Generation of Complexity-Controlled Mathematics Problems for Colleges

Yuki Mori¹ and Shuichi Yukita²
¹Graduate School of Computer and Information Sciences, Hosei University, Tokyo, Japan
²Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan

Abstract—Exercises with adequate complexity motivate students and facilitate a deeper understanding. Manually constructing such problems consumes time that teachers can otherwise use to mentor students. Many software tools and services for automatic generation of mathematics problems are available on the web, but they provide only materials up to high school level. In addition, no standardized methods are provided to evaluate and control the computational complexity of generated problems. In this paper, we propose a framework for evaluating computational complexity from the learners’ perspective, aiming to apply our framework to the automatic generation of college-level mathematics problems with controlled computational complexity. Our framework helps teachers prepare learning materials and thereby save time for mentoring students.

Keywords: Linear Algebra, Computational complexity, Hermitian matrix, Unitary matrix

1. Introduction

We propose a framework for evaluating the computational complexity of college-level mathematics problems, with the aim of applying our framework to the automatic generation of such problems controlled computational complexity.

Providing students with suitably complex practice problems is crucial for motivating them and facilitating deeper understanding. Manually constructing such problems consumes time, which mathematics teachers can otherwise use to mentor students.

Many software tools and services for automatic generation of mathematics problems are available on the web, but they provide only materials up to high school level. In addition, no standardized methods are provided to evaluate and control the computational complexity of the generated problems. Among the popular web sites and services, we list some examples. Wolfram Problem Generator[TM] [1] and Davitily Math Problem Generator[TM] [2] deal with mathematics problems for high school students. SuperKids Math Worksheet Creator[TM] [3] deals with arithmetic problems for children attending elementary schools. However, our framework is new as it deals with math at the college level and introduces suitable methods for evaluating computational complexity from the learners’ perspective.

Figure 1 presents a graphical user interface (GUI) for generating eigenvalue problems using Mathematica[TM]. Here, the user selects parameters such as the algebraic number field used in the calculation, the number of the calculation steps, and the matrix dimension, which determine the computational complexity of the generated problem. The user then selects the problem category and provides the required parameters that control the computational complexity. To avoid excessive selection for busy users, predefined sets of recommended parameters are also stored in the system, and thus ease selection for the user. Problems with the required complexity along with model answers are generated. Eigenvalue is a topic usually taught in linear algebra courses during engineering, however, most textbooks do not provide sufficient practice problems. As a result, teachers must create additional problems to be used in classes, assignment, and exams. In this paper, we present an automatic generation of diagonalization problems for Hermitian matrices to illustrate

---

---
the relevance of our proposed framework.

2. Complexity from Learners’ Perspective

Rigorous concepts of complexity are available in various forms in standard textbooks such as [5]. We deal with a different variety of complexity, subjective complexity, where complexity is measured by the difficulty that learners’ perspective. Designing practice problems that are sufficiently complex, but not excessively, are crucial for keeping a learner motivated. We propose a new framework for estimating such computational complexity and demonstrate its relevance by developing a framework for automatic generation of complexity-controlled practice problems. Our framework enables us to

1) control the number of the calculation steps,
2) limit the height of rational numbers involved in a calculation, and
3) deal with algebraic numbers.

The computational complexity of a generated problems is defined as the approximate sum of the heights of the rational numbers (the maximum ratio of the absolute values of the denominator and numerator) appearing in its model solution. In the hope of extending our work to other mathematics problems, we incorporated algebraic number fields in our system. The user can select the calculation field from the rational number field and other algebraic fields extended by irrational numbers, especially, quadratic irrational numbers, and fourth-power irrational numbers.

3. Automatic Generation of Eigenvalue Problems

Eigenvalue problems are usually taught in linear algebra courses during engineering. Eigenvalue problems appear in two forms: diagonalization of Hermitian matrices and Jordan canonicalization of linear transformations. In this paper, we deal with diagonalization of Hermitian matrices. The process of generating complexity-controlled eigenvalue problems includes

1) predefining unitary matrices,
2) generating eigenvalues,
3) generating \( n \) Hermitian matrices, and
4) selecting matrices suitable for exercises.

We explain each step in detail. First, we generate nearly the entire set of tractable unitary matrices and classify them by algebraic number fields. We define an original function that can extract all of the irrational numbers appearing in the entries of a tentatively generated matrix. For example, this function returns the list \( \{ -1, \sqrt{2}, \sqrt{3}, \sqrt{5} \} \), where

\[
\left( \frac{1}{\sqrt{3}} \quad \frac{\sqrt{3}}{2} \quad \frac{\sqrt{5}}{7} \right)
\] (1)

is entered as a primary material that is later subject to Gram-Schmidt orthonormalization. Diagonalizing a given Hermitian matrix requires

1) calculating eigenvalues,
2) determining the eigenvector that corresponds to each eigenvalue, and
3) constructing a unitary matrix using these eigenvectors.

Hence, the number of calculation steps does not vary once the dimension is determined. General Hermitian matrices can be generated from a diagonal matrix \( D \) and a unitary matrix \( U \) as

\[
H = UDU^†
\] (2)

where \( U^† \) is the conjugate transpose matrix of \( U \). Equation (2) is rewritten as \( D = U^†HU \). The most difficult part is generating a unitary matrix with the specified properties. However, the number of matrices suitable for this purpose is relatively small because the entries of those matrices must be obtained from a given algebraic number field, and the heights of the involved rationals must be restricted, furthermore, all of the column vectors must form an orthonormal system. Therefore it is possible to predefined almost entire sets of unitary matrices that can be used to generate Hermitian matrices that can be diagonalized with specified complexity.

3.1 Generation of Unitary Matrices

This section describes the generation of \( 3 \times 3 \) unitary matrices through examples. The procedure for generating a matrix comprises four major steps:

1) generating a unit column vector,
2) verifying that all the entries belong to the given number field,
3) constructing an orthonormal basis from two other linearly independent column vectors using the Gram-Schmidt procedure, and
4) verifying again that all of the entries of those basis vectors belong to the given number field.

In step 1, we generate various unit vectors that are of the form in Equation (3).

\[
e_1^t = \left( \pm \frac{i}{\sqrt{k}}, \pm \frac{l}{m\sqrt{n}}, \pm \sqrt{1 - \left( \left( \frac{i}{\sqrt{k}} \right)^2 + \left( \frac{l}{m\sqrt{n}} \right)^2 \right)} \right)
\] (3)

where \( i, j, l, \) and \( m \) are rational integers, and \( k \) and \( n \) are 2, 3, 5, 7, or 1. In step 2, all of the irrational numbers such as \( \sqrt{2}, \sqrt{3}, \) and \( \sqrt{-1} \) are extracted from the vector and matrix. We select the vectors whose entries belong to the specified number field. In step 3, \( e_1 \) and two other vectors are orthogonalized. Only an additional two vectors are required to form a linearly independent triple together with \( e_1 \). Hence, for easy calculations, we can take them from sparse matrices, where only the positions of nonzero entries are important. Figure 2 indicates that the field cannot be retained after orthogonalization.
Fig. 2: Example of orthogonalization. The upper part shows failure in retaining the number field after orthogonalization in the case in which $(1, 0, 0)^t$ and $(0, 0, 1)^t$ are appended. The other shows success in retaining the field, when $(0, 0, 1)^t$ and $(0, 0, 1)^t$ are appended.

Fig. 3: Example of expansion of the orthogonalization to imaginary matrices.

We generated three patterns of unitary matrices by changing the position of the non-zero element of each vector. If the user wants a complex number field, then $\sqrt{-1}$ must be added in some entry of an initial vector (see Fig. 3). In step 4, the number field of the components is verified again. This step is necessary because Gram-Schmidt orthogonalization takes square roots which may cause further algebraic extension of fields. We select matrices all of whose entries have rationals of low heights in their subexpressions. These forms the basic set of tractable unitary matrices. Of the 500,000 generated unitary matrices in a preliminary stage, the filter selects 681 matrices according to the criterion described in later sections. The number of predefined matrices are listed in Tables 1 and 2. Though the basic set is relatively small (681), we can generate other tractable matrices by multiplying them among themselves and by taking direct sums as follows: given two matrices of the same dimension

$$U_1, U_2 \in U(n).$$

we obtain

$$U_1 U_2 \in U(n).$$

(5)

Given two matrices of possibly different dimensions

$$U_1 \in U(m) \text{ and } U_2 \in U(n),$$

we obtain

$$U_1 \bigoplus U_2 \in U(m+n).$$

(7)

Note that the number field involved is preserved under both multiplication and direct sum operations. For example,

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$  

(8)

Using such methods, we can obtain sufficient unitary matrices, as presented in Tables 1 and 2.

4. Demonstration

We demonstrate the automatic generation of eigenvalue problems for Hermitian matrices and evaluate the complexity of the generated problems. Figure 4 presents the results of ten generated Hermitian matrices. The dimension of each

$$U_1 \text{ and } U_2 \in U(n),$$

(4)

Table 1: Predefined orthogonal matrices

<table>
<thead>
<tr>
<th>Field</th>
<th>Dimension</th>
<th>$2 \times 2$</th>
<th>$3 \times 3$</th>
<th>$4 \times 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>65</td>
<td>20</td>
<td>4245</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{2})$</td>
<td>27</td>
<td>33</td>
<td>2503</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{3})$</td>
<td>12</td>
<td>10</td>
<td>930</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{5})$</td>
<td>20</td>
<td>25</td>
<td>1714</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{7})$</td>
<td>12</td>
<td>3</td>
<td>927</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Predefined unitary matrices

<table>
<thead>
<tr>
<th>Field</th>
<th>Dimension</th>
<th>$2 \times 2$</th>
<th>$3 \times 3$</th>
<th>$4 \times 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(\sqrt{-1})$</td>
<td>129</td>
<td>20</td>
<td>25066</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{2}, \sqrt{-1})$</td>
<td>65</td>
<td>33</td>
<td>8488</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{3}, \sqrt{-1})$</td>
<td>30</td>
<td>11</td>
<td>2862</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{5}, \sqrt{-1})$</td>
<td>68</td>
<td>26</td>
<td>9068</td>
<td></td>
</tr>
<tr>
<td>$Q(\sqrt{7}, \sqrt{-1})$</td>
<td>24</td>
<td>5</td>
<td>2142</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4: Samples of generated Hermitian matrix
Table 3: Time (sec) for generating problem 1

<table>
<thead>
<tr>
<th>2 x 2</th>
<th>3 x 3</th>
<th>4 x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>5.33</td>
<td>17.14</td>
</tr>
<tr>
<td>Q(\sqrt{2})</td>
<td>702.81</td>
<td>1185.03</td>
</tr>
<tr>
<td>Q(\sqrt{3})</td>
<td>702.00</td>
<td>1324.84</td>
</tr>
<tr>
<td>Q(\sqrt{5})</td>
<td>353.91</td>
<td>1677.19</td>
</tr>
<tr>
<td>Q(\sqrt{7})</td>
<td>320.05</td>
<td>3154.67</td>
</tr>
</tbody>
</table>

Table 4: Time (sec) for generating problem 2

<table>
<thead>
<tr>
<th>2 x 2</th>
<th>3 x 3</th>
<th>4 x 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(\sqrt{-1})</td>
<td>110.61</td>
<td>170.72</td>
</tr>
<tr>
<td>Q(\sqrt{2}, \sqrt{-1})</td>
<td>655.34</td>
<td>2175.56</td>
</tr>
<tr>
<td>Q(\sqrt{3}, \sqrt{-1})</td>
<td>599.29</td>
<td>4585.61</td>
</tr>
<tr>
<td>Q(\sqrt{5}, \sqrt{-1})</td>
<td>691.75</td>
<td>3439.33</td>
</tr>
<tr>
<td>Q(\sqrt{7}, \sqrt{-1})</td>
<td>5543.17</td>
<td>22911.11</td>
</tr>
</tbody>
</table>

matrix, multiplicity of eigenvalues, algebraic number field, height h of numerical calculation, and number n of problems is 3, 1, \sqrt{-1}, 100, and 10, respectively. The system generates n problems on demand. Each matrix

1) has a maximum absolute value of involved rationals less than h,
2) belongs to specified number field, and
3) differs from already generated matrices.

In this case, generating ten problems took 4.16 seconds. The time measurements for generating problems are listed in Tables 3 and 4.

Tests were conducted generating 1,000 problems for each number field. Generating one problem takes 1.18 seconds on average. As the tables indicate, generation consumes more time when the number field is complex because it is because the system needs to decompose all of the matrix entries to check the maximum height of the rationals.

As an experiment, we asked some college science students to solve the generated problems and some non-complexity-controlled (without our methods) problems. On average, the former case, students required 2 min to calculate eigenvalues and 3 min to construct a unitary matrix, while for the latter case students required 6 min to calculate eigenvalues and 7 min to form a unitary matrix. These results demonstrate the validity of our proposed framework. Large-scale verification experiments will be conducted out with the cooperation of university teachers.

5. Conclusion

In this paper, we developed a new framework for evaluating and controlling computational complexity from the learners’ perspective. In addition, we developed an automatic generation system for eigenvalue problems according to our proposed framework. The automatic generation of complexity-controlled eigenvalue problems is one of the sample implementations that validate our proposed framework. Controlling the complexity of eigenvalue problems involves restricting the number of calculation steps, the height of the involved rationals, and the algebraic number field appearing in a model solution. Small-scale experiments illustrated the relevance of our proposed framework. Constructing problems with sufficient computational complexity is essential for maintaining learners’ motivation. Our framework helps teachers prepare learning materials and thereby save time for mentoring students.

We expect our technique to be applicable to other subjects in linear algebra and analysis. Our system paves a path to strict quantitative control on various aspects of complexity from the learners’ perspective. Future work will include the application of our methods to other areas such as differential equations, and number theory. We also intended to show that this system has a positive effect on college-level engineering education. Therefore, demonstration experiments in classroom will be the next step of our research.

References