Vol.2014-SE-183 No.24
2014/3/20

車載制御ソフトウェアモデルのための差分検出アルゴリズム

荒木 円博[1,a)]　加賀 智之[2,b)]

概要：Simulink などの車載制御モデルを対象に，モデル間の差分を漏れなく検出するためのアルゴリズムを開発した．アルゴリズムは，モデルの対に対応したグラフ対を入力とし，最大共通部分グラフの近似解を貪欲法で求める．Simulink モデルを模したグラフ対（エッジ数 200–2000 の 4 分木の対）に，このアルゴリズムを適用したところ，実行時間はエッジ数に対して線形増加の傾向を示した．2000 エッジでの実行時間は，Intel Xeon W3520 2.67GHz の計算機上で 3 秒以下であった．

キーワード：Simulink モデル，最大共通部分グラフ，貪欲法，ジャカード係数

A Difference Detection Algorithm for Automotive Control Software Model

Mitsuhiro Araki[1,a)]　Tomoyuki Kaga[2,b)]

Abstract: On automotive control software model such as Simulink model, we developed model comparison algorithm to detect model differences exhaustively. The algorithm takes graph pair corresponds model pair, then greedily maximize common subgraph. As an experiment, we apply the algorithm to Simulink model like graph pair (4-ary tree pair of edges:200–2000). Elapsed time on the experiment is linearly increased against edges. On 2000 edges tree, the time was in 3 seconds with Intel Xeon W3520 2.67GHz processor.

Keywords: Simulink Model, Maximum Common Subgraph, Greedy Algorithm, Jaccard Index

1. はじめに

車載制御ソフトウェア開発において，ソフトウェアをデータフローの観点でモデル化・シミュレーションする開発スタイル「モデルベース開発」が普及しつつある．モデルは，演算子や定数などを表すブロックをつないだブロック線図として，表現されている．

ソースコードに基づく開発スタイルと同様，モデルベース開発においても，変更箇所レビューの支援などのためにモデル間の差分検出が行われている．既に Simulink モデルを対象とした差分検出ツールとして，Simulink Report Generator[1], [2], ecDIFF[3], [4], SimDiff[5], medini unite[6] が発売されている．これらの市販ツールは，検出時に着目する違いとして，ブロックに付与される識別子 [5], [6] [*1]，レイアウト [4], [6]，およびサブシステム [*2] 階層上の所属箇所 [1] を利用するため，高速である（60000 ブロックで数秒 [5]）．また，表層的な違いを考慮する際に有用である．

一方，表層的な違いを無視して，データ処理の違いに着目したい場合もある．このような場合，サブシステム階層をフラット化したブロック線図において，とり得るデータ処理経路の経路数，各経路の経路長，経路上のブロック出現順が，2 つのモデル間で一致する経路を求め，残りを差分とする検出法が考えられる．この検出法ならば，市販ツールよりも的確な差分が得られる．また経路の中には違いが含まれないので，漏れなく差分検出できることが保証

[1] 豊田中央研究所
Toyota Central R&D Labs., Inc., Aichi 480–1192, Japan
[2] トヨタ自動車
Toyota Motor Corporation, Shizuoka 410–1193, Japan
[a)] e0761@mosk.tytlabs.co.jp
[b)] tomoyuki_kaga@mail.toyota.co.jp

[*1] [5] 上には明示されていないが，問い合わせて確認した．
[*2] 通常のプログラムにおけるサブルーチンや関数に相当

できる．

このような経路検出は，ブロック線図を有向グラフに対応付ければ，2 つの有向グラフの共通部分グラフ検出によって実現できる．ブロック線図から有向グラフへの対応付けは，ブロック，ブロックの種類，ブロックの出力ポートから別のブロックの入力ポートへの結線，および出力ポートの種類と入力ポートの種類 [*3] を，それぞれノード，ノードラベル，エッジ，エッジラベルとすればよい．

差分の誤検出を減らす上では，共通部分グラフをできるだけ大きくして，差分の中に共通部分ができるだけ含まれないようにすべきである．しかし最大共通部分グラフの厳密解法は NP 完全なので [7]，適切に近似する必要がある．本稿では，エッジ数に対して線形となるように構成した近似アルゴリズムを提案する．まず 2 節で類似の問題を対象とした近似アルゴリズムに触れ，3 節で提案アルゴリズムとその計算量について述べる．そして 4 節で Simulink モデルを模したグラフでの実験結果を述べる．

2. 類似問題の近似アルゴリズム

類似問題の近似アルゴリズム 2 つについて述べる．いずれも貪欲法による多項式時間アルゴリズムである．

なお [7] において，共通部分グラフは，2 つの部分グラフ，それぞれとグラフ同型な，別のグラフとして定義されているが，本稿では説明を簡潔にするため，共通部分グラフを，グラフ同型な 2 つの部分グラフの対として定義する．

2.1 Deissenboeck らのアルゴリズム

Deissenboeck らは，Simulink モデルに対応する多重有向グラフ 1 つを入力として，クローン検出の前処理のために，極大共通連結部分グラフの近似解を列挙するアルゴリズムを提案した [8]．

このアルゴリズムでは，まず評価関数値最大のノード対 1 つだけ [*4] からなる共通連結部分グラフを作り，以下の 3 条件をすべて満たすノード対を共通連結部分グラフに追加しながら，共通連結部分グラフを広げ尽くす．

- その対は，片方の部分グラフに隣接するノード [*5] と，もう片方の部分グラフに隣接するノードからなる．
- その対を共通連結部分グラフに追加しても，部分グラフ同士はグラフ同型である．
- 既に共通連結部分グラフに追加済みのノード対 1 つに着目して，対の各ノードに隣接し，かつ上記 2 条件を満たすノード集合間のマッチングを考えた時に，その対が評価関数値を重みとした最大重みマッチングの中に含まれている．

さらに，既に求めた共通連結部分グラフに含まれないノード対について，同様に共通連結部分グラフを求めることを繰り返して，共通連結部分グラフを列挙する．

Deissenboeck らのアルゴリズムは，最大共通部分グラフの近似解検出にも利用できる．この場合には，列挙結果の和集合において，ノード対の片方ともう片方との間に全単射が成り立つように，同型性チェック条件に「対の各ノードが列挙結果に出現していないこと」を加えればよい．

ノード対を選ぶ際の評価関数は，ノード対の周辺 [*6] のノードラベルが一致している度合いであり，高いほど良い．

2.2 Rutgers らのアルゴリズム

Rutgers らは，デジタル回路のネットリストに対応する単純無向グラフ 2 つを入力として，最大共通部分グラフの近似解を検出するアルゴリズムを提案した [10]．

このアルゴリズムでは，まず回路全体の入出力ポートに対応するノードのうち，片方のグラフともう片方とでノードラベルが一致するノード対を共通部分グラフに追加する．そして，Deissenboeck らと類似の下記 3 条件を満たすノード対を追加しながら，共通部分グラフを広げ尽くして，アルゴリズムを終了する．

- その対は，片方の部分グラフに隣接するノードと，もう片方の部分グラフに隣接するノードからなる．
- その対を共通部分グラフに追加しても，部分グラフ同士はグラフ同型である．
- 上記 2 条件を満たす対の中で，評価関数値が最大

すなわち，入出力ポート相当のノードラベルについては，グラフ間で 1 対 1 対応するか，まったく対応しないことが，アルゴリズムの前提となっている．

ノード対を選ぶ際の評価関数は，対の片方のノードと，もう片方とが，1 対 1 対応する度合いの高さである．1 対 1 対応している対は，そのまま共通部分グラフに追加される．1 対 1 対応していない対の場合には，Deissenboeck らの評価関数と同様に，ノード対の周辺 [*7] のノードラベルが一致している度合いが高い対が選ばれる．この周辺ラベルの一致度は候補選択時に候補ごとに算出される．したがって，評価関数値は，1 対 1 対応度，周辺ラベル一致度，いずれも候補選択時に算出されるので，候補となるノード対が少ない場合には，実行時間の点で有利である．さらに，この候補ノード対の数を減らすのに，1 対 1 対応度の高さを優先して評価していることが寄与している．

[10] には，Rutgers らのアルゴリズムが線形時間となること，および約 40000 ノードの実問題に対して実行時間が約 10 秒 [*8] となることが示されている．

[*3] 例：任意個加減算（Sum ブロック）では加算と減算の 2 種類
[*4] ノード対複数の場合には任意の 1 つ
[*5] 正確には，部分グラフに含まれるノードに隣接し，かつそのノード自身は部分グラフに含まれないノードのこと
[*6] 試用版ソースコード [9] によると，ノード対の各ノードから距離 5 以内のノード集合
[*7] [10] の実験では距離 5 以内
[*8] Intel Pentium 4 3.20 GHz の計算機上

3. 提案アルゴリズム

提案アルゴリズムは，Simulink モデルに対応する多重有向グラフ G_{I1} と G_{I2} を入力として，グラフ間の差分検出のために，最大共通部分グラフの近似解を検出するアルゴリズムである．G_{I1}, G_{I2} ともに自己ループは含まない．

3.1 入力グラフと共通部分グラフの定義

定義 1 入力グラフ $G_I = (N_I, E_I, nl_I, el_I)$ は，ノード集合 N_I，有向エッジ集合 $E_I = \{(s_I, t_I) | s_I \in N_I, t_I \in N_I, s_I \neq t_I\}$，ノードラベル関数 $nl_I : N_I \to NL_I$ およびエッジラベル関数 $el_I : E_I \to EL_I$ からなる 4 つ組である．ここで NL_I, EL_I は，それぞれノードラベル集合およびエッジラベル集合である．

定義 2 2 つの入力グラフ $G_{I1} = (N_{I1}, E_{I1}, nl_{I1}, el_{I1})$, $G_{I2} = (N_{I2}, E_{I2}, nl_{I2}, el_{I2})$ において，$|E_{I1}| \leq |E_{I2}|$ であり，同種のラベル関数の値域は同じである．すなわち $nl_{I1} : N_{I1} \to NL_I$, $nl_{I2} : N_{I2} \to NL_I$, $el_{I1} : E_{I1} \to EL_I$, $el_{I2} : E_{I2} \to EL_I$ である．

定義 3 入力グラフのノード $n_I \in N_I$ について，流入および流出エッジ集合関数を，それぞれ $in(n_I)$, $out(n_I)$ と置く．2 つの入力グラフについても，同様に $in1(n_{I1})$, $out1(n_{I1})$, $in2(n_{I2})$, $out2(n_{I2})$ を定義する．

定義 4 入力グラフのエッジ $e_I \in E_I$ について，始点 s_I および終点 t_I を表す関数を，それぞれ $s(e_I)$, $t(e_I)$ と置く．2 つの入力グラフについても，同様に $s1(e_{I1})$, $t1(e_{I1})$, $s2(e_{I2})$, $t2(e_{I2})$ を定義する．

定義 5 G_{I1} と G_{I2} の共通部分グラフ $(G_{I1com}, G_{I2com}) = ((N_{I1com}, E_{I1com}), (N_{I2com}, E_{I2com}))$ において，ラベル付き多重グラフ間のグラフ同型性，すなわちノード間全単射 $bij_N : N_{I1com} \to N_{I2com}$，エッジ間全単射 $bij_E : E_{I1com} \to E_{I2com}$，ノードラベルに関する $\forall_{n_{I1com} \in N_{I1com}} nl_{I1}(n_{I1com}) = nl_{I2}(bij_N(n_{I1com}))$ およびエッジラベルに関する $\forall_{e_{I1com} \in E_{I1com}} el_{I1}(e_{I1com}) = el_{I2}(bij_E(e_{I1com}))$ が成り立っている．

3.2 設計方針

提案アルゴリズムは，線形時間アルゴリズムとすることを狙って，Rutgers らのアルゴリズムをもとに設計した．設計にあたって，以下の 3 つを考慮した．

(1) Rutgers らはエッジの向きやラベルを考慮していないのに対し，提案アルゴリズムではこれらを考慮する必要がある．

(2) 実際の Simulink モデルを [9] によってグラフ化して予備実験した結果，評価関数値として 1 対 1 対応度を優先するよりも，Deissenboeck らと同様に周辺ラベル一致度だけを使う方が，共通部分グラフの規模が大きくなる傾向にある．

(3) 提案アルゴリズムの入力には，入出力ポート相当のノードがない場合や，入出力ポート相当のノードラベルとして同一のものが複数出現する場合があるため，前提が Rutgers らと異なる．

これら 3 点のうち（1）については，共通部分グラフに追加する候補の探索は Rutgers らと同様に無向グラフ（**探索用グラフ**）上で，候補がグラフ同型性を満たすかどうかの判定は入力グラフ上で行うことによって，対応する．探索用グラフは 3.3 節で定義する．

（2）の予備実験結果は，周辺ラベル一致度の高い候補を選ぶことが，先読みに基づく候補選択にあたることから，妥当と考える．そこで，評価関数では，周辺ラベル一致度だけを使うことにする（3.7 節）．

（3）については，アルゴリズム開始時の共通部分グラフを，Deissenboeck らのように評価関数値最大の対からなるように構成することによって，対応する．ただし評価関数値最大の対を，探索用グラフ間でノードラベルが一致しているノード対すべてから探すと，3.10 節で述べるように線形時間アルゴリズムにはならない．そこでアルゴリズム開始時の候補をノードラベルが 1 対 1 対応しているノード対（定義 13）に限定する．アルゴリズム開始時の候補については 3.5 節で述べる．

3.3 探索用グラフの定義

探索用グラフは，入力グラフを無向グラフとみなした時の線グラフであり，共通部分グラフに追加する候補の探索や，評価関数値算出のために用いる．

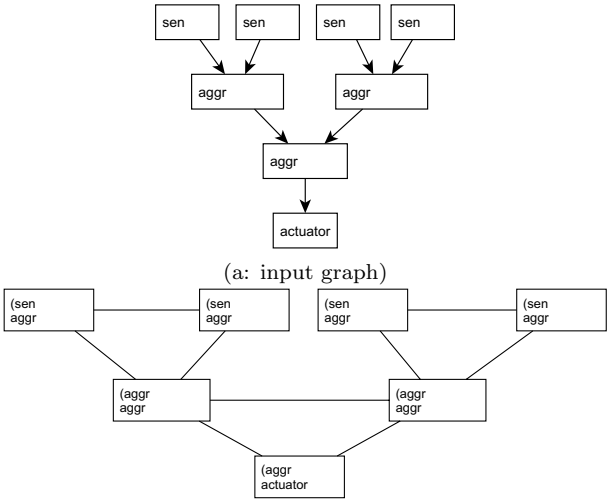
定義 6 探索用グラフ $G_S = (N_S, E_S, l_S)$ は，ノード集合 N_S，無向エッジ集合 E_S およびラベル関数 $l_S : N_S \to L_S$ からなる 3 つ組である．ここで L_S はラベル集合である．

定義 7 探索用グラフ $G_S = (N_S, E_S, l_S)$ において，$n_S \in N_S$ に隣接するノード集合関数を $adj(n_S)$ と置く．

定義 8 入力グラフと探索用グラフの間には，全単射 $bij_S : E_I \to N_S$，ラベルに関する $\forall_{e_I}(nl_I(s(e_I)), el_I(e_I), nl_I(t(e_I))) = l_S(bij_S(e_I))$，および e_I の始終点に接続しているエッジ集合 $E_{incident} = in(s(e_I)) \cup out(s(e_I)) \cup in(t(e_I)) \cup out(t(e_I))$ に関する $\forall_{e_I}(\forall_{e' \in E_{incident} \wedge e' \neq e_I} bij_S(e') \in adj(bij_S(e_I))) \wedge (\forall_{e'' \in E_I \wedge e'' \notin E_{incident} \wedge e'' \neq e_I} bij_S(e'') \notin adj(bij_S(e_I)))$ が成り立っている（図 1 参照）[*9]．

定義 9 2 つの探索用グラフ $G_{S1} = (N_{S1}, E_{S1}, l_{S1})$, $G_{S2} = (N_{S2}, E_{S2}, l_{S2})$ についても定義 7 と同様な隣接ノード集合関数 $adj1$, $adj2$ があり，定義 8 と同様に入力グラフとの全単射 $bij_{S1} : E_{I1} \to N_{S1}$, $bij_{S2} : E_{I2} \to N_{S2}$ などが成り立っている．

[*9] 入力グラフ上で始終点のいずれかを共有しているエッジ同士と，探索用グラフ上で隣接しているノード同士とが対応していること，および裏も真であることを意味する．



(a: input graph)

(b: graph on search)

図 1 入力グラフと探索用グラフの例
Fig. 1 An Example of Input Graph and Graph on Search

定義 10 ノード集合 $N_{S1com} = \{bij_{S1}(e_{I1com}) | e_{I1com} \in E_{I1com}\}$ および $N_{S2com} = \{bij_{S2}(e_{I2com}) | e_{I2com} \in E_{I2com}\}$ それぞれによる誘導部分グラフの対 (G_{S1com}, G_{S2com})[*10] も，本稿では共通部分グラフと呼ぶことにする．

3.4 アルゴリズム概要

提案アルゴリズムは，以下のように動作する．

(1) 共通部分グラフに隣接しない候補ノード対（3.5 節）の中で，評価関数値最大の候補ノード対を 1 つだけ共通部分グラフに追加する．評価関数値最大の候補選択，および共通部分グラフへのノード対追加については，それぞれ 3.7 節，3.8 節で述べる．
(2) 共通部分グラフに隣接する候補ノード対（3.6 節）の中で，評価関数値最大の候補ノード対すべてについて，共通部分グラフへの追加を試みる（3.8 節）．
(3) （2）での候補が得られなくなるまで，（2）を反復する．
(4) （1）での候補が得られなくなるまで，（1）—（3）を反復する．

3.5 共通部分グラフに隣接しない候補

定義 11 共通部分グラフの片方 G_{S1com} に隣接しないノード集合を $N_{S1nadj} = \{n_{S1nadj}\} = (N_{S1} \setminus N_{S1com}) \setminus \bigcup_{n \in N_{S1com}} adj1(n)$ として定義する．もう片方 G_{S2com} に隣接しないノード集合 N_{S2nadj} も同様である．

定義 12 共通部分グラフに隣接しない候補は，N_{S1nadj}，N_{S2nadj} それぞれのノードからなるノード対 (n_{S1nadj}, n_{S2nadj}) のうち，所属元の集合間でラベルが 1 対 1 対応し（定義 13），かつ $E_{I1com} \cup \{bij_{S1}{}^{-1}(n_{S1nadj})\}$ と $E_{I2com} \cup \{bij_{S2}{}^{-1}(n_{S2nadj})\}$ それぞれをエッジ集合とするグラフ同士が同型なノード対である．

定義 13 以下の条件がすべて成り立っている場合に，(n_{S1nadj}, n_{S2nadj}) のラベルが N_{S1nadj}, N_{S2nadj} 間で 1 対 1 対応していると定義する．

- $l_{S1}(n_{S1nadj}) = l_{S2}(n_{S2nadj})$
- $\forall_{n \in N_{S1nadj} \wedge n \neq n_{S1nadj}} l_{S1}(n) \neq l_{S1}(n_{S1nadj})$
- $\forall_{n \in N_{S2nadj} \wedge n \neq n_{S2nadj}} l_{S2}(n) \neq l_{S2}(n_{S2nadj})$

3.5.1 候補の求め方

ノード対 (n_{S1nadj}, n_{S2nadj}) のラベルが N_{S1nadj}, N_{S2nadj} 間で 1 対 1 対応していれば，共通部分グラフの定義（定義 5）のうち，ラベル一致条件を満たす．

次に $E_{I1com} \cup \{bij_{S1}{}^{-1}(n_{S1nadj})\}$ と $E_{I2com} \cup \{bij_{S2}{}^{-1}(n_{S2nadj})\}$ の間にエッジ間全単射が成り立つかどうかに着目する．$e_{I1nadj} = bij_{S1}{}^{-1}(n_{S1nadj})$, $e_{I2nadj} = bij_{S2}{}^{-1}(n_{S2nadj})$ と置くと，$e_{I1nadj} \notin E_{I1com} \wedge e_{I2nadj} \notin E_{I2com}$ なので，$E_{I1com} \cup \{e_{I1nadj}\}$, $E_{I2com} \cup \{e_{I2nadj}\}$ 間では，E_{I1com}, E_{I2com} 間で成り立っていた全単射がそのまま成立し，さらに $bij_E(e_{I1nadj}) = e_{I2nadj}$ が成り立つ．

また，探索用グラフ上でノード同士が隣接していないことが，入力グラフ上においてはエッジ同士の始終点いずれも共有していないことに相当するため，$bij_{S1}{}^{-1}(n_{S1nadj})$ および $bij_{S2}{}^{-1}(n_{S2nadj})$ の始終点は，N_{I1com}, N_{I2com} に含まれない．すなわちノード間全単射も成立する．

したがって，候補としては，N_{S1nadj}, N_{S2nadj} 間でラベルが 1 対 1 対応しているノード対であればよい．

この条件を満たすノード対を線形時間で求めるには，N_{S1nadj} と N_{S2nadj} を走査して，ラベルからそのラベルが対応付けられたノード部分集合へのハッシュ表，すなわち $L_S \to 2^{N_{S1nadj}}$ と $L_S \to 2^{N_{S2nadj}}$ を表すハッシュ表を作ってから，各ラベルについて N_{S1nadj}, N_{S2nadj} それぞれの部分集合の要素数を調べればよい．

3.5.2 共通部分グラフに隣接しないノード集合から候補が得られない場合

アルゴリズム開始時に N_{S1nadj}, N_{S2nadj} 間でラベルが 1 対 1 対応しているノード対がない場合には，共通部分グラフが空のままアルゴリズムが終了する．このような場合で，かつ N_{S1nadj}, N_{S2nadj} 間でラベルが一致するノード対が存在する場合には，明らかに共通部分グラフの検出漏れ（差分誤検出）が生じる．

この問題の解決と，ラベル 1 対 1 対応となる候補生成とを両立させるために，N_{S1nadj}, N_{S2nadj} 間でラベルが 1 対 1 対応しているノード対がない場合には，ノード対の所属元の集合を絞り込んでから，候補を求める．

N_{S1nadj} や N_{S2nadj} を絞り込んだ集合の対として，対応する入力グラフ側エッジ始点の入次数が 0 の $(N_{S1in0nadj}, N_{S2in0nadj})$，および入力グラフ側エッジ終点の出次数が 0 の $(N_{S1out0nadj}, N_{S2out0nadj})$ を考え，それ

[*10] Rutgers らの共通部分グラフに相当

ぞれの集合対から候補を探す．

定義 14 N_{S1nadj} に対応する入力グラフ側エッジ始点の入次数が 0 のノード集合を，$N_{S1in0nadj} = \{n_{S1in0nadj} | n_{S1in0nadj} \in N_{S1nadj} \wedge |in1(s1({bij_{S1}}^{-1}(n_{S1in0nadj})))| = 0\}$ として定義する．$N_{S2in0nadj}$, $N_{S1out0nadj}$, $N_{S2out0nadj}$ も同様である．

3.6 共通部分グラフに隣接する候補

定義 15 共通部分グラフの片方 G_{S1com} に隣接するノード集合を $N_{S1adj} = \{n_{S1adj}\} = (N_{S1} \setminus N_{S1com}) \cap \bigcup_{n \in N_{S1com}} adj1(n)$ として定義する．もう片方 G_{S2com} に隣接するノード集合 N_{S2adj} も同様である．

定義 16 共通部分グラフに隣接する候補は，N_{S1adj}, N_{S2adj} それぞれのノードからなるノード対 (n_{S1adj}, n_{S2adj}) のうち，$E_{I1com} \cup \{{bij_{S1}}^{-1}(n_{S1adj})\}$ と $E_{I2com} \cup \{{bij_{S2}}^{-1}(n_{S2adj})\}$ それぞれをエッジ集合とするグラフ同士が同型なノード対である．

3.6.1 候補の求め方

ノード対 (n_{S1adj}, n_{S2adj}) は，3.5 節と同様に，定義 5 のうちエッジ間全単射条件を満たす．さらに $l_{S1}(n_{S1adj}) = l_{S2}(n_{S2adj})$ ならば，ラベル一致条件も満たす．

一方，3.5 節の場合とは異なり，$e_{I1adj} = {bij_{S1}}^{-1}(n_{S1adj})$ の始終点 $s_{I1adj} = s1(e_{I1adj})$, $t_{I1adj} = t1(e_{I1adj})$ および $e_{I2adj} = {bij_{S2}}^{-1}(n_{S2adj})$ の始終点 $s_{I2adj} = s2(e_{I2adj})$, $t_{I2adj} = t2(e_{I2adj})$ について，必ず始点側か終点側の少なくともどちらかは共通部分グラフに含まれる（$s_{I1adj} \in N_{I1com} \wedge s_{I2adj} \in N_{I2com} \vee t_{I1adj} \in N_{I1com} \wedge t_{I2adj} \in N_{I2com}$）．すなわち常にノード間全単射条件を満たすとは限らない．

例として始点側が共通部分グラフに含まれる場合 $s_{I1adj} \in N_{I1com}$ を考える．この場合，ノード間全単射 bij_N を介して $bij_N(s_{I1adj}) = s_{I2adj}$ となっている時だけ，対を追加した後のノード間全単射が成立する．言い換えると，入力グラフ G_{I2} 側のエッジの始点を $bij_N(s_{I1adj})$ にすればよい．

したがって，$s_{I1adj} \in N_{I1com}$ の場合，以下の条件を満たす e_{I2out} があれば，$(n_{S1adj}, bij_{S2}(e_{I2out}))$ を候補とすればよい．

- $e_{I2out} \in out2(bij_N(s_{I1adj}))$
- $e_{I2out} \notin E_{I2com}$
- $l_{S1}(n_{S1adj}) = l_{S2}(bij_{S2}(e_{I2out}))$
- $(t_{I1adj} \in N_{I1com}) \wedge (t2(e_{I2out}) = bij_N(t_{I1adj})) \vee (t_{I1adj} \notin N_{I1com}) \wedge (t2(e_{I2out}) \notin N_{I2com})$

同様に，$t_{I1adj} \in N_{I1com}$ の場合，以下の条件を満たす e_{I2in} があれば，$(n_{S1adj}, bij_{S2}(e_{I2in}))$ を候補とすればよい．

- $e_{I2in} \in in2(bij_N(t_{I1adj}))$
- $e_{I2in} \notin E_{I2com}$
- $l_{S1}(n_{S1adj}) = l_{S2}(bij_{S2}(e_{I2out}))$
- $(s_{I1adj} \in N_{I1com}) \wedge (s2(e_{I2in}) = bij_N(s_{I1adj})) \vee (s_{I1adj} \notin N_{I1com}) \wedge (s2(e_{I2in}) \notin N_{I2com})$

すなわち，共通部分グラフに隣接する候補を求める際には，N_{S1adj} と N_{S2adj} の組合せを考慮する必要はなく，N_{S1adj} の走査によって求めることができる．

3.7 評価関数と，関数値最大の候補選択方法

3.7.1 評価関数の定義

定義 17 評価関数 $ev(p_c, d)$ は，探索用グラフにおける候補ノード対 $p_c = (n_{S1c}, n_{S2c})$ から距離 d のノード集合 N_{S1cd}, N_{S2cd} 間のノードラベル一致度関数（$[0, 1]$ の範囲の実数値）である．d は $[1, d_{limit}]$ の範囲の整数値をとり，d_{limit} はアルゴリズムのパラメータとして与えられる．

定義 18 ノードラベルの一致度は，N_{S1cd}, N_{S2cd} それぞれに対応するノードラベルの多重集合 [*11] M_{S1cd}, M_{S2cd} 間のジャカード係数 [*12] $|M_{S1cd} \cap M_{S2cd}|/|M_{S1cd} \cup M_{S2cd}|$ として定義する．

3.7.2 候補が 2 つの場合の候補選択方法

2 つの候補ノード対 p_i と p_j から 1 対を選ぶ際には，まず距離 1 で $ev(p_i, 1)$ と $ev(p_j, 1)$ を比較する．評価関数値に違いがあれば値が大きい方を選ぶ．値が同じならば，距離値を d_{limit} に達するまで 1 つずつ増やしながら，同様に比較・選択する．

距離値 d_{limit} において，評価関数値が同じならば，いずれも選択結果とする．

比較の際，p_i について $(M_{S1id} \cup M_{S2id}) = \emptyset$ [*13]，p_j について $(M_{S1jd} \cup M_{S2jd}) \neq \emptyset \wedge ev(p_j, d) < 1$ となる場合がある．この場合には，p_i 周辺のノードラベルが一致していることに着目すると，p_i が選ばれるべきである．そこで評価関数の定義に，以下の場合分けを追加する．

定義 19 $ev(p_c, d)$ におけるジャカード係数式の分母が 0 になる場合，すなわち $(M_{S1cd} \cup M_{S2cd}) = \emptyset$ ならば，$ev(p_c, d) \equiv 1$ とする．

3.7.3 候補が 3 つ以上の場合の候補選択方法

候補ノード対 3 つ以上の場合のノード対選択は，候補のうちの 1 つを暫定的に評価関数値最大として，残りの候補 1 つずつとの比較・選択を 3.7.2 節と同様に行う．

3.8 共通部分グラフへのノード対の追加

共通部分グラフに追加しようとしているノード対が 1 つの場合には，無条件に追加する．

一方，ノード対が複数ある場合，すべて追加しようとすると，定義 5 のグラフ同型性が不成立になり得る．例えば

[*11] 要素の重複を許容する集合．通常の集合と，要素の重複度関数との組として定義されている．
[*12] 集合間の類似度の一種
[*13] p_i のどちらのノードからも d 離れるとノードがない場合，すなわち p_i のどちらのノードも小さい連結成分に含まれている場合に生じ得る．

候補が (na, nb) と (na, nc) の 2 つの場合，両方を共通部分グラフに追加すると，E_{I1} 側の ${bij_{S1}}^{-1}(na)$ が，E_{I2} 側では ${bij_{S2}}^{-1}(nb)$ と ${bij_{S2}}^{-1}(nc)$ の 2 つに対応することになり，エッジ間全単射が成り立たなくなる．

そこでノード対複数ならば，1 つ追加してから，残りのうち以下の条件をすべて満たすものを，共通部分グラフに追加する（追加しようとしているノード対集合を $\{(n_{S1cmax}, n_{S2cmax})\}$，$e_{I1cmax} = {bij_{S1}}^{-1}(n_{S1cmax})$，$s_{I1cmax} = s1(e_{I1cmax})$，$t_{I1cmax} = t1(e_{I1cmax})$ と置く．e_{I2cmax}，s_{I2cmax}，t_{I2cmax} についても同様）．

- $(e_{I1cmax} \notin E_{I1com}) \wedge (e_{I2cmax} \notin E_{I2com})$
- $(s_{I1cmax} \in N_{I1com}) \wedge (s_{I2cmax} = bij_N(s_{I1cmax})) \vee (s_{I1cmax} \notin N_{I1com}) \wedge (s_{I2cmax} \notin N_{I2com})$
- $(t_{I1cmax} \in N_{I1com}) \wedge (t_{I2cmax} = bij_N(t_{I1cmax})) \vee (t_{I1cmax} \notin N_{I1com}) \wedge (t_{I2cmax} \notin N_{I2com})$

3.9 実装

実行時間短縮のため，ノード対の追加に伴って N_{S1nadj}，N_{S2nadj} および N_{S1adj} を更新するようにしている．また集合などをハッシュ表によって表現している．

ハッシュ表は，集合，ラベル多重集合 [*14]，ノード間全単射 bij_N およびエッジ間全単射 bij_E の表現と，以下の関数値のメモ化に利用している．

(1) 評価関数
(2) 探索用グラフにおけるノード $bij_S(e_I)$ から距離 d のノード集合に対応する，ラベル多重集合の関数
(3) 探索用グラフにおけるノード $bij_S(e_I)$ から距離 d のノード集合に対応する，入力グラフのエッジ集合関数．この関数は，$adj(bij_S(e_I))$ の関数値や，評価関数値算出に用いるノード集合 N_{S1cd}，N_{S2cd} を，実際に探索用グラフを構成することなく，入力グラフ上で求めるために利用している．

3.10 計算量

3.9 節で述べた実装での予備実験から，評価関数値の算出が，実行時間の 95%以上を占めることがわかった．

そこで計算量解析にあたって，評価関数に着目する．

3.10.1 評価関数値の算出回数

評価関数値は，候補数 $candidates = |\{(n_{S1c}, n_{S2c})\}|$ に対して，最大で $d_{limit} \times candidates$ 回算出される．したがって評価関数値の算出回数は $O(candidates)$ である．

共通部分グラフに隣接しない候補を，ラベル 1 対 1 対応の対に限定しなければ $candidates = O(|N_{S1}||N_{S2}|)$ となるので，明らかに線形時間アルゴリズムにはならない．

しかし提案アルゴリズムでは，共通部分グラフに隣接しない候補を，ラベルが 1 対 1 対応している対だけに限定しているので，$candidates = O(|N_{S1}|)$ となる．

共通部分グラフに隣接する候補の場合，共通部分グラフに含まれている対 (e_{I1com}, e_{I2com}) 1 つに対する候補数は，それぞれの始点同士を s_{I1com}，s_{I2com}，終点同士を t_{I1com}，t_{I2com} と置くと，$O(|in1(s_{I1com})||in2(s_{I2com})| + |out1(s_{I1com})||out2(s_{I2com})| + |in1(t_{I1com})||in2(t_{I2com})| + |out1(t_{I1com})||out2(t_{I2com})|)$ となる．この次数に基づく候補数を，Rutgers らの解析と同様に定数とみなすと，$candidates$ は，最大で共通部分グラフに含まれ得る対の数に比例するので，$O(|N_{S1}|)$ となる．

したがって評価関数値算出回数は $O(|N_{S1}|)$ となる．

3.10.2 評価関数の計算量

評価関数を求めるには，最大で n_{S1c} や n_{S2c} から，それぞれ距離 d_{limit} 以内に含まれるノードを調べる必要がある．

メモ化を行っていることを考慮すると，時間計算量，空間計算量，ともにこれらのノード数に比例する．

したがって G_{S1} と G_{S2} の直径が d_{limit} 以下の場合の計算量は，時間計算量，空間計算量，ともに $O(|N_{S2}|)$ となる．

一方，探索用グラフのどのノードから見ても距離 d_{limit} より遠いノードがある場合には，調べるべきノード数は，n_{S1c} や n_{S2c} を根とした深さ d_{limit} の全域木の枝と同数になる．したがって探索用グラフにおける次数最大値を deg_{Smax} と置くと，計算量は $O({deg_{Smax}}^{d_{limit}})$ となる．

3.10.3 アルゴリズム全体の計算量

探索用グラフの直径が d_{limit} 以下の場合，時間計算量，空間計算量，ともに $O(|N_{S1}||N_{S2}|) = O(|E_{I1}||E_{I2}|)$ となる．

一方，探索用グラフのどのノードから見ても距離 d_{limit} より遠いノードがある場合には，評価関数の計算量を定数とみなすと，全体としては $O(|N_{S1}|) = O(|E_{I1}|)$ となる．

したがって，後者の場合，すなわち入力グラフ全体のエッジより，評価関数値算出時に考慮するエッジが少ない場合には，提案アルゴリズムは線形時間である．

4. 実験

提案アルゴリズムの実装が，3.10 節での計算量解析結果と整合しているかどうか確認するために，実行時間および使用メモリ量を実測した．実行時間は，アルゴリズム開始時刻と終了時刻の差を測定した．

4.1 実験用のグラフ

実験用グラフは，Simulink モデルを模し，かつ，計算量ができるだけ最悪ケースに近くなるように構成した．

Simulink モデルとして，センサ入力を集約してアクチュエータに出力する例を考える．このモデルを模したグラフとして，葉がセンサ，非終端ノードが集約演算を表すような k 分木の根に，アクチュエータへのエッジを付け加えたグラフが考えられる．

実行時間が，できるだけ最悪ケースに近くなるようにす

[*14] ラベルからラベル出現回数へのハッシュ表

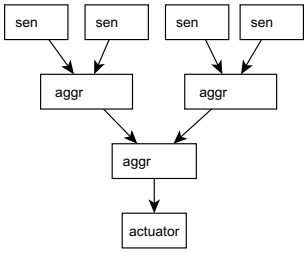


図 2　実験用グラフの例
Fig. 2　An Example of Input Graph on Experiment

るため，グラフのエッジラベル，k 分木の葉同士や非終端ノード同士のラベルを，それぞれ一致させる（図 2）．さらに，アルゴリズムへの入力を，同一のグラフ対とする．このようなグラフ対では，評価関数値を求める際，d を d_{limit} まで増やして求めることが強いられる．

実験用グラフは，エッジ数と k をパラメータとして，所望のエッジ数に達するまで，幅優先探索によって生成した．したがって k 分木の葉によっては，根からの距離が木の深さより 1 つ小さくなり得る．

k が大きくなると，計算量解析において定数とみなした値が大きくなる．共通部分グラフに追加する対の候補数は，共通部分グラフに含まれている対 1 つに対して $O(k^2)$ となる．また，探索用グラフの最大ノード次数は $2k$ なので，評価関数 1 回あたりの計算量は $O(k^{d_{limit}})$ となる．

4.2　実験条件

実験用グラフは，以下のパラメータによる 50 通りとする．

エッジ数:　200, 400, . . . 2000 (10 通り)

k:　2, 4, . . . 10 (5 通り)

実験は，1 つの実験用グラフ対につき，5 回連続して行う．

提案アルゴリズムのパラメータ d_{limit} は 8 とする．

d_{limit} は探索時の先読みの深さに相当するので，ラベルがほとんど同じ場合に正解（すなわち入力全体）を検出するには，大きくする必要がある．d_{limit} が Deissenboeck らや Rutgers らの類似パラメータと同じ 5 の場合には，$k = 2$，すなわち k 分木が深い場合に不正解となった．この不正解は，先読み不十分なために，k 分木の根から葉への深さが異なる経路が，G_{I1com} と G_{I2com} に含まれた結果，生じた．

そこで，グラフと測定回数の組合せ 250 通りすべてにおいて，共通部分グラフ検出結果が正解となるような最小値を求めたところ，8 が得られた．

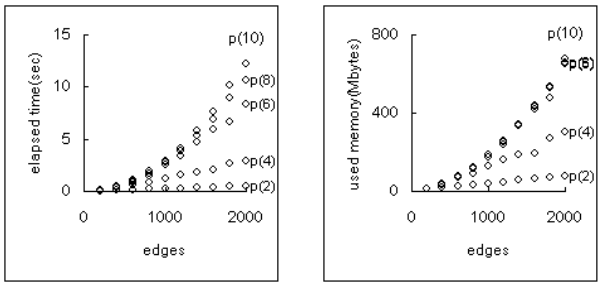
実験には，Java Standard Edition 1.7.0_45 標準の仮想計算機（ヒープ最大サイズ 1400Mbytes）および Intel Xeon W3520 2.67GHz の計算機を用いた．

4.3　実験結果

表 1 と図 3(a) に実行時間，図 3(b) に使用メモリ量を示す．実行時間，使用メモリ量，ともに 5 回の実験のうち，最小値と最大値を除いた 3 回分の平均値を示した．これらの結果から，実行時間，使用メモリ量ともに，エッジ数に対して，$k = 2, 4$ でほぼ線形，$k = 6, 8, 10$ でほぼ二乗で増加していることがわかる．

表 1　提案アルゴリズムの実行時間（秒）
Table 1　Elapsed Time on Proposed Algorithm (in seconds)

$\lvert E_I \rvert$	$k=2$	4	6	8	10
200	0.1	0.1	0.1	0.1	0.2
400	0.1	0.3	0.4	0.4	0.5
600	0.1	0.6	0.9	1.0	1.1
800	0.2	0.9	1.5	1.7	1.9
1000	0.2	1.3	2.5	2.7	3.0
1200	0.3	1.6	3.3	3.8	4.2
1400	0.4	1.8	4.7	5.3	5.9
1600	0.4	2.1	5.9	6.9	7.7
1800	0.5	2.6	6.7	9.0	10.2
2000	0.5	2.9	8.3	10.7	12.3



(a: time)　(b: memory)

図 3　提案アルゴリズムの実行時間と使用メモリ量（カッコ内：k）
Fig. 3　Elapsed Time and Used Memory on Proposed Algorithm (in parentheses: k)

実験結果は，探索用グラフの直径が d_{limit} 以下の場合の $O(|E_I|^2)$，およびどのノードから見ても距離 d_{limit} より遠いノードがある場合の $O(|E_I|)$ と整合している．

表 2 に，探索用グラフの各ノードからの最遠ノードまでの距離分布を最小値，最大値（すなわち直径）として示す．最頻値は最大値と一致しており，$k = 6, 8, 10$ においては中央値も最大値と一致している．$k = 2$, $|E_I| \geq 800$ の場合，どのノードからも d_{limit} より遠いノードがある．この場合の実行時間は，エッジ数に対してほぼ線形に増加している．一方，$k = 6, 8, 10$ の場合には，直径が d_{limit} 以下で，おおむね二乗オーダーとなっている．

5.　まとめ

Simulink などの車載制御モデルを対象に，モデル間の差分を漏れなく検出するためのアルゴリズムを開発した．アルゴリズムは，モデルの対に対応したグラフ対を入力とし，最大共通部分グラフの近似解を貪欲法で求める．

このアルゴリズムは，Simulink モデルが扱える Deissenboeck らのアルゴリズム [8] と，Rutgers らの線形時間

表 2　探索用グラフの最遠ノード距離分布
Table 2　Farthest Distance Distribution in Graph on Search

$\|E_I\|$	$k=2$		4		6		8		10	
	min	max								
200	7	13	4	7	3	5	3	5	2	4
400	8	15	4	8	3	6	3	5	3	5
600	8	16	5	9	4	7	3	6	3	5
800	9	17	5	9	4	7	3	6	3	5
1000	9	17	5	9	4	7	3	6	3	5
1200	9	18	5	9	4	7	4	7	3	6
1400	9	18	5	10	4	7	4	7	3	6
1600	10	19	5	10	4	8	4	7	3	6
1800	10	19	5	10	4	8	4	7	3	6
2000	10	19	5	10	4	8	4	7	3	6

アルゴリズム [10]，それぞれの利点を兼ね備えている．

アルゴリズムの実行時間が，エッジ数に対して線形になるようにするため，以下の方策をとった．

(1) 共通部分グラフに追加するエッジ対を選ぶ際の評価関数は，選択の都度，算出し，メモ化する．
(2) アルゴリズム開始時のエッジ対候補を，始終点およびエッジ自身のラベル 3 つ組が，2 つのグラフ間で 1 対 1 対応しているエッジ対に限定する．
(3) 探索中のエッジ対候補として，共通部分グラフ中のノードに接続するエッジ対を優先する．

これらの方策によって，グラフ全体のエッジより，評価関数値算出時に考慮するエッジが少ない場合には，時間計算量，空間計算量，ともにエッジ数に対して線形増加する．

Simulink モデルを模した同一のグラフ対（エッジ数 200–2000 の 4 分木）に，このアルゴリズムを適用したところ，実行時間はエッジ数に対して線形増加の傾向を示した．

参考文献

[1] MathWorks: Simulink Report Generator, The MathWorks, Inc., available from ⟨http://www.mathworks.co.jp/products/simulink/⟩ (accessed 2013-11-11).
[2] Sudarshan S. Chawathe, Anand Rajaraman, H. G.-M. and Widom, J.: Change Detection in Hierarchically Structured Information, *Proc. 1996 ACM SIGMOD Intl. Conf. on Management of Data*, Montreal, Canada, ACM, pp. 493–504 (1996).
[3] ExpertControl: ecDIFF, ExpertControl GmbH, available from ⟨http://www.expertcontrol.com/en/index.php⟩ (accessed 2013-11-11).
[4] Hsu, R.: Method for detecting differences between graphical programs, Patent 5974254, The United States Patent and Trademark Office (1999).
[5] EnSoft: SimDiff 4 Team, EnSoft Corp., available from ⟨http://www.ensoftcorp.com/simdiff/⟩ (accessed 2013-11-11).
[6] ikv++ technologies: medini unite, ikv++ technologies ag, available from ⟨http://www.ikv.de/index.php/en/products/unite⟩ (accessed 2013-11-11).
[7] Donatello Conte, P. F. and Vento, M.: Challenging Complexity of Maximum Common Subgraph Detection Algorithms: A Performance Analysis of Three Algorithms on a Wide Database of Graphs, *Journal of Graph Algorithms and Applications*, Vol. 11, No. 1, pp. 99–143 (2007).
[8] Florian Deissenboeck, Benjamin Hummel, E. J. B. S. S. W. J.-F. G. and Teuchert, S.: Clone Detection in Automotive Model-Based Development, *Proc. 30th Intl. Conf. on Software Engineering*, Leipzig, Germany, ACM/IEEE, pp. 603–612 (2008).
[9] CQSE: Analysis of Simulink Models, CQSE GmbH, available from ⟨https://www.cqse.eu/en/products/conqat/tutorials/⟩ (accessed 2013-09-23).
[10] Jochem H. Rutgers, Pascal T. Wolkotte, P. K. F. H. J. K. and Smit, G. J. M.: An Approximate Maximum Common Subgraph Algorithm for Large Digital Circuits, *Proc. 13th EUROMICRO Conf. on Digital System Design*, Lille, France, IEEE, pp. 699–705 (2010).

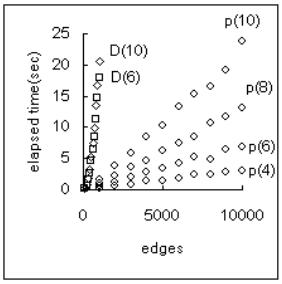


図 **A·1**　実行時間の比較（p: 提案アルゴリズム（$d_{limit}=5$），D: Deissenboeck 改造版，カッコ内：k）
Fig. A·1　Comparison on the Elapsed Time (p: proposed($d_{limit}=5$), D: modified Deissenboeck's, in parentheses: k)

付　　録

$d_{limit}=5$ での提案アルゴリズムと，Deissenboeck 改造版 *15 の実行時間を比較した．

実験用グラフを対象として，検出結果が正解と一致した k での実行時間を図 A·1 に示す *16．エッジ数に対する実行時間の増加傾向は，提案アルゴリズムでほぼ線形なのに対し，Deissenboeck 改造版ではおおむね二乗であった．

一方，公開されている Simulink モデル airlib.mdl*17 中で違いが明らかな 2 つのサブシステム B747 と DT-B747 *18 の場合，2 つのアルゴリズムともに正しく検出し，実行時間は大差なかった（提案アルゴリズム 0.2 秒，Deissenboeck 改造版 0.1 秒）．

*15 最大共通部分グラフ検出向けに改造（2.1 節）
*16 Deissenboeck 改造版において $k=8$ は省略
*17 [9] の例題として付属
*18 どちらもボーイング 747 をモデル化したもので，モデル中の唯一の積分器ブロックだけが異なる．それぞれ Integrator，Discrete-Time Integrator である．[9] によってグラフ化したところ，入力グラフとしてはノード数 183，エッジ数 308，探索用グラフとしては直径 11，次数平均値 10.5，次数最大値 37 となった．