様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

# 科学研究費助成事業　　研究成果報告書

令和　３ 年　 ６ 月　 ６ 日現在

機関番号：１２７０１
研究種目：基盤研究(C)（特設分野研究）
研究期間：2017～2020
課題番号：１７ＫＴ０１２２

研究課題名（和文）派生システム開発の品質変化を予測する複雑ネットワーク指標に基づくメトリクスの開発

研究課題名（英文）Development of Metrics Based on Complex Network Indicators to Predict Quality Changes in Derivative System Development

研究代表者
濱上　知樹（HAMAGAMI, TOMOKI）

横浜国立大学・大学院工学研究院・教授

研究者番号：３０３３４２０４

交付決定額（研究期間全体）：（直接経費）　　3,400,000円

研究成果の概要（和文）：ライフサイクルの長いソフトウェアシステムの品質を維持していくために、派生開発および継続的インテグレーションによる構造変化と品質の関係を評価する複雑ネットワークメトリクスを開発した。そして、機械学習による予測手法によって自律的なソフトウェア発展に貢献する。まず、メトリクスによるソフトウェア構造の分析を行い、改変前後のソフトウェアにおける複雑ネットワークの指標値を計測した。次に、このメトリクスを用いた機械学習によるバグ予測を行い有効性を確認した。さらに、ブロック型ニューラルネットワークによるFPGA実装においてメトリクスを適応度とする進化的手法を用いて高速な設計を行う拡張BBNNの技術を実現した。

研究成果の学術的意義や社会的意義
近年のソフトウェア開発においては，新規開発に比べ既存のソフトウェアを改変する派生開発の比重が増している。特に組み込みソフトウェアのように，ライフサイクルが比較的長く，機能追加のための改変が頻繁に行われる開発形態では，改変時の品質管理が重要な課題となる。かかるソフトウェアの発展において、将来的な品質を予測し問題が起きる前に対処することは重要な課題である。本研究では、複雑ネットワーク指標を用いたメトリクスによって改変後の品質に与える影響が予測できることを示すとともに、FPGAと機械学習を用いた実験により発展的ソフトウェアの品質評価と継続的インテグレーションの方法を明らかにした。

研究成果の概要（英文）：In order to maintain the quality of software systems with long lifecycles, we have developed complex network metrics to evaluate the relationship between quality and structural changes caused by derivative development and continuous integration. Then, we contribute to autonomous software development by using machine learning prediction methods. First, we analyzed the software structure using the metrics and measured the index values of the complex network in the software before and after the modification. Next, we performed bug prediction by machine learning using these metrics and confirmed its effectiveness. In addition, we realized an extended BBNN technique for a fast design using an evolutionary method with metrics as the adaptive degree in FPGA implementation with block neural networks.

研究分野：知能システム

キーワード：ソフトウェア品質管理　複雑ネットワーク　機械学習　ニューラルネットワーク

様　式　Ｃ－１９，Ｆ－１９－１，Ｚ－１９（共通）

１．研究開始当初の背景

近年のソフトウェア開発においては，新規開発に比べ既存のソフトウェアを改変する派生開発の比重が増している。一方で，度重なる改変が行われたシステムは，保守性の低下や品質の劣化が生じやすくなることが知られている。特に近年の組み込みソフトウェアのように，ライフサイクルが比較的長く，機能追加のための改変が頻繁に行われる開発形態では，改変時の品質管理が重要な課題となる。

この課題に対し，これまでにもソフトウェア構造に関するメトリクスと改変時に発生する欠陥との相関について，多くの研究成果が報告されてきた(①②)。伝統的な研究例としてプログラムの最大ネスト数や分岐数に着目した Cyclomatic Number がある。Cyclomatic Number が改変時の品質に大きな影響を与えるという報告もあるが，オープンソースを対象に分析した結果，ほとんど影響はないという報告(③)もあり，一定以上の成熟された開発プロセスを適用して開発されたソフトウェアでは，小さなプログラムの個々の複雑さはさほど品質に影響を与えないことが予想される。

以上の従来研究は主に改変後の欠陥の発生数を予測することを目的として改変前のネットワーク指標値と欠陥の発生状況の関係を分析している。しかし，欠陥の発生にはどのような改変が行われたかに依存する。特に近年のライフサイクルの長いソフトウェアシステムの品質を維持していくためには，各回の改変が構造に及ぼす影響を測り，その構造変化と品質の関係を評価するメトリクスと継続的インテグレーションが可能な自律性を実現するしくみが必要である。

２．研究の目的

本研究では，ネットワーク機器の組み込みソフトウェアを対象に，３つの目的を設けた。

(1) 複雑ネットワーク(④⑤)の指標値によるソフトウェア構造の分析：改変前後のソフトウェアにおける複雑ネットワークの指標値を計測し，同時にその改変で生じた欠陥数との関係を分析することで改変に適したソフトウェア構造，および欠陥を生じる可能性が高い改変の特性を明らかにすることを目的とする。

(2) 複雑ネットワーク指標メトリクスを用いた機械学習によるバグ予測：このメトリクスを用いたバグの発生を機械学習によって予測する方法を明らかにする。

(3) ブロック型ニューラルネットワークによる FPGA 実装(⑥)：さらにこれらのメトリクスを応用したソフトウェアの自律的な再構築を FPGA によって実装しその効果を確認することをめざした。本研究の基本的な考え方を示す。

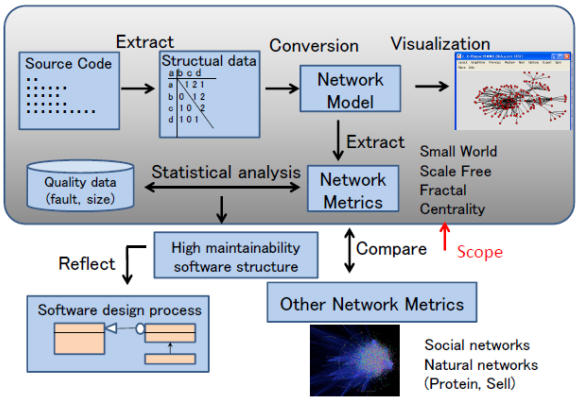


図１　本研究の全体像

３．研究の方法

(1) 複雑ネットワークの指標値によるソフトウェア構造の分析

本研究では，実験対象とするソフトウェアプロジェクトが，C 言語で実装されたモジュール構造であることを想定し，図２に示すデータフローダイアグラム（DFD）に準じたネットワーク化の考え方を採用した。ここでノードとは「プログラム中の関数およびグローバルデータ」，「リンク」はノード間の依存関係を表す有向リンクと定義する。

ノード間の依存関係としては，一般に（1）関数-関数の呼び出しと（2）関数-グローバルデータへのアクセス，オブジェクト指向言語ではそれらに加えて（3）関数（クラス）間の継承関係があるが，本研究では（1),(2）のみを対象とした。

ソフトウェアネットワークを分析するにあたり，改変時の品質に関連するネットワーク指標値として，リンク密度，媒介中心性，分散中心性の分散，到達平均ノード間距離に着目する。

（i）リンク密度（Link density: LD）はノード数とリンク数の比率である。一般に LD が高いネットワークは，変化に伴う影響が伝搬しやすく，改造時に欠陥が発生しやすい。

（ii）媒介中心性（Betweenness centrality: CB）は情報伝達におけるフローコントロールの可能性を示す指標である。ノード v の CB（CB(v)）は，v を除く他ノード間最短パスのうち，v を経由するパスが存在する比率と定義される。

（iiI）分散中心性の分散（Betweenness centrality dispersion: CBD）は，(3) 式で示すように，ネットワーク内で最大の CB（CBmax = maxCB(v),v∀）を持つノードと全ノードの CV (v) の差分の合計と定義する。

（iV）到達平均ノード間距離（Average distance: AD）は，到達するパスが存在する 2 ノード間の平均距離である。母体の AD が長いほど改変の影響が遠くに波及する可能性が高い。また改変によって AD が長くなる場合，構造変化に伴う影響を確認・試験することが考えられる。そのため，AD が長い母体や，改変によって長くなる場合に，欠陥が生じやすいことが予想される。

以上の指標を用いることでソフトウェア構造の複雑性を評価する。

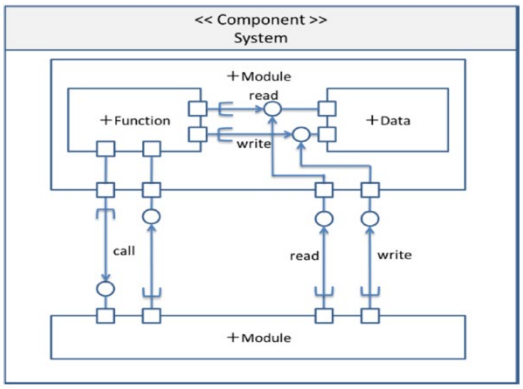


図２　ソフトウェアの階層構造モデル

(2) 複雑ネットワーク指標メトリクスを用いた機械学習によるバグ予測

表１に分析対象とした 6 種類のソフトウェアの緒元を示す。評価対象のソフトウェアはいずれも 3 回の改変(4ver.)が行われている。各緒元は最新のバージョンにおける値である。このソフトウェアの改変に伴う品質変化を分析するにあたり，表 2 に示したソフトウェアを構成するモジュールの中から，2 種類のモジュール（高品質モジュール（HQM），低品質モジュール（LQM））を抽出した。

改変によるネットワーク指標の変化分析 HQM, LQM それぞれ 17 モジュールについて，各改変によって生じた変化を，最初のバージョン(Ver.1)を 1.0 とした場合の相対的な増減として分析する。ことでメトリクスとの関係性を分析し，機械学習よりバグ頻度を予測する。

表１　評価対象のソフトウェアモジュール

| Software | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Nodes | 23,478 | 9,370 | 6,786 | 19,942 | 3,942 | 3,889 |
| Links | 66,548 | 26,831 | 50,932 | 124,073 | 13,866 | 6,526 |
| Steps(k) | 4,141 | 1,159 | 749 | 796 | 724 | 1,358 |
| Modules | 24 | 9 | 42 | 32 | 37 | 31 |
| HQMs | 3 | 1 | 4 | 3 | 3 | 3 |
| LQMs | 2 | 1 | 4 | 3 | 3 | 4 |

(3) ブロック型ニューラルネットワークによる FPGA 実装

ここまでに得られたメトリクスの効果を，実機で確認するために FPGA 上に実装されるソフトウェアを対象に，メトリクスに基づく自律的再構成の実験を行う。ここで，FPGA 上のネットワーク実装モデルとして BBNN(Block Based Neural Network)を採用し，非同期型基本ブロックとパイプライン処理を実現することで，評価値に基づく再構成方法を明らかにする。

本研究で対象とする BBNN は複数の基本ブロックで構成された各ステージと複数のステージが接続された 2 次元構造ネットワークである。基本ブロックを組み合わせることで，所望の動作をするよう設計が行われる。BBNN のネットワーク構造と各基本ブロックの重みを同時に扱うために，これらの組み合わせを 1 つの染色体としてエンコードを行う。エンコードされた染色体は遺伝的アルゴリズム（GA）を用いて，訓練データの結果に近づけることで最適な構造と重みの組み合わせを求める。

本研究では，メトリクス（評価関数）に基づくリアルタイム処理を可能にするための非同期型基本ブロック(Asynchronous Block)とパイプライン処理を組み合わせた高速化されたブロック構造ネットワーク(Asynchronous BBNN)を提案する。ABBNN は BBNN の左右ノードを二重に拡張することで，すべてのリソースを有効に使用しながら左右信号処理を並列に実行する。これにより，ステージ内クリティカルパスを短縮し，処理の高速化が可能になる。また，各ステージの出力にレジスタを挿入することでパイプライン処理を実現する。これによりステージにまたがるクリティカルパスを短縮することでステージ間処理の高速化が可能になる。さらに，各ステージの出力に接続されるレジスタ層にはマルチプレクサを挿入する「学習モード」を設け，学習モードの間はレジスタをバイパスするようにする。

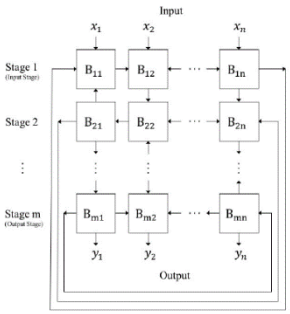


図３　FPGA 上の実装される BBNN の構造

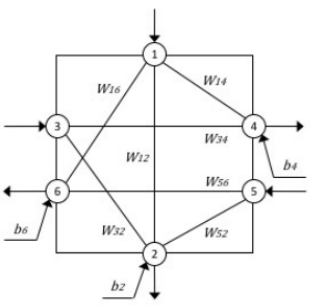


図４　ABBNN 基本ブロック

４．研究成果

(1,2)複雑ネットワークの指標値によるソフトウェア構造の分析および機械学習によるバグ予測

改変によるネットワーク指標の変化分析 HQM, LQM それぞれ 17 モジュールについて，各改変によって生じた変化を，初期バージョン(Ver.1) を 1.0 とした場合の相対的な増減として分析を行った。図５に，それぞれソフトウェア規模の変化（Ratio of size），ノード数の変化(Ratio of node num），リンク数の変化(Ratio of link num），リンク密度の変化(Ratio of LD），媒介性中心度の変化(Ratio of CBN) を，平均ノード間距離(Ratio of AD) の変化を示す。

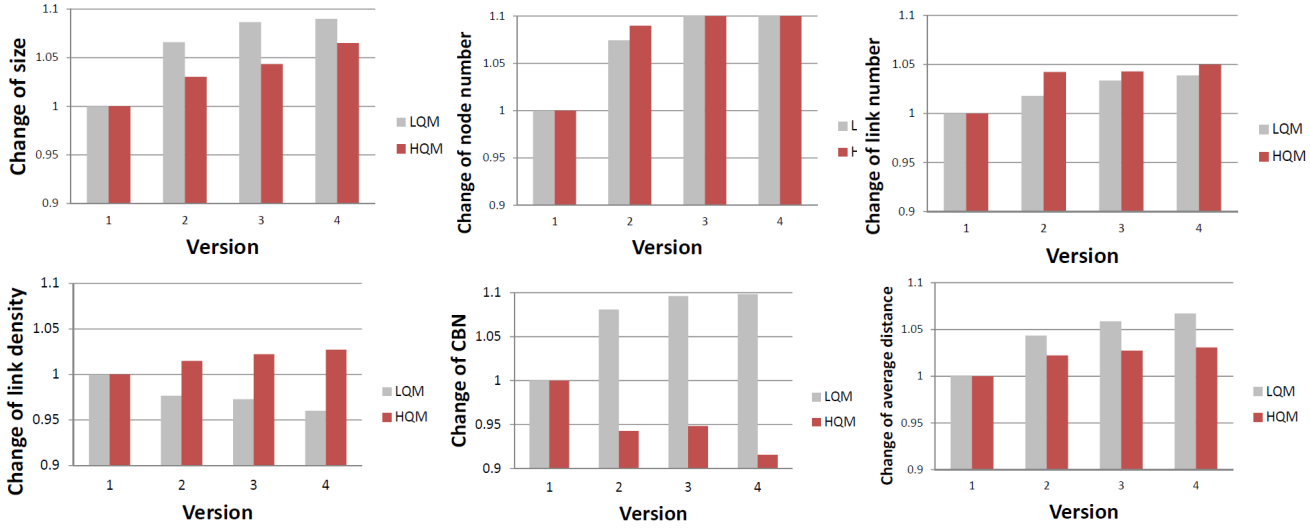


図５　各メトリクスの評価結果

得られたネットワークメトリクスを用いて，バグ変数を目的変数とした回帰分析を，重回帰分析およびサポートベクタ回帰(SVR) を用いて試みた。用いたメトリクスは以下の通りである。

(A) 規模(steps)　(B) ネットワークメトリクス　(C) ネットワークメトリクスの差

表 2 に重回帰分析の結果を，表 3 に SVR を用いた結果を示す。

表２　重回帰分析によるバグ予測結果

| | (A) | (A)+(B)+(C) | (B)+(C) | (C) |
|---|---|---|---|---|
| 残差 | 1329.7 | 1071.4 | 1543.0 | 1762.8 |
| 平均誤差 | 10.1 | 8.1 | 11.7 | 13.4 |
| 平均誤差率 (%) | 53.6 | 43.2 | 62.2 | 71.1 |

表３　ＳＶＲによるバグ予測結果

| | (A) | (A)+(B)+(C) | (B)+(C) | (C) |
|---|---|---|---|---|
| 残差 | 947.0 | 736.2 | 592.2 | 1076.5 |
| 平均誤差 | 7.2 | 5.6 | 4.5 | 8.2 |
| 平均誤差率 (%) | 38.2 | 29.7 | 23.9 | 43.4 |

重回帰分析において(A) による予測誤差より，(B)+(C)に基づく予測誤差が大きくなるのはメトリクス相互の関係や設計アーキテクチャの特性により，バグへの影響に非線形あることを示唆している。これに対し，SVR による予測はいずれも重回帰分析精度を上回っており，さらに(B)+(C)による精度が最も高いことがわかる。また(C) のみによる予測精度に比べ(B) を含めた方が精度がよいこと，また規模に関する情報はネットワークメトリクスを用いた予測には必要なく，むしろ過学習によって予測精度が下がることが明らかになった。

(3) ブロック型ニューラルネットワークによる FPGA 実装

ABBNN が，BBNN と同様に進化的手法によって構築可能であることをシミュレーションによって確認した。実験環境には SystemVerilog で設計した回路のシミュレーションを用いた。SystemVerilog はハードウェア記述言語である VerilogHDL を拡張した言語であり，VerilogHDL に比べて検証に関する機能が大幅に拡張・統合されている。さらにオブジェクト指向プログラミング（Object Oriented Programming，OOP）を可能にする class 構文の採用と制約付きランダムテスト関連機能に優れているため，多く普及しているハードウェア記述言語である。

表４に，2x2～3x3 の各サイズにおける ABBNN のシミュレーション結果を示す。シミュレーションでは閉路を形成しないすべての構造を網羅し，進化の収束した平均世代数(Avg.$gene)を求めた。Valid structure rate は閉路を形成しない進化可能な構造の比率，Success rate は進化に成功した割合，Total#structure は存在する構造の数である。2x3 構造の場合，進化可能な構造は全体の 17%であり，進化が収束するまでに必要な世代数は 2779 であった。これらの結果から，進化可能な ABBNN 構造を発生させながら，それぞれの構造に対応した重みを進化させることで，目的とする ABBNN が得られることが明らかとなった。

つぎに，進化的手法で得られた ABBNN の論理合成を行い，パイプライン化の効果を評価した。2x3 の ABBNN モデルから 4x8，4x16，8x16 構造 ABBNN の論理合成を行い，パイプラインありとなしの ABBN における動作周波数の変化を確認する。

表５にパイプラインの有無による動作周波数の違いを示す。パイプラインがある場合，ネットワークの規模が大きくなっても回路の動作周波数は変わらない。すなわち，ネットワーク規模の増加に伴う動作周波数の低下は見られない。一方，パイプラインがない場合，ネットワーク規模

の増加に伴い，急激に動作周波数が低下していく。なお，4x16 と 8x16 のパイプラインなしの場合の動作周波数が同じであることから，動作周波数は同ステージ上のブロックの数より，ステージの数に多く影響を受けることがわかる。以上の結果から，パイプライン化の有効性が明らかとなった。

さらに，論理合成の結果から動作周波数を確認し，BBNN の規模と動作周波数の関係について確認した。そして，同じ規模のパイプライン BBNN と非パイプライン BBNN の動作周波数を比較することで，提案したパイプライン構造が BBNN の高速化に貢献できていることを確認した。

表６に 8x16ABBNN の論理合成後の面積を比較した結果を示す。パイプライン化をした場合，しない場合に比べてよりレジスタの使用量が増加するものの，チップ全体の面積に対する割合としてはその増加は限定的である。

表４　シミュレーション結果　表５パイプラインの有無の評価　　表６　8x6ABBN 評価結果

| Size of BBNN | 2×2 | 2×3 | 3×2 | 3×3 |
|---|---|---|---|---|
| Avg. #gene | 1983 | 2779 | 5045 | 4383 |
| Valid structure rate | 32% | 17% | 59% | 45% |
| Success rate | 100% | 100% | 100% | 100% |
| Total #structure | 256 | 4,906 | 4,906 | 262,144 |

| Size of ABBNN | 2×3 | 4×8 | 4×16 | 8×16 |
|---|---|---|---|---|
| Pipeline ABBNN | 30.3MHz | 30.3MHz | 30.3MHz | 30.3MHz |
| Non-pipeline ABBNN | 13.25MHz | 4.78MHz | 2.41MHz | 2.41MHz |

| Resource (Available) | Pipeline | Non-Pipeline |
|---|---|---|
| IOs (1120) | 53.12% | 53.21% |
| LUTs (650,440) | 3.66% | 3.66% |
| Register (650,440) | 0.77% | 0.41% |
| Memory (23,685,120) | 0.00% | 0.00% |
| DSP (960) | 100% | 100% |

図６に ABBNN の収束世代数（Number of Generation）と進化に成功した学習可能構造数（Number of BBNN）を示す。2x2ABBNN は４つの基本ブロックから構成され，それぞれ４タイプの基本ブロックが配置されうるので，構造のパターンは 256 である。2x2ABBNN の学習可能構造数は総構造の 32%であり，学習が成功するまでの平均世代数は 1,983 である。2x3 と 3x2　ABBNN では 4,096 構造の内，それぞれ 17%の 704 構造，59%の 2,401 構造が学習可能構造で，学習成功までの平均世代数は 2,779 と 5,045 である。3x3ABBNN の総構造数は 262,144 から 45%の 118,024 構造が学習可能構造で学習成功平均世代数は 4,383 である。なお，実験対象となっている 2x2～3x3 の全ての構造において進化可能構造の内，約 65%の構造が平均世代数以下で収束していることから，相対的に進化が収束しやすい ABBNN 構造が存在していると考えられる。

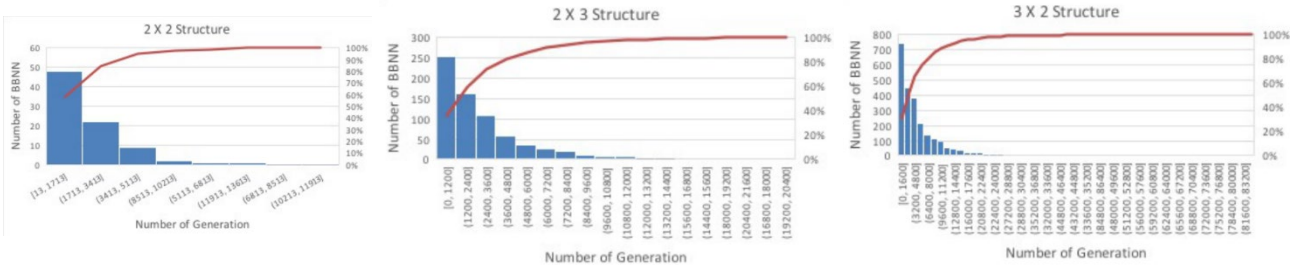


図６　各構造における構造の成功数・収束世代数

以上のように，BBNN に非同期型基本ブロック構造とパイプライン処理を組み合わせた ABBNN を，メトリクスを適応度として最適化することが可能になった。最適化にあたっては，閉路が生じないパターンを予め生成し，生成された構造に対して遺伝的アルゴリズムによる重みの最適化を行った。シミュレーション実験および論理合成の結果，従来手法より動作周波数の高速化がはかれた。

＜引用文献＞


① McCabe, Thomas J. and Butler, Charles W. “Design Complexity Measurement and Testing, ” Communications of the ACM 32, 12 pp.1415-1425, 1989.

② C.Jones, “Applied Software Measurement Second Edition” 1991 McGraw-Hill.

③ Coverity Home Page, ” Open Source Report 2008”. http://scan.coverity.com/report/Coverity Whit Paper-Scan Open Source Report 2008.pdf

④ G.Paul, T.Tanizawa, S. Havlin, H. E. Stanley, “Optimization of Robustness of Complex Networks,”, The EuropeanPhysical Journal B - Condensed Matter and Complex Systems Volume 38, Issue 2,pp.187{191, 2004.

⑤ Gregory Madey, Vincent Freeh, and Renee Tynan, “The open source software development phenomenon: An analysis basedon social network theory,” Proceedings of the Americas Conference on Information Systems, pp.1806{1813, 2002.

⑥S.W.Moon and S.G.Kong : ”Block-Based Neural Networks”, IEEE trans. Neural Networks, Vol.12, No.2 pp.307-317, 2001.

５．主な発表論文等

〔雑誌論文〕　計1件（うち査読付論文　1件／うち国際共著　0件／うちオープンアクセス　0件）

| | |
|---|---|
| １．著者名<br>Lee Kundo、Hamagami Tomoki | ４．巻<br>140 |
| ２．論文標題<br>Block-Based Neural Network Optimization with Manageable Problem Space | ５．発行年<br>2020年 |
| ３．雑誌名<br>IEEJ Transactions on Electronics, Information and Systems | ６．最初と最後の頁<br>68～74 |
| 掲載論文のDOI（デジタルオブジェクト識別子）<br>10.1541/ieejeiss.140.68 | 査読の有無<br>有 |
| オープンアクセス<br>オープンアクセスではない、又はオープンアクセスが困難 | 国際共著<br>－ |

〔学会発表〕　計11件（うち招待講演　0件／うち国際学会　3件）

| |
|---|
| １．発表者名<br>Li Xin、Hamagami Tomoki |
| ２．発表標題<br>An Improved Auto-encoder Based on 2-Level Prioritized Experience Replay for High Dimension Skewed Data |
| ３．学会等名<br>Intelligent and Evolutionary Systems 2019, Springer（国際学会） |
| ４．発表年<br>2019年 |

| |
|---|
| １．発表者名<br>Lee Kundo、Hamagami Tomoki |
| ２．発表標題<br>Block-Based Neural Network High Speed Optimization |
| ３．学会等名<br>Intelligent and Evolutionary Systems 2019, Springer（国際学会） |
| ４．発表年<br>2019年 |

| |
|---|
| １．発表者名<br>Li Xin,Hamagami Tomoki |
| ２．発表標題<br>Prioritized Sampling Method for Autoencoder to Reduce Loss Rate for Skewed Data |
| ３．学会等名<br>電気学会　システム研究会　ST-18-076 |
| ４．発表年<br>2018年 |

１．発表者名
岡崎雅也，濱上知樹

２．発表標題
教師なしランダムフォレストを用いた多変量時系列データの類型化

３．学会等名
電気学会　システム研究会　ST-18-115

４．発表年
2018年

１．発表者名
Kundo Lee, Tomoki Hamagami

２．発表標題
Performance oriented Block-Based Neural Network Model by parallelized neighbor's communication

３．学会等名
IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2018 ) (国際学会)

４．発表年
2018年

１．発表者名
濱上知樹

２．発表標題
機械学習によるソフトウェア品質メトリクスの研究

３．学会等名
電気学会システム研究会 ST-18-010

４．発表年
2017年

〔図書〕　計1件

| １．著者名<br>濱上知樹 | ４．発行年<br>2017年 |
|---|---|
| ２．出版社<br>情報機構 | ５．総ページ数<br>337 |
| ３．書名<br>機械学習・人工知能　業務活用の手引き～導入の判断・具体的応用とその運用設計事例集～第３章　機械学習とそのアルゴリズム | |

〔産業財産権〕

〔その他〕

―

６．研究組織

| | 氏名<br>（ローマ字氏名）<br>（研究者番号） | 所属研究機関・部局・職<br>（機関番号） | 備考 |
|---|---|---|---|

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

| 共同研究相手国 | 相手方研究機関 |
|---|---|