International Journal of Informatics Society

03/21 Vol. 12 No.3 ISSN 1883-4566

Informatics Society

Aims and Scope

The purpose of this journal is to provide an open forum to publish high quality research papers in the areas of informatics and related fields to promote the exchange of research ideas, experiences and results.

Informatics is the systematic study of Information and the application of research methods to study Information systems and services. It deals primarily with human aspects of information, such as its qu ality and value as a resource. Informatics also referred to as Information science, studies t he structure, algorithms, behavior, and interactions of natural and a rtificial systems that store, process, access and communicate information. It also develops its own conceptual and theoretical foundations and utilizes foundations developed in other fields. The advent of computers, its ubiquity and ease to use has led to th e study of info rmatics that has computational, cognitive and social aspects, including study of the social impact of information technologies.

The characteristic of informatics' context is amalgamation of technologies. For creating an informatics product, it is necessary to integrate many technologies, such as mathematics, linguistics, engineering and other emerging new fields.

Guest Editor's Message

Takayasu Yamaguchi

Guest Editor of Thirty-fourth Issue of International Journal of Informatics Society

We are delighted to have the Thirty-sixth issue of the International Journal of Informatics Soci-ety (IJIS) published. This issue includes selected papers from the Twelfth International Workshop on Informatics (IWIN2019), which was held at Hamburg, Germany, Sept. 8-11, 2019. The workshop was the thirteenth event for the Infor-matics Society. It was intended to bring together researchers and practitioners to share and ex-change their experiences, discuss challenges and present original ideas in all aspects of informat-ics and computer networks. In the workshop, 25 papers were presented in seven technical ses-sions. The workshop was successfully finished with precious experiences provided to the partic-ipants. It highlighted the latest research results in the area of informatics and its applications that include networking, mobile ubiquitous systems, data analytics, business systems, education sys-tems, design methodology, intelligent systems, groupware and social systems.

Each paper submitted IWIN2019 was re-viewed in terms of technical content, scientific rigor, novelty, originality and quality of presen-tation by at least two reviewers. Through those reviews, 20 papers were selected for publication candidates of IJIS, and they were further re-viewed as a journal paper. We have three catego-ries of IJIS papers, Regular papers, Industrial pa-pers, and Invited papers, each of which were re-viewed from the different points of view. This volume includes seven papers among those ac-cepted papers, which have been improved through the workshop discussion and the re-viewers' comments.

We publish the journal in print as well as in an electronic form over the Internet. We hope that the issue would be of interest to many research-ers as well as engineers and practitioners over the world.

Takayasu Yamaguchi received B.E., M.E. and Ph.D. degrees from the University of Electro-Communications in 1999, 2001 and 2019, re-spectively. After his working in NTT DOCOMO, Inc. from 2001 through 2020, he has been a pro-fessor in the Department of Management Sci-ence and Engineering, Faculty of Systems Sci-ence and Technology, Akita Prefectural Univer-sity, since 2020. He obtained nineteen patents for information technologies such as object iden-tification using a mobile camera, vast multime-dia contents search engine, and mobile spatial statistics grasping dynamic population in Japan while preserving their privacy. He received the IPSJ Kiyasu Special Industrial Achievement Award in 2015, the IPSJ Outstanding Paper Award in 2016, and IWIN2017 Best Paper Award in 2017. He is a member of the Infor-mation Processing Society of Japan (IPSJ). His research interests include artificial intelligence, cyber-physical system, and financial technology.

Regular Paper

Proposal and Evaluation for A Method to Verify Equivalence of Specifications of C and Java Functions with Recursive Data Structures by SAW: Case Studies of Linear Structures and Binary Trees

Rin Karashima[†], Kozo Okano[‡], Shinpei Ogata[‡], Satoshi Harauchi[§], and Toshifusa Sekizawa*

[†]Faculty of Engineering, Shinshu University, Japan
[§] Mitsubishi Electric Corporation Advanced Technology R&D Center
* Faculty of Informatics, Nihon University, Japan
19w2036c@shinshu-u.ac.jp[†]
{okano, ogata}@cs.shinshu-u.ac.jp[‡]
Harauchi.Satoshi@bc.MitsubishiElectric.co.jp[§]
sekizawa@cs.ce.nihon-u.ac.jp*

Abstract - A process to improve the internal structure of a software system while keeping its external behavior is called refactoring. When performing refactoring, programmers need to verify equivalence of the new and old programs by checking that the functions of both satisfy the given specifications to avoid unexpected bugs. Generally, programmers perform unit testing by using test cases like JUnit, but it has the problems of lack of completeness and high time costs. In the Software Analysis Workbench (SAW) verification tool, programmers use the SAWScript scripting language to define verification properties that target functions must satisfy and formal verification is performed by using SMT solvers and symbolic execution. With the symbolic execution of SAW, programmers can perform unit testing completely and effectively without test cases.

This paper proposes a method for verifying the specifications of target functions by generating a helper function and calling a target function as its return value. By using this method, programmers can define the values of class objects that the target function receives in a helper function and make SAW verification possible. As a case study, we performed SAW verification for Java functions that receive recursive class objects as arguments and showed that property verification and equivalence checking is actually possible.

Keywords: SAW, verification, equivalence, refactoring, unit test, recursive data structures

1 INTRODUCTION

Both Java and C programs are widely used for modern information systems. Sometimes, changes of the working environment force engineers to develop new software code with the same functions as the old codes. For example, C programs working in an old environment are sometimes changed to new Java programs that must work in a new environment. In these situations, programmers have to ensure that the revised Java programs preserve the behavior of the old C versions.

As another example, a programmer might have to develop new C programs for an embedded system, where the CPU and memory resources are limited, based on an existing Java program which runs in a newer environment with rich resources. In such a case, programmers have to ensure that the revised C programs preserve the behavior of the old Java programs. The process of improving the internal structure of a software system while maintaining its external behavior is called refactoring [1].

Usually, regression testing is performed to check whether different versions of a program have the same behavior. In the above situation, however, we usually cannot use the same regression test-suites because they use different programming languages. Software Analysis Workbench (SAW) [2], [3] provides the ability to verify programs written in C and Java by symbolic execution.

Formal approach techniques might help in such a situation. These techniques will find potential bugs by checking the conformance to program specifications with adequate efficiency. We call this kind of verification formal conformance verification (FCV). Recent tools, however, do not fully support programs dealing with dynamic data structures especially recursive data structures. For example, our previous research [4] uncovered the possibility of FCV for C programs with recursive data structures.

In this work, we attempt to resolve this problem for Java programs by using the idea of bounded model checking. However, for Java programs, we do not know the possibility of applying FCV. The reasons are summarized as follows.

- SAW retrieves information for verification from the byte codes of C or Java and the difference of the codes affects the checking algorithm.
- Java is based on the Java Virtual Machine while C has no reference machine to interpret.

In order to avoid the halting problem (e.g., if verification is performed without defining the end of a linear list structure, the verification may fail without completing the recursion and a state explosion could occur), our proposed method is based on the bounded model verification technique [5], [6]. In this work, we also perform experimental evaluation using SAW, a recent formal verification tool.

From the experimental results, we find that it is partially possible to perform verification of functions dealing with recursive data structures in Java. This observation supports the possibility of using FCV between C and Java with recursive data structures.

The rest of this paper is organized as follows. Section 2 gives the preliminaries and Section 3 describes the proposed method. Sections 4 and 5 describe the experimental evaluation and results. Section 6 discusses the results. Finally, Section 7 summarizes this paper.

2 PRELIMINARIES

2.1 Equivalence Between Two Functions

Refactoring is defined as the process of improving the internal structure of a software system while keeping its external behavior [1]. Thus, when we perform refactoring on a program function, we need to check that the external behaviors of the new and old functions are equivalent.

In this research, we defined external behaviors as the inputs and outputs of program functions. When two functions f and g are equivalent, expression (1) holds.

$$\forall x \in A : f(x) = g(x) \tag{1}$$

We say these functions are equivalent for any x that satisfies A, where A specifies the range and type of the variable x. For example, if x belongs to an integer type, then $A = \mathbb{Z}_{32}$ holds, where $\mathbb{Z}_{32}$ represents the set of signed integers with 32-bit width. In this paper, we use the following notations to show the integer types $\mathbb{Z}_{16}$, $\mathbb{Z}_{32}$, $\mathbb{Z}_{64}$, $\mathbb{Z}_{8^u}$, $\mathbb{Z}_{16^u}$, $\mathbb{Z}_{32^u}$, and $\mathbb{Z}_{64^u}$ which correspond to the signed integers of 16-, 32-, and 64-bit width, and the unsigned integers of 8-, 16-, 32-, and 64-bit width, respectively.

In this work, we consider inputs and outputs as the requirements of equivalence and do not consider program size, algorithm, execution time, type of programming language, etc.

The parameter x can be easily extended to a parameter vector with the same signatures. Here, a signature is a list of types corresponding to each element of the parameters.

The programming languages for f and g are not fixed, but in this paper, we use the case where f is implemented in Java and g is implemented in C.

2.2 SAW

The unit testing framework xUnit[7][8] is the most popular method for verifying the specifications of a program. However, verifying programs by using test cases has the difficuly of having to create all possible cases and the time cost problem of generation. Checking specifications of a program automatically is an important theme and there exists much research about it especially concerning SAT and SMT solvers [9]-[11].

The Software Analysis Workbench (SAW) is an open-source formal verification tool developed by Galois [3]. The SAW tool automatically generates a formal model from the byte code of a target function and by using symbolic execution, it can verify whether the model satisfies a verification property for any set of inputs and finds a counterexample if it does not.

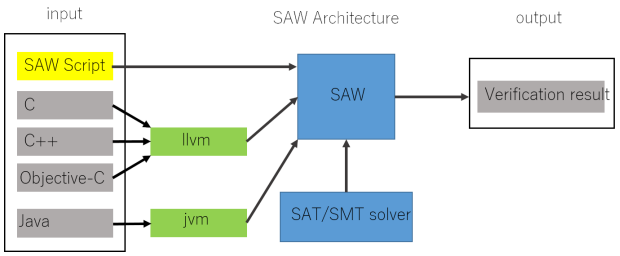


Figure 1: SAW Architecture

Users can define verification properties by using the special SAWScript scripting language which we explain in the next section.

The SAW architecture is shown in Fig. 1. SAW can analyze Low Level Virtual Machine (LLVM) [12], Java Virtual Machine (JVM) [13] byte code, and Cryptol [14], [15]. High level languages like C or Java must be compiled before performing SAW verification.

2.3 SAWScript

According to [2], "the process of model generation and transformation in SAW, and the interaction with third-party proof tools, is coordinated by a scripting language called SAWScript" and users can specify a .saw file to control or command SAW execution, e.g., extract target functions from .bc or .class files, define verification properties, generate SMT expressions and prove them by solvers.

We explain the SAW verification procedure with a simple example. We prove the function `oldJavafunc` in Listing 1 returns $2x$ when it receives x as any integer type. The SAWScript for this verification is shown in Listing 2. First, we describe explicitly the verification properties and which SMT solver to use in the JavaSetup block. In this case, the java_var command in the second line generates integer type symbolic inputs to correspond to the Java variable x as $java\_input$. The java_return command in the third line defines the expected return value of the target function as $2java\_input$. The java_verify_tactic command in the fourth line specifies the SMT solver to use. Next, the load .class file operation is indicated by the java_load_class command in the sixth line as $java\_mul$. Finally, we verify whether or not `oldJavafunc` of the Mul class satisfies the verification properties by the java_verify command in the seventh line. If the verification is valid, z3 proves that `oldJavafunc` the verification property described in the JavaSetup block and `newJavafunc` fulfill expression (2). Thus, `oldJavafunc` always returns $2 * x$ for any input x when x is an integer type.

$$\begin{aligned} &\forall java\_input : oldJavafunc(java\_input) \\ &\quad = 2java\_input \\ &\wedge java\_input \in \mathbb{Z}_{32}, java\_input = x \\ &\vdash \forall x \in \mathbb{Z}_{32} : oldJavafunc(x) = 2x \end{aligned} \tag{2}$$

Listing 1: Java Mul Class

```
class Mul{
  int oldJavafunc(int x){
    return x + x;
  }
}
```

Listing 2: SAWScript for Java Mul Class

```
let java_spec : JavaSetup () = do {
    java_input <- java_var "x" java_int;
    java_return {{java_input*2}};
    java_verify_tactic z3;
};
java_mul <- java_load_class "Mul";
java_verify java_mul "oldJavafunc" [] java_spec
    ;
```

2.4 SAT/SMT Solvers

A SATisfiability problem (SAT) searches for all combinations of boolean variables that make a propositional formula true [16]. In particular, programs that solve CNF propositional formulas automatically are called SAT solvers.

We explain the process of proving propositions using SAT solvers. If a propositional statement is a tautology, it is satisfiable for any status of its boolean variables. That is, the negation of the propositional logic is UNSAT for any combination of its boolean variables. Thus, proving that the negation of a propositional logic leads to UNSAT means that the propositional logic is a tautology.

There are many proposed SAT solvers for quickly solving propositional logic. However, SAT solvers handle only propositional logic and do not have the definitions of background knowledge like in predicate logic [17]. Satisfiability Modulo Theories (SMT) solvers have been proposed to resolve this problem. In addition to the definitions of predicate logic, SMT solvers can define integer types, arrays, etc. that are used in recent computer science and allow users to describe propositions abstractly and effectively.

SAW supports ABC [18], Boolector [19], CVC4 [20], MathSAT [21], Yices [22], and Z3 [23]. In this work, we used all of the above SMT solvers except for Boolector which does not run on Windows.

2.5 Problems with Handling Class Objects

The SAW uses the prove command to verify equivalence between two functions by comparing the formal models of the semantics of the two functions that are extracted by the java_extract command. This method can easily verify equivalence of the two functions from intermediate code. However, since this method does not perform assertion checks, we cannot find a counterexample regarding the two functions that has the same bugs. Thus we need to devise a specification based method to check equivalence. Another problem of this method is that the java_extract command cannot model functions that have array type arguments and we need to devise a new method to do this.

The SAW tool uses the java_verify and llvm_verify commands to prove equivalence between a C or Java function and a specification. The java_verify command is used for proving the equivalence between JVM code and a specification that users describe in the JavaSetup block of a SAWScript. The llvm_verify command and LLVMSetup block are for LLVM code. In the JavaSetup block, users can create variables that show the entire set of values of inputs by the java_var command and use it as a verification property. The java_var command can handle array type variables, however when a variable is declared, the concrete size of the array must be given by a parameter. Thus, SAW verification is valid only for an array specified with a given size in SAWScript. Due to the static symbolic execution of SAW, increasing of the size of an array leads to a state explosion.

In this work, we want to verify C and Java functions that handle data structures as arguments. In C, data structures are defined by struct, thus, users need to define struct variables as a verification property in SAWScript. In this case, verification can be performed by allocating memory spaces for data structures by the llvm_alloc command and assigning their heap values by the llvm_points_to command. However, SAW currently has no command to assign symbolic variables for fields of class objects that are allocated by the jvm_alloc_object command. Thus, we need to devise a method to assign symbolic variables and to perform SAW verification on Java functions.

3 PROPOSED METHOD

3.1 Equivalence Checking by Using SAW

In this section, we first explain how to perform an equivalence verification with the prove command using the two functions shown in Listing 3. The SAWScript is shown in Listing 4. The java_extract command in the second and third lines extract Java functions from the .class file and create formal models that show these semantics. The fourth line defines a verification property that states the return values of the two formal models are equivalent for any argument x. The fifth line verifies the property with the Z3 solver by the prove command which verifies the given assertion and generates a counterexample if it is invalid. In this case, SAW does not find any counterexamples. Thus the two functions are equivalent.

The prove command provides an easy method for checking the equivalence of two functions. However, the java_extract command cannot model functions that have indefinite size arguments like array types because this command generates a formal model by performing symbolic simulation. When a function that has an array type argument is assigned to the java_extract command, SAW will terminate. This method has a demerit that the prove command will only check the equivalence of two functions regardless of whether or not these functions satisfy the required specification.

We introduce a method to verify equivalence of functions that have array type arguments by employing an assertion check with the verify command. The conceptual diagram is shown in Fig. 2. In this method, users describe appropriate properties that target functions need to satisfy in the SAWScript and perform verification with the java_verify or llvm_verify commands for each function. When two functions satisfy the same properties, in other words, SAW does

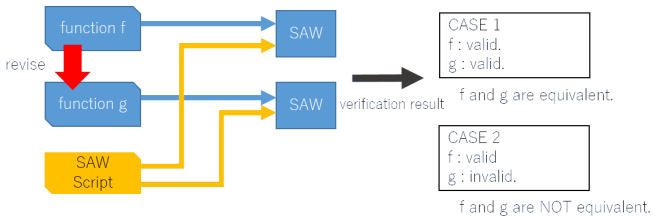


Figure 2: Equivalence checking by SAW

not find a counterexample, the two functions are judged as equivalent.

Listing 3: Revised Java Mul Class

```
1  class Mul{
2    int oldJavafunc(int x){
3      return x + x;
4    }
5
6    int newJavafunc(int x){
7      return x << 1;
8    }
9  }
```

Listing 4: Equivalence Verification with the prove Command

```
1  java_mul <- java_load_class "Mul";
2  old <- java_extract java_mul "oldJavafunc"
       java_pure;
3  new <- java_extract java_mul "newJavafunc"
       java_pure;
4  let thm = {{ \x -> old x == new x }};
5  result <- prove z3 thm;
6  print result;
```

We show a concrete verification example using the target code shown in Listing 3 and the SAWScript is shown in Listing 5. The JavaSetup block of the SAWScript is the same as in Listing 2. Line 7 verifies `oldJavaFunc` and the Line 8 verifies `newJavaFunc`. Both verifications are valid, thus these functions are equivalent meaning that they have the same property of returning $2x$ for any integer type argument x.

The merits of this method are shown below. By using the verify command, users can verify not only equivalence between functions but also equivalence between target functions and specifications. The verify command can handle functions that have array type arguments by defining their sizes in SAWScript as well as reuse verified specifications for other verifications. This is effective in shortening verification times for complex functions.

Listing 5: SAWScript for Revised Java Mul Class

```
1  let java_spec : JavaSetup () = do {
2      java_input <- java_var "x" java_int;
3      java_return {{java_input*2}};
4      java_verify_tactic z3;
5  };
6  java_mul <- java_load_class "Mul";
7  java_verify java_mul "oldJavafunc" [] java_spec
       ;
8  java_verify java_mul "newJavafunc" [] java_spec
       ;
```

If the programming languages that implement the functions are different, the method for proving equivalence requires an additional step since we also need to prove that the verification properties of both program functions are equivalent. We created a C program shown in Listing 6 by revising Listing 1 and the SAWScript for verifying it is shown in Listing 7. In the Java verification, SAW uses the java_int command when creating an integer type symbolic input for Java verification, llvm_int for C. C verification uses LLVMSetup to describe a verification property, whereas Java uses JavaSetup. This is due to the difference of the compiling methods between LLVM and JVM. Therefore, we cannot use the verification property shown in Listing 5 for functions written in a language other than Java and we need to check that the verification properties have the same inputs/outputs in each range. In this case, we defined $\mathbb{Z}_{32}$ that the two functions receive as expression (3).

$$\begin{aligned} \forall x \in \mathbb{Z} \land -2147483648 \leqq x \leqq 2147483647 \\ \Leftrightarrow x \in \mathbb{Z}_{32} \end{aligned} \tag{3}$$

It is apparent that ints of both languages are integer types from their specifications [24], [25]. The range of values changes according to the number of bits of the OS or environment of the compiler. The min/max values of the integer type can be checked by using limits.h for C and Integer.MAX_VALUE for Java. The environment of the experiment is explained in Section 5. As a result, the ranges of int of both languages satisfy expression (3). Thus, we consider that the int of Java and C are the same.

As a result of SAW verification, we confirmed that `newCfunc` satisfies the verification property of Listing 7. Therefore, expression (4) holds and the two functions satisfy the definition of equivalence.

$$\begin{aligned} \forall java\_input : oldJavafunc(java\_input) = 2java\_input \\ \land \forall c\_input : newCfunc(c\_input) = 2c\_input \\ \land java\_input = c\_input \in \mathbb{Z}_{32} \\ \vdash \forall x \in \mathbb{Z}_{32} : \\ oldJavafunc(x) = newCfunc(x) = 2x \end{aligned} \tag{4}$$

Listing 6: C Mul Function

```
1  #include <stdint.h>
2
3  int newCfunc(int x){
4    return x << 1;
5  }
```

Listing 7: SAWScript for C Mul Function

```
1  let c_spec : LLVMSetup () = do{
2    c_input <- llvm_var "x" (llvm_int 32);
3      llvm_return {{ c_input*2 : [32] }};
4      llvm_verify_tactic z3;
5  };
6  c_mul <- llvm_load_module "mul.bc";
7  llvm_verify c_mul "newCfunc" [] c_spec;
```

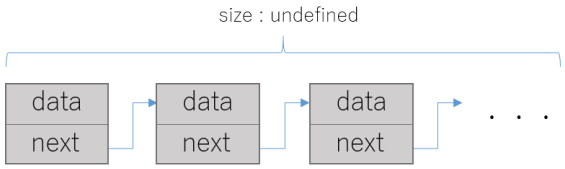


Figure 3: Linear List with Unspecified Size

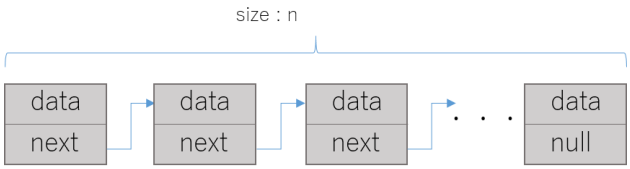


Figure 4: Linear List with Specified Size

3.2 Bounded Model Checking

The java_var command can handle array type variables by using the java_arrray command. However, when users declare array type variables in SAWScript, the concrete size of an array must be given by a parameter. This means there is no method to verify the properties of target functions valid with any array size. Owing to the static symbolic execution of the SAW, increasing the size of data leads to the verification failing without completing the recursion and ending in a state explosion.

In this paper, in order to solve this problem, we verify the data of such structures using the bounded verification method. The bounded verification method verifies that the specified verification property is satisfied for all states obtained by state transitions from the initial state to a certain given number n, using the assumption that the execution of a program is regarded as a state transition [5]. In general, bounded verification methods are realized by using an unfolding technique for a finite number of application steps such as loops. In this paper, we define n to be the size of data structure elements to be given to the function to be verified as shown in Fig. 4.

3.3 Introduction of Helper Functions to SAWScript

The SAW generates symbolic states from intermediate code and looks for combinations of input and output that can disable assertions in SAWScript. Thus, to verify functions that have properties that are difficult to describe in SAWScript, users need to devise alternative methods. We consider for example Java functions that handle linear lists that are defined by class objects. The SAW 0.2 does not have a method to assign symbolic variables to fields of allocated class objects. Thus, the SAW tool cannot directly verify Java functions that handle class object type arguments.

To solve this problem, we devised a method of using helper functions. Helper functions are defined as wrapper functions that call the target functions. Helper functions absorb differences between properties of the target functions and assertions that can be described in SAWScript and make verifying possible. Introducing helper functions to SAW has a merit of being able to perform verifications without changing any code of the target functions.

A simple example of verifying with a helper function is shown in Listings 8 and 9. The target function returns the value of a field x of a class object that receives it as an argument. The argument of the target function is a class object, thus we create a helper function called `testme` which receives an integer type argument and creates an instance for the target function. The helper function will return the return value of the target function, if the target function satisfies the property shown in Listing 9. In this case, the helper function resolves the problem of SAWScript mentioned above by allocating values in it.

Listing 8: Example of Helper Function

```
1  public class Helper {
2    int x;
3
4    Helper(int x){
5      this.x = x;
6    }
7
8    static int target(Helper foo){
9      return foo.x;
10   }
11
12   int testme(int x){
13     Helper help = new Helper(x);
14
15     return target(help);
16   }
17 }
```

Listing 9: SAWScript for Helper Function

```
1  let java_spec : JavaSetup () = do {
2    java_input <- java_var "x" java_int;
3    java_return {{java_input}};
4    java_verify_tactic z3;
5  };
6
7  java_mul <- java_load_class "Helper";
8  java_verify java_mul "testme" [] java_spec;
```

4 EXPERIMENTS

The following two research questions were established for conducting the evaluation experiments.

RQ1 With the proposed method, can we use SAW to evaluate the behavior of Java functions that handle recursive structures?

RQ2 How does the verification time change depending on the solver used and the size of input data?

In order to investigate these questions, we implemented two programs that deal with linear data list structures and binary tree structures, which are typical for handling recursive data structures. Unlike C, Java generally uses classes to implement data structures.

Due to the limitation of SAW, we use a single Java class to implement the data structures.

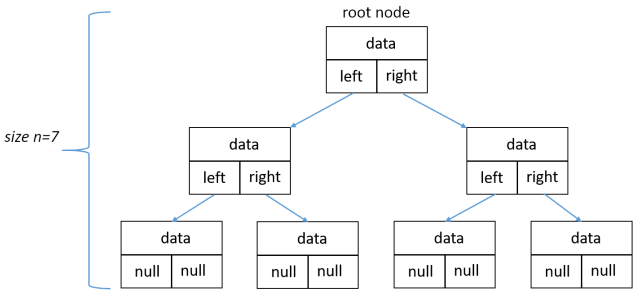


Figure 5: Perfect Binary Tree with size $n = 7$

4.1 Binary Trees

We performed SAW verification for Java functions that can receive binary trees and return the sum of all of the nodes. A binary tree is a data structure that has nodes defined recursively. The example in Fig. 5 shows a perfect binary tree with size $n = 7$.

In this paper, all nodes of a binary tree are Node objects with an integer type variable and two Node objects as their fields. For convenience, we call the nodes "right" and "left" and the size of a binary tree "n".

First, we created a definition of a binary tree shown in Listing 10 by using the Alloy Analyzer [26]. Only one root node exists and it is defined as a subset of Node. All the nodes have lone relationships with other nodes to "right" and "left" and a relationship with Value. The Value signature is $\mathbb{Z}_{32}$. The Alloy Analyzer has an Int signature of integers, however, its range is from -32 to +31. We consider that each node in a binary tree has a value, thus we defined this abstractly by using a Value signature. If data structure S satisfies the constraint below, it is called $S \in Binary\ Tree$.

- Each Node has only one variable.
- All of the Nodes and the Values are able to be referred to from the Root node.
- Each Node has only one route from the Root node.
- Nodes do not loop.

Listing 10: Alloy Code for Binary Trees

```
1  sig Node {
2    val: one Value,
3    left: lone Node,
4    right: lone Node
5  }
6  one sig Root extends Node {}
7  sig Value {}
8
9  fact {
10   #Node = #Value
11   all v: Value, r: Root | v in r.*(right + left
         ).data
12   no n: Node, n': n.right.*(right + left), n'': 
          n.left.*(right + left) | n' = n''
13   all n: Node | n not in n.^(right + left)
14 }
```

The Java program that handles a binary tree is shown in Listing 11. The Node and Value signature of the Alloy script

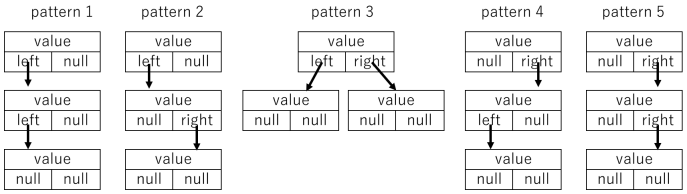


Figure 6: All Patterns of Binary Trees with size $n = 3$

corresponds with the objects of this class and the var in the second line. Generally, a node class is defined in another class, however, we use a single class to simplify the SAW verification in this case.

We consider the case in which we refactored `oldFunc` to create `newFunc` and we need to prove that these functions are equivalent.

When each of the functions receives the top node of a binary tree, they return the sum of all of the values in it, but their algorithms are different. Their return values are integer types, however, the arguments of both are defined as class objects. Since it is difficult to define a class object in SAWScript, we created a helper function based on the proposed method and verified it.

We explain the behavior of the helper function `testme`. When it receives an integer type array, it generates three class objects of `JavaBTree`.

The node2 object is assigned to node1.left and node3 is assigned to node1.right. The data structure generated by the `testme` function is a perfect binary tree with size $n = 3$ that satisfies the definition of a binary tree shown in Listing 10. The `helperFunc` calls either of the target functions as its return value.

The possible binary tree structures with size $n = 3$ are the five patterns shown in Fig. 6. The structure of the binary tree generated by `testme` equals Pattern 3. To say that the target function satisfies the verification property with size $n = 3$ we need to check for all patterns. Therefore, we created and proved helper functions corresponding to each of the binary tree structures. Since implementation is possible by simply switching the node indicated by the pointers, we omit the source code here.

We explain the SAWScript shown in Listing 12. Since the size of the array that `testme` assumes is 3, we defined an integer type array with size $n = 3$ as input and the sum of its elements as output in the SAWScript. The same SAWScript is used when the size of the binary tree is 3. The size of the array increases or decreases based on the size of the binary trees that `testme` assumes.

Listing 11: Java Program for Pattern 3

```
1  public class JavaBTree{
2    int val;
3    JavaBTree left;
4    JavaBTree right;
5
6    JavaBTree(int val){
7        this.val = val;
8        left = null;
9        right = null;
```

```
10    }
11
12    int oldFunc(JavaBTree now){
13      if(now != null && now.right != null && now.
            left != null){
14        return now.val + oldJavaBTreeFunc(now.
              right) + oldJavaBTreeFunc(now.left);
15      }else if(now != null && now.right == null
            && now.left != null){
16        return now.val + oldJavaBTreeFunc(now.
              left);
17      }else if(now != null && now.right != null
            && now.left == null){
18        return now.val + oldJavaBTreeFunc(now.
              right);
19      }else if(now != null && now.right == null
            && now.left == null){
20        return now.val;
21      }else{
22        return 0;
23      }
24    }
25
26    int newFunc(JavaBTree now){
27      if(now == null){
28        return 0;
29      }
30      return now.val + newJavaBTreeFunc(now.right
            ) + newJavaBTreeFunc(now.left);
31    }
32
33    int testme(int[] x){
34      JavaBTree node1 = new JavaBTree(x[0]);
35      JavaBTree node2 = new JavaBTree(x[1]);
36      JavaBTree node3 = new JavaBTree(x[2]);
37      node1.left = node2;
38      node1.right = node3;
39
40      return oldFunc(node1);
41      //return newFunc(node1);
42    }
43  }
```

Listing 12: SAWScript for Java BTree Program

```
1  let linear_spec : JavaSetup () = do {
2    ary <- java_var "x" (java_array 3 java_int);
3    java_return {{ ary@0 + ary@1 + ary@2 }};
4    java_verify_tactic yices;
5  };
6  linear_java <- java_load_class "JavaBTree";
7  java_verify linear_java "
       helperFunc_size3_pattern3" [] linear_spec;
```

We performed the following two experiments.

- SAW verification for all binary tree structures with size $1 \leq n \leq 3$. The numbers of possible binary tree structures are 1 when size $n = 1$, 2 when size $n = 2$, and 5 when size $n = 3$. The verification result is shown in Section 6.1.1.
- SAW verification for all perfect binary trees with size $1 \leq n \leq 63$ to investigate the gains of verification time by size. The verification result is shown in Section 6.1.2.

4.2 Linear Lists

We performed SAW verification for C and Java functions that receive linear lists and return the sum of all of the nodes.

A linear list is a data structure that has nodes defined recursively. The example in Fig. 7 shows a linear list with size $n = 3$. In this paper, all nodes of linear lists are Node objects and they have an integer type variable and one Node object as their fields. For convenience, we call nodes indicated by the pointer as "next" and the size of a linear list as "n".

As with the binary trees, we created a definition of a linear list shown in Listing 13 by using the Alloy Analyzer. Only one top node exists and it is defined as a subset of Node. All nodes have lone relationships with other nodes as "next" and a relationship with Value. The Value signature is taken as $\mathbb{Z}_{32}$ and if the data structure S satisfies the constraint below, it is called $S \in Linear\ List$.

- Each Node has only one variable.
- All of the Nodes and the Values are able to be referred to from the Top node.
- Nodes do not loop.

Listing 13: Alloy Code for Linear Lists

```
1  sig Node {
2    val: one Value,
3    next: lone Node
4  }
5    one sig Top extends Node {}
6    sig Value {
7  }
8
9  fact {
10   #Node = #Value
11   all v: Value, t: Top | v in t.*(next).val
12   no n: Node | n in n.^next
13 }
```

4.2.1 Java Linear List

A Java program for defining a linear list class is shown in Listing 14. We performed SAW verification for all linear list structures with size $1 \leq n \leq 10$. The verification result is shown in Section 6.2.1. As described above, we implemented the program in a single Java class for simplicity since we are focusing on a single function.

We assume that we need to prove the property of the function `javaLinearFunc`. When this function receives the top node of a linear list, it returns the sum of its data by calling itself recursively. The data type of the argument of the target function is a class object of `JavaLinear`, which is difficult to define in SAWScript as input.

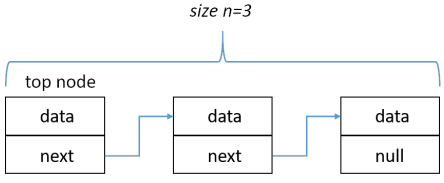


Figure 7: Linear List Structure

Thus, we create `helperFunc` based on the proposed method. This function generates a linear list with size $n = 3$ that satisfies expression (5). Since the function structure is the same as Listing 11, we omit the explanation.

$$now = node1(x[0], node2(x[1], node3(x[2], null))) \quad (5)$$

As a result, all possible linear list patterns that can be constructed with size $n = 3$ can be completed with one helper function. This is a benefit of symbolic execution. The SAWScript is shown in Listing 15.

Listing 14: Java Linear class

```
1  public class JavaLinear{
2    int val;
3      JavaLinear next;
4
5      JavaLinear(int val){
6        this.val = val;
7        next = null;
8      }
9
10   int javaLinearFunc(JavaLinear now){
11     if(now.next != null){
12       return now.val + javaLinearFunc(now.next)
             ;
13     }else{
14       return now.val;
15     }
16   }
17
18   int helperFunc(int[] x){
19     JavaLinear node1 = new JavaLinear(x[0]);
20     JavaLinear node2 = new JavaLinear(x[1]);
21     JavaLinear node3 = new JavaLinear(x[2]);
22     node1.next = node2;
23     node2.next = node3;
24     return javaLinearFunc(node1);
25   }
26 }
```

Listing 15: SAWScript for Java Linear Program

```
1  let java_spec : JavaSetup () = do {
2    ary <- java_var "x" (java_array 3 java_int);
3    java_return {{ary@0 + ary@1 + ary@2}};
4    java_verify_tactic mathsat;
5  };
6  linear_java <- java_load_class "JavaLinear";
7  java_verify linear_java "helperFunc" []
       java_spec;
```

4.2.2 C Linear List

We revised the previous Java program of Listing 14 to a C program shown in Listing 16. We performed SAW verification for all linear list structures with size $1 \leq n \leq 10$. The verification result is shown in Section 6.2.2. A node is defined by using struct, has an integer type variable and a pointer to the next node. The target function is `cLinearFunc` and its algorithm is the same as `javaLinearFunc` in Listing 14.

The SAWScript is shown in Listing 17. Defining a struct of C in SAWScript is easily possible by using the crucible_alloc and crucible_points_to commands.

The crucible_alloc command allows SAW to reserve memory space. In this experiment, we need to perform SAW verification on a linear structure with size $n = 3$. The description from Lines 6 to 8 let SAW reserve memory space of struct named node1, 2, and 3.

The crucible_points_to allows SAW to specify the destination of a pointer in the memory space. By using this, users can create data structures that the target function receives in SAWScript. The description from Lines 10 to 15 determines the specification of node1: its field is val1, next node is node2.

The crucible_execute_func in Line 29 allows SAW to determine the argument that the target function receives. In this case, the target function needs to receive the top node of a linear list.

Listing 16: C Linear Program

```
1  #include <stdint.h>
2  typedef struct NODE{
3    int val;
4    struct NODE *next;
5  }node_t;
6
7  int cLinearFunc(node_t *now){
8    if(now->next != '\0'){
9      return now->val +cLinearFunc(now->next);
10   }else{
11     return now->val;
12   }
13 }
```

Listing 17: SAWScript for C Linear Program

```
1  let linear_spec = do{
2    val1<-crucible_fresh_var "val1" (llvm_int 32)
         ;
3    val2<-crucible_fresh_var "val2" (llvm_int 32)
         ;
4    val3<-crucible_fresh_var "val3" (llvm_int 32)
         ;
5
6    node1<-crucible_alloc (llvm_struct "struct.
         NODE");
7    node2<-crucible_alloc (llvm_struct "struct.
         NODE");
8    node3<-crucible_alloc (llvm_struct "struct.
         NODE");
9
10   crucible_points_to node1 (
11     crucible_struct [
12     crucible_term val1,
13     node2
14     ]
15   );
16   crucible_points_to node2 (
17     crucible_struct [
18     crucible_term val2,
19     node3
20     ]
21   );
22   crucible_points_to node3 (
23     crucible_struct [
24     crucible_term val3,
25     crucible_null
26     ]
27   );
28
29   crucible_execute_func [node1];
30   crucible_return(crucible_term{{val1+val2+val3
         }});
```

```
31  };
32
33  print "llvm_load start";
34  linear_c <- llvm_load_module "liner.bc";
35  crucible_llvm_verify linear_c "cLinearFunc" []
        false linear_spec abc;
36  print "Done.";
```

4.2.3 Increases of Verification Time

As shown in Tables 1 and 2, timeout (T/O) verification times greater than 600 sec. did not occur with the verifications by CVC4, Yices, and Z3. Therefore, we performed additional tests with these solvers to investigate the increases in verification time with size. We generated test cases of linear lists with size $100 \leq n \leq 2000$ in 50 node increments and performed SAW verification. The results are shown in Section 6.2.3.

5 ENVIRONMENT OF THE EXPERIMENTS

The following summarizes the specifications of the PC used for the experiments.

- PC : TOSHIBA Dynabook T55/76MG
- OS : Windows 10 64-bit
- CPU : Intel Core i7 4510U
- RAM : 8GB DDR3

The versions of the tools used in the experiments are as follows.

- SAW : 0.2
- Clang : 3.7.1
- Java : 1.8.0_211
- ABC : 1.0.1
- CVC4 : 1.5
- Z3 : 4.6.0
- Yices : 2.5.4
- MathSAT : 5.5.1

We used the older LLVM 3.7.1, due to the SAW limitation.

6 EXPERIMENT RESULTS

In this section, we show the results of experiments for the various solvers. The numbers in the tables represent seconds. Verification times greater than 600 seconds were judged as timeouts (T/O).

6.1 Verification Results for Binary Trees

6.1.1 Verification Result with Size $1 \leq n \leq 3$

We performed SAW verification for all binary tree structures with size $1 \leq n \leq 3$. As a result, SAW verification for both functions was successful for all possible binary tree patterns in this range.

6.1.2 Verification Times Changing the Number of Elements

We performed SAW verification for all perfect binary trees with size $1 \leq n \leq 63$. The verification times in seconds are shown in Table 1.

6.2 Verification Results for Linear Lists

6.2.1 Verification Times for Java Linear List

We performed SAW verification for the `javaLinearFunc` function in Listing 14 by changing the size of the linear list. The verification times in seconds are shown in Table 2.

6.2.2 Verification Times of C Linear List Structures

The result of SAW verification for Listing 16 is shown below. The value of the counterexample is 1 more than INT_MAX.

> SolverStats solverStatsSolvers = fromList [”ABC”], solverStatsGoalSize = 45
>
> ————-Counterexample————-
>
> (“val2”,2147483648)
>
> (“val3”,2147483648)

When we changed the type of variable and function from int to uint32_t, SAW verification was successful. Thus, we performed a modified SAW verification for that program and the results are shown in Table 3.

6.2.3 Increases of Verification Times

The results of the experiment in Section 4.2.3 are shown in Figs. 6.2.3 and 6.2.3. Figure 6.2.3 shows the change in verification time of the Java program of Listing 14. No timeouts occured with any of the three SMT solvers and SAW verifications were successful, taking less than 30 seconds with size $n = 2000$.

Table 1: Verification Times for Binary Trees

size n	ABC	CVC4	MathSAT	Yices	Z3
1	0.203	0.340	0.319	0.358	0.315
3	0.212	0.454	0.415	0.526	0.423
7	133.626	0.462	T/O	0.504	0.431
15	T/O	0.494	T/O	0.536	0.469
31	T/O	0.635	T/O	0.616	0.571
63	T/O	0.634	T/O	0.706	0.571

Figure 6.2.3 shows the change of verification time of the C program of Listing 16. To perform SAW verification, we changed the type of variable and function from int to uint32_t. timeouts occur with CVC4 when size $n = 1400$, Yices and Z3 when size $n = 1350$.

7 DISCUSSION

7.1 Property Proving for Java Binary Tree Programs

When function `helperFunc` of Listing 11 satisfies the verification property shown in Listing 12, expression (6) holds.

$$\begin{aligned} &\forall x \in \mathbb{Z}_{32}\ Array, size = 3 : \\ &helperFunc(x) = \sum x = x[0] + x[1] + x[2] \end{aligned} \tag{6}$$

This means that when `helperFunc` receives any integer type array with size $n = 3$, it returns the sum of its elements. Property proving for the target function must be given by users.

According to the result of the experiment, when the return value of `helperFunc` is either target function, it satisfies the verification property. Thus, the return value of `helperFunc` is the same as the return value of `oldJavaBTreeFunc(node1)` and `newJavaBTreeFunc(node1)`. Also, the argument node1 that the target functions receive is the root node of any binary tree with $size\ n = 3$ and $Pattern = 3$ in Fig. 6. Thus, expression (7) holds.

$$\begin{aligned} &\forall x \in \mathbb{Z}_{32}\ Array \wedge size = 3 \\ &\wedge \forall node1 \in BinaryTree \wedge size = 3 \wedge Pattern = 3 : \\ &helperFunc(x) = oldJavaBTreeFunc(node1) \\ &= newJavaBTreeFunc(node1) \\ &= \sum x \end{aligned} \tag{7}$$

When instance creation occurs in `helperFunc`, an integer value is given for the val of each node as data by a constructor. From the description of Listing 11, expression (8) holds.

Table 2: Verification Times for Java Linear List Structures

size n	ABC	CVC4	MathSAT	Yices	Z3
1	0.186	0.205	0.312	0.202	0.221
2	0.081	0.206	0.177	0.224	0.214
3	0.278	0.205	0.301	0.208	0.211
4	1.184	0.208	6.573	0.216	0.219
5	5.978	0.212	114.327	0.219	0.213
6	54.713	0.214	332.668	0.214	0.214
7	144.602	0.211	T/O	0.215	0.228
8	T/O	0.221	T/O	0.214	0.228
9	T/O	0.252	T/O	0.238	0.266
10	T/O	0.224	T/O	0.225	0.231

Table 3: Verification Times for C Linear List Stuctures

size n	ABC	CVC4	MathSAT	Yices	Z3
1	0.512	0.462	0.447	0.463	0.459
2	0.568	0.508	0.539	0.540	0.505
3	0.766	0.506	0.617	0.494	0.505
4	1.823	0.514	6.478	0.495	0.606
5	7.234	0.513	105.291	0.502	0.502
6	58.827	0.521	315.102	0.505	0.509
7	134.956	0.514	T/O	0.507	0.525
8	256.111	0.526	T/O	0.508	0.516
9	T/O	0.523	T/O	0.509	0.529
10	T/O	0.525	T/O	0.512	0.526

$$\begin{aligned} &node1.val = x[0] \wedge node2.val = x[1] \wedge node3.val = x[2] \\ &\vdash \sum x = \sum_{i=1}^{n} node_n.val \end{aligned} \tag{8}$$

From expressions (7) and (8), it is proved that when both target functions receive the root node of any binary tree with $size\ n = 3$ and $Pattern = 3$, they return the sum of all node values.

According to the result of the experiment, SAW verifications succeeded for all possible binary tree structures with size $1 \leq n \leq 3$. The proof for the other structure patterns is possible in the same way, therefore expression (9) holds.

$$\begin{aligned} &\forall now \in BinaryTree \wedge 1 \leq n \leq 3 : \\ &oldJavaBTreeFunc(now) \\ &= newJavaBTreeFunc(now) \\ &= \sum_{i=1}^{n} node_n.val \end{aligned} \tag{9}$$

In other words, two functions being equivalent means that they return the sum of all elements of a binary tree when they receive any binary tree with size $1 \leq n \leq 3$.

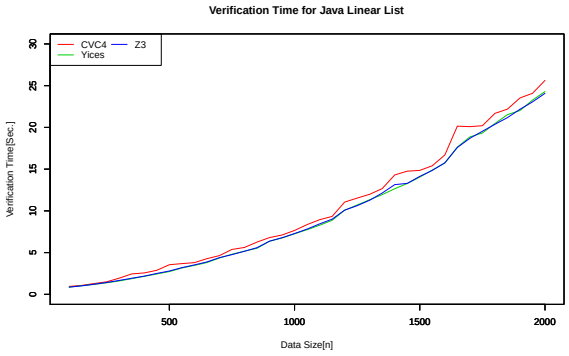


Figure 8: Verification Time for Java Linear List Program

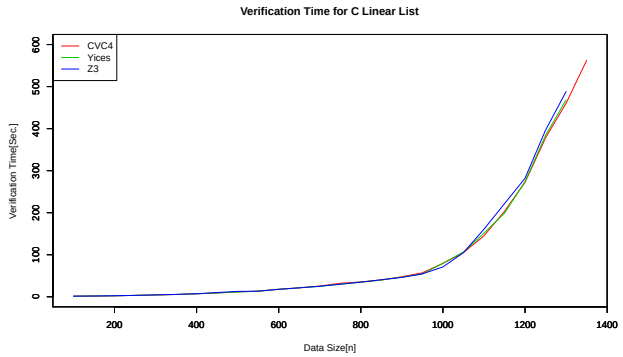


Figure 9: Verification Time for C Linear List Program

7.2 Property Proving for Java and C Linear List Programs

Since the number of possible linear list structure orderings does not increase with additional nodes, verification is easy compared with binary tree structures.

7.2.1 Java Linear List

The function `javaLinearFunc` of Listing 14 satisfies expression (10).

$$\begin{aligned} &\forall now \in LinearList \wedge 1 \leq size\ n \leq 10 : \\ &= javaLinearFunc(now) = \sum_{i=1}^{n} node_n.val \end{aligned} \tag{10}$$

In other words, when `javaLinearFunc` receives any linear list with size $1 \leq n \leq 10$, it returns the sum of its elements. The proof is similar to that in Section 7.1 and we omit it here.

7.2.2 C Linear List

As shown in Section 6.2.2, we changed the type of variable and function from int to uint_32 to enable a SAW verification. Thus, the definition of a linear list was changed to expression (11).

$$\begin{aligned} \forall f(x,n) : x \in \mathbb{Z}_{32^u} \wedge n ::= f(x,n)|null \\ \Leftrightarrow f(x,n) \in C\ Linear\ List \end{aligned} \tag{11}$$

It is possible for SAWScript to define a struct of C for C verification. Thus, expression (12) holds.

$$\begin{aligned} &\forall now \in C\ Linear\ List \wedge 1 \leq size\ n \leq 10 : \\ &= cLinearFunc(now) = \sum_{i=1}^{n} node_n.val \end{aligned} \tag{12}$$

7.2.3 Equivalence between Java and C Functions

The value of the counterexample shown in Section 6.2.2 was 1 more than INT_MAX and SAW verification was successful when we changed the type of variable and function from int to uint32_t. Thus, we assume that llvm_int 32 of SAWScript means uint32_t of C and we implemented an intentional bug of Listing 6 so that it returns x when x==0xFFFFFFFF. The type of its variable and function were changed into uint32_t. The result of SAW verification is shown below.

> Proof of return value failed.
> ————-Counterexample————-
> %x: 4294967295
> return value
> Encountered: 4294967295
> Expected: 4294967294
> Proof failed.

From this result, we determined that llvm_int 32 defines the symbolic variable of uint32_t. However, it is not clear why the SAW verification of the original program of Listing 6 was successful in spite of the types of the target function and its argument being int. Currently, we consider this behavior of SAW as a bug and we need to investigate further and clarify the specification of SAW.

The types of the target functions and data values of linear lists are $\mathbb{Z}_{32}$ in Java, $\mathbb{Z}_{32^u}$ in C. These differences make it difficult to prove the two functions are equivalent in a strict sense. However, it can at least be said that the two functions meet the same specification in that they return the sum of a linear list when they receive the top node of it.

7.3 Verification Time

In this section, we discuss the verification times. SAW automatically generates formal models based on target functions and descriptions of SAWScript and then solves them using SMT solvers. Thus, the time or ability of calculation is strongly influenced by the performances of the solvers.

There are pros and cons of the various SMT solvers. For example, in Table 2, the verification time with size $n = 5$ of Yices is over 100 sec. less than that of MathSAT. It is known as a demerit of SMT verification that estimating the verification time before performing verification is difficult due to the complex implementation of SMT solvers [27]. However, we can change which SMT solver to use in SAW just by editing the description of SAWScript and users can investigate the differences of verification time easily.

Due to the symbolic execution of SAW, we reduced the number of required test cases for verifying functions that handle recursive data structures to the number of possible patterns of data structures.

The number of possible binary tree structures with size n is shown in expression (13) and is called a Catalan Number[28].

$$P(n) = \frac{(2n)!}{(n+1)!n!}(n \geq 0) \tag{13}$$

This shows that the time complexity of verifying a binary tree function is $O(n!)$. Table 1 shows that Z3 verifies a perfect binary tree with size $n = 15$ in 0.469 sec., however, the

total number of possible binary tree patterns with size $n = 15$ is about 9.7 million. In spite of using symbolic execution, verifying a function with recursive data structures still needs enormous time. Thus, we need to devise a method that verifies more effectively.

7.4 Threats to Validity

7.4.1 Proposed Method

In this paper, we proposed a method to perform SAW verification for functions that receive class objects by using helper functions. This method is useful in defining arguments of the target function that cannot be defined in SAWScript. However, SAW can only verify the property of the helper functions, which means users need to prove properties of the target function manually like in Section 7.1. This dependency is a threat to internal validity and a way of generalizing the method is needed.

7.4.2 Definition of Equivalence

It is obvious that there are various requirements for the definition of equivalence [29], e.g., time complexity, readability, requests of memory space, and power consumption. However, in this paper, we adopted the description in [1] as the definition of refactoring and expression (1) as the definition of equivalence between two program functions. The reason for this is due to the specification of SAW. SAW automatically generates formal models of target functions and solves them by using SMT solvers. However, in SAWScript, users can define inputs and outputs as verification properties of the target function by this method, and we used SAW as a black-box unit testing tool. To evaluate the requirements shown above, SAW is not an adequate tool and users need to adopt other approaches.

8 CONCLUSION

In this work, we applied our proposed method to Java programs dealing with two types of recursive data structures and verified them using SAW to show that verification is actually possible. For the verification of linear lists, a comparison with the verification for C in our previous research was discussed.

We also proposed a method for simplifying the description of SAWScript and performing verification inductively by creating a helper function and defining a structure piece by piece in it. We confirmed that equivalence verification by SAW can be performed in the sense that two functions written in C and Java satisfy the same verification property.

In our future work, we will conduct further evaluation experiments and devise a method to verify the equivalence of the behavior of functions which have more complex algorithms like sorting algorithms. In addition, we would like to investigate the influence of language and algorithm on the solvers, and consider a method to find the optimal solver in equivalence verification.

ACKNOWLEDGEMENT

Funding from Mitsubishi Electric Corp. is gratefully acknowledged.

The research is being partially conducted as Grant-in- Aid for Scientific Research C (16K00094), C (17K00111) and A (19H01102).

REFERENCES

[1] M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, and Gamma E, *"Refactoring: Improving the Design of Existing Code"*. Addison-Wesley Professional, July 8, (1999).

[2] Dock A., J. Hendrix, B. Huffman, D. McNamee, and A. Tomb, "Constructing semantic models of programs with the software analysis workbench". In *11th Working Conference on Verified Software: Theories, Tools, and Experiments*, pages 56–72, (2016).

[3] K. Carter, A. Foltzer, J. Hendrix, B. Huffman, and A. Tomb, "Saw: The software analysis workbench". In *Proceedings of the 2013 ACM SIGAda Annual Conference on High Integrity Language Technology*, HILT '13, pages 15–18, (2013).

[4] R. Karashima, S. Harauchi, K. Okano, and S. Ogata, "Proposal and evaluation for equivalence checking for program with recursive data using saw". In *25th Workshop of Fundamentals of Software Engineering*, pages 91–96, (2018).

[5] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving". *Formal Methods in System Design*, 19(1):7–34, (2001).

[6] L. Tianhai, N. Michael, and T. Mana, "Bounded program verification using an smt solver: A case study". In *5th International Conference on Software Testing, Verification and Validation*, pages 101–110, (2012).

[7] G. Meszaros, *"xUnit Test Patterns: Refactoring Test Code"*. Addison-Wesley Signature Series. Addison-Wesley.

[8] S. Gulati and R. Sharma, *"Java Unit Testing with JUnit 5: Test Driven Development with JUnit 5"*. Apress, (2017).

[9] S. Gupta, A. Saxena, A. Mahajan, and S. Bansal, "Effective use of smt solvers for program equivalence checking through invariant-sketching and query-decomposition". In *Theory and Applications of Satisfiability Testing – SAT 2018*, pages 365–382. Springer International Publishing, (2018).

[10] Y. Sasaki, K. Okano, and S. Kusumoto, "A design of analyzer for java program using smt solver". *Foundation of Software Engineering XIX*, pages 33–38, Dec. (2012).

[11] D. R. Cok and J. R. Kiniry, "Esc/java2: Uniting esc/java and jml". In *Proceedings of the 2004 International Conference on Construction and Analysis of Safe, Secure, and Interoperable Smart Devices*, CASSIS'04, pages 108–128.

[12] C. Lattner and V. Adve, "Llvm: A compilation framework for lifelong program analysis & transformation". In *Proceedings of the International Symposium on Code*

Generation and Optimization: Feedback-directed and Runtime Optimization, pages 75–88, (2004).
[13] T. Lindholm, F. Yellin, G. Bracha, and A. Buckley, *"The Java Virtual Machine Specification, Java SE 8 Edition"*. Addison-Wesley Professional, 1st edition, (2014).
[14] R. L. Jeffrey and A. M. Brad, "Cryptol: high assurance, retargetable crypto development and validation". *IEEE Military Communications Conference*, Vol.2:820–825, (2003).
[15] L. Erkök and J. Matthews, "High assurance programming in cryptol". In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, pages 60:1–60:2, (2009).
[16] A. Biere, *"Handbook of Satisfiability*. IOS Press, (2009).
[17] K. Iwanuma and H. Nabeshima., "Smt: Satisfiability modulo theories". *Journal of Japanese Society for Artificial Intelligence*, 25(1):86–95, Jan. (2010).
[18] R. Brayton and A. Mishchenko, "Abc: An academic industrial-strength verification tool". In *Computer Aided Verification*, pages 24–40, (2010).
[19] N. Aina, P. Mathias, and B. Armin, "Boolector 2.0". *Journal on Satisfiability, Boolean Modeling and Computation*, 9:53–58, (2014).
[20] C. Barrett, Christopher L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli, "Cvc4". In *Computer Aided Verification*, pages 171–177, (2011).
[21] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani, "The mathsat5 smt solver". In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 93–107, (2013).
[22] B. Dutertre, "Yices 2.2". In *Computer Aided Verification*, pages 737–744, (2014).
[23] L. de Moura and Nikolaj Bjørner, "Z3: An efficient smt solver". In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, (2008).
[24] J. Gosling, B. Joy, G. L. Steele, G. Bracha, and A. Buckley, *"The Java Language Specification, Java SE 8 Edition"*. Addison-Wesley Professional, 1st edition, (2014).
[25] ISO. *"ISO/IEC 9899:2011 Information technology — Programming languages — C"*. International Organization for Standardization, Geneva, Switzerland.
[26] D. Jackson, "Alloy: A lightweight object modelling notation". *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290, (2002).
[27] Y. Takano, H. Sakaji, and S. Sato, "Toward automatic selection and automatic tuning of smt solver using machine learning". In *"35th Japan Society for Software Science and Technology Annual Conference*, Aug. (2018).
[28] R. P. Stanley, *"Catalan Numbers"*. Cambridge University Press, (2015).
[29] K. Maruyama, S. Hayashi, N. Yoshida, and E. Choi, "Frame-based behavior preservation in refactoring". In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 573–574, Feb. (2017).

(Received December 10, 2019)
(Revised July 30, 2020)

Rin Karashima is a graduate student of Shinshu University. His areas of interest include formal verification and STAMP/STPA.

Kozo Okano received his BE, ME, and PhD degrees in Information and Computer Sciences from Osaka University in 1990, 1992, and 1995, respectively. From 2002 to 2015, he was an Associate Professor at the Graduate School of Information Science and Technology of Osaka University. In 2002 and 2003, he was a visiting researcher at the Department of Computer Science of the University of Kent in Canterbury, and a visiting lecturer at the School of Computer Science of the University of Birmingham, respectively. Since 2015, he has been an Associate Professor at the Department of Electrical and Computer Engineering, Shinshu University. His current research interests include formal methods for software and information system design. He is a member of IEEE, IEICE, and IPSJ.

Shinpei Ogata is an Assistant Professor of the Graduate School of Science and Technology in Shinshu University, Japan. He received a PhD from Shibaura Institute of Technology, Japan in 2012. His current research interests include model-driven engineering for information system development. He is a member of IEEE, ACM, IEICE, and IPSJ.

Satoshi Harauchi received BE and ME degrees in Information Sciences from Kyoto University in 1996 and 1998, respectively. Since 1998, he has been at the Advanced Technology R&D Center of Mitsubishi Electric Corporation and is currently interested in software engineering for social infrastructure systems. He is a member of IEICE and JSASS.

Toshifumi Sekizawa received his MSc degree in physics from Gakushuin University in 1998, and Ph.D. in information science and technology from Osaka University in 2009. He previously worked at Nihon Unisys Ltd., Japan Science and Technology Agency, National Institute of Advanced Industrial Science and Technology, and Osaka Gakuin University. He is currently working at College of Engineering, Nihon University. His research interests include model checking and its applications.

Regular Paper

Analytical Method using Geotagged Tweets Developed for Tourist Spot Extraction and Real-time Analysis

Masaki Endo[*], Munenori Takahashi[**], Masaharu Hirota[***],
Makoto Imamura[****], and Hiroshi Ishikawa[*****]

[*] Division of Core Manufacturing, Polytechnic University, Tokyo, Japan
[**] Graduate Student in Electro-info, Polytechnic University, Tokyo, Japan
[***] Faculty of Informatics, Okayama University of Science, Okayama, Japan
[****] School of Information and Telecommunication Engineering, Tokai University, Tokyo, Japan
[*****] Graduate School of System Design, Tokyo Metropolitan University, Tokyo, Japan
endou@uitec.ac.jp

Abstract – The popularization of social network services (SNSs) has made it possible to acquire large amounts of data in real time. For this reason, various studies are being conducted to analyze social media data and to extract information about real-world events. Among them, a salient advantage of analysis using positional information is that one can accurately extract events from target areas of interest. We propose real-time analysis of tourist information using a simple method that incorporates a moving average of geotagged tweet messages on Twitter (Twitter Inc.), a popular SNS. Nevertheless, social media data that include position information are few: the data might be insufficient for analysis. Therefore, we are assessing a real-time analytical method using appropriate data accumulated over a longer period of time. For this study, we detect tourist spots using a regional analysis method that uses location information from Twitter tweets and time-series changes. The experimentally obtained results demonstrate that our method is useful as a complement to our previously proposed moving-average method. As described herein, we propose a method of extracting tourist attractions from the accumulated tweets and report the results of analysis of the extracted destinations. Herein, we explain results of sightseeing spot analysis of cherry blossoms obtained using geotagged tweets from Tokyo.

Keywords: big data, location information, social media, time series, Twitter

1 INTRODUCTION

From everyday life, because of the wide dissemination and rapid performance improvement of various devices such as smartphones and tablets, vast amounts of diverse data are generated and transmitted via the internet. Social network services (SNSs) have become especially popular media because users can post data and various messages easily. Twitter (Twitter Inc.)[1], a popular SNS that provides a micro-blogging service, is used as a real-time communication tool. Numerous tweets are posted daily by vast numbers of users worldwide. Twitter is therefore a useful medium to obtain, from huge amounts of information posted by many users, real-time information corresponding to the real world.

By analyzing the information distributed by these SNSs, useful information is obtainable in real time. We are conducting research related to provision of tourist information to travelers. Therefore, this study specifically examines the provision of real-time sightseeing information.

Herein, we describe the provision of information to tourists using web contents. Such information is useful for tourists. However, providing timely and topical travel information entails high costs for information providers because they must update the information continually. Today, reliable information for local travel is demanded strongly not only by tourists, but also by local governments, tourism organizations, and travel companies, which are all burdened by the high costs of providing such information.

For the reasons explained above, ascertaining changes of information according to seasons and time zones of the tourism region and then providing current, useful, real-world information for travelers is important for the travel industry. Disseminating information using popular SNSs can be done, but organizations able to do such work are constrained by human resource and cost limitations. Analysis using an SNS can overcome these difficulties by providing useful data facilitating real-time information provision.

To accomplish this task, much research is currently underway to analyze SNS data. Research using Twitter is one branch of investigation. Tweet messages are short sentences from which a location can be inferred if a tweet includes a place name or a facility name. If such information is not included in the message, then identifying a location from a tweet might be difficult. Therefore, research is underway to use tweets with location information or tweets which give location information in the tweet itself. Geotagged tweets can identify places: they are useful for analysis. Nevertheless, few geo-tagged tweets exist among the total information contents of tweets. Therefore, analyzing all regions is not possible. We also use geotagged tweets to conduct research using information interpolation to infer positions around an area that are not specified by position information [2].

Currently, the authors are assessing a real-time analytical method that requires collection of temporal and spatial information from geotagged tweets over a period of time. This report presents this experimental approach to use small

amounts of information accumulated over longer periods. However, earlier methods are based on the assumption that the traveler knows the tourist spot because the analysis is based on a destination-specific analysis. Although this is effective in the case of familiar tourist spots, it is difficult to use this method in areas that are being visited for the first time or in places for which no prior knowledge exists. As described in this paper, we propose a method for extracting tourist attractions and for analyzing them in real time in areas with tourist attractions for which no prior information is known to the user. The proposed method is one way to help tourists to collect information related to sightseeing in areas for people with no prior information related to the sightseeing spots.

The remainder of the paper is organized as explained hereinafter. Chapter 2 presents earlier research efforts exploring this topic. Chapter 3 explains a real-time analytical method using data collected for a certain period. Chapter 4 describes experimentally obtained results for our proposed method, with related discussion. Chapter 5 presents a summary of the study contributions and indicates some expected avenues of future work.

2 RELATED WORK

Various studies are being conducted using SNS position information. Omori et al. [3] described a method of extracting geographical features such as coastlines using tags of photo-sharing sites with geotags. Sakaki et al. [4] assessed a method to detect events such as earthquakes and typhoons based on a study estimating real-time events from Twitter. By analyzing the Twitter text stream, Pratap et al. [5] explained a solution to optimize traffic control by incorporating earlier traffic analysis methodologies and complementary real-time social data in one analytical system. Various analytical methods have been proposed for analyzing SNS data using position information and time series information. However, those studies mainly address analysis of data for which large amounts of position information and time series information exist. Few research efforts have examined information using only a few data.

Some research efforts have examined visualization. Nakaji et al. [6] proposed the use of a geotagged and visual features of photographs. They suggested a way to select photographs related to a given real event from geotagged tweets. Their developed system can visualize real-world events on online maps. Through the GeoNLP Project [7], we are developing a geotagging system that extracts location descriptions such as place names and addresses included in natural language sentences. Offered as open source software, it provides metadata of places described in text. Although these studies are very useful for extraction of specific designated events and for analysis of preregistered places, further discussion must be held about automatic extraction of events and identification of new places.

In light of the efforts described above, the present study using geotagged tweets for places with small information amounts and new events and places represents a new approach. This research was conducted to achieve real-time identification of events and places in space–time space based on accumulated information and differences.

3 PROPOSED METHOD

This chapter presents a description of real-time analysis of position information and time series information as a target data collection method.

3.1 Data Collection

Here, we explain the data collection target for this research. Geotagged tweets distributed through Twitter are the collection target. The range of geotagged tweets includes the Japanese archipelago (120.0°E ≤ longitude ≤ 154.0°E and 20.0°N ≤ latitude ≤ 47.0°N) as the collection target. These data were collected using a streaming application programming interface (API) [8] provided by Twitter Inc.

Next, we describe the number of collected data. According to a report by Hashimoto et al. [9], among all tweets originating in Japan, only about 0.18% are geotagged tweets: they are rare among all data. However, the collected geotagged tweets number about 70,000, even on weekdays. On some weekend days, more than 100,000 such messages are posted, constituting about 423 million geotagged tweets from 2/17/2015 through 12/26/2018. For these analyses, we examined 19 million geotagged tweets from Tokyo.

3.2 Preprocessing

This chapter presents a description of preprocessing after data collection. Preprocessing includes reverse geocoding and morphological analysis, with database storage for data collected using the process.

Reverse geocoding was sufficient to identify prefectures and municipalities by town name using latitude and longitude information from individually collected tweets. For this process, we use a simple reverse geocoding service [10] available from the National Agriculture and Food Research Organization: e.g., (latitude, longitude) = (35.7384446°N, 139.460910°W) by reverse geocoding becomes (Ogawanishi-machi 2-chome, Kodaira-shi, Tokyo). In addition, based on latitude and longitude information of the collected tweets, data from the same place are accumulated. As data accumulate, the data obtained over time are saved in mesh form.

Morphological analysis divides the collected geo-tagged tweet morphemes. We use the Mecab™ morphological analyzer [11]. As an example, “桜は美しいです” (“Cherry blossoms are beautiful.” in English) is divisible into "(桜 / noun), (は / particle), (美しい / adjective), (です / auxiliary verb), and (。 / symbol)".

The preprocessing involves storing the necessary data based on the results of data collection, reverse geocoding, and morphological analysis process. Data used for this study are the tweet ID, tweet posting time, tweet text, morpheme analysis result, latitude, and longitude.

3.3 Analytical Method

This chapter presents a description of the method of real-time analysis using position information and time series information. The analytical method we proposed has the following two stages.

Stage 1. Extraction of places by fixed point observation

The steps in Stage 1 are the following.

1. After a user specifies the two points of latitude and longitude in the southeasternmost and northwestern regions of the rectangle to collect tourist attraction information, keywords to be extracted (e.g. cherry blossoms) are input.

2. The specified area is divided into 25 and 16 equal parts east–west and north–south, respectively.

3. Tweets in the mesh including keywords are extracted. Then the numbers of tweets for meshes are observed.

Stage 2. Analysis considering the time series based on Stage 1

The steps in Stage 2 are explained below.

1. Numbers of tweets observed per mesh in Stage 1 are analyzed by year and month. At this stage, we manually designate them according to the tourism keyword event. For example, for cherry blossoms, the analysis will be performed on a monthly basis.

2. An event will be determined as a seasonally appropriate tourist destination if it appears only at a specific time of year. Then, using our existing method [12], we make the spot a target for real-time analysis to ascertain whether it is currently in its best season.

Therein, Stage 1 is an estimate of the location derived from stationary observation. At such spots, even in places with few tweets, one can discover the location through long-term observation. This method accomplishes spot extraction by accumulating geotagged tweets including specific keywords over long periods at every latitude and longitude. Concretely, one selects an arbitrary area to be observed on the map. Then the selected area is divided equally into north–south and east–west meshes. Although another method of meshing using JISX0410 standardized regional mesh codes exists, an arbitrary area was selected and meshed by hand as an experiment. By long-term accumulation of numerous tweets within this meshed area over a long period, one can infer that the place is a tourist spot. Stage 2 is extraction of new spots using spot information accumulated during a long period as a baseline, by considering the time series, and by finding differences. Through analyses using these proposed methods, we aim to capture real-time changes in specific areas. This goal was achieved by checking the change of specific keywords in the chronological order of each year for the selection in Stage 1. With Stage 2, use of only those tweets with location information was sufficient to extract the spots automatically. The extracted spots can be judged in real time using the estimation method we have developed to date. Through analysis using these proposed methods, we aim to ascertain real-time changes in tourist spots in a specific region.

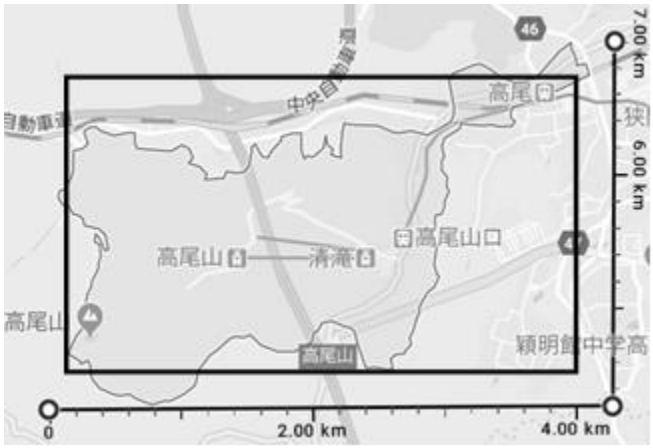


Figure 1: Target of Takao-machi, Hachioji City.

4 EXPERIMENTS

This chapter presents a description of a real-time analysis experiment that was conducted using the method explained in Chapter 3.

4.1 Dataset

Datasets used for this experiment were collected using streaming API, as described for data collection in Sec. 3.1. The data are geo-tagged tweets from Tokyo during 2/17/2015 – 7/29/2019, including about 48 million items. We use these data for experiments to assess the two methods.

4.2 Experiment Method

Experiments designed to assess the proposed method described in Chapter 3 are explained in Sec. 4.2.1 – Sec. 4.2.3.

4.2.1 Extraction of Places by Fixed Point Observation on Stage 1

We conducted a preliminary experiment to ascertain whether spots can be found from the collected tweets, or not. This experiment was conducted for Takao-machi, Hachioji, Tokyo: an area of about 4 km east–west and about 2.5 km north–south, as presented in Fig. 1. Experimentally obtained results described later are included within the thick frame depicted in Fig. 1. As a process for Stage 1, we used a mesh divided into 25 equal north–south mesh sections and 16 equal east–west mesh sections. For this area, we conducted an extraction experiment with the target word as "cherry blossom" in Japanese as "桜", "さくら", or "サクラ". In all, 65 tweets were found to include a target word.

4.2.2 Time Series Analysis of Stage 2

Spot extraction was performed using time series analysis of Stage 2 for the area shown in Sec. 4.2.1. This analysis is aimed at adjusting the units of the period in the future automatically according to event characteristics, to reflect monthly, weekly, and hourly scheduling. Cherry blossoms bloom once each year. Therefore, we applied a time-series analysis for each year and analyzed them by year. In addition, the analysis for Takao-machi, Hachioji City, Tokyo was conducted for each mesh section in Sec. 4.2.1.

4.2.3 Experiment in Tokyo

We conducted an experiment examining tweets from Tokyo to confirm the accuracy of the proposed method presented in Sec. 4.2.1 and Sec. 4.2.2. This experiment was conducted to

check the accuracy of the spots extracted by Sec. 4.2.1 and Sec. 4.2.2. For comparison, this time, the correct answer spots were 64 spots corresponding to cherry blossom viewing spots in Tokyo, which were listed in the Walker + 2019 edition of Cherry-blossom viewing in Japan. Then, we checked whether known tourist spots can be extracted using the proposed method. The following existing methods [12] were used to find the correct spots.

As a determination method, a 3D map was generated using a spreadsheet program (Excel 2016; Microsoft Corp.). A heat map was used to display the results. A square 100-m on a side was prepared as a spot point centered on the latitude and longitude reported by Walker +. A 1 km diameter circle was also prepared. Next, the following three spot determination methods were used.

1. Places indicated by many tweets are included in the square.
2. Places indicated by many tweets are included in the circle.
3. Places indicated by medium numbers of tweets are in the circle.

Judgment was made based on these three judgment criteria.

4.3 Experiment Results

This chapter presents results obtained from the experiments described in Sec. 4.2.1 – Sec. 4.2.3.

4.3.1 Experiment in Takao-machi, Hachioji, Tokyo

The distribution of geotagged tweets from Takao-machi, Hachioji, including cherry blossoms, as obtained from the experiments described in Sec. 4.2.1 and Sec. 4.2.2, are presented in Fig. 2 for 2017 and Fig. 3 for 2018. The interior area of the bold frame in Fig. 1 is described in the table: it is about 265 m measured east–west and about 85 m measured north–south. This area is obtained by dividing the maximum value and the minimum value of latitude and longitude into 25 and 16, portions respectively, for geotagging tweets. For the current experiment, the number of divisions was specified manually because the target area differs for each experiment. The more closely the color of the mesh section approaches black, the more data are associated with that area.

The tweet data extracted for this experiment were very few: 65 for the entire collection period. However, in 2017 and 2018, we confirmed tweets at JR Takao Station, Takao Yamaguchi Station, Takao Ropeway Station, and Takao Mountain. The correlation coefficient between the extracted spots in 2017 and 2018 was 0.769: high positive correlation was found. The correlation coefficient is calculated using equation (1).

$$\mathrm{Correl}(X-Y)=\frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum(X-\bar{X})^2\sum(Y-\bar{Y})^2}} \quad (1)$$

$$\mathrm{X}: Number\ of\ Tweets\ of\ 2017_{(latitude,longitude)}$$

$$\mathrm{Y}: Number\ of\ Tweets\ of\ 2018_{(latitude,longitude)}$$

Results of the time-series analysis by year for the data collection period in the experimental area are portrayed in Fig. 4, which shows that most of the tweets were collected in March and April. Therefore, one can identify the cherry blossom viewing related spots and analyze the cherry blossom season. Although some tweets are related to cherry blossoms during the season of autumn leaves, the tweets generally include messages about "cherry blossoms of the four seasons" and messages about seasons when people view cherry blossoms. Although accuracy improvement through noise removal is necessary, we limited the recommended seasons even for small numbers of tweets.

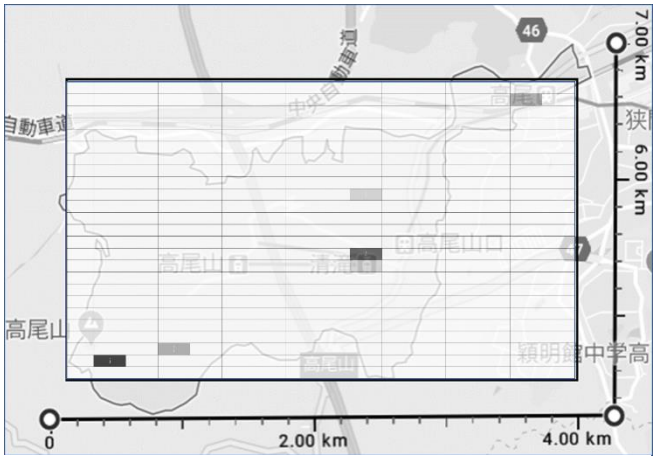


Figure 2: Number of Tweets including target words in Takao-machi in 2017.

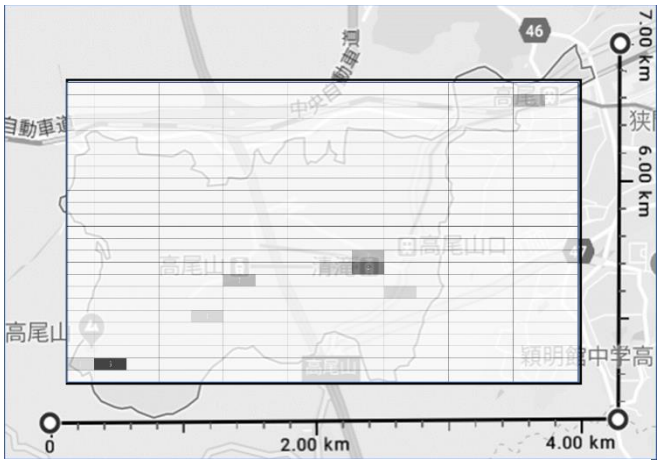


Figure 3: Number of Tweets including target words in Takao-machi in 2018.

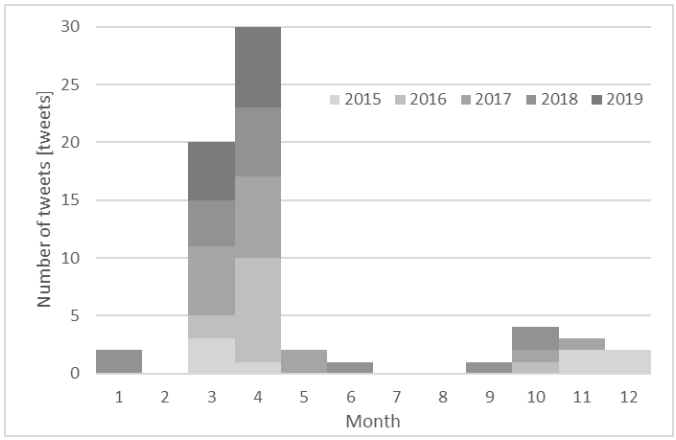


Figure 4: Results of the time-series analysis in Takao-machi.

Spot extraction is possible even with few data. Moreover, various spots can be extracted when using longer periods. Therefore, fixed point extraction of sightseeing spots is regarded as possible through continuing observation of geotagged tweets.

4.3.2 Spot Extraction Experiment Results in Tokyo

We present results obtained from the experiment in Tokyo for the method described in Sec. 4.2.3. This section presents results of spot extraction yielded using the proposed method with 64 correct spots in the 2019 version of Walker + cherry blossom viewing in Tokyo. Table 1 presents results found using the method described in Sec. 4.2.3: 1. A spot with many tweets is included in the square is “Contain Gray below"; 2. A spot with many tweets is included in the circle is “Gray within 1 km”; 3. The circle includes locations where the tweets are medium – "White within 1 km".

This result shows few spots for which tweets are concentrated in a square denoted as 1 because the latitude and longitude designated as spots in Walker + are assigned one point. The park has a larger area. Additionally, it is assumed that cherry blossoms in the park are not all at one point. Instead, they are distributed widely throughout the park. Cherry blossoms rarely exist only in a 100 m square area. Therefore, the spot area size can strongly affect results. For this reason, the numbers of spots found within a 1 km circle of 2 or 3 are greater. Some places were not extracted as spots by any determination method, probably because tweets are posted widely from many locations in the park. They might be few compared to those found for the entirety of Tokyo in a relative judgment based on the heat map. Results demonstrate that tweets were few in parks with low visibility. Moreover, extraction using the proposed method was difficult.

4.3.3 Spot Detection Rate in Tokyo

The spot detection rate in Tokyo is presented next. Table 2 presents a comparison between the 64 spots in Sec. 4.3.2 and the extraction results obtained using three determination methods. The spot detection rate is shown as equation (2).

$$\text{Spot Detection Rate} = \frac{\text{Matching}}{\text{Total Spots}} \qquad (2)$$

From these results, it was difficult to ascertain the correct data based on the latitude and longitude of one point, but results suggest that accuracy of 0.8 was obtainable within a 1 km range using the detection method. Setting of the park size range presents issues for future studies, but the capability of extracting major spots using the proposed method was confirmed. This result can engender automation of cherry blossom spot extraction using tweets.

Table 1: Extraction results for Tokyo obtained using the determination method

	Contains Gray below	Gray within 1 km	White within 1 km		Contains Gray below	Gray within 1 km	White within 1 km
Ark hills	0	0	0	Chidorigafuchi park	1	1	1
Asukayama park	1	1	1	Central park	0	0	0
Senzoku pond park	0	0	1	Tokyo midtown	1	1	1
Ikegami honmon temple	1	1	1	Toshimaen sakura festival	0	0	1
Inokashira gift park	1	1	1	Toneri park	0	0	1
Ueno gift park	1	1	1	Toyama park	1	1	1
Ukima park	0	1	1	Sakuragaoka park	0	0	1
Baigan temple	1	1	1	Sayama park	0	0	1
Tamagawadai park	0	1	1	Jindai botanical garden	1	1	1
Lake okutama	0	0	0	Hikarigaoka park	0	1	1
Otonashishinsui park	1	1	1	Yoyogi park	1	1	1
One Green road	0	1	1	Nishiarai park	0	0	1
Kamanofuchi park	0	0	0	Hibiya park	1	1	1
Former iwasaki garden	0	1	1	Hamarikyu gardens	0	1	1
Former shibarikyu garden	1	1	1	Hamura weir	1	1	1
Former furukawa garden	0	1	1	Fujimori park	1	1	1
Kiyosumi garden	0	1	1	Houmyou temple approach	1	1	1
Koishikawa korakuen	0	1	1	Mizumoto park	0	0	0
Koganei park	0	1	1	Myoujinsita park	0	0	0
Showa kinen park	0	0	1	Mukoujima hundred gaeden	0	0	0
Kotta river	0	0	0	Musashin park	0	0	1
Komazawa plympic park	0	1	1	Meijijinguugaien	0	1	1
Sun shine city	1	1	1	Meguro river	1	1	1
Shiotakouchitsutsumi	0	0	0	Roppongi mohri garden	1	1	1
Shiba park	1	1	1	Yaesu sakura street	1	1	1
Shakujii park	0	0	0	Yomiuri land	0	0	0
Shinjuku gyoen	1	1	1	Rikugien	1	1	1
Sumida park	1	1	1	Sotobori park	1	1	1
Sendaiborigawa park	0	0	0	Kinuta park	0	0	1
Zenpukuji river green	0	0	0	Tatsuminomori green park	0	0	1
Takiyama park	0	0	1	Harimazaka	1	1	1
Tama river embankment	0	0	1	Yasukuni shrine	1	1	1

Table 2: Spot detection rate results for Tokyo

	Contains Gray below	Gray within 1km	White within 1km
Matching	27	39	51
Total Spots	64	64	64
Spot Detection Rate	0.42	0.61	0.80

From these results, it was difficult to ascertain the correct data based on the latitude and longitude of one point, but results suggest that accuracy of 0.8 was obtainable within a 1 km range using the detection method. Setting of the park size range presents issues for future studies, but the capability of extracting major spots using the proposed method was confirmed. This result can engender automation of cherry blossom spot extraction using tweets.

4.3.4 Multiple Spots can be Dense

This section presents a description of the heat map results obtained when multiple spots are dense. As an example, Fig. 5 depicts the area around Kita-ku, Tokyo. Spots presented as circles in the figure are cherry blossom spots that were correct answer data. In the figure, A–E respectively stand for Kita-ku Chuo Park, Otonashi Shinsui Park, Asukayama Park, Former Furukawa Garden, and Rikugien Garden.

From Table 1, spots other than Kita-ku Chuo Park in A can be extracted as spots using the proposed method. Regarding A, the tweet exists in the circle, but it was not judged to be a Gray part or a White part because of the small number of tweets compared to those of Tokyo overall.

Additionally, tweets are widely distributed in places other than spots. Of these, tweet origins are most concentrated around stations such as Oji Station, Komagome Station, and Sugamo Station. This concentration is characteristic of areas near stations. Many people can tweet there while waiting for a train or a bus or when loitering around the station when shopping or eating meals. Therefore, when performing an analysis including tweets around the station, the number of excluded spots such as A can be expected to increase.

4.3.5 Time Series Analysis Results

This section presents results of estimation obtained after considering time series for spots that can be detected as cherry blossom spots. As an example, Meguro River results obtained for 2017–2019 are presented in Figs. 6–8. Meguro sightseeing spots can be detected every year, as described in the preceding section. The number of tweets varies every year, but cherry blossom tweets are readily detectable.

Figures 9–11 present results obtained from performing best-time estimation using the proposed method to assess tweets of each year. It tends to blossom at the end of March every year, but each year can produce a different outlook estimate. The accuracy of this best-time estimation result was confirmed using information related to the flowering and full bloom of the Meguro River in 2019, as shown for the Sakura channel of Weathernews as an example [13]. According to the site, the Meguro River in 2019 will bloom on March 21 and will be in full bloom on March 30. The estimation

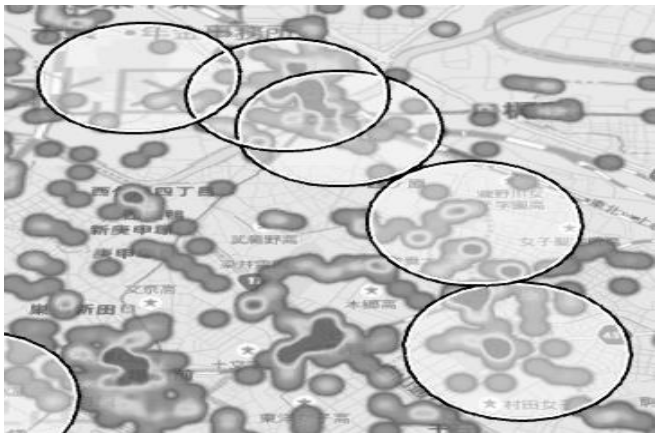


Figure 5: Example of multiple spot judgment results.

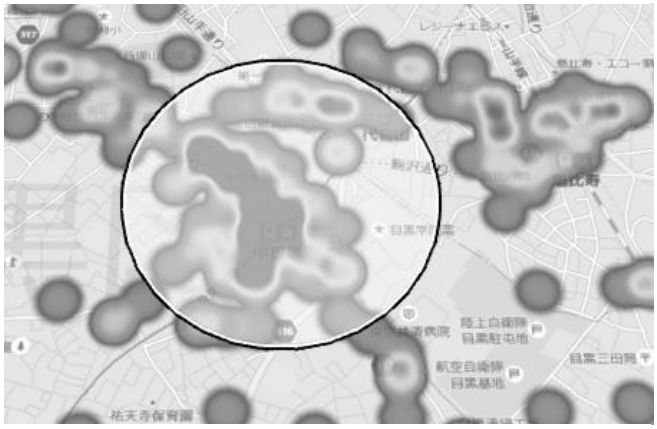


Figure 6: Meguro River judgment results of 2017.

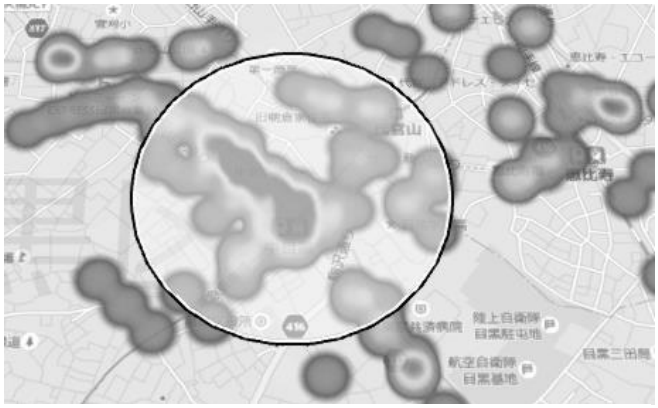


Figure 7: Meguro River judgment results of 2018.

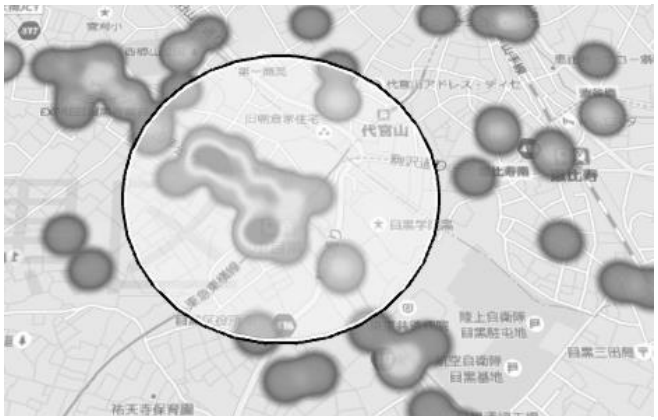


Figure 8: Meguro River judgment results of 2019.

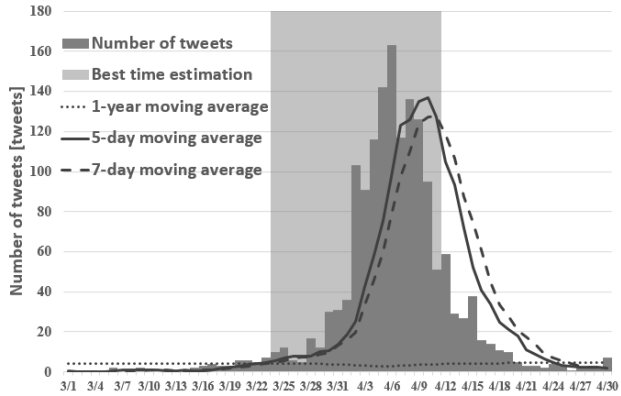


Figure 9: Estimation result of the cherry blossoms in Meguro River in 2017.

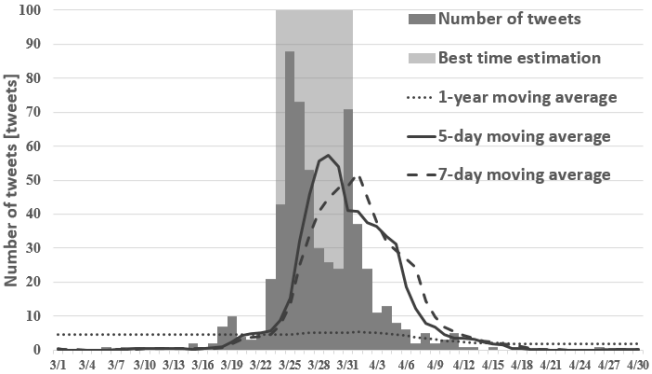


Figure 10: Estimation result of cherry blossoms in Meguro River in 2018.

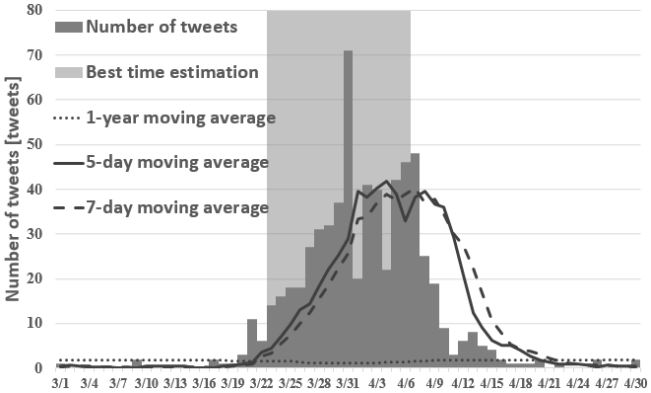


Figure 11: Estimation result of cherry blossoms in Meguro River in 2019.

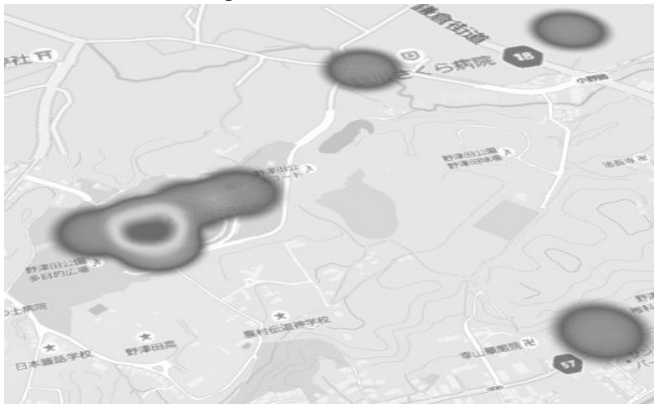


Figure 12: Nozuta Park, newly detected in this experiment

achieved using our proposed method estimates blooming around March 22 – April 1. These results confirmed that the proposed method can estimate flowering to full bloom with sufficient accuracy.

4.3.6 Extracting Spots Other than Correct Answers

Here, places other than those in Table 1, the correct answer spots, are described. After extraction using the heat map, many spots other than the correct spots are extracted in the experiment area of Tokyo.

As described in Sec. 4.3.4, tweets are most concentrated in places where people gather, such as around stations. Next, famous spots are extracted as cherry blossom spots, but not all tourist spots are listed in tourist information magazines and websites. However, extracted locations at which tweets are concentrated might not only be possible for locations that many people know but also locations at which a certain number of tweets are obtainable, such as locally famous locations.

As an example, Fig. 12 presents an example of Nozuta Park. In this figure, no circle signifying a correct answer is displayed, but a visualization result suggesting a correct answer spot was obtained. These results demonstrated that real-time analysis of tweets can automatically extract not only widely known locations but also local cherry blossom viewing spots. The analysis can provide such information. However, when using the proposed method, uniform spot extraction of the experiment range using a heat map might cause errors in local spot extraction because of the influence of tweet amounts around a station. Therefore, improvement of the extraction method is left as a topic for future study.

5 CONCLUSION

As described in this paper, we evaluated a regional analysis method based on location and time-series changes by providing real-time tweets with location information from Twitter. Our existing method provides an estimation of the viewing time for a specific tourist spot. Therefore, the method was constrained to using cases in which travelers have information about the travel area. As described in this paper, as stages 1 and 2, we propose a method for extracting information related to sightseeing spots using geotagged tweets. Subsequently, we demonstrate, experimentally, a method to extract information related to sightseeing spots.

To confirm the usefulness of the proposed method, we conducted experiments demonstrating the automatic extraction of tourist attractions using the proposed method, and experiments for estimating best times using existing methods. As a result, we demonstrated that even a small number of geotagged tweets can be useful to extract spots using location information accumulated over time. Additionally, we showed that the correct spots in Tokyo can be extracted with accuracy of 0.8. Furthermore, we confirmed that the proposed method can extract tourist spots other than the correct answer, such as famous local spots. It was also shown that the existing method is useful for spots extracted using the proposed method to estimate the best time for each sightseeing spot.

These results demonstrate the usefulness of SNS to provide real-time information. Therefore, we were able to demonstrate the possibility of spot extraction using the proposed method, but the scope of its application must be verified further. Moreover, although it is possible to estimate the best time for a spot that can be extracted using the proposed method, we confirmed that the existing methods are not accurate in places where there are tweets of many different types of tweets in areas with high concentrations of people, such as around stations. In future research, it will be necessary to examine a better spot extraction method combining the proposed method described herein with conventional methods and to consider automating tourist spot extraction for real-time analysis.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number 18K13254.

REFERENCES

[1] Twitter, It's what's happening, URL < https://Twitter.com/> (2015).
[2] M. Endo, S. Ohno, M. Hirota, D. Kato, and H. Ishikawa, "Examination of Best-time Estimation for Each Tourist Spots by Interlinking using Geotagged Tweets," International Journal on Advanced in Systems and Measurements, Vol.10, No.3–4, pp.163–173, IARIA (2018).
[3] M. Omori, M. Hirota, H. Ishikawa, and S. Yokoyama, "Can geo-tags on flickr draw coastlines?," In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14), pp.425–428, ACM (2014).
[4] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes Twitter users: real-time event detection by social sensors," pp.851–860, WWW 2010 (2010).
[5] A. R. Pratap, J. V. D. Prasad, K. P. Kumar, and S. Babu, "An investigation on optimizing traffic flow based on Twitter Data Analysis," 2018 Second International Conference on Inventive Communication and Computational Technologies, pp.320–325, ICICCT (2018).
[6] Y. Nakaji, and K. Yanai, "Visualization of Real-World Events with Geotagged Tweet Photos," 2012 IEEE International Conference on Multimedia and Expo Workshops, pp.272–277, IEEE (2012).
[7] GeoNLP Project, A place name information processing system which maps sentences automatically, URL < https://geonlp.ex.nii.ac.jp/> (2019).
[8] Twitter Developers, Twitter Developer official site, URL <https://dev.twitter.com/> (2015).
[9] Y. Hashimoto, and M. Oka, "Statistics of Geo-Tagged Tweets in Urban Areas (<Special Issue>Synthesis and Analysis of Massive Data Flow)," Vol.27, No.4, pp.424–431, JSAI (2012) (in Japanese).
[10] National Agriculture and Food Research Organization, Simple reverse geocoding service, URL <http://www.finds.jp/wsdocs/rgeocode/index.html.ja> (2015).
[11] MeCab, Yet Another Part-of-Speech and Morphological Analyzer, URL < http://mecab.google-code.com/svn/trunk/mecab/doc/index.html> (2015).
[12] M. Takahashi, M. Endo, S. Ohno, M. Hirota, and H. Ishikawa, "Automatic detection method of tourist spots using SNS," In Proceedings of the 10th International Conference on Web Intelligence, Mining and Semantics (WIMS 2020), pp.91–96, ACM (2020).
[13] WEATHERNEWS Inc., Sakura ch., URL < https://weathernews.jp/s/sakura/spot/4450/> (2019).

(Received January 29, 2019)
(Revised August 21, 2019)

Masaki Endo earned a B.E. degree from Polytechnic University, Tokyo and graduated from the course of Electrical Engineering and Computer Science, Graduate School of Engineering Polytechnic University. He received an M.E. degree from NIAD-UE, Tokyo. He earned a Ph.D. Degree in Engineering from Tokyo Metropolitan University in 2016. He is currently an Associate Professor of Polytechnic University, Tokyo. His research interests include web services and web mining. He is also a member of DBSJ, NPO STI, IPSJ, and IEICE.

Munenori Takahashi is currently studying at Polytechnic university.
His research areas include big data and web mining.

Masaharu Hirota received a Doctor of Informatics degree in 2014 from Shizuoka University. After working for the National Institute of Technology, Oita College, he has been working as an Associate Professor in the Faculty of Informatics, Okayama University of Science since April, 2017. His research interests include photograph, GIS, multimedia, and visualization. He is a member of ACM, DBSJ, and IPSJ.

Makoto Imamura He received a M.E. degree from Kyoto University of Applied Mathematics and Physics in 1986 and a Ph.D. degree from Osaka University of the Information Science and Technology in 2008. During 1986–2016, he worked for Mitsubishi Electric Corp. In April 2016, he moved to the school of Information and Telecommunication Engineering at Tokai University as a Professor. His research interests include machine learning, model-based design and Prognostics and Health Management (PHM).

Hiroshi Ishikawa earned B.S. and Ph.D. degrees in Information Science from The University of Tokyo. After working for Fujitsu Laboratories and becoming a full Professor at Shizuoka University, he became a full Professor at Tokyo Metropolitan University in April, 2013. His research interests include databases, data mining, and social big data. He has published actively in international refereed journals and conferences such as ACM TODS, IEEE TKDE, VLDB, IEEE ICDE, and ACM SIGSPATIAL. He has authored several books: Social Big Data Mining (CRC Press). He is a fellow of IPSJ and IEICE and is a member of ACM and IEEE.

Regular Paper

An Incremental Approach for Optimal Feature Selection in Regression: A Case Study of Wagyu Proteome Analysis

Nanami Higashiguchi†, Masatsugu Motohiro†, Haruka Ikegami*,
Tamako Matsuhashi*, Kazuya Matsumoto*, and Takuya Yoshihiro‡

†Graduate School of Systems Engineering, Wakayama University, Japan
*Graduate School of Biology-Oriented Science and Technology, Kindai University, Japan
‡Faculty of Systems Engineering, Wakayama University, Japan
‡tac@wakayama-u.ac.jp

Abstract - Sparse modeling has attracted significant attention as big data analysis goes popular. LASSO is one of the sparse modeling techniques to retrieve a set of features correlated to a target function. LASSO runs in very low computational time to obtain near-optimal set of features even if the data set is very large. However, due to its own regularization term, optimization errors are not negligible. In several practical scenes, it is required to obtain more optimal solutions within feasible computational time. However, the solution has not been presented clearly. In this paper, we present a new heuristic approach called IFS (Incremental Feature Selection) that starts from a collection of singleton feature sets, and increases features in a set one by one to finally obtain a quasi-optimal M-element feature set. The proposed technique IFS is applied to the analysis of Wagyu proteome expression data set, and we proved that IFS performs better than LASSO. Simultaneously we introduce a technique to find commonly-correlated features for the same objective function among multiple groups of data sets. We can use Multi-task LASSO for this purpose, but since it does not aware of the uniformity of the effects on each group, it is not enough to identify commonly correlated feature sets among all the groups. Our technique in IFS uses a fairness index to tackle the problem. We applied IFS to the Wagyu data set and showed that IFS ensures to retrieve a quasi-optimal feature set whose fairness index among correlation of those groups is larger than the given threshold.

Keywords: Sparse Analysis, LASSO, Feature Selection, Wagyu, Proteomics

1 INTRODUCTION

Sparse modeling has attracted significant attention in face of big data analysis. Recently, large dimensional data sets are easily obtained such as biological data represented by gene or protein expression profiles, and are recognized as very useful sources to analyze valuable properties of creatures. However, in computational analysis with those data, variable selection that retrieves a variable set that highly correlates the target trait from a vast amount of features is an essential task to pursuit. With the naive execution, the computational time of this task explodes to exponential and usually not feasibly solved with the current computers. To solve the task within feasible time, LASSO (Least Absolute Shrinkage and Selection Operator) [1], which retrieves a quasi-optimal variable set that minimizes the square errors in multiple regression analysis, is one of the well-recognized feature selection methods from vast amount of features included in the original data set. LASSO applies L1-norm regularization term in optimization formula to obtain a quasi-optimal feature set within feasible time. However, LASSO has a problem that the obtained feature set in many cases has a considerably large error and sometime far from optimal. To obtain a feature set closer to the optimal within feasible time is one of the solicited research tasks in this field.

In this study, we tackle this problem with a case study of Wagyu analysis, in which we try to find a small feature set from hundreds of proteins in a given protein profile that significantly correlates with Wagyu beef quality. Additionally, because the data set includes samples (i.e., beef cattle) from multiple Wagyu regions, we try to separate the common trends and each regional trend. We propose a new method for these tasks to treat a data set with hundreds of features.

Wagyu is known as a high-quality branded beef of Japan, with a feature of soft and tender meats due to fats mixed in the meat. There are several regions famous for Wagyu in Japan, and each region has different policy of breeding sires and beef cattle to produce larger amount of higher-quality meat. This contention among regions has improved the methodology of breeding beef cattle so far. However, since they mostly depend on traditional methods based on statistics on bloodlines or breeding experience, there is an apparent limitation in beef-quality improvement.

Recently, several comprehensive analyses in genomics or proteomics have been developed, for example, gene and protein expression profiles that include expression values of so many genes and proteins are available with smaller cost than ever. Specifically, we have a large number of explanation variables retrieved from each sample, which potentially makes us predict beef quality of each beef cattle in the early stage of beef-cattle breeding. This also could lead to the innovative methodology of breeding beef cattle to improve its beef quality.

Here, the first problem is that the variables in genes or protein profiles are so many that we can hardly select the optimal variable set to predict beef quality. The second prob-

lem is that beef cattle of distinct regions has different trend on its data so some analytic methods to treat this problem is required. As for the first problem, recently sparse analyses have been developed in which near optimal feature selection is possible with small computational cost. Especially, if we intend to perform multiple regression, LASSO is often used. LASSO minimizes MSEs (Mean Square Errors) in the form of multiple linear regression, in which by using L1 regularizer most of the coefficients are to shrink to zero. LASSO actually selects a near-optimal variable set within feasible time even if the number of available variables is very large. However, LASSO has a problem in Wagyu analysis that it cannot catch up with the trend of each branded Wagyu regions.

Multi-task LASSO(MT-LASSO) [2], which considers multiple objective functions in selecting a variable set has been proposed. MT-LASSO applies L1/L2 penalty to retrieve a variable set that commonly explains the multiple objective functions. This by definition can be used to explain the trend of each region of Wagyu brand by retrieving the common variables that explain trends of the all target regions. However, MT-LASSO retrieves a variable set without considering the balance of effects among multiple regions so that it may select a variable set that strongly effects on a region while weakly effects on other regions. Furthermore, it is known in both LASSO and MT-LASSO that the selected variables are not always optimal in terms of multiple regression so that we can hardly retrieve the optimal set of variables that explains the target traits of Wagyu beef [3]. Methods to retrieve the optimal variable set while considering multiple Wagyu brand regions are required.

In this paper, we present a solution for this problem, i.e., we propose a variable selection method IFS (Incremental Feature Selection) that retrieves an optimal commonly effecting variables among multiple Wagyu regions within a feasible computational time. We first exploit a single regression results, i.e., correlation coefficients, and fairness indices among them to retrieve a small number of variables as a candidate of selected variables. Second, we make a pair of those selected variables, and retrieve a certain number of pairs from them using the multiple correlation coefficients and the fairness indices among regions. We repeat this process to make candidate combinations including a larger number of variables. By testing all combinations of the candidate variables with multiple regression and fairness indices, we finally retrieve the best variable set within feasible time.

This paper is organized as follows. In Section 2, we describe the trend of Branded Wagyu beef. In Section 3, we introduce LASSO and MT-LASSO. In Section 4, we present a proposed method and its algorithms, and its computational complexity is analyzed in Section 5. After the evaluation results shown in Section 6, finally we conclude the work in Section 7.

2 BRAND WAGYU BEEF

Japanese Black Cattle is a beef cattle peculiar to Japan, which produces various brand beef called Wagyu such as Kobe beef, etc. There are many regional brand beefs in Japan, each of which has its own way to breed cattle, and apply its own criterion to authorize whether each head of cattle is sold under the name of the brand beef. As the authorization criteria, there are several items, e.g., the birth of cattle, the way to raise cattle, the rating of beef, etc. Among them, the rating of beef is the most important. The rating criteria include various values, and especially 6 items among them are regarded as the most important ones to judge whether a head of cattle is authorized as brand beef [5]. The 6 items, which we call *economical traits*, are CW (Carcass Weight), BMS (Beef Marbling Standard), YE (Yield Enhancement), RT (Rib Thickness), SFT (Subcutaneous Fat Thickness), and REA (Rib-Eye Area). Basically from these criteria, the price of beef in the market is determined. Therefore, the farmers of brand beef have been made a great endeavor to produce quality beef.

Wagyu farmers take various methodologies to produce quality beef stably. One of the most important methods is to control bloodline so as to have better values of the economical traits. Since the bloodline is known to have close relationship with economical traits, efficient inbreeding by producing and identifying genetically excellent individual cattle has a significant importance to improve the value of brand beef site. Each brand-beef site usually breeds several head of cattle called sires that have excellent genetic ability [6], [7]. From sires, we take sperms and freeze them, and sell them to farmers. With this system, excellent bloodline of sires is distributed to farmers and generate thousands of children cattle from an excellent sire. Note that, in brand-beef sites, father of each beef cattle is called '1-generation ancestor' and the father of beef cattle is one of the most important criteria to predict economical traits of beef cattle.

As a statistical methodology to predict economical traits of beef cattle from past records, the breeding values are usually used in brand beef sites. The breeding values are calculated for each economical trait, which represent the ability to improve the trait values compared to the average ability in the group. There are two kinds of breeding values, i.e., estimated breeding values and expected breeding values. The former is calculated for sires who have descendants with carcass characteristic scores and represents the ability to improve 6 economical traits. In contrast, the latter is calculated for each beef cattle that does not have enough number of descendants to estimate breeding values.

We have several variations of bloodline models used to compute breeding values. Currently, the most frequently used model is so called 'animal model,' which considers all the relative relationship including brothers of beef cattle that have the same mother. With a bloodline model and the data set, BLUP method calculates the breeding values in a statistical manner as the genetic ability inherited through bloodlines [8]. The expected breeding value for each beef cattle is calculated as the average of its two parents.

On the other side, raising method to produce high-value beef cattle stably also has been studied so far. However, methods in this area are mostly depends on experiences of farmers, and are not based on any scientific results or real data. For example, livestock associations or stock farmers have accumulated their experience to raise high-value beef cattle as know-how or some kind of manuals. This kind of information

has wide variations from direct methods such as how to feed cattle to indirect methods such as the structure of cowsheds. As for the academic results, a few studies have been published on the relationship between raising methodology and economical traits. For example, there is a study on improving BMS values by controlling the concentration of vitamin A [9]. However, in the current state, we have still too little knowledge to actually control economical traits in raising in livestock farms.

3 LASSO AND MULTI-TASK LASSO

LASSO [1] is a well-known technique for feature selection from the large number of features based on linier regression models. Let S be the given set of samples, and F be that of features. Let $x_{sf}(s = 1, 2, \ldots, |S|, f = 1, 2, \ldots, |F|)$ be the measured feature value of sample s on feature f, where $|S|$ and $|F|$ are the number of elements of S and F, respectively. Hereafter we may write just S and F in place of $|S|$ and $|F|$ for conciseness. Let $\mathbf{x}_s = (x_{s1}, x_{s2}, \ldots, x_{sF})^\mathsf{T}$, be the measured vector for each sample $s \in S$, where A^T denotes a transposed matrix of A. Let $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_S]$ be the matrix of the feature data. Let y_s be the measured trait values for each sample s, and $\mathbf{y} = (y_1, y_2, \ldots, y_S)$ be the trait vector. Then, LASSO is formulated as follows:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}}(\|\mathbf{y} - \boldsymbol{\beta}\mathbf{X}\| + \lambda|\boldsymbol{\beta}|) \tag{1}$$

where λ is a non-negative regularization parameter, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_F)$ is a coefficient vector for X. Additionally, $\|\boldsymbol{\beta}\|$ represents the L2 norm of a vector $\boldsymbol{\beta}$ defined as $\|\boldsymbol{\beta}\| = \sqrt{\sum_{f\in F}\beta_f^2}$, and $|\boldsymbol{\beta}|$ represents the L1 norm defined as $|\boldsymbol{\beta}| = \sum_{f\in F}|\beta_f|$ Due to the effect of L1-norm penalty with λ, most of β_f converges to zero during the computation of the optimal solution. As a result, we have a small number of non-zero coefficients, and this process works as a feature selection from a large number of feature variables.

Multi-task LASSO [2] is an extension of LASSO, which treats multiple objective functions. Let us denote T as the set of tasks (i.e., set of objective functions), and also let us define a feature matrix and a trait vector for each task. Namely, we let $x_{sf}^{(t)}$ be the measured feature values for task $t \in T$. Also, let $y_s^{(t)}$ be the measured trait values for task $t \in T$. Similarly, we also write $\mathbf{x}_f^{(t)}$, $\mathbf{X}^{(t)}$, $\mathbf{y}^{(t)}$, $\boldsymbol{\beta}^{(t)}$ etc. Note that the number of samples for each task $S^{(t)}$ could be different. Then, the MT-LASSO is expressed as follows:

$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}}(\sum_{t\in T}\|\mathbf{y}^{(t)} - \boldsymbol{\beta}^{(t)}\mathbf{X}^{(t)}\| + \lambda|\mathbf{w}|), \tag{2}$$

where W is the coefficient matrix that combines all coefficient vectors, defined as $W = [\boldsymbol{\beta}^{(1)}, \boldsymbol{\beta}^{(2)}, \ldots, \boldsymbol{\beta}^{(T)}]$, and $\mathbf{w}$ is the vector of L2 norms of the coefficients where $\mathbf{w} = (\|\mathbf{w}_1\|, \|\mathbf{w}_2\|, \ldots, \|\mathbf{w}_F\|)$, and $\mathbf{w}_f$ for $f \in F$ is defined as $\mathbf{w}_f = (\beta_f^{(1)}, \beta_f^{(2)}, \ldots, \beta_f^{(T)})$. Note that the regularization term is the combination of L1 and L2 norms, as shown in Fig. 1. In MT-LASSO, the coefficients $\beta_f^{(t)}$ are defined for each $f \in F$ and $t \in T$. To proceed feature selection and coefficients optimization altogether, MT-LASSO uses the combination of L1/L2 regularization. First, L2-norm of w_f, the coefficients vector of the same feature is computed, and second, L1-norm of those are used in the regilarization term. This enables us to select the commonly effective features for all tasks first, and then to optimize coefficients of the selected features within each task.

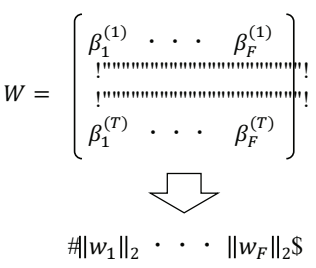


Figure 1: L1/L2 Regularization in Multi-task LASSO

MT-LASSO can be applied to our problem. Note that, in each region t of brand Wagyu, distinct samples, i.e., heads of beef cattle, are grown up so that we have measured feature sets $\mathbf{X}^{(t)}$ for each region t. As for the trait, we apply the same trait such as BMS, but each region has their own beef cattle, so that the data is expressed as $\mathbf{y}^{(t)}$. By solving MT-LASSO with the above $\mathbf{X}^{(t)}$ and $\mathbf{y}^{(t)}$, we can obtain the commonly effective feature set among multiple regions within the framework of MT-LASSO. However, the problem is that MT-LASSO does not consider the balance of effects among multiple regions. Additionally, MT-LASSO lacks optimality so that a non-optimal set of features would be selected frequently. In this paper, we try to improve the optimality utilizing correlation coefficients between $\mathbf{y}^{(t)}$ and $\mathbf{x}_f^{(t)}$ while balancing the effects on multiple regions using a fairness index.

4 THE PROPOSED METHOD

4.1 Problem Formulation

In this study, with a given set of regions T, we retrieve a set of features that has comparably high correlation for all region $t \in T$. For each region $t \in T$, for a given measured trait set $\mathbf{y}^{(t)}$, and a measured feature set $\mathbf{X}^{(t)}$ for features F, we try to find a feature set $F' \subseteq F$ that minimizes total MSE under the constraint that the fairness index of MSEs among all regions is larger than a given threshold. We use Jain's fairness index [4] to measure the 'fairness,' namely, to measure the uniformity in the effect of those feature set. This index takes one when all the values are the same, and takes n^{-1} in the worst case, where n is the number of input values (i.e., regions in this study). In the proposed method, the fairness in MSE among regions are defined as follows.

$$FI_{F'} = J(E_{F'}^{(1)}, \ldots, E_{F'}^{(T)}) = \frac{(\sum_{t\in T} E_{F'}^{(t)})^2}{n\sum_{t\in T}(E_{F'}^{(t)})^2} \tag{3}$$

Here, for the retrieved feature set F', we let $E_{F'}^{(t)}$ be the

MSE in region $t \in T$, and let $FI_{F'}$ be the fairness index of MSEs among all regions. Also, we let $E^{(all)}_{F'}$ be the MSE computed from all samples s in all regions T. Then, the problem formulation to solve in this paper is shown as follows.

Problem Formulation

Given M, the number of features to retrieve, and J, the least required value of $FI_{F'}$, find the feature set F' that minimizes $E^{(all)}_{F'}$ under constraints $|F'| = M$ and $FI_{F'} \geq l$.

4.2 Proposed Algorithm

As shown above, we propose an algorithm to compute a feature set that marks equally high correlation in every region $t \in T$. Specifically, since LASSO minimizes MSE, we also use MSE as the performance index, and compute a feature set that takes equally small MSEs for the given feature values $\mathbf{x}^{(t)}$ for each region $(t \in T)$. Generally, the computational complexity explodes when we select a set of M features that leads minimum MSE from a large number of features because we must compute MSEs through multiple regression for every combination of M features in the data set. In our algorithm, we solve this problem by increasing the number of features step by step. Namely, we start from a set of single features, next we make pairs of features, and then triads of features, and so on. Every time we increase the member of the sets, we filter the sets to limit the number of sets in order to limit the computational time. Generally, in multiple regression analysis, MSE values for a set of features tends to be smaller when a part of them leads to low MSE values. By making use of this property, we repeat increasing the member of features in the combinations and filter them, and finally obtain the best set of M features within feasible computational time.

Specifically, to retrieve several features out of hundreds or thousands of features, we first make a single regression analysis, compute MSEs, and filter them with those MSEs. With the reduced number of features, we make all possible pairs of the features, and filter them to retrieve pairs that have uniformly high MSEs in multiple regression analysis. Next, we make all possible triads of features by combinatorially adding one feature to the pairs, and filter them in the same way to retrieve a feasible number of triads. By repeating those, we finally obtain a set of M features that has good MSEs as well as good uniformity in MSEs among regions. By limiting the number of feature sets in each stage, we provide a guarantee on computational time to be feasible.

As aforementioned, we denote the given regions of Wagyu brand by $t \in T$, feature set by F, measured value vector for feature $f \in F$ by $\mathbf{x}^{(t)}_f$, measured value matrix for all features by $\mathbf{X}^{(t)}$, and measured trait vector by $\mathbf{y}^{(t)}$. For each stage i of our algorithm, G_i is input and G_{i+1} is output, where G_i is a family of feature sets represented by $G_i = \{F_{1,i}, F_{2,i}, \ldots, F_{k,i}\}, (1 \leq k \leq N)$, in which $F_{k,i} \subseteq F$, $|F_{k,i}| = i$, and N is a predefined natural number.

We present the algorithm to obtain the solution F' in the following.

1. Initialize with $i = 1$ and $G_i = F$.
2. Compute MSEs by applying multiple regression analysis between $\mathbf{y}^{(t)}$ and $F_{k,i} \in G_i$, and get $E^{(t)}_{F_{k,i}}$ for each $t \in T$ and $F_{k,i} \in G_i$. If $i = 1$, since the number of features in $F_{k,i}$ is one, we apply single regression analysis between each feature $f \in F_{k,i}$ and $\mathbf{y}^{(t)}$ instead.
3. Compute the fairness index $FI_{F_{k,i}}$ from $E^{(t)}_{F_{k,i}}$ $(^\forall t \in T)$ for each $F_{k,i} \in G_i$.
4. Compute MSEs with all samples of all regions, i.e., compute $E^{(all)}_{F_{k,i}}$ for each feature set in $F_{k,i}$.
5. Obtain a family of feature sets $G'_i \subseteq G_i$ by retrieving feature sets $F_{k,i}$ with $FI_{F_{k,i}} \geq Jl$.
6. If $|G'_i| > N$, limit the number of elements in G_i: Obtain $G''_i \subseteq G'_i$ by retrieving N-smallest feature sets in G'_i with respect to $E^{(all)}_{F_{k,i}}$. Otherwise, $G''_i = G'_i$.
7. Obtain the family of feature sets G_{i+1} as follows.
   (a) Create a set of features included in G''_i. Specifically, retrieve all features included in some feature sets in G''_i, and make a feature set $H_i = \{f_{1,i}, f_{2,i}, \ldots, f_{n,i}\}$.
   (b) Create a family of feature sets G_{i+1} by making all combinations between $G''_i = \{F_{1,i}, F_{2,i}, \ldots, F_{k,i}\}$ and $H_i = \{f_{1,i}, f_{2,i}, \ldots, f_{n,i}\}$. Namely, $G_{i+1} = \{F_{1,i} \cup \{f_{1,i}\}, F_{1,i} \cup \{f_{2,i}\}, \ldots, F_{1,i} \cup \{f_{n,i}\}, F_{2,i} \cup \{f_{1,i}\}, \ldots, F_{2,i} \cup \{f_{n,i}\}, \ldots, F_{N,i} \cup \{f_{1,i}\}, \ldots, F_{N,i} \cup \{f_{n,i}\}\}$.
   (c) Remove duplicated elements in G_{i+1}.
8. If $i < M$, do $i = i + 1$ and return to step 2.
9. Select the least MSE feature set F' that satisfies $FI_{F'} \geq J$ from G''_M, and output it.

We explain each steps of the algorithm. See Table.1 for definitions of variables.

First of all, in Step 1, we initialize variables i and G_i.

In Step 2, we compute MSEs by applying multiple regression for all feature sets in G_i and the target trait. Specifically, for each region $t \in T$ and feature set $F_{k,i} \in G_i$, we perform multiple regression analysis between $x^{(t)}_{F_{k,i}}$ and $\mathbf{y}^{(t)}$, and obtain the MSE value $E^{(t)}_{F_{k,i}}$ as a result.

In Step 3, we compute the fairness index for each feature set in G_i based on the formula (3), which represents the uniformity of the effects of $F_{k,i}$ on each region.

In Step 4, we apply multiple regression and compute MSE for each $F_{k,i}$ again, but using all samples $S^{(t)}$ in all regions in T altogether. We let $\mathbf{x}^{(all)}_F = (\mathbf{x}^{(1)}_F, \mathbf{x}^{(2)}_F, \ldots, \mathbf{x}^{(s)}_F)$ be the feature matrix and let $\mathbf{y}^{(all)} = (\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(s)})$ be the trait vector. Then we can compute MSE denoted by $E^{(all)}_F$ as a result of multiple regression of $\mathbf{x}^{(all)}_F$ and $\mathbf{y}^{(all)}$. Note that $E^{(all)}_{F_{k,i}}$ is computed for each $F_{k,i} \in G_i$.

Table 1: Notations

Symbol	Description
i	Current processing stage.
F	Set of Features.
T	Set of Wagyu region t
$\mathbf{x}_f^{(t)}$	Measurement vector of feature f in region t.
$\mathbf{y}^{(t)}$	Measurement vector of a trait with region t.
G_i	Family of feature sets in i-th stage.
H_i	Set of features included in G_i.
$E_{F_{k,i}}^{(t)}$	MSE value in i-th stage computed from regression with $x_F^{(t)}$ and $\mathbf{y}^{(t)}$ in region t.
$E_{F_{k,i}}^{(all)}$	The MSE in i-th stage computed with all samples in all regions.
$FI_{F_{k,i}}$	Jain's fairness index computed for $F_{k,i}$.
J	Threshold on fairness index given as constraint for output.
l	Margin for J to allow possible candidates with smaller value than J. In algorithm, we apply Jl as the threshold.
N	The number of feature sets to be selected on each stage.
M	The number of features to be selected finally.

In Step 5, we filter the feature set in G_i using Jain's fairness index to exclude the feature set unlikely to be a candidate for final output. With Jain's fairness index, we examine the uniformity of MSEs for each region $t \in T$. If the uniformity is high, the feature set is said to have correlation equally to all regions, meaning that the feature set includes general effect for Wagyu, not depending on regions. Through preliminary test, we found that the MSEs and fairness indices computed with a feature set F has small difference from those computed from the feature set $F' = F - \{f\}$ for $f \in F$. Thus, our basic strategy is to keep feature sets whose fairness indices are high enough in each stage of the algorithm. We apply threshold $J \times l$ to $FI_{F_{k,i}}$ and obtain the feature set $G_i' \subset G_i$. The threshold J and l are predefined constants, where J represents the requirement on fairness index for the final output of the algorithm. l provides a mergin for threshold J to allow feature sets with a little smaller value than J included in a candidate feature set G_i. Note that a feature set in stage i that has a little smaller fairness index than J can increase its fairness index in stage $i+1$ by adding one feature. The margin l works to keep the potential candidates for the next stage.

In Step 6, we limit the number of feature set in G_i' up to N elements. This step aims at ensuring the computational time to be feasible. As we mention later, the computational time highly depends on N. This is done by using MSE values $E_{F_{k,i}}^{(all)}$ computed in Step 4, i.e., if $|G_i'| > N$, we select top-N feature sets in terms of MSE, and create G_i''. Note that MSE is the most important selection criteria in the objective of this paper, and in this paper, we intend to obtain the minimum MSE feature set under the constraint that the fairness index is larger than J.

In Step 7, we create a family of feature sets for the next stage, i.e., G_{i+1}. When we increment the processing stage from i to $i+1$, the number of features in the feature set also incremented by one. Our basic strategy is to make all combinations of adding one feature to each feature set $F \in G_i''$. As candidate features to add, we make a feature set H_i that consists of all features included in the feature set family G_i'', and make all combinations of H_i and G_i''. By removing duplicated feature sets, we regard the set as G_{i+1}.

In Step 8, we repeat the above process until the number of elements in the feature set reaches M.

Finally in Step 9, we select the best feature set from G_M'' and output it. The best feature set is the one that have minimum MSE value among the sets whose fairness indices are larger than or equal to J.

5 COMPUTATIONAL COMPLEXITY

In this section, we analyze the computational complexity of the algorithm. The computational complexity depends on S, the number of all samples in all regions, N, the maximum number of feature sets in each stage, M, the number of stages, and T, the number of regions. The initialization process in Step 1 is clearly $O(1)$. In Step 2, we compute MSE for each feature set and for each region. Since the number of feature sets is less than N^2, and the complexity of multiple regression is $O(S)$, the complexity of this part is $O(SN^2)$. In Step 3, we compute fairness index for each feature set. The complexity to compute a fairness index is $O(T)$ when MSE values for each region is given. Thus, the complexity of this part if $O(TN^2)$. Since $T \ll S$, this can be regarded as $O(N^2)$. In Step 4, we do multiple regression with all samples in all regions for each feature sets, taking $O(SN^2)$ time. In Step 5, we remove feature sets whose fairness indices are less than Jl from G_i, which takes $O(N^2)$ time. In Step 6, we sort the feature sets by MSE, and select top-N sets. Since we sort at most N^2 elements, the complexity is $O(N^2 logN^2) = O(N^2 logN)$. In Step 7, we create the family of feature sets G_{i+1}. Since the number of elements is at most N^2, the complexity is $O(N^2)$. Step 8 and 9 apparently takes $O(1)$ time. The above is the complexity for one stage. Since we have M stages, the total complexity is $O(SMN^2 logN)$.

In Fig. 2, we show the real computational time in our evaluation described later. The parameters are $S = 96, M = 6$, and $N = (50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550)$, and the proposed method is executed on a general personal computer equipped with Intel i5 2.8GHz CPU and 8GB Memory. The results show that the computational time increases as N increases, but the slope is not steep. Although the number of samples in this data set is not large, the computational time of the proposed algorithm is feasible for a certain magnitude of data size.

6 EVALUATION

6.1 Data Description

We evaluate the proposed method compared with the result of Multi-task LASSO (MT-LASSO). The data consists of 3

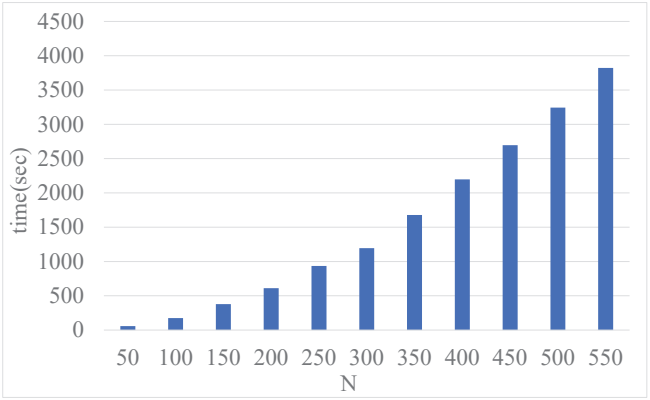


Figure 2: Execution Time of Proposed Algorithm

regions of branded Wagyu, which we refer region A, B, and C, and each region has 51, 10, and 35 beef cattle (i.e., samples) in the data. Each beef cattle has been grown up in one of the regions, and the 6 economical traits were measured before slaughtered and sold as meat. As aforementioned, the 6 economical traits are CW (Carcass Weight), REA (Rib-Eye Area), RT (Rib Thickness), SFT (Subcutaneous Fat Thickness), YE (Yield Enhancement), and BMS (Beef Merbling Standard). As a result, our data has 6 trait values for each sample from 3 regions.

The feature data set is a proteome expression profile of serum; for each beef cattle, serum is taken with the interval of 3-4 months, which are analyzed by SWATH-MS [10] (Sequential Window Acquisition of all Theoretical fragment ion spectra Mass Spectrometry) method with our own preprocessing treatment. In this method, we got the expression levels of 135 proteins for each sample. As a result, we got 135 protein expression values for 6 periods of time, so we have $135 \times 6 = 810$ features for each sample from 3 regions. After removing the features that contain null values, we have 580 features to apply the proposed method.

6.2 Evaluation Methods

We applied the proposed method and MT-LASSO to the data set described above. MT-LASSO originally is a feature selection method based on multiple objective functions, but it can be applied to our problem, i.e., it treats the same objective function for different data groups. To the best of our knowledge, MT-LASSO is the only method that treats multiple regions under a single objective function.

As evaluation criteria, we use MSE and the fairness index to compare those two methods. We select MSE rather than correlation coefficients to compare performance of the two because LASSO (as well as MT-LASSO) is an optimization scheme based on MSE. We also use Jain's fairness index [4] to measure the uniformity of the effects (i.e., correlation measured by MSE) among multiple Wagyu regions. When the fairness index takes high value, i.e., close to 1, we can regard that the selected feature set has uniformly the same level of correlation in all regions, meaning that the effect is not specific in a particular region, but expresses a general property in Wagyu. To separate the general effect from regional ones is the objective of this study.

As parameters, we set $M = 6$, i.e., we retrieve a set of 6 features to explain target traits. We choose this value considering the number of samples in each region. We also set $J = 0.8$, $l = 0.8$, and $N = (50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550)$, which are determined through preliminary tests. In MT-LASSO, to compare the performance with the proposed method, we adjust the parameter value λ to select exactly 6 features, and used the results in our comparison. We tried to use MT-LASSO implementation included in scikit-learn [12], but unfortunately, this cannot treat our dataset; it does not support the case with the same objective function applied to multiple groups. However, when we focus on the feature selection function, the mechanism of MT-LASSO is exactly the same as LASSO. (Notice that MT-LASSO first makes a feature selection based on L1 norm penalty, and then determine coefficients for each objective function based on L2 norm.) Thus, we apply LASSO implementation included in Python scikit-learn library for the results of MT-LASSO. Additionally, to expect the fairness in comparison, we do not use the MSE value obtained directly from MT-LASSO. To get rid of the effect of penalty terms, we made a multiple regression analysis with the features retrieved by MT-LASSO, and used the MSE values in our comparison.

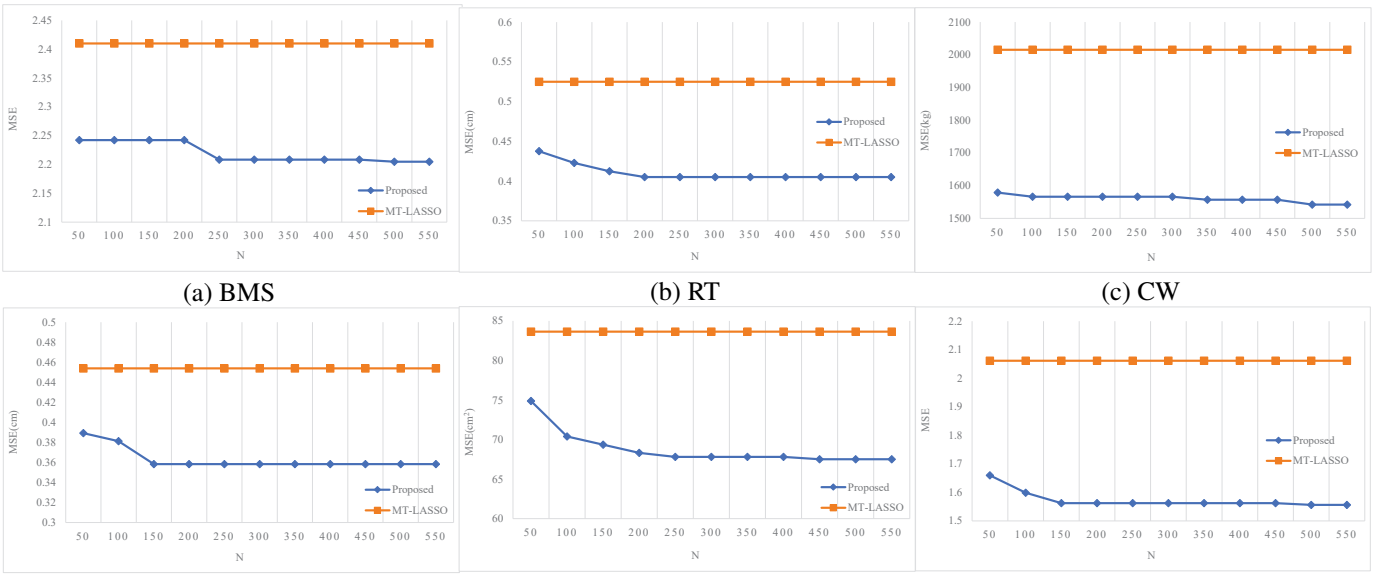
6.3 Evaluation Results

In Fig. 3, we show the comparison results on MSE values for each target trait. We see that the proposed method outperforms MT-LASSO in every trait. The cause of errors in MT-LASSO is the L1-norm regularization term. In contrast, the proposed method selects feature sets in the stepwise manner during which we keep good combinations of features as candidates of the solution. These results show that our strategy clearly works, and the performance is better than MT-LASSO (as well as LASSO) even if we keep only 50 ($N = 50$) candidates in each stage. Figure3 also shows that the performance goes better when N is increased, although the performance improvement is not significant. Note that, in this figure, MT-LASSO is shown to always take the same value since it does not have parameter N.

In Fig. 4, we show the results on fairness index. We see that, in most cases, the proposed method presents better performance than MT-LASSO, meaning that the proposed method surely retrieves feature sets that uniformly effects on all regions. We see that the performance of the proposed method involves some fluctuation, which is not seen in MSE performance. This is because the primal objective function is not fairness index, but MSE. It is natural that the best-MSE feature sets do not always have the best fairness index value.

More importantly, we see that the proposed method always keeps the constraint of fairness index, i.e., the value must be larger than $J = 0.8$. Recall that the proposed method intends to minimize MSE under the constraint on fairness index. We succeeded to keep the constraint and to ensure that the fairness index is above the preconfigured threshold $J = 0.8$.

In summary, we showed that the proposed method not only ensures the minimum bound on fairness index, but also mark better optimality on MSE compared with MT-LASSO as well as LASSO.



(a) BMS (b) RT (c) CW

(d) SFT (e) REA (f) YE

Figure 3: Results on MSE

7 DISCUSSION ON APPLICATION

In this section, we discuss how to apply the proposed methods in practice. We proposed two different techniques in this paper. One is IFS, which takes the incremental approach to select optimal set of features, and the other is an extension of IFS, which enable us to retrieve a commonly effective feature set among multiple groups. The first method IFS solves the same problem as well-known LASSO, which is used widely in variation of feature selection cases. Thus, IFS is also applicable a wide variety of practical cases to obtain the optimal set.

The second method, the extension of IFS, is compared with Multi-task LASSO in our evaluation. However, Multi-task LASSO originally is a method to select a feature set that explains more than one objective variables. In other words, the task we are focusing on, i.e., retrieving a commonly effective feature set among many features under a single objective variable, has not been tackled so far. Nevertheless, this new task may be required in several cases, e.g., analyzing causes of a pandemic disease among countries or regions could be a target. There are many candidate features that causes a disease spreading. To identify the common causes among regions and separate them from those specific to the region may be a useful application.

8 CONCLUSION

In this paper, we proposed a new feature selection method IFS (Incremental Feature Selection) that incrementally increases the number of elements in the candidate feature sets in order to finally select a quasi-optimal feature set that minimizes MSE in multiple regression analysis. In addition, we considered to retrieve commonly effective feature sets among different regions of Wagyu brands to identify the generally effective features that explain Wagyu beef quality regardless of Wagyu regions. We proposed to apply fairness indices among MSE values of multiple regions to limit the least allowable fairness among MSEs as a constraint.

Through evaluation compared with LASSO and Multi-task LASSO, we showed that the proposed method IFS outperforms those conventional methods. From the result, we proved that the incremental approach works more effectively to decrease error in MSE than the well-known LASSO. Although the computational time of IFS is larger than LASSO, it is still feasible if the given feature set size is in the order of hundreds or thousands. Additionally, we showed that IFS enables us to provide a constraint of fairness in MSEs among different Wagyu regions. We outperform MT-LASSO in retrieving a fairly effective feature set not only optimality in minimizing MSEs but also in that it guarantees the minimum fairness in MSE among regions.

As future work, how to determine the parameters J and l would be interesting. In addition, designing a method for predicting beef-quality traits from the common and regional feature sets retrieved from IFS would be one of the challenging topics.

Acknowledgment

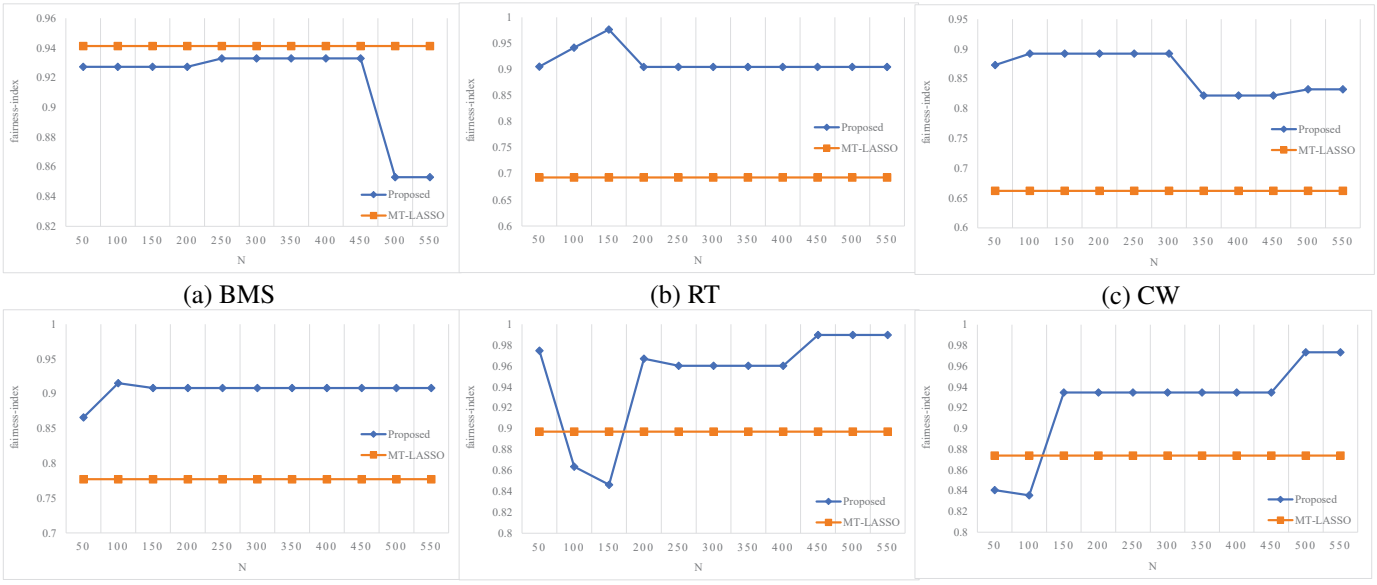
This work is supported by “the Program for Promotion of Stockbreeing” of JRA (Japan Racing Association).

REFERENCES

[1] R. Tibshirani, Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society, Series B, 58(1), pp.267–288 (1996).

[2] G. Obozinski, B. Taskar, and M. Jordan., Multi-task feature selection, In Technical Report, Department of Statistics, University of California, Berkeley, (2006).

[3] S. Hara and T. Maehara, “Enumerate Lasso Solutions for Feature Selection,” In Proc. AAAI2017, (2017).



(a) BMS (b) RT (c) CW

(d) SFT (e) REA (f) YE

Figure 4: Results on Fairness

[4] R. Jain, D.M. Chiu, W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems," DEC Research Report TR-301, (1984).
[5] Japan Meat Grading Association, "The Manual of Standard in Dealing Pork and Beef," (2001) (In Japanese).
[6] A Guide of Japanese Black Cattle Sires, http://liaj.lin.gr.jp/index.php/detail/data/m/803237096 (referred in October 2020) (In Japanese).
[7] Wagyu Registry Association,"Compliation of Sires of Japanese Black Cattle," (2003) (In Japanese).
[8] N. D. Cameron, "Selection Indices and Prediction of Genetic Merit in Animal Breeding," CAB International (1997).
[9] A. Oka, T. Dohgo, M. Juen, and T. Saito, "Effects of vitamin A on beef quality, weight gain, and serum concentration of thyroid hormones, insulin-like growth factor-I, and insulin in Japanese black steers," Animal Science and Technology (1998).
[10] C. Ludwig, L. Gillet, G. Rosenberger, S. Amon, B.C. Collins, R. Aebersold, "Data-independent acquisition-based SWATH-MS for quantitative proteomics: a tutorial," Molecular Systems Biology (2018) DOI 10.15252/msb.20178126.
[11] R-Project, https://www.r-project.org/, (referred in October 2020)
[12] Scikit-learn, https://scikit-learn.org/ (referred in October 2020)

(Received May 14, 2020)
(Revised October 28, 2020)

Nanami Higashiguchi received the B.E. degree from Wakayama University in 2019. She is currently a Master-course student in Wakayama University. She is interested in Data Science, Statistical Modeling, and Wa-gyu beef improvement. She is a student menber of IPSJ.

Masatsugu Motohiro received the B.E. and M.E. degree from Wakayama University in 2018 and 2020, respectively, and he is now with Hibiya Computer Systems, Co., Ltd. He is interested in Data Science, Statistical Modeling, Agricultural IoT, and so on.

Haruka Ikegami received her B.E. from Kindai University in 2004. She was a researcher in Kindai University from 2004 to 2020. She is interested in agriculture proteomics, bioinformatics, mass spectrometry, and so on.

Tamako Matsuhashi received her B.A. and M.S. from Hokkaido University in 1997 and 1999, then received her Ph.D. (Doctor of Veterinary) degrees from The University of Tokyo in 2006. She was a chief scientist and then research specialist in Gifu Prefecture from 2006 and 2014, respectively. She is a junior associate professor in Kindai University from 2016. She is interested in reproduction and bioinformatics of livestock in animal science.

Kazuya Matsumoto Received his B.A. degree from Utsunomiya University in 1984, and then M.S. and Ph.D. (Doctor of Agriculture) degrees from Kyoto University in 1986 and 1989, respectively. He was a staff scientist in NT. Science and Tosoh Co. from 1989 to 1995, and visiting scientist in The Institute of Medical Science, The University of Tokyo from 1995 to 1998. In 1998, he moved as an Assistant Professor to Kindai University. He is a Professor in Kindai University. One of his major research themes is applications of bioinformatics analysis for animal science.

Takuya Yoshihiro received his B.E., M.I. and Ph.D. degrees from Kyoto University in 1998, 2000 and 2003, respectively. He was an assistant professor in Wakayama University from 2003 to 2009. He has been an associate professor in Wakayama University from 2009. He is currently interested in the graph theory, distributed algorithms, computer networks, medial applications, and bioinformatics, and so on. He is a member of IEEE, ACM, IEICE, and Senior member of IPSJ.

Regular Paper

Visual Appearance is not Always Useful: Voice is Preferred as a Mediating Conversational Agent to Support Second Language Conversation

Tomoo Inoue,* Kumiko Kawai,** Boyu Sun**, and Kosuke Sasaki**

*Faculty of Library, Information and Media Science, University of Tsukuba, Japan
**Graduate School of Library, Information and Media Studies, University of Tsukuba, Japan
*inoue @slis.tsukuba.ac.jp

Abstract – Two representations of a conversational agent to support second language communication were compared, one of which was a CG character agent and the other was a voice agent. Except for the appearance, these agents were identical in terms of the function to intervene in the conversation between the native speaker and the non-native speaker and to pass the next speaking turn to the non-native speaker. A few distinctions were found between the two in the influence on the second language conversation. The voice agent was more effective in intervention, and was perceived more useful by the participants.

Keywords: Conversational agent, Second language communication, Voice user interface, Non-native speaker, Turn taking

1 INTRODUCTION

In recent years, globalization has increased the opportunities for conversation in the second language among people from different countries. Cross-cultural communication has disincentives such as differences in language and culture. Especially when a non-native speaker (hereinafter NNS) conducts conversation in a second language with a native speaker (hereinafter NS), he/she often face problems in listening comprehension of NS speech as well as in speaking, because his/her oral skills and knowledge in the second language are limited and thus difficult to make a story smoothly. As a result, there are differences in the amount of speech between NS and NNS [1], and the tendency of NS to take the initiative in conversation [2].

To improve these biases, methods and tools to support more participation of NNS in conversation have been proposed [3]-[6]. However, most of those are not very easy to use in everyday life in terms of portability and equipment cost, as they require a PC with a display or the installation of a robot, and so on [7]. More convenient form of a support tool is desired with the expansion of opportunities for second language conversation. Audio support without video can be considered along this line. In this research, we compared the voice agent with the existing CG character agent which works in the same way.[1]

2 RELATED WORK

2.1 Second Language Conversation Support Systems

Various studies on supporting second language conversation have been conducted recently. Majority of them are on presenting visual information such as text and images. For example, a face-to-face cross-cultural communication support system was proposed that displayed related information on the nouns appeared in a conversation [3]. A keyword sharing system to promote mutual understanding in a remote conversation between NS and NNS has been studied where keywords in the voice conversation are entered by NS for accurate and efficient content summarization and they are shown in each screen for better understanding and participation of NNS [4]. In the speech speed awareness system, the speech speed of the speaker is constantly detected and when it becomes too fast for NNS, the system notifies it through a screen [5]. Guo et al. has developed a CG character agent with voice to support second language conversation [6]. When a long silence occurs in the conversation between NS and NNS, the agent takes the speaking turn and talks to the NNS to pass the next speaking turn to him/her. It aims to encourage NNS utterance for better participation, which would result in better productivity in conversation.

All of these second language conversation support systems require a screen and a PC, and are not very portable. Considering that the second language conversation takes place in various places, it is desirable to develop a small portable system.

2.2 CG Character Agents and Robots for Intervention in Conversation

A CG character agent and a robot are often introduced to support conversation. Huang et al. proposed a CG character agent that participated in human conversation and presented information during the conversation [8]. From a WoZ experiment of intervention to conversation, the following four types of interventions was shown to be effective. No.1 was "provide-topic" that provides new topic to the conversation.

[1] This is an extended version of the paper presented at IWIN 2019.

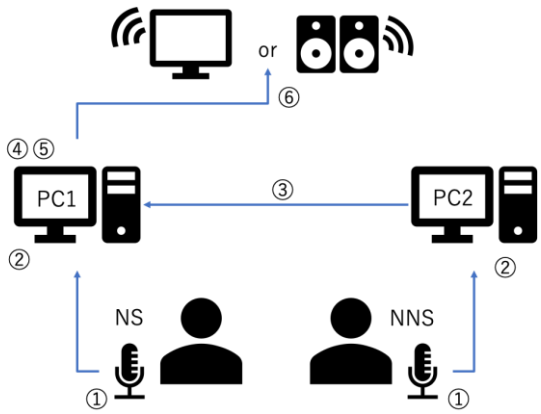


Figure 1: Configuration of the agent system.

No.2 was “more-information” that provides additional information to the topic of the conversation. No.3 was “recall-support” that supplements the information that the user has forgotten. No.4 was “discussion-support” that organizes the content of the conversation so far. In particular, the rating of “provide-topic” was high, which indicated the demand for keeping conversation.

Akiba et al. developed a robot agent to be a facilitator in order to eliminate social imbalances in multi-party conversation [9]. The robot was life-size and intervened in the conversation between the two participants who were leading it when there was a less involved participant.

Those agents and robots also have visual appearances and need bulky equipment.

2.3 Representation of Agent and Its Psychological Influence

Research has also been conducted on the psychological effects of the form of an agent on users. Naito et al. [10] compared with a robot, a CG character, and a voice agent on the user's attitude toward information acceptance. A robot significantly increased the user's acceptance attitude, and there was no difference between a CG character and a voice agent on the information acceptance attitude. There have been other experiments to improve affinity and communication by making the agent look and act like human. According to Takeuchi et al., a human-like agent could promote relaxed dialogue with little psychological burden, but at the same time, could cause excessive expectations for its behavior [11]. Hara et al. conducted a study asking if it is possible to give more feeling of being watched by making the appearance of a camera robot eyeball-shaped. It was shown that the eyeball-shaped camera robot could make people feel more nervous and afraid than the normal camera robot. As a result, it was shown that the presence of the eye could make people feel more nervous and afraid than the camera [12].

Those studies indicate representation of an agent has psychological effects and influences on human behavior. In this research, we are interested in the influence of appearance of an agent that facilitates second language conversation between NS and NNS.

Table 1: Agent intervention patterns.

Last speaker	Intervention pattern after 2 seconds of silence
NS	Ask a specific participant for an opinion ・What do you think, NNS?
	Ask for a new opinion ・Do you have any opinion, NNS?
NNS	Ask for an utterance that supplements own opinion ・I see NNS. Could you tell me more?
	Ask for a new opinion ・Do you have any other opinion, NNS?

3 CONVERSATIONAL AGENT

The agent used in this study is a second language conversation support agent based on the turn taking rule developed by Guo et al. [6]. It participates in conversation and try to pass the speaking turn to NNS according to the next speaker selection technique and the turn taking rule in conversation between NS and NNS.

Figure 1 shows the system configuration. The voices of NS and NNS are by taken by microphones (1). The volume of each voice is detected by the voice detection module in PCs (2). The voice detection module notifies to the silence detection module whether received sound exceeds pre-determined threshold value (3). The silence detection module gathers information from the voice detection modules to know silence, and notifies to the Wizard, an experimenter, when silence continues for more than 2 seconds, because it is regarded as another turn even if the former speaker speaks again [13] (4). The Wizard selects an intervention speech which is most appropriate in a next turn (5). The agent starts performing intervention by the selected speech (6).

As the agent's intervention, Guo et al. applied the adjacent pair pattern "Question-Response" (e.g. "What do you think, NNS?") and the additional question (e.g. “You think so, don’t you, NNS?”) based on the next speaker selection technique [14]. However, it might be unnatural when the agent intervenes after the NNS utterance. Therefore, in this research, the intervention patterns based on the “Question-Response” shown in Table 1 were used with reference to the intervention by Yoshino et al. [15]. In this study, we used the WoZ method to manually determine the intervention content, because it was decided based on the last speaker before the silence.

4 EXPERIMENT

We investigated the influence of representation of our conversational agent that facilitates second language conversation between NS and NNS. Specifically, it was whether the voice agent and the CG agent provided the same support and the same psychological effect on the participants.

(a) CG condition

(b) Voice condition

Figure 2: Scenes from the experiment.

4.1 Participants

The participants were a total of 48 people of 24 pairs each consisting of a native Japanese speaker and a non-native Japanese speaker. JLPT N2 was required as the language skill of a non-native Japanese speaker to meet our target who could speak Japanese but not fluently like native speakers. JLPT is a standardized criterion-referenced test to evaluate and certify Japanese language proficiency for non-native speakers [16]. N2 level certifies the ability to understand Japanese used in everyday situations, and in a variety of circumstances to a certain degree [17]. They were recruited from a Web bulletin board, and were undergraduate or graduate university students. Each pair was new to each other and of the same sex in this experiment.

4.2 Design

In order to investigate the influence of agent representation, we conducted within-subject experiments under the following three conditions.

1) CG condition

The CG character agent displayed on the screen intervenes in the conversation.

2) Voice condition

The voice agent from the loudspeaker intervenes in the conversation.

3) FTF condition

Participants talk face-to-face without the agent.

Figure 2 shows the scenes from the experiment.

The CG agent was implemented using the MMD agent, which is an open-source toolkit for building voice interaction systems with a CG character [18]. An agent built by the MMD agent has a basic function to perform language and non-verbal interaction with people. The appearance of the agent is shown in Fig. 2(a). For our purpose, specific motion of the agent including facial expression was not used. What was used in this study was its default behavior and facial expression, which was to sit still with slight swaying and normal facial expression to avoid giving a strange impression. By adding a socket plugin-in to the MMD Agent, the agent was able to be controlled with a Python program. For speech detection, one of audio I/O libraries PyAudio was used [19]. For speech synthesis, a Japanese text-to-speech system Open JTalk was used [20]. It provides the voice of a female speaker model called the "May" character. The voice agent was the same with the CG agent but only used its voice part.

Each pair had free conversation in 5 minutes for each condition. The topic was different each time. The order of participation in the conditions was balanced by the Latin square method.

4.3 Data

1) Conversational behavior

The video and audio data were collected from three cameras, the panoramic camera that captures all of NS, NNS, and the agent, and the two cameras that capture each speaker from the front. From this video and audio, we analyzed the conversational behavior of NS, NNS and the agent. The total length of data used for analysis was 3 conditions * 24 pairs * 5 minutes = 360 minutes.

From the collected video data, speech segments were extracted under the condition of minimum silent interval of 400 milliseconds [21] using the segmentation function of the annotation tool ELAN. The part where the utterances overlapped and the part where the voice was low were segmented manually to avoid incorrect segmentation. Then the speech segments excluding non-verbal speech segments such as laughter, cough and sigh were classified as NS speech, NNS speech, and agent speech.

2) Questionnaire

The questionnaire consisted of the following items. They were the items such as naturalness of communication, ease of speech, for evaluating overall impression of communication and conversation [3], [22]-[24], items regarding the appropriateness of the agent intervention and usefulness, items for measuring stress in conversation, and items regarding the agent's impression for evaluating the influence of the agent representation on the participants [11], [25]-[26]. Items related to an agent were asked to the CG condition and the voice condition, and not to the FTF condition. All items were measured in 7-point Likert scale where 1 corresponded to strongly

Table 2: Categories of intervention.

	Effective	Ineffective	Total
CG (#)	17	9	26
CG (%)	65.4	34.6	100
Voice (#)	37	4	41
Voice (%)	90.2	9.8	100

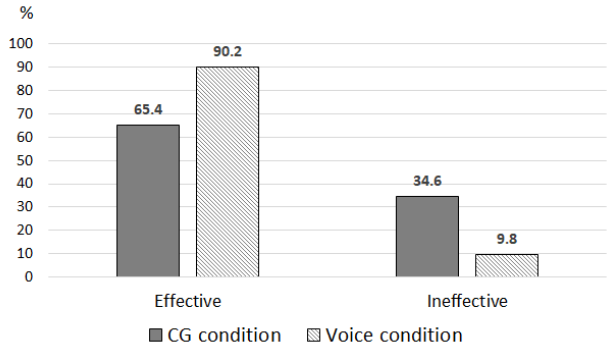


Figure 3: Ratio of interventions by the agent.

disagree to 7 corresponded to strongly agree. It is better evaluated in the item #1 to #18 and #22 to #23 as having higher scores, while in the item #19 to #21 as having lower scores.

5 RESULT

5.1 Number of Intervention by the Agent

We found a mistake in the operation of the wizard in a pair, whose data was removed.

The total number of agent interventions in the 23 pairs was 70, of which 29 times in the CG condition and 41 times in the voice condition. Paired t-test found a marginally significant difference between the conditions (t(23)=-1.96, p = .06).

5.2 Categories of Intervention by the Agent

Interventions were categorized into 2 types of effective intervention and ineffective intervention, where NNS took the next speaking turn in effective intervention while he/she did not take it in ineffective intervention.

We removed 3 interventions, all of which were in the CG condition, from the analysis. For one, the system stopped right after the intervention due to malfunction. For other 2, the interventions were the end of the sessions.

Table 2 and Fig. 3 show the result. The effective intervention was 65.4% in the CG condition and 90.2% in the voice condition.

A two-way ANOVA was used to compare whether there are differences in the number of interventions between the conditions of CG and voice, and between the categories of effective intervention and ineffective intervention. The result showed a marginally significant difference for the main effect between conditions (F(1,88) = 3.13, p = .08) and a significant difference for the main effect between categories (F(1,88) = 23.4, p <.01). A significant difference was also found for interactions (F(1,88) = 8.70, p = .004), but since there was a sufficient difference in the number of interventions between the categories, simple main effects were examined by a Bonferroni-corrected t-test using the number of interventions in each category.

As a result, significant differences were found between the number of effective interventions in the voice condition and the following numbers: the effective interventions in the CG condition (t(22) = -3.66, p = .008), the ineffective interventions in the CG condition (t(22) = -5.18, p = .0002), the ineffective interventions in the voice condition (t(22) = 5.13, p = .0002).

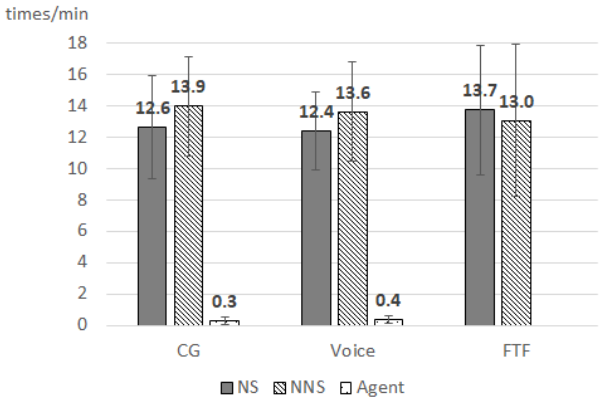


Figure 4: Speech frequency.

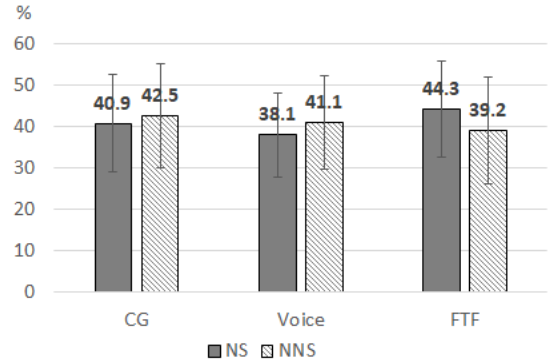


Figure 5: Speech time ratio.

5.3 Participants' Speech

The speech frequency is shown in Fig. 4. A two-way ANOVA comparing between the speakers of NS and NNS, and between the conditions of CG, voice, and FTF revealed no significant differences in the main effect and interaction (Inter-speaker main effect, F(1,132)=0.98, p=.32; inter-condition main effect, F(2,132)=0.14, p=.87; interaction, F(2,132)=1.09, p=.34).

The ratio of the time each speaker was speaking during the conversation (speech time / conversation time * 100) is defined as the speech time ratio. Figure 5 shows the speech time ratio of each NS and NNS for each condition. A two-way

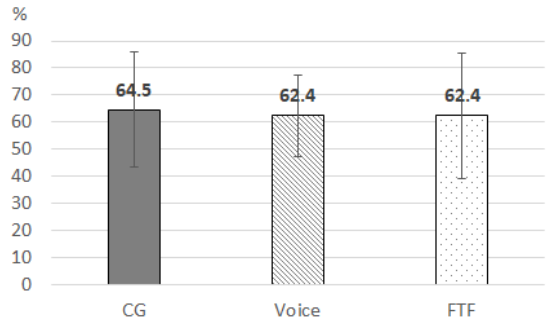


Figure 6: Speech balance.

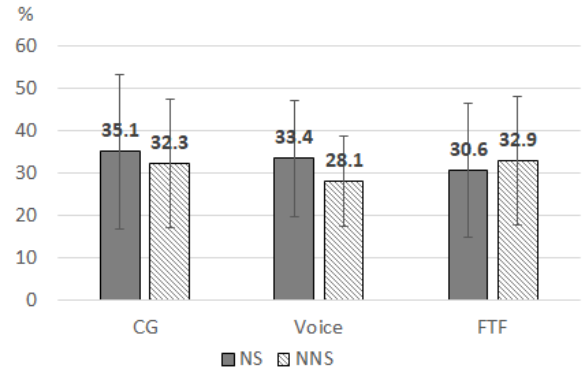


Figure 7: Back channeling rate.

ANOVA comparing between the speakers of NS and NNS, and between the conditions of CG, voice, and FTF revealed no significant differences in the main effect and interaction (Inter-speaker main effect, $F(1,132)= 0.006$, $p=.94$; inter-condition main effect, $F(2,132)= 0.62$, $p=.62$; interaction, $F(2,132)= 1.49$, $p=.23$).

Figure 6 shows the results of the speech balance, which is defined as (the number of utterances of the less talkative speaker) / (the number of utterances of the more talkative speaker) * 100 (%) [27]. A one-way ANOVA comparing between the conditions of CG, voice, and FTF revealed no significant difference ($F(2,69)=0.09$, $p=.92$).

Back channeling is a short expression sent by the listener while the speaker is using the right to speak [28]. To compare the degree that the participant plays the role of the listener, the back channeling rate (the number of back channeling / the number of utterance * 100) was calculated. Back channeling was judged according to the definition of the back channeling expression by Yoshida et al. [29]. The results are shown in Fig. 7. A two-way ANOVA comparing between the speakers of NS and NNS, and between the conditions of CG, voice, and FTF revealed no significant differences in the main effect and interaction (Inter-speaker main effect, $F(1,132)=0.52$, $p=.47$; inter-condition main effect, $F(2,132)=0.44$, $p=.64$; interaction, $F(2,132)= 0.74$, $p=.48$).

5.4 Questionnaire

Figure 8 shows the items for impression of communication and conversation. Figure 9 shows the items for the usefulness of the agent. Figure 10 shows the items for the impression of the agent. Figure 11 shows the items for the stress in conversation.

To examine whether there were differences in the results of the questionnaire between conditions, Friedman tests were performed for questions #1 to #9 and #19 to #23, which asked participants in all three conditions, and Wilcoxon's signed-rank tests were performed for questions #10 to #18, which asked participants in the CG and voice conditions. As a result, we found that the scores of NNS in the following items were significantly or marginally significantly lower in the CG condition than in the voice condition. They are the item #10 "I felt the agent was involved in the conversation" ($Z = -2.07$, $p =.04$), the item #13 "I could speak a lot because of the agent intervention." ($Z = -2.03$, $p = .04$), and the item #14 "I felt the agent's utterance timing was appropriate." ($Z = -1.75$, $p = .08$).

6 DISCUSSION

6.1 Influence of the Agent Representations on Conversation

It was surprising that the influence of the agent representation seemed differ in some aspects.

The tendency of more number of interventions in the voice condition in 5.1 indicates more tendency of long silence occurred at the same time. Although the differences were not statistically significant, it seems to be consistent with the fact that the voice condition had smaller values than the CG condition in all of Fig. 4, 5, and 7 which related to participants' conversation.

Figure 7 shows that the intervention was relatively effective in both the voice and CG conditions, and that the intervention resulted in a higher proportion of NNS speech in the voice condition, which might not be meet our expectation.

Again, although the difference was not statistically significant, the following can be read from Fig. 4, 5 and 7. The speech frequency of NNS seemed slightly higher than that of NS in the CG condition and the voice condition, while that of NNS seemed lower than that of NS in the FTF condition. Also, the speech time ratio of NNS seemed slightly higher than that of NS in the CG condition and the voice condition, while that of NNS seemed lower than that of NS in the FTF condition. Moreover, the back channeling rate of NNS seemed slightly lower than that of NS in the CG condition and the voice condition, while that of NNS seemed higher than that of NS in the FTF condition. It is interesting that there is such consistency in the trend of conversation change. NNS seemed to be more of a listener in the FTF condition. But agent intervention seemed to have reversed it.

Overall, the result could assure the feasibility of a voice agent.

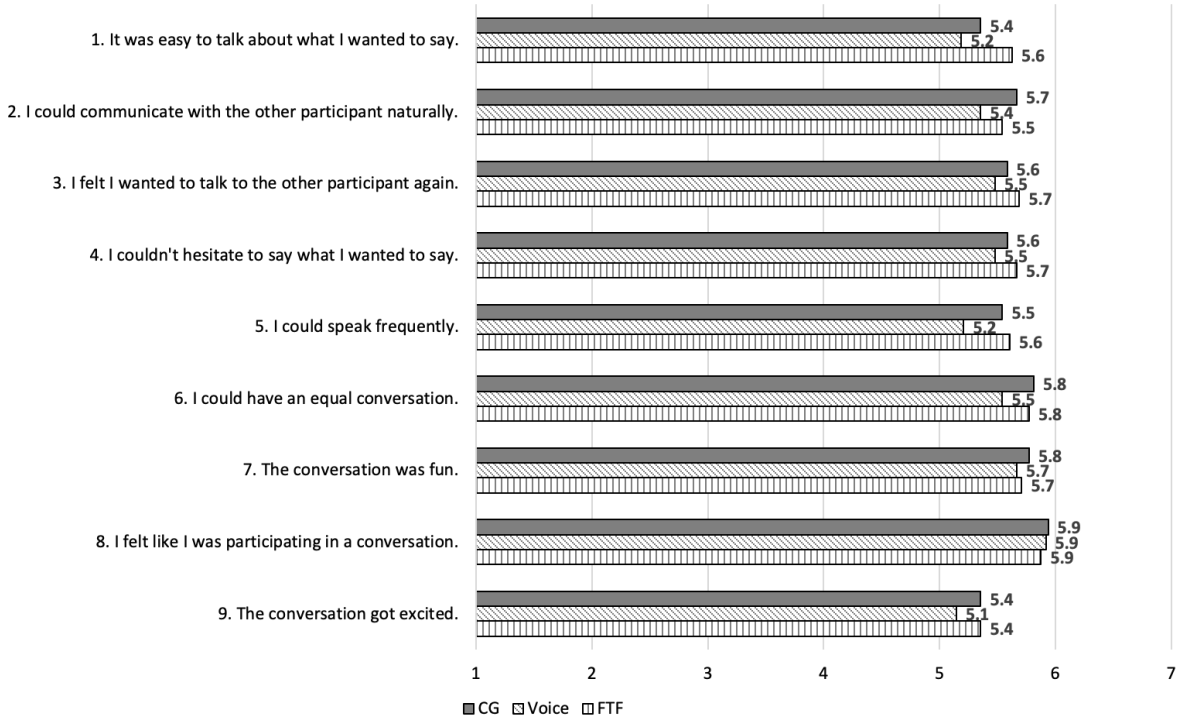


Figure 8: Questionnaire result for conversation.

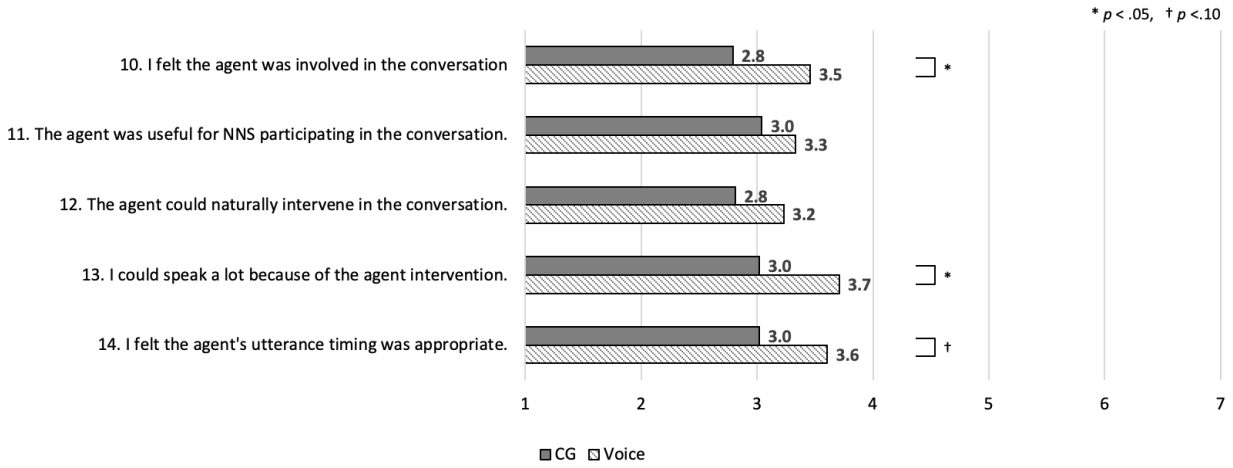


Figure 9: Questionnaire result for the agent usefulness.

6.2 Influence of the Agent Representations on Subjective Impression

From the questionnaire, the voice agent was better accepted than the CG agent in some question items, which related to the agent usefulness. Nevertheless, the average scores for the voice agent were lower than neutral, so they were not highly rated; the both agents were rated low, meaning that CG agents were rated even lower.

It might be possible that these lower ratings for the agents, especially for the CG character agent, were the reflection of the over expectation for an agent with very limited capability mentioned in Section 2.3.

For other question items, the significant difference was not found between the conditions. One of the possible reasons may be that the CG character appearance does not play any role in this conversational agent system. Actually, only the voice element of the agent system was used for intervening in the conversation, which could make the visual element not useful.

There were no significant differences between any of the agent conditions in comparison to the FTF. In other words, the agents did not have a subjectively noticeable effect on

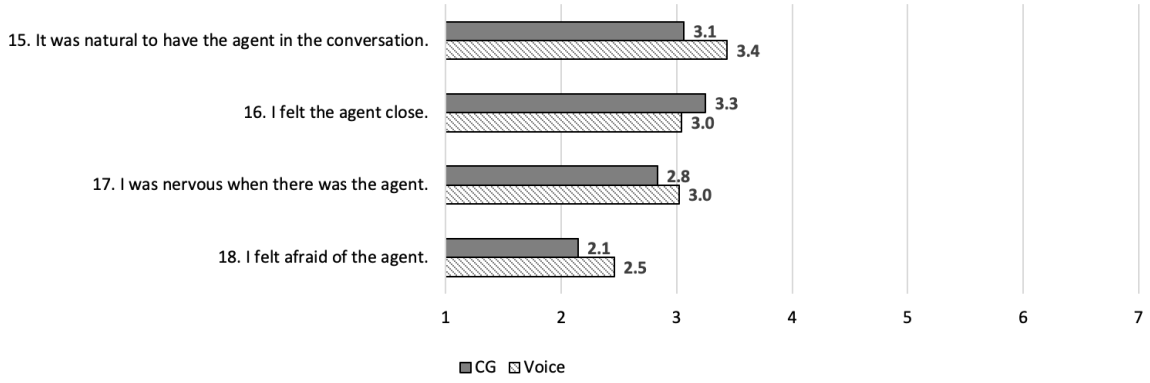


Figure 10: Questionnaire result for the agent impression.

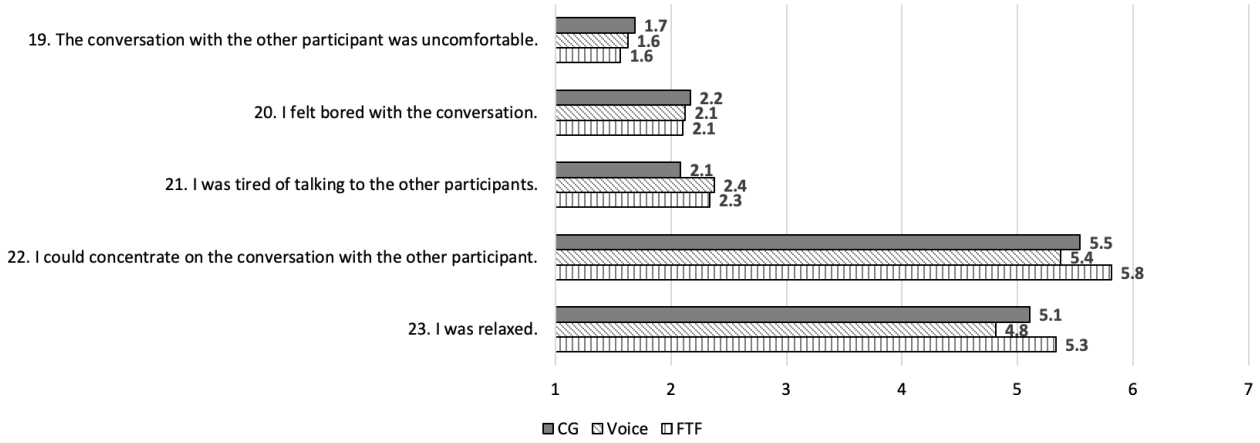


Figure 11: Questionnaire result for stress.

conversation or stress. The objective data showed that the agents worked in a certain way on the conversation, but were not very aware of it by the participants.

Another possibility that the agents did not get noticeable difference might be the positions where the agents were placed. The agents were placed on the side of the participants as in Fig. 2. This setting could certainly affect the magnitude of an agent's impact, in particular the magnitude of the visual agent's impact, and the results. Thus the agent placement is the information needed to reproduce the results.

6.3 A Mediating Conversational Agent to Support Second Language Conversation

A summary of the findings of this study are as follows.

1) In many cases, NNS actually acquires a speaking turn and starts the next utterance by an agent who participates in the conversation between NNS and NS and takes a simple action of giving the right to speak to NNS.

2) The voice agent is more effective than the CG agent in transferring the right to speak, and is also perceived to be more useful.

3) The addition of the agent (who intervenes only during long silences of more than 2 seconds) does not affect the conversation very much, nor does the participants feel any effect.

In this study, as a second language conversation support system, we investigated the influence of the representation of the agent who intervenes in the conversation, based on the related research that the agent representation has a psychological effect. It was possible that the influence of CG was not seen from the similarity with [10], the results were somewhat surprising and the CG agent performed worse. As a study with similar direction of the result, in classical mediated communication experiments of problem-solving tasks, it is known that audio is an essential medium and video has little impact [30]. It can be said that the usefulness of media such as audio and video depend largely on what is done. CG was not only unnecessary in the agent's role in this experiment, but it may also have undercut expectations of competence for human-like CG.

Of course, the results of this experiment may also depend on the appearance and behavior of the CG agent used in this experiment. The visual impact may be increased by using an agent that gives a stronger impression, or an agent that has

facial expressions or gestures. The same applies to the agent placement described in 6.2. This is the limitation of this study.

How can we use such an agent? The good news is that a voice agent would be sufficient for this purpose, as you don't have to make any visual parts (if it's not a special one). If you just want to output sound, you only need a loudspeaker, which makes it more portable. For example, you don't have to sit down at a dedicated conference table where a CG agent shows up to start a conversation. You can use it for a little standing talk by running it on a smartwatch. Since conversations occur everywhere, it's nice to have the system easier to use. If it becomes easier to use, it is conceivable to improve the practicality by working on a method of voice intervention in the future.

7 CONCLUSION

In this research, we compare a CG character agent and a voice agent to support second language conversation. In the evaluation by 24 pairs of participants in the experiment, the intervention in conversation by the agent generally resulted in passing the speaking turn to NNS successfully, but more when the agent was voice-only. From the results of questionnaire survey, the voice agent gave an impression that it was more useful than the CG character agent. This finding indicates that visuals are not always important when designing agents to intervene with people, but rather demonstrates the availability of agents in environments where providing sufficient visuals is not easy, such as mobile environments.

REFERENCES

[1] W. Zhu, “Interaction and Feedback in Mixed Peer Response Groups”, Journal of Second Language Writing, Vol.10, No.4, pp.251-276 (2001).

[2] T. Hifumi, “Relationship of Linguistic Processing and Psychological Processing in Conversations Between Native Speakers of Japanese and Nonnative Speakers”, Japanese Journal of Educational Psychology, Vol.47, pp.490-500 (1999).

[3] K. Okamoto, and T. Yoshino, “Development of Face-to-face Intercultural Communication Support System using Visualized Keyword of Conversation”, Proceedings of Forum on Information Technology, Vol.8, No.3, pp.393-396 (2009).

[4] H. Hanawa, X. Song, and T. Inoue, “Communication Support by a Native Speaker’s Entering Keywords on Discussion with a Non-native Speaker - Effects of Entering Keywords by a Native Speaker on Audio Conference -”, IEICE Technical Report, Vol.116, No.31, pp.139-144 (2016).

[5] J. Ye, and T. Inoue, “A Speech Speed Awareness System for Non-Native Speakers”, Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion, pp.49-52 (2016).

[6] Z. Guo, M. Tang, and T. Inoue, “A Conversational Agent for a Non-native Speaker to Talk with a Native Speaker”, IEICE Technical Report, Vol.117, No.509, pp.107-112 (2018).

[7] S. Yamada, “Are Robots Necessary for Human-Agent Interaction?”, Proceedings of the 23rd Annual Conference of the Japanese Society for Artificial Intelligence, Vol.23, No.1G2-OS3-11 (2009).

[8] H. Huang, S. Otogi, R. Horita, and K. Kawagoe, “Toward a Guide Agent Who Actively Provides Information in Multi-user Conversation – A Trial in Finding the Timings for Intervention”, Transactions of Japanese Society for Artificial Intelligence, Vol.31, No.1, 10.1527/tjsai.DSF-514 (2016).

[9] I. Akiba, Y. Matsuyama, and T. Kobayashi, “Procedures of Obtaining Initiatives for Multiparty Conversation Facilitation Robots”, IPSJ SIG Technical Reports, Vol.2013-SLP-97, No.10 (2013).

[10] H. Naito, S. Kawamura, and Y. Takeuchi, “An Empirical Study of Indicating Reliable Messages in Real World Interaction”, The Transactions of the IEICE. A, Vol.92, No.11, pp. 840-851 (2009).

[11] Y. Takeuchi, and Y. Katagiri, “Inducing Social Agreement Responses Toward Interface Agent”, IPSJ Journal, Vol.41, No.5, pp.1257-1266 (2000).

[12] T. Hara, and K. Terada, “Gankyu robot ni yoru hishikan ga doutokusei koujou ni ataeru eikyo”, IPSJ SIG Technical Reports, Vol.2015-HCI-164, No.1 (2015).

[13] J. Rubin, “A Review of Second Language Listening Comprehension Research”, The Modern Language Journal, Vol.78, No.2, pp.199-221 (1994).

[14] H. Sacks, E. A. Schegloff, and G. Jefferson, “A Simplest Systematics for the Organization of Turn-Taking for Conversation”, Language, Vol.50, No.4, Part1, pp.696-735 (1974).

[15] T. Yoshino, M. Yatsushiro, Y. Takase, and Y. Nakano, “Intervening in Multiparty Conversations by Conversational Agents based on Dominance Estimation”, Proceedings of the 29th Annual Conference of the Japanese Society for Artificial Intelligence, Vol.29, No.1I4-4, pp.1-3 (2015).

[16] “Objectives and History | JLPT Japanese-Language Proficiency Test”, http://www.jlpt.jp/e/about/purpose.html, last accessed 2020/09/11.

[17] “Competence Required for Each Level | JLPT Japanese-Language Proficiency Test”, https://www.jlpt.jp/e/about/levelsummary.html, last accessed 2020/09/11.

[18] “mmdagent.jp”, http://www.mmdagent.jp/, last accessed 2020/09/11.

[19] “PyAudio Documentation”, https://people.csail.mit.edu/hubert/pyaudio/docs/, last accessed 2020/09/11.

[20] “Open JTalk - HMM-based Text-to-Speech System”, http://open-jtalk.sp.nitech.ac.jp/, last accessed 2020/09/11.

[21] B. F. Freed, N. Segalowitz, and D. P. Dewey. “CONTEXT OF LEARNING AND SECOND LANGUAGE FLUENCY IN FRENCH: Comparing Regular Classroom, Study Abroad, and Intensive Domestic Immersion Programs”, Studies in Second Language Acquisition, Vol.26, No.2, pp.275-301 (2004).

[22] R. Liu, and T. Inoue, “Application of an Anthropomorphic Dining Agent to Idea Generation”, Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp.607-612, (2014).
[23] T. Inoue, “Naturalistic Control of Conversation by Meal: Induction of Attentive Listening Attitude through Uneven Meal Distribution in Co-dining”, Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp.601-606 (2014).
[24] T. Takahashi, M. Hamasaki, and H. Takeda, “Web Community System Mediated by Avatar-like Agents”, Proceedings of the 15th Annual Conference of the Japanese Society for Artificial Intelligence, 1F1-10 (2001).
[25] Y. Hayashi, and K. Miwa, “Cognitive and Emotional Characteristics in Human-Human / Human-Agent Interaction”, The Transactions of Human Interface Society, Vol.10, No.4, pp.445-455 (2008).
[26] Y. Hayashi, E. Cooper, V. Kryssanov, A. Urao, and H. Ogawa, “Psychological Characteristics on Communication with a Conversational Agent - A Study on Schema and Embodiment -”, Transactions of Japan Society of Kansei Engineering, Vol.11, No.3, pp.459-467 (2012).
[27] A. Shamekhi, Q. V. Liao, D. Wang, R. K. E. Bellamy, and T. Erickson, “Face Value? Exploring the Effects of Embodiment for a Group Facilitation Agent”, Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp.391-403 (2018).
[28] S. K. Maynard, Discourse Analysis, Kuroshio Shuppan (1993).
[29] N. Yoshida, K. Takanashi, and Y. Den, “Taiwa ni okeru aiduchi hyougen no nintei to sono mondaiten ni tsuite”, Proceedings of the Annual Meeting of the Association for Natural Language Processing, Vol.15, pp.430-433 (2009).
[30] A. Chapanis, “Interactive Human Communication”, Scientific American, Vol.232, pp.36-42 (1975).

(Received November 12, 2019)
(Revised April 6, 2020)

Tomoo Inoue is Professor of the Faculty of Library, Information and Media Science of University of Tsukuba, Japan. His research interests include Computer-Supported Cooperative Work, Human-Computer Interaction and Educational Technology. He received his Ph.D. in Engineering from Keio University in 1998. He is a recipient of awards including Outstanding Paper Award, Activity Contribution Award and SIG Research Award from Information Processing Society of Japan (IPSJ). He has served a number of academic committees, including IPSJ Journal Group Editor-in-Chief, IPSJ Transactions on Digital Content Editor-in-Chief, IEICE Technical Committee on Human Communication Science Chair, IPSJ SIG Groupware and Network Services Steering Committee Secretary, ACM CSCW Papers Associate Chair, CollabTech Steering Committee, and IEEE TC CSCWD. He has co-authored and edited "Idea Generation Methods and Collaboration Technologies (Kyoritsu Shuppan)(in Japanese)," and "Communication and Collaboration Support Systems (IOS Press)" among others.

Kumiko Kawai received her M.Sc. in Informatics from University of Tsukuba in 2019. She is currently with Ateam Inc. Her research interests include human interface and interaction.

Boyu Sun received his B.Sc. in Engineering from Huaqiao University in 2018. He is currently a graduate student in University of Tsukuba, Japan. His research interests include human-computer interaction.

Kosuke Sasaki received his M.Sc. in Informatics from University of Tsukuba in 2016. He is currently a Ph.D student in University of Tsukuba, Japan. His research interests include Computer-Supported Cooperative Work.

Submission Guidance

About IJIS

International Journal of Informatics Society (ISSN 1883-4566) is published in one volume of three issues a year. One should be a member of Informatics Society for the submission of the article at least. A submission article is reviewed at least two reviewer. The online version of the journal is available at the following site: http://www.infsoc.org.

Aims and Scope of Informatics Society

The evolution of informatics heralds a new information society. It provides more convenience to our life. Informatics and technologies have been integrated by various fields. For example, mathematics, linguistics, logics, engineering, and new fields will join it. Especially, we are continuing to maintain an awareness of informatics and communication convergence. Informatics Society is the organization that tries to develop informatics and technologies with this convergence. International Journal of Informatics Society (IJIS) is the journal of Informatics Society.

Areas of interest include, but are not limited to:

- Internet of Things (IoT)
- Smart Cities, Communities, and Spaces
- Big Data, Artificial Intelligence, and Data Science
- Network Systems and Protocols
- Computer Supported Cooperative Work and Groupware
- Security and Privacy in Information Systems
- Intelligent Transportation System
- Distributed Computing
- Multi-media communication
- Information systems
- Mobile computing
- Ubiquitous computing

Instruction to Authors

For detailed instructions please refer to the Authors Corner on our Web site, http://www.infsoc.org/.

Submission of manuscripts: There is no limitation of page count as full papers, each of which will be subject to a full review process. An electronic, PDF-based submission of papers is mandatory. Download and use the LaTeX2e or Microsoft Word sample IJIS formats.

http://www.infsoc.org/IJIS-Format.pdf

LaTeX2e

LaTeX2e files (ZIP) http://www.infsoc.org/template_IJIS.zip

Microsoft Word™

Sample document http://www.infsoc.org/sample_IJIS.doc

Please send the PDF file of your paper to secretariat@infsoc.org with the following information:

Title, Author: Name (Affiliation), Name (Affiliation), Corresponding Author. Address, Tel, Fax, E-mail:

Copyright

Publisher

Address: Informatics Laboratory, 3-41 Tsujimachi, Kitaku, Nagoya 462-0032, Japan

E-mail: secretariat@infsoc.org

CONTENTS