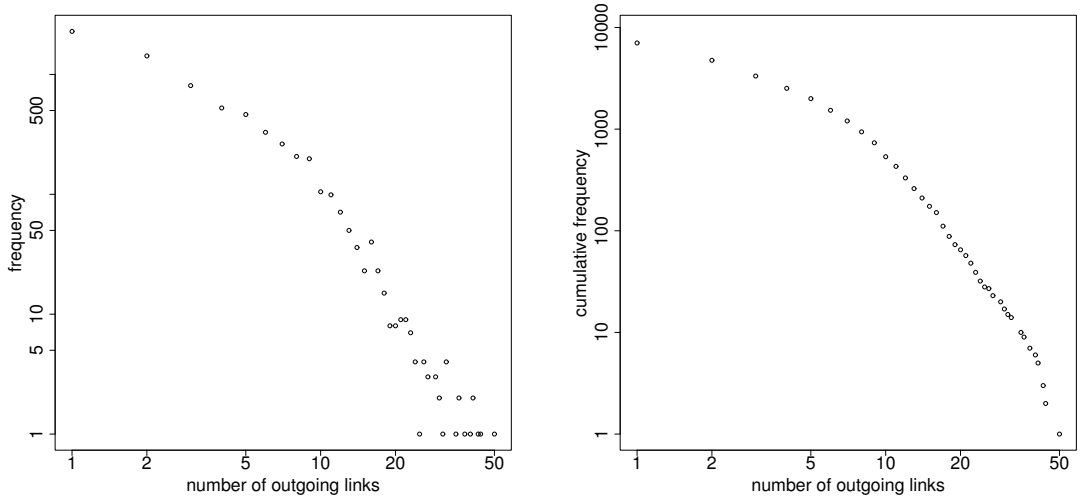


図 42: 実験 3 出力次数の分布 (10)

C10



C11



C12

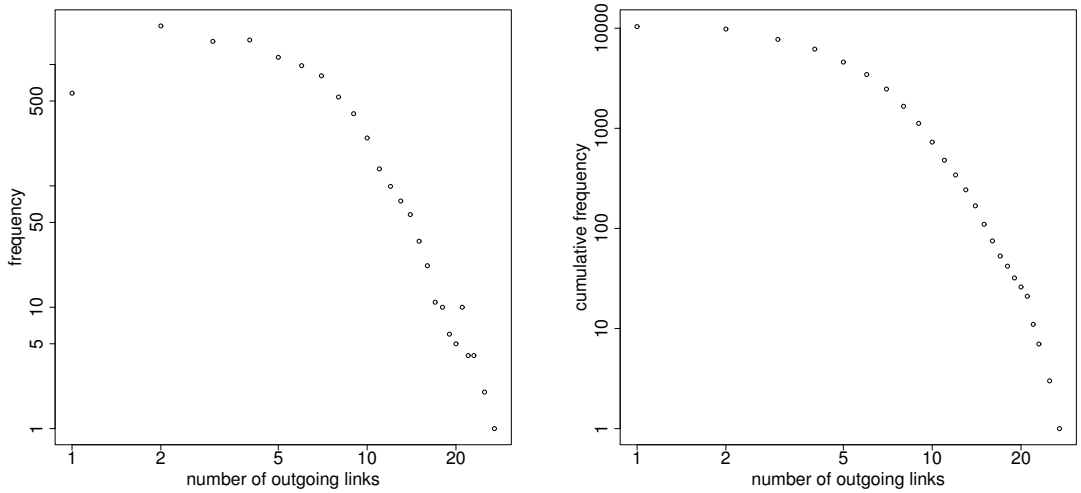


図 43: 実験 3 出力次数の分布 (11)