圧縮アルゴリズムに基づく相似性と類似ファイル数を併用したプロダクト派生関係の推定手法の提案

索手　一平†　　早瀬　康裕††　　北川　博之††

† 筑波大学情報学群情報科学類　〒 305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学システム情報系情報工学域　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †ippei.nawate@kde.cs.tsukuba.ac.jp, ††{hayase,kitagawa}@cs.tsukuba.ac.jp

あらまし　ソフトウェアプロダクトは，その派生関係を辿ることで，保守を効率的に行うことができる．しかし，開発の過程で派生関係の記録が失われてしまい，開発者は正確に派生関係を辿ることができず，効率的な保守が困難となるという問題がある．この問題に対し，失われた派生関係のグラフを推定する手法として，プロダクト間の類似性を類似ファイル数に基づいて計測する PRET と，可逆圧縮に基づいて計測する EEGL の二つが提案されている．EEGL の評価実験において，二つの手法の推定精度は同程度であるものの，推定結果の誤りの傾向に違いがあることがわかっている．このことから，2 つの手法を組み合わせることで，互いの誤りを減らし，推定精度を改善できると考えられる．そこで本研究では，PRET と EEGL の誤り傾向の違いに基づき，それらの構成要素を組み合わせることで，推定精度の改善を行う派生関係推定手法 SPG を提案する．2 種類のデータセットに対して SPG と既存手法を比較評価する実験を行った結果，提案手法の推定精度が，既存手法と比べて有意に高いことが確認できた．

キーワード　ソフトウェア進化，圧縮アルゴリズム，類似ファイル

1. 序　　論

新しいソフトウェアプロダクトは，既存のプロダクトをコピーし，そこに変更を加えることで開発されることがある．これは，既存のリソースを再利用することで，類似するプロダクトを効率的に開発すること目的としている．

このようにして開発されたプロダクトは，どのプロダクトを基に開発されたのか (派生関係) を辿っていくことで，保守を効率的に行うことができる。例えば，派生関係にあるプロダクト群は，互いに多くの実装を共有しているため，複数のプロダクトに同じ修正を適用しなければならない場合がある．このとき開発者は，プロダクトの派生関係を辿ることで，修正対象のプロダクトを効率的に把握することができる．

しかし，開発の過程で派生関係の記録が失われてしまい，開発者は正確に派生関係を辿ることができず，効率的な保守が困難となるという問題がある．多くのソフトウェアプロジェクトでは，プロダクトの派生関係が記録されておらず，開発者の記憶に頼る形で保守が行われている [1]．しかし多くの場合に，開発者の記憶は完全ではない. [2]．そのため，失われた派生関係の記録を復元することが重要な課題となっている．

この問題に対し，既存のプロダクトからそれらの派生関係のグラフ (派生関係グラフ) を推定する手法として，プロダクト間で類似するソースファイル数に基づいて推定する PRET [3] [4] と，プロダクトの可逆圧縮後のデータ量に基づいて推定する EEGL [5] の二つが提案されている．どちらもプロダクトの集合を入力とし，図 1 のような派生関係グラフを出力する．グラフの頂点はプロダクトを表しており，辺はその派生関係を表す．

EEGL の評価実験 [5] において，PRET と EEGL は推定精度は同程度であるものの，誤りの傾向に違いがあることがわかっている．PRET は辺を逆向きに接続してしまう傾向がある．EEGL は直接的な派生関係のないプロダクト間を接続してしまう傾向があり，推定に用いる圧縮アルゴリズムによって誤りの内容が変化する．これらのことから，PRET と EEGL を，誤り傾向の違いを考慮して組み合わせることで，推定結果の誤りを減らすことができる可能性がある．

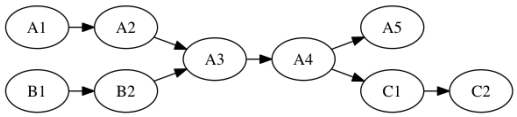


図 1　派生関係グラフの例

本論文では，PRET と EEGL の誤り傾向の違いに基づき，それらの構成要素を組み合わせることで，推定精度の改善を狙う派生関係推定手法 SPG(Stacking Product derivation Graph estimators) を提案する．本手法では，構成要素を組み合わせる枠組みとして，スタッキング [6] を使用する．これによって，出力される誤りを減らし，推定精度の改善を行う．

本稿の構成は，次のようになっている．まず 2. 章で，既存手法である PRET と EEGL の概要と，それらの誤りの傾向の違いについて述べる．3. 章では，提案手法について述べる．4. 章では，提案手法の評価実験について述べる．5. 章では，関連研究を紹介する．最後に 6. 章で，本論文の内容をまとめる．

2. 既 存 手 法

本章では，提案手法の構成要素となる既存のプロダクト派生関係推定手法の概要と，それらの違いについて述べる．既存手法である PRET と EEGL は，プロダクト間の距離と派生方向

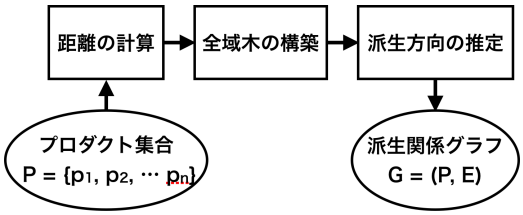


図 2　PRET の推定手順

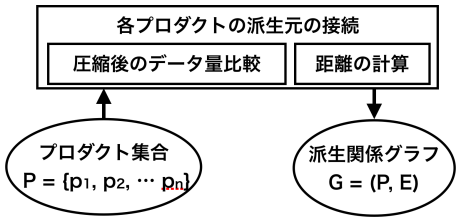


図 3　EEGL の推定手順

を求めることで派生関係グラフを推定する．どちらの手法も推定結果に誤りを含むが，その傾向は異なる．2.1 節と 2.2 節では，それぞれ PRET と EEGL について述べる．2.3 節では，両手法の誤り傾向の違いについて述べる．

2.1　PRET

神田らは，プロダクト間で類似するソースファイルの組 (類似ファイル) を基に，派生関係グラフを推定する手法 PRET を提案した [3] [4]．この手法では，プロダクト間の距離と派生方向を，類似ファイルを基にして求める．類似ファイルは，プロダクトに含まれるソースファイルを，トークン単位で比較することで発見される．

推定手順の概要を図 2 に示す．PRET の推定手順は大きく三つの段階に分かれている．まず，P に含まれるプロダクトの全組み合わせについて，プロダクト間の距離を計測する．次に，計算された距離に基づいて，プロダクトの全域木を構築する．最後に，全域木の各辺に対し，接続されているプロダクトの派生方向を推定することで，方向を決定する．これらの手順を経て派生関係グラフを構築する．

プロダクト間の距離は，類似ファイルの類似度を合計することで計算される．計算式を式 1 に示す．

$$Nw(p_i, p_j, th) = \sum_{(a,b)\in S(p_i,p_j,th)} \mathrm{sim}(a,b) \tag{1}$$

$\mathrm{sim}(a,b)$ は類似ファイル (a,b) の類似度であり，S は p_i, p_j 間に存在する類似ファイルの集合である．th は類似ファイルの発見に用いるしきい値であり，この値によって PRET の推定結果は変化する．

派生方向の推定は，プロダクトの組 (p_i, p_j) について，それぞれを派生元としたときの類似ファイルのトークン増加量 $\mathrm{ADD}(p_i, p_j)$，$\mathrm{ADD}(p_j, p_i)$ を比較することで行われる．トークン増加量の比較に基づく，PRET の派生方向推定の規則を式 2 に示す．

$$\begin{cases} p_i \to p_j & \mathrm{ADD}(p_i,p_j) > \mathrm{ADD}(p_j,p_i) \\ p_i - p_j & \mathrm{ADD}(p_i,p_j) = \mathrm{ADD}(p_j,p_i) \\ p_i \leftarrow p_j & \mathrm{ADD}(p_i,p_j) < \mathrm{ADD}(p_j,p_i) \end{cases} \tag{2}$$

$p_i - p_j$ は，派生方向が推定できなかったことを表している．

2.2　EEGL

我々の研究グループは，プロダクト間のコルモゴロフ複雑性 [7] の考え方に基づく情報量の差分を求めることで，派生関係グラフを推定する手法 EEGL を提案した [5]．コルモゴロフ複雑性とは，ある文字列を出力する最短のプログラムの長さを，その文字列の複雑さ (情報量) だとするものである．コルモゴロフ複雑性は計算不可能であることが知られており，一般に文字列の可逆圧縮後のデータ量によって近似される．この手法では，プロダクト間の距離と派生方向を，可逆圧縮後のプロダクトのデータ量を基にして求める．

推定手順を図 3 に示す．EEGL の推定手順は PRET とは異なり，一つの段階で完結する．具体的には，P に含まれる全ての p_i の派生元のプロダクト (派生元プロダクト) を発見・接続することで，派生関係グラフを構築する．p_i の派生元プロダクトには，可逆圧縮後のデータ量 $C(p)$ が $C(p_i)$ より小さいプロダクトのうち，最も距離の近いプロダクトが選択される．

プロダクト間の距離は，プロダクトの派生時に増加した情報量を測ることで計算される．派生元プロダクトを p_i，派生先のプロダクト (派生先プロダクト) を p_j としたときの増加情報量 $I(p_i, p_j)$ は式 3 で計算される．

$$I(p_i, p_j) = C(p_i, p_j) - C(p_j) \tag{3}$$

$C(p_i, p_j)$ は p_i と p_j をまとめて可逆圧縮したときのデータ量である．

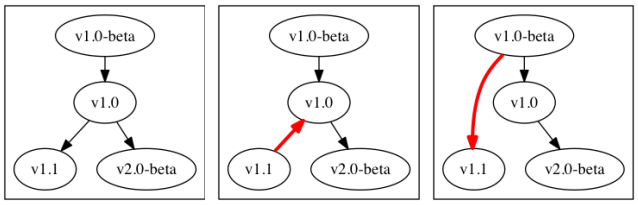
EEGL は明確な派生方向推定の規則を持たないが，圧縮後のデータ量を比較することで派生方向を推定している．そのため，EEGL の派生方向推定の規則は，式 4 のように解釈できる．

$$\begin{cases} p_i \to p_j & C(p_j) > C(p_i) \\ p_i - p_j & C(p_j) = C(p_i) \\ p_i \leftarrow p_j & C(p_j) < C(p_i) \end{cases} \tag{4}$$

また，EEGL は圧縮アルゴリズムの差し替えが可能であり，用いる圧縮アルゴリズムによって推定結果が変化する．これは，圧縮アルゴリズムによってコルモゴロフ複雑性の近似度が異なるためである [5] [8]．

2.3　誤り傾向の違い

過去に行われた EEGL と PRET を評価する実験 [5] によって，二つの手法は全体的に近い精度を持つが，誤りの傾向の違いがあることがわかっている．PRET は，辺を本来の派生関係とは逆向きに接続してしまう (**方向誤り**) 傾向があり，出力される誤り数は EEGL の約 1〜3 倍であった．EEGL は，辺を直接の派生関係にあるプロダクトではなくより上流のプロダクトと接続してしまう (**スキップ誤り**) 傾向があり，出力される誤り数は PRET の約 1〜22 倍であった．また，スキップ誤りの量は圧縮アルゴリズムによって異なっていた．グラフの同一部分における誤りの例を図 4 に示す．図中の赤い辺は，推定結果の誤りを示している．図 4(b) は方向誤りの例であり，v1.0 から v1.1 に向けて接続されるはずの辺が逆向きに接続されている．



(a) 本来の派生関係グラフ　(b) 方向誤りの例　(c) スキップ誤りの例

図 4　推定結果の違いの例

図 4(c) のスキップ誤りの例であり，v1.1 の派生元には本来の派生元 (v1.0) ではなく，より上流の 1.0-beta が接続されている．

3. 提 案 手 法

本論文では，PRET と EEGL の構成要素を組み合わせることで，派生関係グラフを推定する手法 SPG を提案する．PRET と EEGL は，2.3 節で述べたように，誤りの傾向が異なる．そのため，両手法の構成要素を組み合わせることで，誤りを減らし，推定精度を改善することができると考えられる．

基本的なアイデアは，PRET と EEGL(本手法ではこの二つを**弱推定器**と呼ぶ) の構成要素を 2.3 節に述べた誤りの違いを考慮してスタッキング [6] することで，推定結果の誤りを減らすというものである．スタッキングとは，複数の予測アルゴリズムの組み合わせ方を学習することで，より高精度な予測アルゴリズムを構築するという手法である．スタッキングする構成要素とは，プロダクトの距離計算と派生方向推定の仕組みのことである．それらの出力した値を，重みを付けて別々に合成することで，プロダクトの距離と派生方向を求める．距離の合成によってスキップ誤りが，派生方向の合成によって方向誤りが減ることが期待される．各構成要素のスタッキング方法の詳細は，3.1 節と 3.3 節で述べる．重みの最適化方法については，その詳細を 3.4 節で述べる．

本手法では PRET を基本とした推定手順を採用する．この理由は 2 点ある．1 点目は，推定手順を既存手法と同じにすることで，スタッキングによる推定精度の改善度を純粋に比較できる点である．2 点目は，プロダクトの距離計算と派生方向推定の手順が明確に分かれており，提案手法のアイデアを実現するのに最適である点である．推定手順の概要を図 5 に示す．基本的な手順や処理方法は PRET と同じであり，距離計算と派生方向推定をスタッキングによって行う点が異なっている．

3.1 手順 1: 距離の計算

距離の値は連続値であることから，弱推定器の計測する距離を重み付きで足し合わせることで推定結果のスキップ誤りが少なくなるようにプロダクト間の距離を計算する．弱推定器に与えられる重みは，スキップ誤り数が少ない弱推定器ほど大きいくなると期待される．つまり，PRET は重みが大きくなり，EEGL は重みが小さくなることが期待される．

計算手順の概要を図 6 に示す．まず，弱推定器を用いてプロダクト間の距離を計測し，その値を正規化する．正規化後の値は (i) 値域が [0,1] である (ii) プロダクトが類似しているほど小さくなる という二つの性質を満たす．PRET の計測するプロダクト間の類似度 Nw は，式 5 を用いて正規化する．

$$1 - \frac{Nw(p_i, p_j, th)}{\max\limits_{q \in P, r \in P, q \neq r} Nw(q, r, th)} \tag{5}$$

EEGL の計測するプロダクト間の増加情報量 I は，式 6 で正規化する．

$$\frac{I(p_i, p_j)}{\max\limits_{q \in P, r \in P, q \neq r} I(q, r)} \quad (C(p_i) < C(p_j), C(q) < C(r)) \tag{6}$$

次に正規化して得られた値を重みを付けて足し合わせることで，プロダクト間の距離を計算する．計算式を式 7 に示す．

$$\mathrm{dis}(p_i, p_j) = \sum_k w_k \mathrm{dis}_k(p_i, p_j) \tag{7}$$

w_k は k 番目の弱推定器 (弱推定器 k) に与えられる重みである．$\mathrm{dis}_k(p_i, p_j)$ は弱推定器 k によって計測されるプロダクト間の距離であり，式 5，6 で計算される．

3.2 手順 2: 全域木の構築

プロダクト間の距離を計算した後，プロダクトの全域木を構築する．具体的には，まず，任意のプロダクトを選択し，頂点が一つの木を構築する．次に，木に含まれないプロダクトのうち，木の頂点からの最も近いプロダクトを選択し，木に接続する．この処理を，全ての頂点が木に含まれるまで繰り返すことで全域木を構築する．

3.3 手順 3: 派生方向の推定

派生方向の推定結果は離散値であることから，弱推定器の推定した方向の重み付き多数決を取ることで推定結果の方向誤りが少なくなるように派生方向を推定する．弱推定器に与えられる重みは，方向誤り数が少ない弱推定器ほど大きいくなると期待される．つまり，PRET は重みが小さくなり，EEGL は重みが大きくなることが期待される．

派生方向の推定手順の概要を図 7 に示す．まず，弱推定器によるプロダクト間の派生方向推定を行う．PRET の派生方向推定は式 2 に，EEGL は式 4 に従って行う．次に，各弱推定器の推定結果を用いた重み付き多数決によって派生方向を決定する．重み付き多数決による各派生方向の評価値を式 8 で計算する．

$$\mathrm{vote}_{\mathrm{dir}} = \sum_{k:\mathrm{dir}_k(p_i, p_j) == \mathrm{dir}} v_k \tag{8}$$

v_k は弱推定器 k に与えられる重みである．$\mathrm{dir}_k(p_i, p_j)$ は弱推定器 k の推定した派生方向であり，式 2，4 に従って決定される．各方向の評価をもとに，式 9 に示す規則に従って派生方向を決定する．

$$\begin{cases} p_i \to p_j & (\mathrm{vote}_{p_i \leftarrow p_j} < \mathrm{vote}_{p_i \to p_j}) \\ p_i - p_j & (\mathrm{vote}_{p_i \leftarrow p_j} = \mathrm{vote}_{p_i \to p_j}) \\ p_i \leftarrow p_j & (\mathrm{vote}_{p_i \leftarrow p_j} > \mathrm{vote}_{p_i \to p_j}) \end{cases} \tag{9}$$

3.4 重みの最適化

この節では各弱推定器に与える重みの組 $W = (w_1, w_2 ...)$ と $V = (v_1, v_2 ...)$ を最適化する手順について述べる．W と V の最適化は，w_k と v_k の組合せ最適化問題だと考えることができ

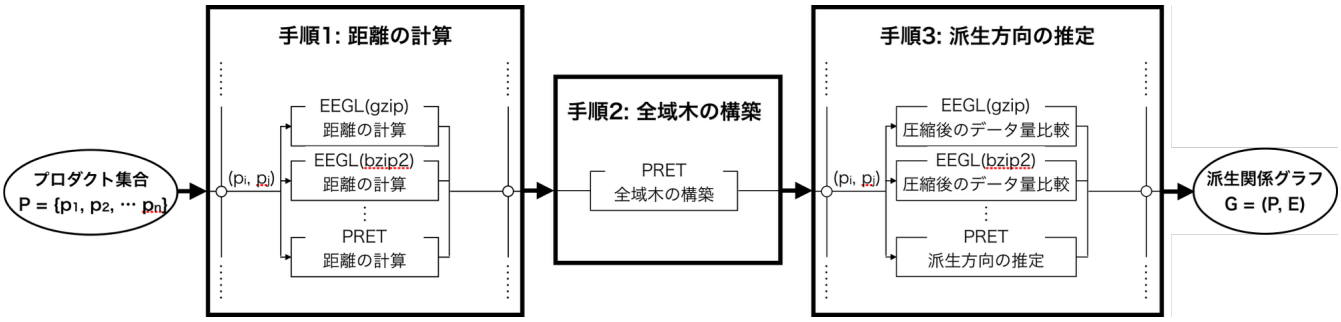


図 5　SPG の推定手順

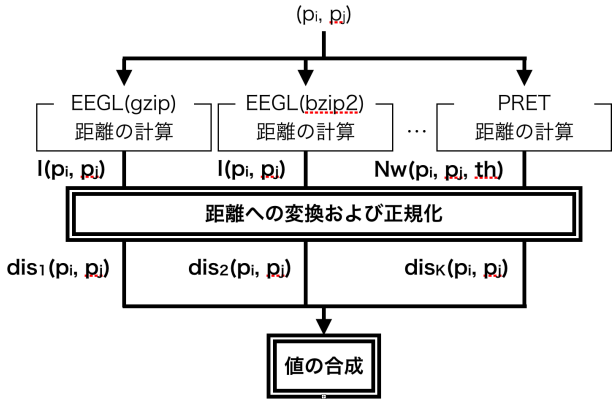


図 6　手順 1: 距離の計算手順の概要

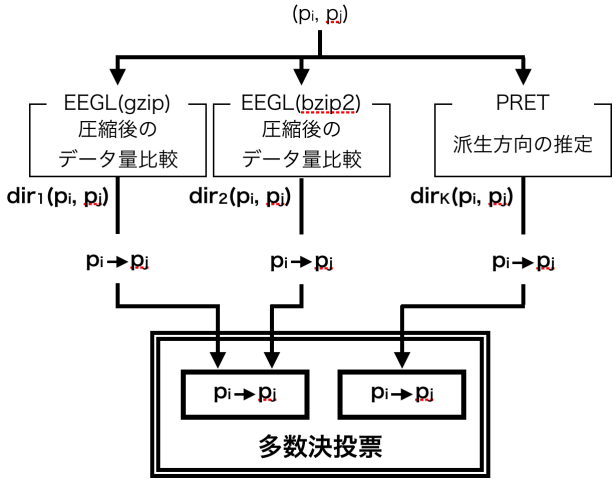


図 7　手順 3: 派生方向の推定手順の概要

る．w_k と v_k は値域に制限がないため，本手法ではメタヒューリスティクスを用いて最適化を行う．

以降の説明で必要となるため，プロダクト集合の集合 (データセット)$D = \{P_1, P_2 ... P_i\}$ を定義する．P_i はそれぞれ，本来の派生関係グラフ $G_i = (P_i, E_i)$ と，G_i の無向グラフ $Gn_i = (P_i, En_i)$ を持つとする．

まず，W と V のメタヒューリスティクスに与える初期値を決定する．w_k と v_k は値域に制限がないため，一定の範囲を総当り的に探索することで初期値を決定する．具体的には各値を $[-1, 1]$ の範囲で 0.1 刻みに変化させ，評価が最大となる重みの組み合わせを選択する．W の評価は，各 P_i を入力として手順 1〜2 を実行し，得られた全域木 $Gn_{(i,W)} = (P_i, En_{(i,W)})$ と Gn_i の類似度の総和をとることで行う．W の評価式を式 10 に示す．

$$\sum_i \frac{|En_i \cap En_{(i,W)}|}{|En_{(i,W)}|} \tag{10}$$

V の評価は，各 En_i に対し 3.3 節に述べた方法を用いて各辺の方向を推定し，得られた有向辺集合 $E_{(i,V)}$ と E_i の類似度の総和をとることで行う．V の評価式を式 11 に示す．

$$\sum_i \frac{|E_i \cap E_{(i,V)}|}{|E_i|} \tag{11}$$

次に，求まった初期値を用いて，Nelder-Mead 法 [9] による最適化を行う．この手法に与える初期値には，W と V の相互作用を考慮し，二つの初期値をまとめた $WV = (w_1, w_2 ..., v_1, v_2 ...)$ を使用する．目的関数は，各 P_i を入力としたときの推定結果の適合率の総和をとし，これを最大化する．適合率とは，推定されたグラフの辺のうち，本来の派生関係グラフ含まれた辺の割合のことである．P_i に対する推定結果のグラフを $G_{(i,WV)} = (P_i, E_{(i,WV)})$ としたときの適合率は，式 12 によって計算される．

$$\frac{|E_i \cap E_{(i,WV)}|}{|E_{(i,WV)}|} \tag{12}$$

3.5　弱推定器の推定結果に応じた重みの動的調整

本節では，弱推定器の推定結果に応じて，重み v_k を動的に調整することで，SGP の推定精度をより高める推定手法 SPG-f を提案する．v_k は弱推定器 k の推定した方向が誤りである場合，より小さい値となることが望ましい．推定された方向によって，派生関係を表す辺は異なる内容を持つことが考えられる．その内容を考慮することで，v_k の値を動的に調整し，推定精度の改善を行う．

本手法では，v_k の動的調整に，推定された辺の持つ性質を利用する．辺の性質とは，辺の表す派生関係において，プロダクトの派生時に加わる変更のことである．ある有向辺 $p_i \rightarrow p_j$ の持つ M 個の性質を以下のように定義する．

$$F(p_i, p_j) = \{f_1(p_i, p_j), f_2(p_i, p_j), .. f_M(p_i, p_j)\}$$

f はそれぞれ，異なる尺度で変更の量を測った時の値である．提案手法ではこの尺度を，変更の単位と種類の組み合わせによって定義する．例えば,「行の削除量」という変更の場合，単位は「行」であり，種類は「削除量」である．提案手法で用いる，プロダクトの変更の単位と種類を表 1 に示す．ディレク

表 1　プロダクトの変更の単位と種類

(a) 変更の単位

単位	
ディレクトリ	LOC
ファイル	拡張子数
行	言語数
サイズ	

(b) 変更の種類

種類	
増加量	増加率
追加量	追加率
削除量	削除率
追加・削除量	追加・削除率
共有量	共有率

トリとファイルは，それぞれプロダクトに含まれるディレクトリとファイルの総数である．行とサイズは，それぞれプロダクトに含まれる全てのファイルの行数とサイズの合計である．LOC(LinesOfCode) は，プロダクトに含まれる全てのファイルの LOC の合計であり，CLOC [10] を用いて計測する．拡張子数はプロダクトに含まれるファイルの拡張子の種類の数である．言語数とは，プロダクトに含まれる各ファイルの記述プログラミング言語の種類の総数のことであり，CLOC を用いて推定する．変更の種類の計算方法は，派生元プロダクトと派生先プロダクトにおけるある単位 (例:ディレクトリ) の集合を N_B, N_D とすると，増加量は $|N_D| - |N_B|$，追加量は $|N_D \setminus N_B|$，削除量は $|N_B \setminus N_D|$，追加・削除量は $|N_D \triangle N_B|$(△ は 2 つの集合の対象差を表す)，共有量は $|N_D \cap N_B|$ で表される．増加率，追加率，削除率，変更率，共有率はそれぞれ対応する量を $|N_B|$ で割った値である．

v_k を調整は，弱推定器 k の推定した方向が正しい確率 (正解確率) を求めることで行う．推定された方向が $p_i \to p_j$ であった場合に与える重みは式 13 で計算される．

$$v_k(p_i, p_j) = u_k b_k(p_i, p_j) \tag{13}$$

u_k は k 番目の弱推定器に固定して与えられる重みである．3.4 に述べた重み最適化を行う際には v_k の代わりに u_k を最適化する．$b_k(p_i, p_j)$ は弱推定器 k の正解確率である． $b_k(p_i, p_j)$ の値は，ロジスティック回帰モデルを用いて，$F(p_i, p_j)$ の値から回帰することで求める．回帰モデルの学習データには，3.4 で使用した D の各 E_i について，弱推定器 k による辺の方向の決定を行い，その結果が正いかどうかを F にラベル付けしたデータを使用する．

4. 評 価 実 験

提案手法の有効性を評価するため，2 種類のデータセットを使用して評価実験を実施し，その結果を考察する．有効性の評価は，提案手法を用いて派生関係グラフを推定し，その精度を既存手法と比較すること行う．

4.1 実験の目的と方法

提案手法と既存手法の推定結果を比較し，提案手法の有効性を評価する．まず，SPG と既存手法の推定結果を比較し，スタッキングによる推定精度の改善度を評価する．このとき，3.4 節に述べた重み最適化の効果を評価するため，重み最適化なしで SPG を用いた場合との比較も行う．次に，SPG と SPG-f の推定結果を比較し，辺の性質をスタッキングの学習に組み込むことによる推定精度の改善度を評価する．SPG-f の推定精度は，回帰モデルの汎化性能が高いほど改善すると考えられる．そこで，回帰モデルの AIC [11] を最小化し，汎化性能の高い回帰モデルを用いた場合の SPG-f との比較も行う．

表 2　PRET の評価実験で使用されたプロダクト集合

No	プロダクト数	分岐数	併合数	言語
1	14	0	0	C
2	144	5	0	C
3	38	0	0	C
4	25	0	0	C
5	16	1	0	C
6	16	5	2	C
7	37	5	0	Java
8	62	3	0	Java
9	16	0	0	Java

提案手法の推定精度は，leave-one-out 交差検定によって計測する．leave-one-out 交差検定とは，複数のデータの中から一つをテストデータとして取り出し，残りを訓練データとする．これを全データが 1 回ずつテストデータとなるように繰り返すというものである．本実験では，訓練データを重みの最適化や回帰モデルの学習に使用し，その結果を用いてテストデータの派生関係グラフを推定する．

提案手法で組み合わせる弱推定器には，新しい PRET と，五つの圧縮アルゴリズムをそれぞれ用いる EEGL を使用する．PRET に与えるしきい値 th は 0.9 とする．EEGL で用いる圧縮アルゴリズムは gzip(deflate(LZ77 [12] + Huffman coding [13]))，bzip2(ブロックソート法 [14] + Huffman coding)，xz(LZMA(LZ77 の変種 + Range Encoder [15]))，PPMd [16]，RePair [17] の五つとする．

本研究では，2 種類のデータセットを使用して評価実験を実施した．まず，PRET の評価実験 [4] と同じデータセットを使用し，PRET と同条件での評価を行った．データセットに含まれるプロダクト集合の一覧を表 2 に示す．このうち，3 と 4 のプロダクト集合は 2 のサブセットである．”分岐数”とはプロダクト集合の持つ派生関係グラフの頂点うち，出次数が 2 以上の頂点の (出自数-1) の総和である．”併合数”とはプロダクト集合の持つ派生関係グラフの頂点うち，入次数が 2 以上の頂点の (入次数-1) の総和である．表 2 のプロダクト集合は，それぞれ一部のソースファイルのみが含まれるような加工が施されている．次に，より一般性の高い評価結果得るため，データセットを独自に作成して評価を行った．作成したデータセットは，表 2 よりも大規模なものとし，人工的な加工が少なくなるようにした．作成手順を以下に記す．

（1） 以下 (a)〜(d) の手順で，開発言語が C 言語と Java のリポジトリを 10 個づつ取得する
  (a) Github の star 数上位のリポジトリを取得する．
  (b) 各リポジトリから参照コミットの重複するタグを取り除く．このとき作成日が新しいタグから取り除く．
  (c) タグ数が 14 未満のリポジトリを取り除く．
  (d) (c) 終了時点で，star 数上位のリポジトリを選択する．
（2） リポジトリからデータセットに使用するタグを選択する．選択数の上限は 60 とする．61 個以上のタグが含まれている場合，参照するコミットの時系列順にタグを並べ，一定間隔で 30〜60 個を選択する．

表 3　作成したプロダクト集合

No	リポジトリ	タグ数	分岐数	併合数	言語
1	redis	34 (176)	6 (15)	0 (5)	C
2	git	44 (590)	1 (254)	2 (255)	C
3	emscripten	40 (216)	0 (2)	0 (0)	C
4	php-src	45 (726)	34 (232)	17 (113)	C
5	the_silver_searcher	40 (40)	0 (0)	0 (0)	C
6	wrk	18 (18)	0 (0)	0 (0)	C
7	libgit2	36 (36)	2 (2)	0 (0)	C
8	h2o	33 (35)	2 (2)	0 (0)	C
9	macvim	53 (54)	0 (0)	0 (0)	C
10	firefox-ios	18 (19)	0 (0)	0 (0)	C
11	elasticsearch	60 (140)	15 (19)	0 (0)	Java
12	Android-Universal-Image-Loader	15 (15)	0 (0)	0 (0)	Java
13	RxJava	50 (135)	1 (1)	0 (0)	Java
14	retrofit	32 (33)	1 (1)	0 (0)	Java
15	picasso	20 (20)	0 (0)	0 (0)	Java
16	okhttp	26 (29)	3 (4)	0 (0)	Java
17	libgdx	34 (35)	0 (0)	0 (0)	Java
18	spring-framework	36 (83)	6 (6)	0 (0)	Java
19	zxing	19 (19)	0 (0)	0 (0)	Java
20	ActionBarSherlock	32 (32)	0 (0)	0 (0)	Java

（3） 選択されたタグのスナップショットをプロダクトとする．スナップショットからは，開発言語のソースファイル以外を削除する．C であれば，拡張子が.c，.h のファイル以外を削除する．Java であれば，拡張子が.java のファイル以外を削除する．

取得するリポジトリの情報は 2015/12/24 時点のものを利用した．リポジトリの開発言語の定義は，Github の API に従った．作成したプロダクト集合の一覧を表 3 に示す．セルの括弧内の値は，タグ選択が行われる前の値を表している．プロダクト集合のうち，括弧内外の値が等しくない項目があるものはタグの選択が行われている．C のデータセットを作成する際，上位のリポジトリであった torvalds/linux と WhisperSystems/Signal-Android は例外的に取り除いた．torvalds/linux は，一つのプロダクトのサイズが 200MB〜600MB と大きく，推定時間がかかりすぎてしまうため取り除いた．WhisperSystems/Signal-Android は，開発の過程で主要言語が Java から C 言語へと切り替わっており，ほとんどのプロダクトに C 言語のソースファイルが含まれていないため取り除いた．また，このデータセットは Web 上で公開されており [18]，他の研究者による利用が可能となっている．

4.2　データセット 1: PRET の評価実験時のデータセット

表 2 のデータセットを用いて，評価実験を行った．3 と 4 のプロダクト集合は 2 のサブセットであることから，1,2,5〜9 のみを用いて leave-one-out 交差検定を行う．その後，2 をテストデータとしたときに最適化された重みを用いて，3,4 に対する提案手法の精度を計測する．

推定結果の精度を図 8，その平均を表 4 に示す．図 8 は，推定手法ごとの適合率 (式 12) の分布を表した散布図である．横軸はプロダクト集合の番号を表しており，縦軸は適合率を表している．表 4 の横軸は推定手法を表している．各行について，最も良かった値が太字で，すべての既存手法を上回った提案手法の値が下線で強調されている．

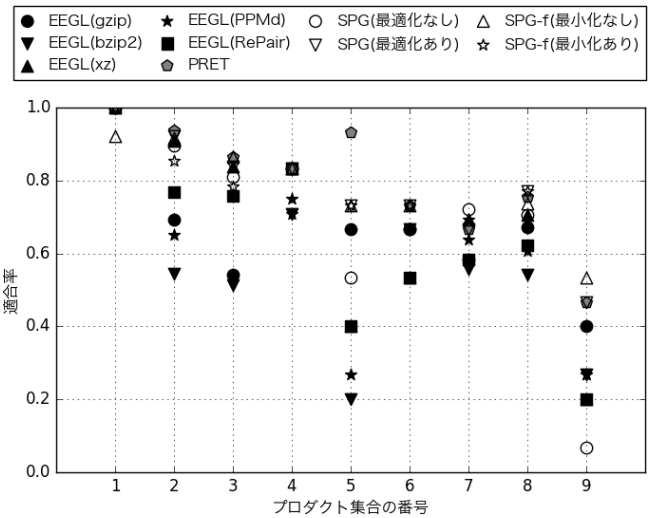


図 8　データセット 1: 推定精度の分布

4.2.1　SPG と既存手法の比較

図 8 より，7 と 8 のプロダクト集合に対して SPG が既存手法の推定精度を上回っていることがわかる．表 4 より，SPG の精度は PRET よりも平均的に低いことがわかる．しかし，その差は 0.025 と小さく，大きく劣っているわけではない．SPG (最適化なし) と SPG (最適化あり) を比較すると，重みを最適化した場合に最適化なしの精度を平均的に上回っていることがわかる．このことから，重みの最適化を行うことで推定精度を向上することができたと考えられる．これらの結果から，SPG で用いる重みの最適化をより進めることで，より良い精度を期待できるといえる．

4.2.2　SPG と SPG-f の比較

表 4 より，SPG (最適化なし) と SPG-f (最小化あり) は精度の平均値が等しいことがわかる．図 8 より，個々のプロダクト集合についてみると，精度を向上できているものもあるが，全体として精度を向上できているとはいえない．また，SPG-f (最小化なし) と SPG-f (最小化あり) を比較すると，AIC を最小化した場合に平均的な精度が低くなっていることがわかる．このことから，回帰モデルの汎化性能と SPG-f の精度の関係性は薄いと考えられる．

4.3　データセット 2: Github 上から作成したデータ集合

表 3 のデータセットを用いて，評価実験を行った．このデータセットでは，プロダクト集合が互いに独立していることから，そのすべてを leave-one-out 評価検定に使用する．

推定結果の精度を図 9 に示す．図の形式は図 8 と同じである．また，適合率と誤り数の平均を表 5 に示す．横軸はプロダクト集合の番号を表している．

4.3.1　SPG と既存手法の比較

図 9 より，多くのデータセットで SPG の既存手法の精度を上回っていることがわかる．図 4 の結果と比較すると，SPG と EEGL が図の上方に分布しており，PRET は中央や下方辺りに分布することが多くなっている．表 5 より，SPG が精度の平均的で既存手法を上回っていることがわかる．既存手法の値を表 4 の結果と比較すると，EEGL の値が全体として高くなっていることがわかる．特に EEGL (PPMd) はその差が 0.078 と

表 4　データセット 1: 推定精度の平均

	EEGL					PRET	SPG		SPG-f	
	gzip	bzip2	xz	PPMd	RePair		最適化なし	最適化あり	最小化なし	最小化あり
平均	0.673	0.555	0.679	0.598	0.633	**0.799**	0.693	0.774	0.774	0.727

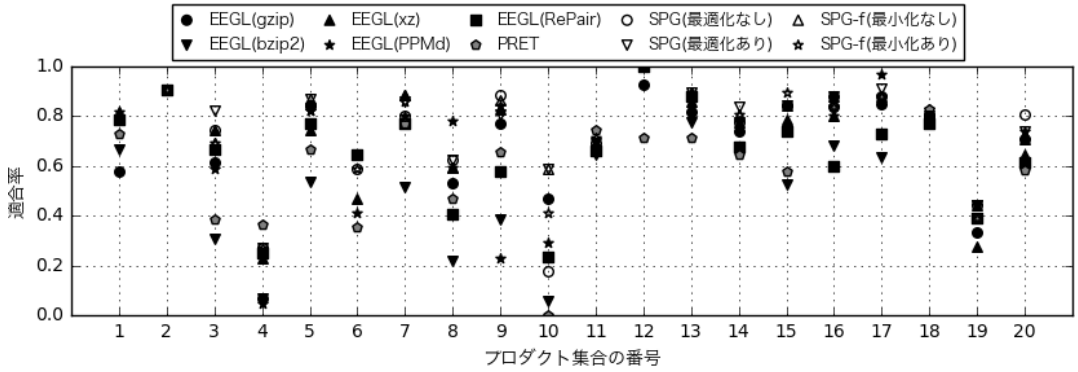


図 9　データセット 2: 推定精度の分布

表 5　データセット 2: 推定精度と誤り数の平均

	EEGL					PRET	SPG		SPG-f	
	gzip	bzip2	xz	PPMd	RePair		最適化なし	最適化あり	最小化なし	最小化あり
適合率	0.688	0.559	0.692	0.676	0.654	0.611	0.730	**0.762**	0.752	0.735
スキップ誤り数	4.750	8.000	3.850	6.400	5.550	3.950	2.950	**2.650**	2.700	2.700
方向誤り数	1.650	2.950	1.900	**1.250**	1.750	4.000	2.000	1.650	1.900	2.250
その他の誤り数	3.950	3.800	3.850	**3.300**	3.950	4.000	3.550	3.550	3.550	3.550

大きく向上している．逆に PRET は差が −1.88 と大きく低下している．また，SPG の出力する誤り数は既存手法よりも平均的に少ないことがわかる．スキップ誤りについてみると，SPG の誤り数はどの既存手法よりも平均的に少ない．方向誤りについては，SPG の誤り数は EEGL (bzip2) と PRET よりも平均的に少ない．その他の誤りについても，EEGL (xz) を除き，既存手法よりも平均的な誤り数は少ない．

4.3.2　SPG と SPG-f の比較

表 5 より，SPG が SPG-f の平均値を上回っていることがわかる．SPG-f と SPG-f (AIC 最小化) については，4.2.2 節の結果と同様，AIC を最小化することで，平均的な精度が低下することとなった．これらの結果から，辺の性質を用いて推定精度を改善することができなかったといえる．

4.3.3　実験結果の検定

本実験の結果について，SPG と既存手法の有意差を検定した．検定手法はウィルコクソンの符号順位検定とした．検定結果を表 6 に示す．1 列目の括弧内の記述は，帰無仮説の条件を表している．各セルの値は検定結果の p 値であり，有意水準 5%で有意だった値を下線で，有意水準 1%で有意だった値を太字で強調している．1 行目より，適合率について，すべての既存手法に対して 1%有意であることがわかる．このことから，提案手法の推定精度は既存手法よりも有意に高いといえる．2 行目より，スキップ誤り数について，EEGL に対して 1%有意であることがわかる．このことから，提案手法のスキップ誤り数は PRET よりも有意に少ないといえる．3 行目より，方向誤り数について，PRET に対して 1%有意であることがわかる．このことから，提案手法の方向誤り数は PRET よりも有意に少ないといえる．これらの結果から，提案手法では，各既存手法の出力傾向の強い誤りを相互に減らすことで，推定精度を高めることができたのだと考えられる．これは，提案手法の基となったアイデアと同じであり，提案手法の設計がうまく機能したのだといえる．

4.4　考　　察

方向誤り数とその他の誤り数は，SPG の平均値が EEGL (PPMd) よりも多かったことから，EEGL (PPMd) に与える重みをより最適化することで，誤りをさらに減らすことができた可能性がある．しかし，図 9 からもわかるように，EEGL (PPMd) は推定精度の振れ幅が大きく，データセット全体を通して適切な重みを与えることは難しいと考えられる．

4.3 節では 4.2 節と比べて PRET の精度が大きく低下したが，これには二つの要因が考えられる．1 点目は PRET に与えるしきい値 th の大きさである．本実験では th に 0.9 を利用したが，この値が本実験で利用したデータセットに対して不適切であった可能性がある．2 点目はデータセットの作成方法である．表 2 のデータセットは一部のソースファイルのみを使用しているが，表 3 のデータセットは全てのソースファイルを使用している．そのため，プロダクトの機能とは関係のないファイル (e.g. サンプルコード) が多く含まれており，それがノイズとなって精度を低下させている可能性がある．

SGP-f の平均的な推定精度は SPG 以下であった．このことから，プロダクトの変化量を辺の性質として用いることは，推定精度の改善に効果的ではなかったと考えられる．そのため，辺の性質を異なる観点から定義することができれば，SPG-f の精度を改善できる可能性がある．また，辺の性質の定義に関わらず，確率を用いて重みを最適化すること自体が不適切であったことも考えられる．

表 6　データセット 2: 検定結果の p 値

	EEGL					PRET
	gzip	bzip2	xz	PPMd	RePair	
適合率 (SPG > 既存手法)	**0.0000**	**0.0000**	**0.0003**	**0.0091**	**0.0000**	**0.0002**
スキップ誤り数 (SPG < 既存手法)	**0.0018**	**0.0001**	**0.0040**	**0.0057**	**0.0002**	0.1365
方向誤り数 (SPG ≠ 既存手法)	1.0000	0.0179	0.7390	0.1658	0.8906	**0.0027**
その他の誤り数 (SPG ≠ 既存手法)	0.0938	0.2363	0.1855	0.9795	0.1023	0.3760

5. 関 連 研 究

本研究以外にも，プロダクト間の差異を計測することで，ソフトウェアプロジェクトを分析した研究が行われている．計測方法には様々なものがあり，情報理論やソースコードの差分を利用したものが多い．

情報理論を利用する計測方法には，コルモゴロフ複雑性を利用したものがある．Arbuckle は，コルモゴロフ複雑性に基づくオブジェクト間の距離 NCD(Normalized Compression Distance) [19] を用いて，ソフトウェアの変化を解析する方法を提案している [8]．Steven らは，コルモゴロフ複雑性を用いて，ソフトウェアの難読性を解析している [20]．

ソースコードの差分を利用する計測方法には，行やトークン単位での差分を利用したものがある．Yamamoto らは，プロダクトクローン検出の技術と最長共通部分列を用いてプロダクト間の類似度を計測するツール SMAT を開発し，プロダクトの派生関係を可視化した [21]．Tian らは，トピックモデルの一つである LDA(Latent Dirichlet Allocation) を用いて，ソースコード上から頻繁に現れるトークンを抽出することで，ソフトウェアをカテゴライズする手法 LACT を提案した [22]．

6. ま と め

本研究では PRET と EEGL の構成要素をスタッキングすることで，プロダクトの派生関係グラフを推定する手法 SPG を提案した．2 種類のデータセットに対して SPG と既存手法を比較評価する実験を行った結果，提案手法の推定精度が，既存手法と比べて有意に高いことが確認できた．加えて，提案手法の出力した誤り数は既存手法と比べて有意に少ないことが確認できた．

今後の課題としては，様々な特徴を持ったデータセットに対して派生関係の推定を行い，推定結果の違いを分析することで，データセットの特徴と推定結果の関係を明らかにすることが挙げられる．また，辺の性質を用いた推定精度の改善方法についても，引き続き検討を行っていく必要がある．例えば，辺の性質の新たな利用方法を提案することや，新たな辺の性質を提案することが挙げられる．

文　　献

[1] T. Lavoie, F. Khomh, E. Merlo, and Y. Zou. Inferring repository file structure modifications using nearest-neighbor clone detection. In *Reverse Engineering (WCRE), 2012 19th Working Conference on*, pp. 325–334, 2012.

[2] D.L. Parnas. Software aging. In *Proceedings of the 16th international conference on Software engineering*, pp. 279–287, 1994.

[3] T. Kanda, T. Ishio, and K. Inoue. Extraction of product evolution tree from source code of product variants. In *Proceedings of the 17th International Software Product Line Conference*, SPLC '13, pp. 141–150, 2013.

[4] T.Kanda, T.Ishio, and K.Inoue. Approximating the evolution history of software from source code. *IEICE Transactions on Information and Systems*, Vol. E98.D, No. 6, pp. 1185–1193, 2015.

[5] Y. Hayase, T. Kanda, and T. Ishio. Estimating product evolution graph using kolmogorov complexity. In *Proceedings of the 14th International Workshop on Principles of Software Evolution*, IWPSE 2015, pp. 66–72, 2015.

[6] D.H. Wolpert. Stacked generalization. *Neural networks*, Vol. 5, No. 2, pp. 241–259, 1992.

[7] A.N. Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376, 1963.

[8] T. Arbuckle. Studying software evolution using artefacts' shared information content. *Sci. Comput. Program.*, Vol. 76, No. 12, pp. 1078–1097, December 2011.

[9] J. A Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, Vol. 7, No. 4, pp. 308–313, 1965.

[10] CLOC. http://cloc.sourceforge.net.

[11] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pp. 199–213. 1998.

[12] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, Vol. 23, No. 3, pp. 337–343, 1977.

[13] D.A. Huffman, et al. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, Vol. 40, No. 9, pp. 1098–1101, 1952.

[14] M. Burrows and D.J. Wheeler. A block-sorting lossless data compression algorithm. 1994.

[15] G.N.N. Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, 1979.

[16] D. Shkarin. Ppm: one step to practicality. In *Data Compression Conference, 2002. Proceedings. DCC 2002*, pp. 202–211, 2002.

[17] N.J. Larsson and A. Moffat. Offline dictionary-based compression. In *Data Compression Conference, 1999. Proceedings. DCC '99*, pp. 296–305, 1999.

[18] SPG. http://www.kde.cs.tsukuba.ac.jp/materials/SPG/.

[19] R. Cilibrasi and P. Vitanyi. Clustering by compression. *Information Theory, IEEE Transactions on*, Vol. 51, No. 4, pp. 1523–1545, 2005.

[20] S.R.Kirk and S.Jenkins. Information theory-based software metrics and obfuscation. *Journal of Systems and Software*, Vol. 72, No. 2, pp. 179–186, July 2004.

[21] T. Yamamoto, M. Matsushita, T. Kamiya, and K. Inoue. Measuring similarity of large software systems based on source code correspondence. In *Product Focused Software Process Improvement*, pp. 530–544. 2005.

[22] K. Tian, M. Revelle, and D. Poshyvanyk. Using latent dirichlet allocation for automatic categorization of software. In *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pp. 163–166, 2009.