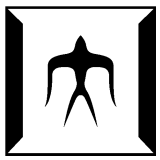
論文 / 著書情報
Article / Book Information

題目(和文)	化合物群の部分構造の共通性に着目した高速バーチャルスクリーニング手法の開発
Title(English)	Fast structure-based virtual screening with commonality of compound substructure
著者(和文)	柳澤渓甫
Author(English)	Keisuke Yanagisawa
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11198号, 授与年月日:2019年3月26日, 学位の種別:課程博士, 審査員:秋山 泰,篠田 浩一,村田 剛志,関嶋 政和,石田 貴士
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11198号, Conferred date:2019/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Fast structure-based virtual screening with commonality of compound substructure

Keisuke Yanagisawa



Department of Computer Science
School of Computing
Tokyo Institute of Technology

Supervisor: Yutaka Akiyama

A Thesis Submitted for the Degree of *Doctor of Engineering*

20 March 2019

This dissertation partly used the published articles:

- Chapters 2, 3: © Yanagisawa *et al.*, 2017. Published by Oxford University Press. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/).
- Chapters 4, 5: © Yanagisawa *et al.*, 2018. Published by Elsevier Ltd. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/).

Abstract

Virtual screening (VS) is a computational method which is widely used to evaluate the binding affinity between a vast number of compounds and a protein before conducting *in vitro* assays. In particular, structure-based VS (SBVS), which uses protein tertiary structures, is attracting attention because it can consider the binding affinity in a physico-chemical manner. However, SBVS is too computationally heavy to evaluate 10 million-order of compounds. Pre-screening methods filtering out unfeasible compounds have been proposed to avoid the problem of computational cost. Most of them are ligand-based, prediction from known inhibitors, and structure-based pre-screening methods are still needed to be accelerated.

In this dissertation, we designed a pipeline for faster virtual screening with a fundamental idea: substructure commonality. Commonality can be used to reuse of intermediate results. We firstly defined "fragment" to maximize meaningful common substructures and we revealed our definition express compounds with much smaller number of fragments than the RECAP, an existing decomposition rule. The fact leads to faster calculation of pre-screening and docking calculation.

As a first step of the structure-based virtual screening pipeline, we proposed the pre-screening method, Spresso. It evaluates compounds based on the binding scores of their fragments, and thus the calculation results of fragments are reused to multiple compounds. It is capable of exhaustive primary screening and completed pre-screening approximately 200 times faster than the conventional method (Glide HTVS) while precision is slightly compromised. When Spresso was applied to a more than 26 million compounds, the calculation time was within a day.

The second step of the pipeline is docking calculation. We designed a docking calculation procedure with reuse of intermediate results. The procedure is based on fragment-based anchor and grow algorithm.. We found that one of the intermediate information, called the fragment grid, is an essential factor while it spends huge memory space; therefore, we formulated and reduced the problem of optimization of the fragment grids reuse as the minimum cost flow problem, which is known as a polynomial-time problem. Because the reduced graph of the problem is topological sortable and sparse, we finally proposed approximately 9 times faster algorithm.

Acknowledgements

I would like to express my sincere gratitude to Professor Yutaka Akiyama for providing me an excellent research environment and continuous guidance and encouragement throughout my research work. I also thank Assistant Professor Masahito Ohue for his valuable comments and discussions on my research. Associate Professor Takashi Ishida also provided valuable suggestions and supports. I also thank all members of Professor Akiyama's laboratory for helpful discussions and for the wonderful time spent together. I wish to express my appreciation to Mr. Shunta Komine, Mr. Shogo D. Suzuki, and Mr. Rikuto Kubota for considerable suggestions to my research work. I would also like to thank Ms. Kanako Ozeki, the secretary of the Akiyama laboratory, for her much appreciated support during my six years of laboratory life.

I also would like to thank Professor Daisuke Kihara at Purdue University, West Lafayette, USA for helpful discussions about my research related to compound decomposition. I wish to express my gratitude to my dissertation committee, Professor Koichi Shinoda, Associate Professor Tsuyoshi Murata, and Associate Professor Masakazu Sekijima at the Tokyo Institute of Technology for their inspiring feedback and valuable comments on my research and the dissertation.

I would like to express my sincerest appreciation to Japanese Society for the Promotion of Science (JSPS) Research Fellow (DC2), Ministry of Education, Culture, Sports, Science and Technology (MEXT) Program for Leading Graduate Schools "Education Academy of Computational Life Sciences (ACLS)" led by Professor Yutaka Akiyama, Japan Science and Technology Agency (JST) Core Research for Evolutionary Science and Technology (CREST) strategic basic research program "EBD: Extreme Big Data - Convergence of Big Data and HPC for Yottabyte Processing" led by Professor Satoshi Matsuoka at Tokyo Institute of Technology, JST Research Complex Program "Wellbeing Research Campus: Creating new values through technological and social innovation", MEXT Regional Innovation and Ecosystem Formation Program "Program to Industrialize an Innovative Middle Molecule Drug Discovery Flow through Fusion of Computational Drug Design and Chemical Synthesis Technology",

and AMED Platform Project for Supporting Drug Discovery and Life Science Research (Basis for Supporting Innovative Drug Discovery and Life Science Research) for financial support throughout my research works. Moreover, this study was supported in part by a Grant-in-Aid for Scientific Research (A) 24240044 from the MEXT program, a Grant JPMJCR1303 from the JST CREST program, and a Grant-in-Aid for JSPS Fellows 17J06897 from the MEXT program.

Last but certainly not the least, I would like to extend a special thanks to my family, Kazuyo Yanagisawa and Takeyuki Yanagisawa. Without their kind support and constant encouragement, this work would have not been possible. Finally, I dedicate this work to my family.

Contents

List of Figures

List of Tables

Part I

General Introduction

Chapter 1

Introduction

1.1 Drug Development

The development process of novel drugs is called "Drug development" or "Drug design", and pharmaceutical companies spends much money for R&D, research and development. The process can be mainly divided into three parts:

Discovery and Development The aim of this step is obtaining potential drug candidates, and it consists of below steps.

- **Target selection** a step to find a target protein related to a specific disease. The function of protein is also considered in terms of safety.
- **Lead generation** a step to find some feasible compounds which bind to the target protein from huge number of compounds. The step is also called "Hit identification".
- **Lead optimization** a step to optimize feasible compounds' efficacy, bioavailability and safety.

Preclinical Research The potential drug candidates are checked their efficacy and safety in cells and animals.

Clinical Research This step is to assess the efficacy and safety of drug candidates in humans. The step is separated into three steps, Phase I–III.

According to an article [1], these steps takes 4.5 years, 1.0 year, and 6.5 years, respectively. Furthermore, the estimated cost for a new approved compound is more than 1 billion dollars [2]. Preclinical Research and Clinical Research are based on *in*

vivo experiences and are unavoidable, thus much of computational effort for obtaining potential drugs and optimization of found candidates has been done. These methods are named as Computer-aided Drug Design (CADD).

1.2 Computer-aided Drug Design (CADD)

Thanks to the enlarged databases of compounds, protein tertiary structure, expression profiles, and assay results, vast number of computational methods have been proposed to improve the Discovery and Development step in the basis of accuracy, cost and speed. For the prediction of efficacy in the Lead generation step, virtual screening (VS) is proposed and done for lots of drug development campaigns [3, 4, 5, 6, 7]. On the other hand, ADMET prediction [8] and off-target prediction [9] has been proposed for the prediction of safety.

In this dissertation, we focused on VS because much more compounds are processed with it compared to safety prediction and thus VS step is usually computationally more expensive.

1.3 Virtual Screening

The identification of potential drug compounds is said as "finding needles in a haystack" [10]; thus, estimation of the likelihood for a compound to become a viable drug before conducting *in vitro* assays is critical in enhancing the effectiveness of searches. To estimate drug likelihood, virtual screening methods have been improved with the enlargement of available databases [11].

Virtual screening methods can be divided into three:

- Structure-based virtual screening (SBVS)
  SBVS methods utilize target protein tertiary structures. It does not require known compounds, and thus outputs are not biased toward knowns.
  - Protein–ligand docking [12, 13]
  - Molecular dynamics [13, 14]
  - Receptor-based pharmacophore [15, 16]
- Ligand-based virtual screening (LBVS)
  LBVS methods utilize known active and inactive compounds. It is considered as more accurate than SBVS methods.

  - Quantitative Structure-Activity Relationship (QSAR) [17, 18]
  - Machine learning [4, 19]
  - Similarity search [20, 21]
  - Ligand-based pharmacophore [16, 22]

- Chemical genomics-based virtual screening (CGBVS)
  It also called as drug-target interaction prediction. CGBVS methods utilize Protein–ligand interaction bipartite network. It predicts relasionship between multiple proteins and multiple compounds, thus it is mainly applied to off-target prediction and drug repositioning [23, 24, 25].

In particular, structure-based VS (SBVS) is attracting attention because it can consider the binding affinity in a physico-chemical manner and thus does not require any known drugs beforehand, whereas LBVS and CGBVS approaches need experimental data for similar protein-compound pairs. The absence of bias toward experimental compounds is also a strength of SBVS. It means the methods will output more diverse potential drugs, and thus researchers are able to select new scaffolds to avoid side effects or existing patents [26]. Additionally, the prediction ability of binding mode for each compound is also a reason SBVS is attracting attention, while most of ligand-based VS methods cannot estimate binding poses. The 3D structural information enables us to interpret experimental results in atomic-level, and plan which part of compound will be modified [6, 7].

Moreover, the availability of protein tertiary structures has increased in recent years. For example, the Protein Data Bank (PDB), which is the most popular public database of protein structures, contains $>$ 136,000 entries, a 9% increase in 2017 [27].

1.3.1 Structure-Based Virtual Screening (SBVS)

As we already mentioned, SBVS is based on 3D structures of a protein and compounds. Most of SBVS methods consider physical interaction, shape complementarity between a protein and a compound [28, 29, 30, 31]. Especially, protein-ligand docking is performed to estimate binding affinities and plausible binding modes for many drug candidates [12, 13, 32].

1.4 Protein-Ligand Docking

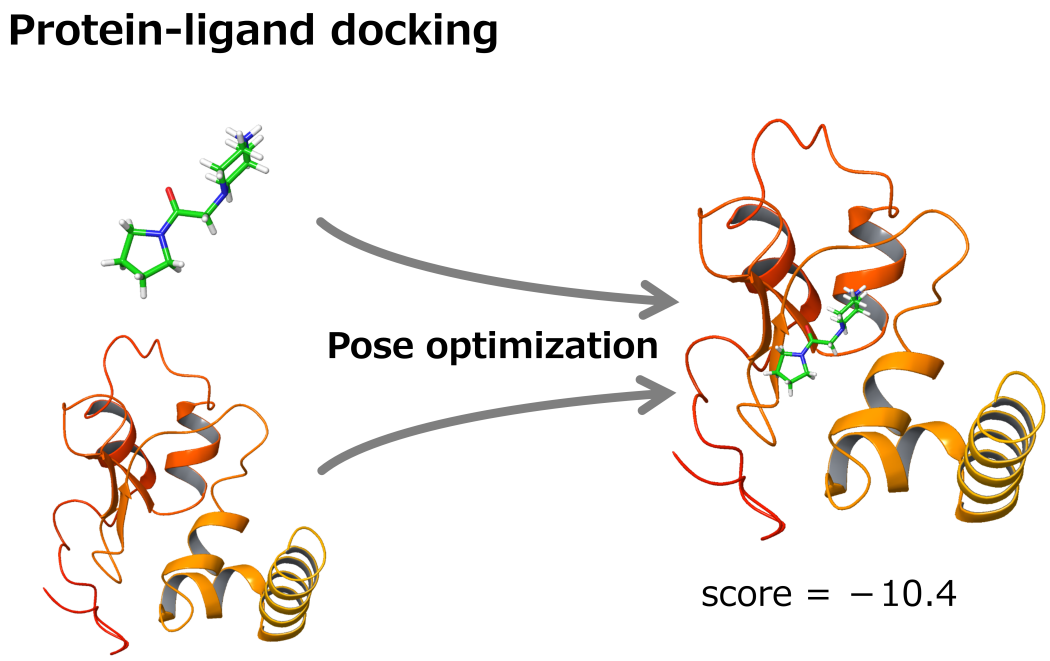
Protein-ligand docking is the most popular method in the SBVS methods since the calculation uses atom positions and bond information between atoms, thus it can consider physical interaction and compound's flexibility explicitly. For instance, hydrogen bond is one of the important factor of ligand binding, and it depends on the position of atoms and directions of bonds. This atomic-level evaluation is informative to understand what binding factors are important and to modify compounds more feasible and effective.

The inputs are a tertiary structure of target protein and a 3D-structure of a compound, while the outputs are predicted binding mode and its score, which usually mimics the binding energy. In order to output most stable binding mode, the docking method is a optimization problem (Fig. 1.1). The screening is done by considering docking scores [33] or re-ranking with predicted binding modes, including visual inspection [34, 35, 36].

Proprietary docking tools such as Glide [37], eHiTS [38], and open-sourced tools such as AutoDock [39], AutoDock Vina [40], DOCK [41] have been developed. Table 1.1 is a list of well-known docking tools. Glide, a proprietary tool, is one of the most accurate tool [42, 43] and is still improved in terms of scoring function [44], thus it is widely used [45].

This process, however, is computationally expensive [26]. The reason is docking is an optimization problem. The internal degrees of freedom of a compound is a significant factor associated with the search space and computation time required for docking simulations. Furthermore, the calculation speed highly depends on the algorithms and tools. For example, AutoDock 4.0 [39] spends $\sim$ 500 CPU core seconds per compound [40], whereas the commercial docking tool Glide [37] is 50-fold faster than AutoDock 4.0. It should be noted that the speediness of Glide is still not enough to evaluate millions of compounds. Additionally, parallel computation of some proprietary tools, such as Glide, is highly restricted in terms of license.

Researches to accelerate docking have been done [40, 46, 47]; however, its use is still not applicable to a huge collection of compounds such as ZINC ($>$ 20 million compounds) [48], PubChem ($>$ 60 million comounds) [49], ZINC15 ($>$ 120 million compounds) [50]. Furthermore, GDB-17, a virtually enumerated compound library have been also proposed ($\approx$ 166.4 billion compounds) [51]. These facts indicates that an acceleration of SBVS procedure has become a critical matter.



Scoring function

ex) Chemscore [51]

$$\Delta G_{bind} = \Delta G_{H\_bond} \sum_{H\_bond} f(\Delta R, \Delta\alpha) + \Delta G_{metal} \sum_{metal} f(\Delta R, \Delta\alpha) + \Delta G_{lipo} \sum_{lipo} f(\Delta R) + \Delta G_{rotor} \sum_{rotor} f(P_{nl}, P'_{nl}) + \Delta G_0$$

Figure 1.1: An example of protein-ligand docking. The factors of docking are pose optimization and scoring.

1.5 Pre-Screening

Because of slowness of docking calculation, the virtual screening campaigns often screen out compounds prior to the docking calculation, termed "pre-screening" [53]. The pre-screening methods can be divided into three types: 1. pre-screening by physicochemical descriptors, 2. pre-screening by 2D-structure of compounds, 3. pre-screening by rough docking calculation.

1. Pre-screening by physicochemical descriptors
   Existing drugs have some specific physicochemical values, e.g. molecular weight, water-octanol partition coefficient (logP), and thus there are filtering methods

Table 1.1: Examples of docking tools

Name	Compound-based or Fragment-based	Method	Free?	Articles
AutoDock 4.0	Compound-based	Genetic Algorithm	✓	[39]
AutoDock Vina	Compound-based	Simulated Annealing	✓	[40]
DOCK	Fragment-based	Anchor and Grow	✓	[41]
eHiTS	Fragment-based	Maximum Clique Finding	–	[38]
FlexX	Fragment-based	Anchor and Grow	–	[54]
FRED	Compound-based	Conformer Docking	–	[55]
Glide	Compound-based	Systematic Search	–	[37]
GOLD	Compound-based	Genetic Algorithm	–	[56, 57]
ICM	Compound-based	Biased Probability Monte Carlo	–	[58]
rDock	Compound-based	Genetic Algorithm	✓	[59]

with those descriptors.

- Lipinski's rule of five [60]
  It consists of four physicochemical rules to be feasible as orally available compounds.
- Quantitative Estimate of Druglikeness (QED) [61]
  It is a scoring method of druglikeness, which is based on the distributions of descriptors of known drugs.

These methods are independent from the target protein, and they evaluate bioavailability and safety of compounds.

2. Pre-screening by 2D structure of compounds
   Structurally similar compounds tend to bind to same targets, thus 2D structure of compound is informative to predict efficacy if there are known binders to the target protein. The prediction strategies are similar to the LBVS methods because of its speediness.

   - Fingerprint-based pre-screening [62]
     Fingerprint is a binary vector of a compound. Each bit represents a substructure of a compound. Similarity search and machine learning techniques are used with fingerprints for pre-screening.
   - Pharmacophore-based pre-screening [63]

It considers functionality of substructures, e.g. aromatic, hydrophobic, hydrogen bond acceptor, hydrogen bond donor.

3. Pre-screening by roughly docking calculation
   As we mentioned in section 1.4, docking is basically time consuming; however, rough optimization of conformation search enables compound pre-screening. For instance, Glide has a HTVS mode to calculate 10-fold faster than ordinal docking mode (SP mode), thus the HTVS is highly used as pre-screening, followed by SP docking calculation [64, 65, 66, 67].

Physicochemical-based and 2D structure-based approaches are widely used as pre-screening methods and can deal with vast numbers of compounds, since the approaches are computationally less expensive than docking-based approaches. Physicochemical-based approaches, however, filter out compounds not from protein-ligand binding point of view. Additionally 2D structure-based approaches rely on known active/inactive compounds. It means pre-screened compounds are biased toward known drugs and thus the strength of SBVS will be vanished.

Docking-based methods avoid this problem, but still require large computation times. For example, Glide HTVS and Panther [68] spends approximately 1 CPU core sec per compound, resulting in spending 4 CPU core months for 10 million compounds. For these reasons, a much faster docking-based method sufficient to evaluate all compounds in ZINC library [48] or any other compound libraries is urgently needed, despite its limited screening accuracy. In addition, it is not necessary for pre-screening methods to output structural conformation information because pre-screened candidates will subsequently undergo more expensive, more detailed docking simulations.

1.6 Fragment-based method

A promising way to achieve acceleration is to reuse the calculation results because chemical compounds often partially share the same substructures, we called fragments. There are some fragment-based docking tools as we have shown in Table 1.1. Thus the reuse of fragment result is possible to apply compound evaluation and docking.

On the other hand, substructure-based methods have been adopted to calculate compound properties. For instance, topological polar surface area (TPSA) [69] is an estimation method of molecular polar surface area (PSA) that sums the substructure contributions and there is also a compound volume estimation method by counting

each type of atom [70]. These methods correctly estimate the properties independent from 3D conformation despite of the conformation dependencies.

The binding affinity highly depends on the binding pose of ligands toward the target protein while binding pose optimization is time consuming. If compound pose optimization-free procedure can predict the binding feasibility, it can be used as a pre-screening method and performs drastically faster than other structure-based methods.

1.7 Purpose of Study

In this thesis, we describe the structure-based pre-screening method and procedure of fragment extension-based docking in order to accelerate the compound evaluation speed. To realize the acceleration, we firstly proposed compound decomposition method, then we adopted it to structure-based virtual screening methods: pre-screening and docking.

The ultrafast structure-based pre-screening method called Spresso evaluates compounds by decomposition of them and fragment docking, followed by compound scoring from fragments scores. We also evaluated its accuracy and proposed the filtering method to improve it.

Fragment-based docking calculation procedure is also designed with reuse of intermediate results. We found that one of the intermediate result, fragment grid, is an essential factor toward acceleration while it spends huge memory space that cannot save all of them at once. Therefore, we formulated and reduced the problem as the minimum cost flow problem, and proposed faster algorithm utilizing graph characteristics.

1.8 Summary of Contributions

The contributions of this thesis are classified into three categories: (1) proposal of concept of decomposition into fragments and reuse of intermediate results, (2) development of a novel structure-based pre-screening method that is up to 300 times faster than the conventional procedure, (3) formulation of memory use optimization for fragment reuse and development of faster algorithm to solve the problem. We now describe these in more detail.

- We proposed the fundamental concept of compound decomposition and reuse of intermediate results, especially for structure-based virtual screening methods.

We also developed an decomposition procedure that generates several common substructures among compounds. We unveiled that the number of kinds of fragments can be one-tenth compared to the number of substructures generated by RECAP, an existing decomposition method and up to one-hundredth or less compared to the number of compounds.

- We developed Spresso, Speedy PRE-Screening method by Segmented cOmpounds. It only depends on protein tertiary structures, and it does not depend on known binders (e.g. inhibitors). The docking calculation is only used for each fragments, not for all compounds. Additionally, the compound conformation-free evaluation scheme is hired. Because of these reasons, the calculation speed is extremely faster than conventional, docking-based pre-screening.

- We designed protein-ligand docking procedure with reuse of intermediate results. The idea of result reuse of fragments was already proposed and implemented in eHiTS [38] but they achieved only up to 4-fold acceleration. We pointed out the drawback of the implementation and we proposed better procedure, which uses main memory, not hard disk. Additionally, we found new, another reusable intermediate result in fragment extension-based docking procedure.

- We formulated the memory consumption problem into the minimum cost flow problem. The memory consumption problem is arisen from intermediate result storage because intermediate result of fragment, named fragment grid, consumes more than 100 MB per fragment. Since the evaluation order of compounds and extension order of fragments are determined in deterministic manner, it can be formulated as a weighted offline cache problem, and can be reduced to a minimum cost flow problem. The generated graph of the problem has characteristics, and we developed faster algorithm specified for the graph.

1.9 Thesis Organization

The remaining chapters of this thesis are organized as follows: Chapter 2 depicts the compound decomposition procedure which used in Chapter 3 and after. Chapter 3 describes a novel structure-based pre-screening method, called Spresso (Speedy PRE-Screening method with Segmented cOmpounds). All experiments were updated from the article [71], thus previous results are written in Appendix A. Novel protein-ligand docking concept with reuse of intermediate results is described in Chapter 4, and

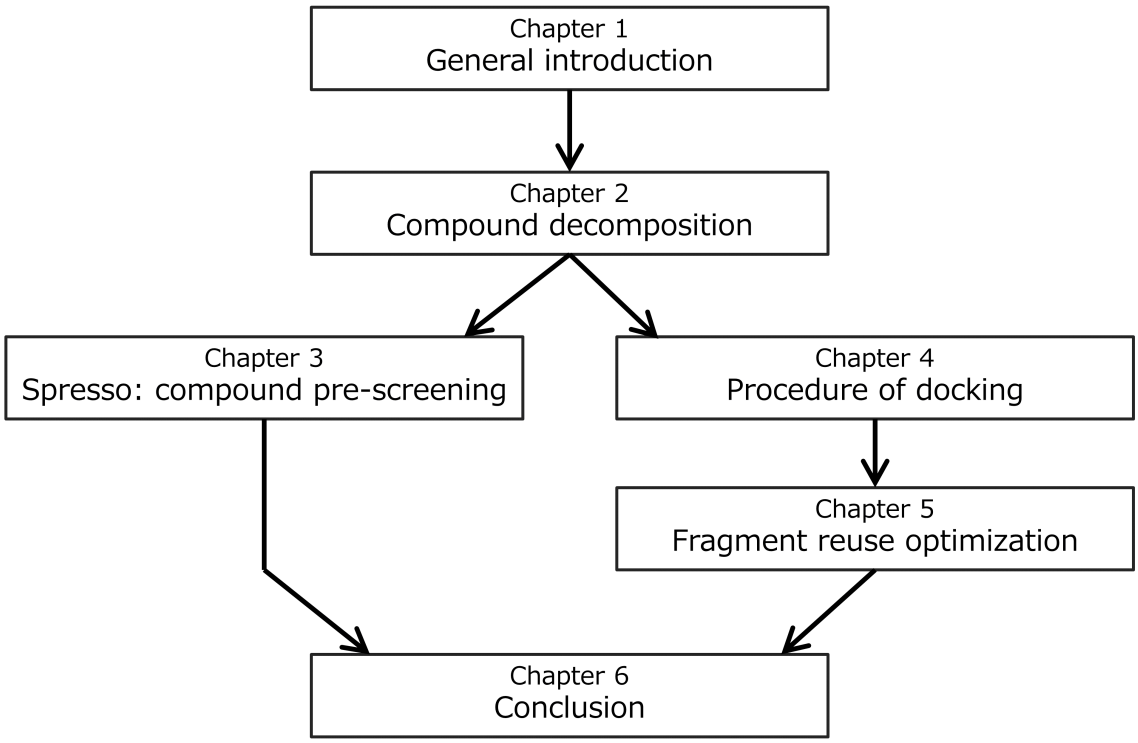


Figure 1.2: The relationship between chapters

Chapter 5 explains the optimization method of memory usage for fragment grids and the procedure of new algorithm to optimize it. Comclusions are presented in Chapter 6 together with future work and discussion. In addition, we report detailed information of each target in DUD-E and prediction accuracies of Glide SP, Glide HTVS, and Spresso in Appendix B. The relationship between chapters is shown in Fig. 1.2.

This thesis is based on the following publications by the author: [71, 72, 73, 74]

Part II

Compound Decomposition

Chapter 2

Compound Decomposition

2.1 Introduction

In this Chapter, we described compound decomposition procedure used in Chapter 3: Spresso and Chapter 4: novel protein-ligand docking procedure.

Medicinal chemists try to know which substructures of a compound are important for binding to optimize compounds, and thus substructure evaluation is common idea, such as group efficiency (GE) [75]. Additionally, linking multiple small structures (called "fragments") has been done as a part of fragment-based drug design (FBDD) [76, 77, 78, 79]. These two research directions imply the feasibility of computational methods with fragment-based evaluation.

Unlike FBDD, the virtual screening step evaluates existing compounds and does not generate new compound structures; however, fragment-based evaluation also has an large advantage in computational point of view. Organic compounds have lots of common substructures since they are composed of specific functional groups (e.g. Phenyl group, Carboxyl group, Imine group). If calculation results of substructures can be shared among compounds, the total amount of calculation will be decreased.

Obviously, the algorithm of decomposition highly affects the amount of calculation. From the retro-synthesis point of view, decomposition methods such as RECAP [80], BRICS [81] have been proposed. The aim of these methods is to virtually generate new compound structures, not to accelerate virtual screening. On the other hand, a fragment-based docking tool, eHiTS [38], decomposes compounds into rigid fragments and linkers between fragments, resulting in huge diversity of linker structures.

Thus, we firstly defined the fragment, followed by proposal of the decomposition method, which is related to eHiTS's method, but linkers are also decomposed into

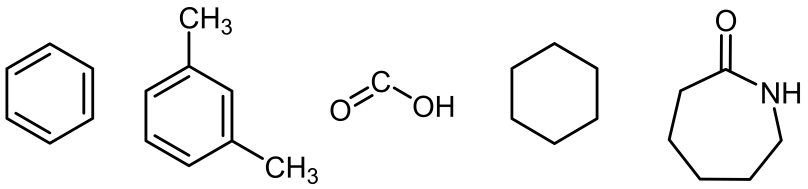


Figure 2.1: Example of fragments

rigid fragments in order to maximize the commonality of fragments. We also revealed that the number of kinds of fragments generated by our method are quite fewer than the number of compounds.

2.2 Definition of fragment

As mentioned in Introduction section, generating rigid substructure is important from the docking calculation point of view. On the other hand, too much fragmentation will break a functional groups into multiple substructures. Because of the reasons, we define that fragment is a substructure which has no (or restricted) flexibility in terms of the heavy atoms (except for hydrogen atoms).

No flexibility double bond, triple bond, and specific resonance bonded atoms such as phenyl-group make substructure rigid. Atoms which connected with these bonds each other are treated as a fragment.

Restricted flexibility some resonance bond such as peptide bonds, and cyclic bonds make substructure less flexible. Atoms which connected with these bonds each other are also treated as a fragment even though they have partial flexibility.

Fig. 2.1 shows some examples of fragment. Single atom is smallest fragment since it has no flexibility in the structure, and thus fragments have inclusion relationship.

It is noted that the size of fragments in this dissertation is relatively small compared to the definition of FBDD's "fragments" (e.g. rule of three [82]).

2.3 Decomposition method

To decompose compounds into fragments, we proposed the two-step decomposition method. The overview of the steps is shown in Fig. 2.2. It is noted that cleaved bonds are attached hydrogen atoms.

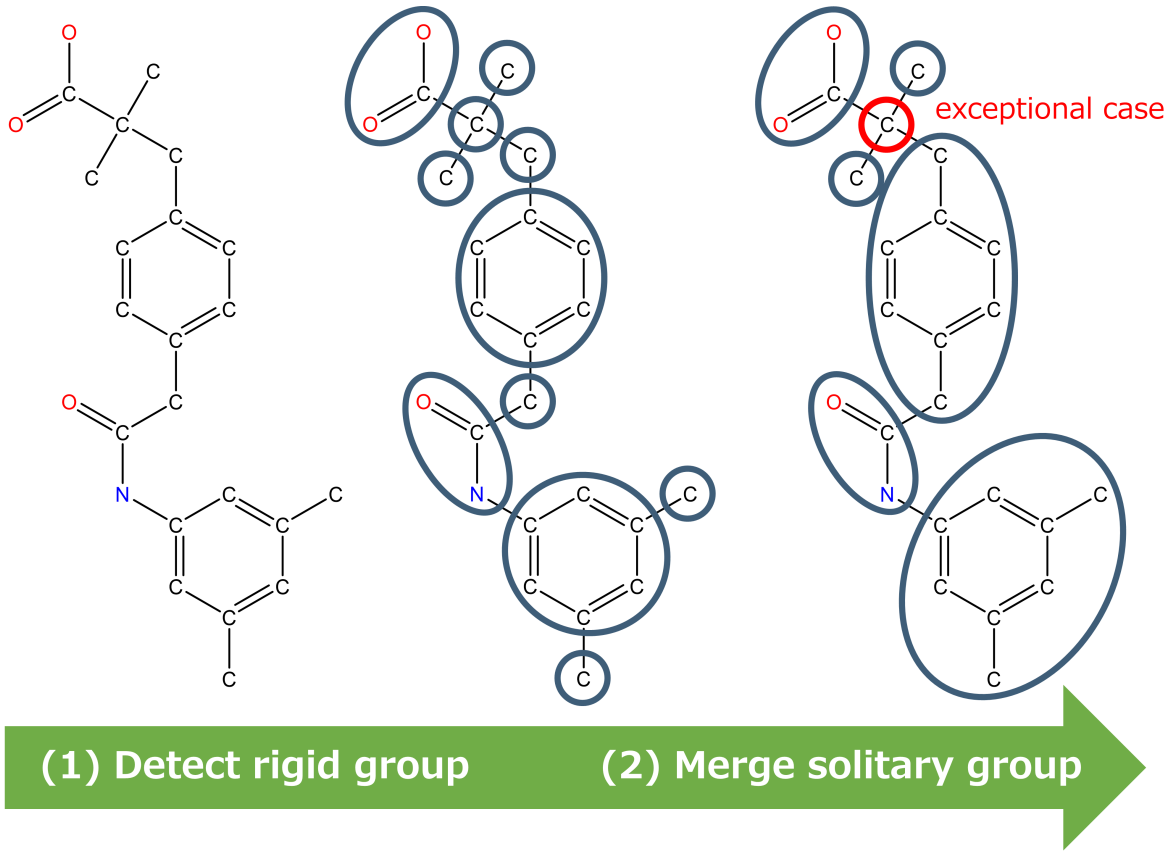


Figure 2.2: An example of compound decomposition. The carbon moiety in the structure on the right has four adjacent groups; therefore, it is not merged into any adjacent groups.

Step1: Detect rigid groups Firstly, rigid groups in each compound are detected. Rigid groups are double, triple, or resonance bonded atoms, ring groups (even in the case of cyclohexane, which has partial flexibility).

Step2: Merge solitary groups The second step involves merging solitary groups (single-atom fragments). Each non-solitary group and its adjacent solitary group are merged, except for solitary groups having three or more adjacent groups. The exceptional case is also shown in Fig. 2.2.

The program was coded with C++ and Open Babel library [83]. The code is open-sourced at `http://www.bi.cs.titech.ac.jp/spresso`.

2.4 Experiments

It is important to check how many fragments will be generated from some compound libraries. In this section, we showed decomposition statistics compared to RECAP [80] implemented in RDKit [84].

2.4.1 Dataset

There are several compound libraries because of different purposes. ZINC is a library of purchasable compounds [48] while the ChEMBL library contains compounds which are associated with other information, such as scientific literatures, patents, assay data (via PubChem Bioassays) [85]. The differences may make the difference of compound structures, thus we utilized three libraries: ZINC (all purchasable and all boutique subsets with ZINC ID unification, 28,629,602 compounds), ChEMBL (version 21, 1,583,897 compounds), and PubChem (accessed at 2016-12-12) [49]. As for the PubChem library, compounds having $\geq 1,000$ Da are rejected because of calculation cost, thus the number of compounds was 88,527,810.

2.4.2 Comparison of decomposition methods with the ZINC library

We firstly compared how many fragments were generated with proposed method and RECAP. The decomposition results are shown in Table 2.1. Proposed method expressed the ZINC library by only 263,319 fragments, while RECAP generated > 10 times more fragments. The expression of compounds by fewer fragments leads more reuse of intermediate results, thus we concluded our proposed method is better than existing method, RECAP.

2.4.3 Difference between libraries

Secondly, we decomposed the libraries by our methods. Table 2.1 shows the results. The results indicates that the number of fragments is dependent on the database; for example, the compounds of ChEMBL library were decomposed into 127,360 fragments (a 12-fold reduction). As for the PubChem, the compounds were decomposed into 2,082,185 fragments (a 43-fold reduction).

Table 2.1: The number of kind of fragments of several libraries and decomposition methods

Compound Library	#compounds	#kinds of fragments	
		Proposed	RECAP [80]
ZINC all purchasable & all boutique	28,629,602	263,319	3,161,753
ChEMBL v21	1,583,897	127,360	–
PubChem (2016-12-12) (MW < 1000 Da)	88,527,810	2,082,185	–

Table 2.2: The numbers of kinds of fragments among different numbers of compounds. Compounds were randomly sampled from ZINC all purchasable subset 5 times for each. The values are averages and standard deviations.

#compounds	#kinds of fragments
100	161.0 ± 3.9
300	338.8 ± 9.7
1,000	762.6 ± 33.1
3,000	1,602.8 ± 38.4
10,000	3,455.0 ± 56.6
30,000	6,754.0 ± 32.9
100,000	13,376.8 ± 70.8
300,000	24,172.2 ± 61.2
1,000,000	45,900.6 ± 65.1

2.5 Discussions

2.5.1 The relationship between the number of compounds and fragments

According to the Table 2.1, the ratio between the number of compounds and the number of fragments varies from 10-fold to 100-fold. For the further investigation, different sizes of libraries were generated by randomly sampled from ZINC library, and decomposed into fragments. Table 2.2 shows the results of the decomposition. The number of kinds of fragments is larger than the number of compounds for the small dataset, while the ratio goes to 22-fold for the large dataset. Interestingly, the increasing ratio of the number of kinds of fragments becomes slower when the number of compounds is larger. The tendency is obviously seen in Fig. 2.3. Thus our decomposition method is more effective when a compound library is larger.

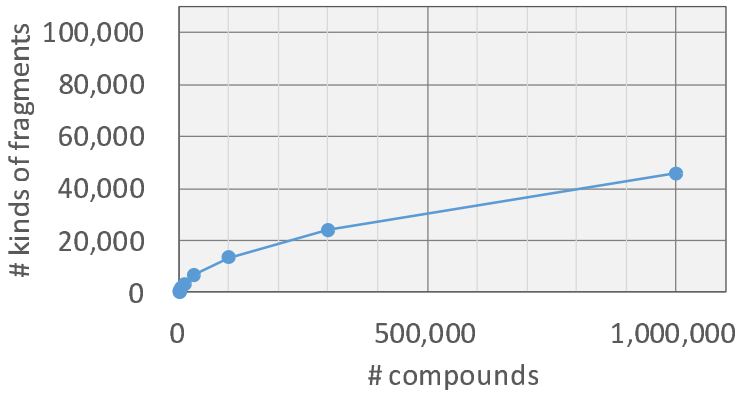


Figure 2.3: The relationship between the number of compounds and the number of kinds of fragments. The values are same as Table 2.2. All compound libraries are generated by randomly sampled from ZINC all purchasable subset.

2.5.2 The frequencies of fragments

The frequencies of each fragment highly varies, as shown in Fig. 2.4. As a further investigation, we confirmed what kind of fragments are frequently emerged and rarely emerged. Frequent fragments are shown in Fig. 2.5, while rare fragments are shown in Fig. 2.6. Almost half of frequent fragments are ring structures, both aromatic and aliphatic rings. Trifluoromethyl group, carboxyl group, and sulfo group are also frequently emerged. On the other hand, most of rare fragments are fused ring and contains chirality, it leads to its uniqueness.

2.6 conclusion

In this Chapter, we described the decomposition method which is utilized in all of our proposed method. The decomposition method can generate fragments which have no rotatable bond except for ring bonds.

The total number of kinds of fragments was decreased compared to the number of compounds. The ratio varies from 10-fold to 100-fold, related to the size of libraries. The fragments generated by our method is more common among compounds than RECAP, an existing decomposition method.

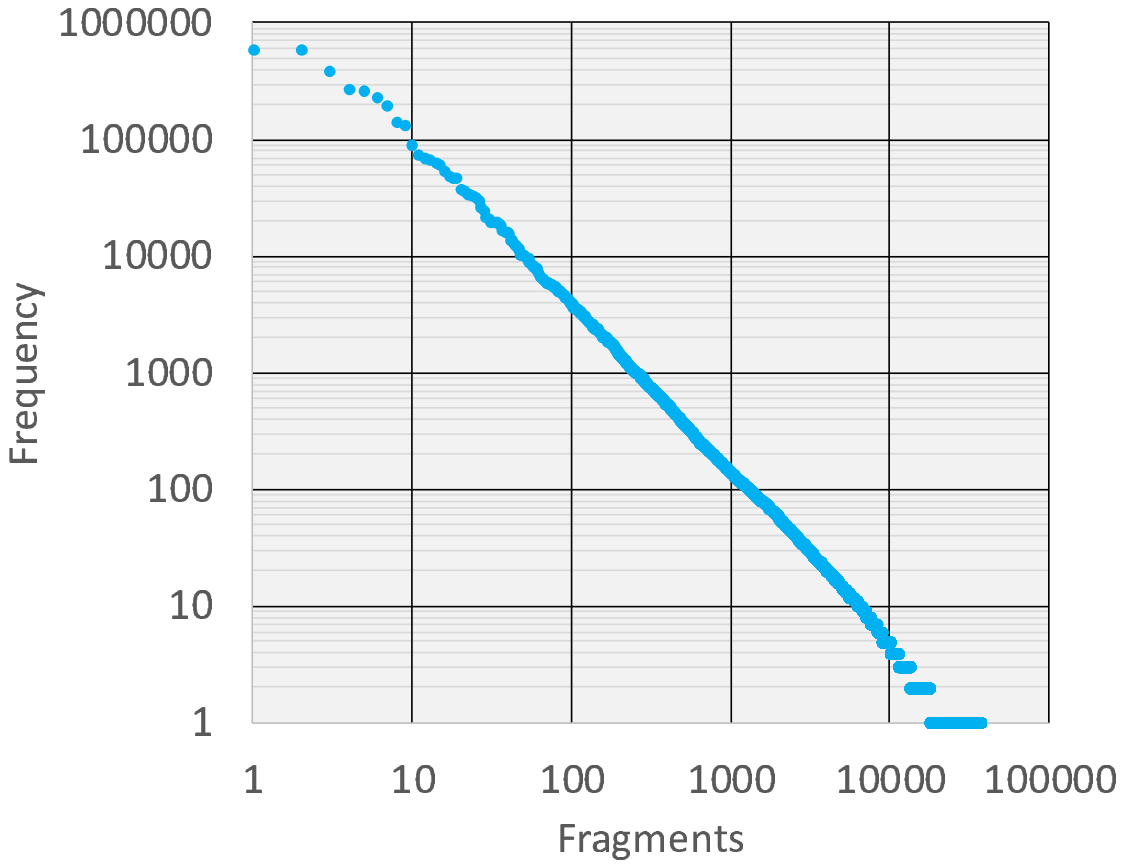


Figure 2.4: A scatter plot of appearance frequencies. The frequencies were calculated by decomposition of 1,000,000 compounds randomly sampled from ZINC all purchasable library.

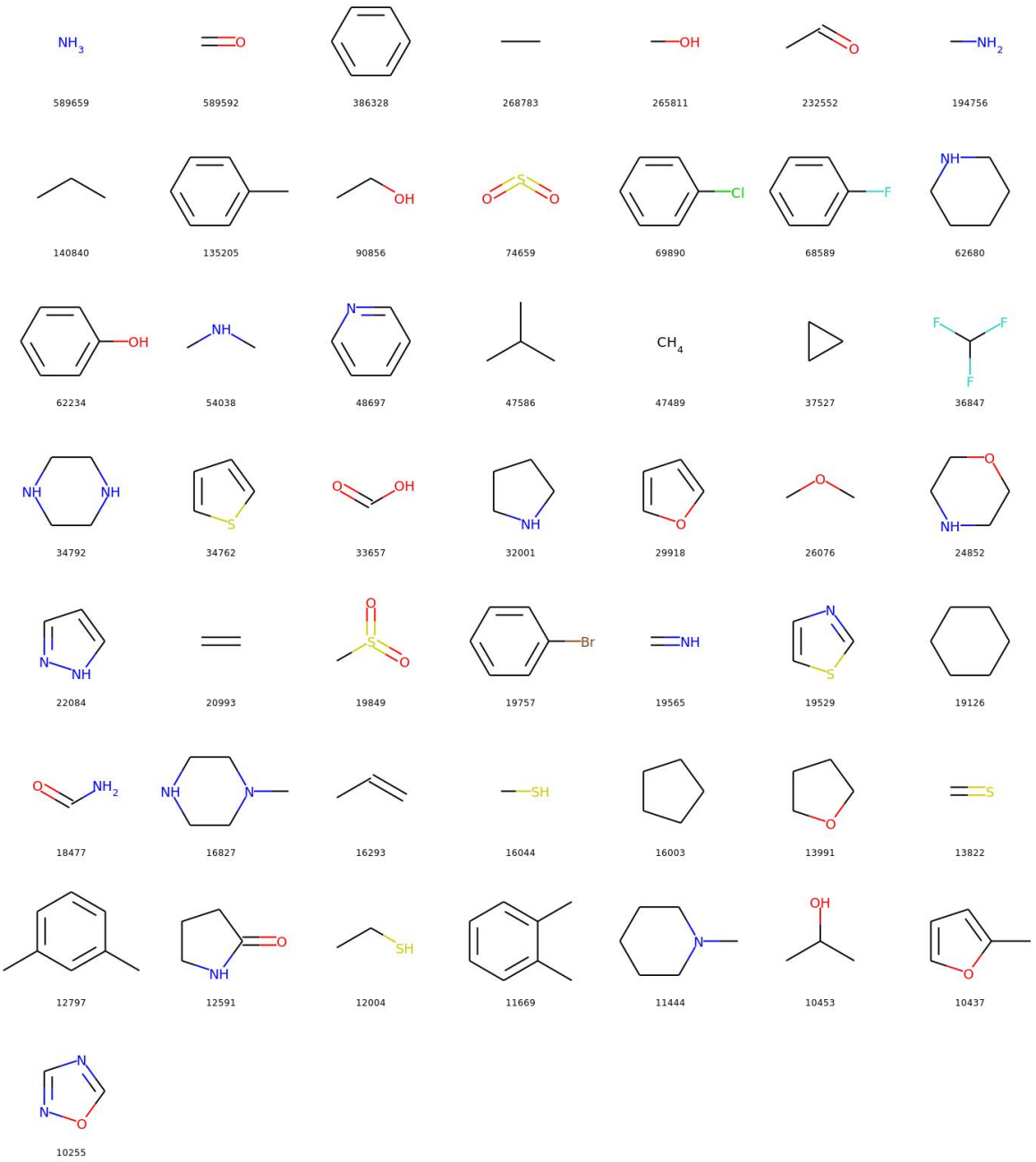


Figure 2.5: Frequent fragments of a decomposition result. These are emerged $> 10{,}000$ times in the decomposition of 1,000,000 compounds randomly sampled from ZINC all purchasable library. The numbers are the frequencies of fragments.

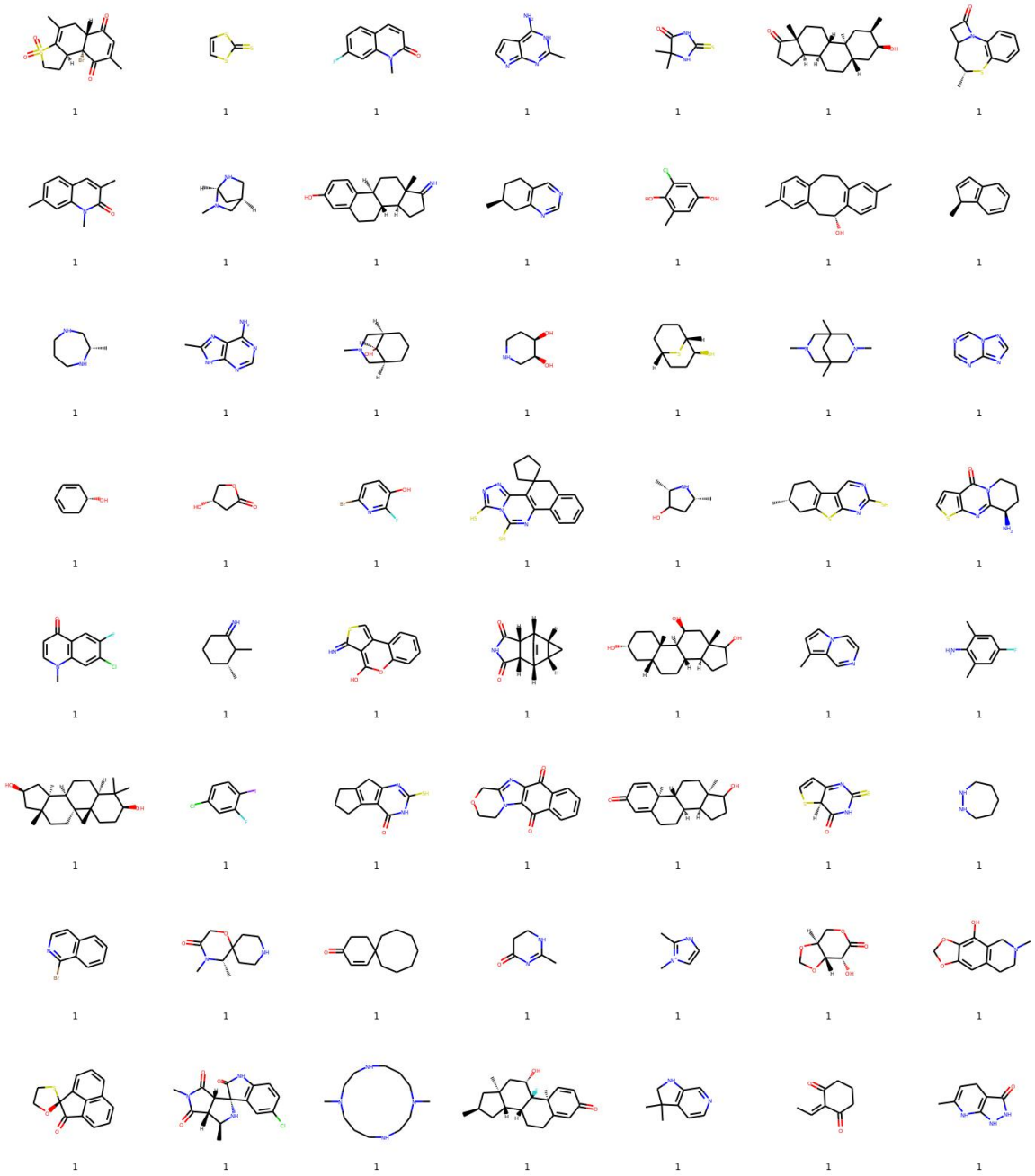


Figure 2.6: Rare fragments of a decomposition result. These are a part of fragments emerged only one time in the decomposition of 1,000,000 compounds randomly sampled from ZINC all purchasable library. The numbers are the frequencies of fragments.

Part III

Acceleration of Pre-screening

Chapter 3

Spresso: Development of an Ultrafast Compound Pre-Screening Method

3.1 Introduction

As we mentioned in Chapter 1, fast pre-screening method which based on protein tertiary structure is needed. Additionally, pre-screening methods are not required to output binding poses of each compound. It means reconstruction of compound structure is not needed. Because of these reasons, we proposed a structure-based pre-screening method called Spresso (Speedy PRE-Screening method with Segmented cOmpounds, pronounced like "espresso") that decomposes all candidate compounds into fragments with no internal degrees of freedom. These fragments are docked into target proteins individually, and compounds are scored based on the results of fragment docking. Spresso performs ultrafast compound evaluation without protein-ligand conformation prediction.

3.2 methods

The procedure of Spresso is comprised of three key steps summarized in Fig. 3.1: 1) compound decomposition, 2) fragment docking, and 3) fragment-based evaluation of each compound score.

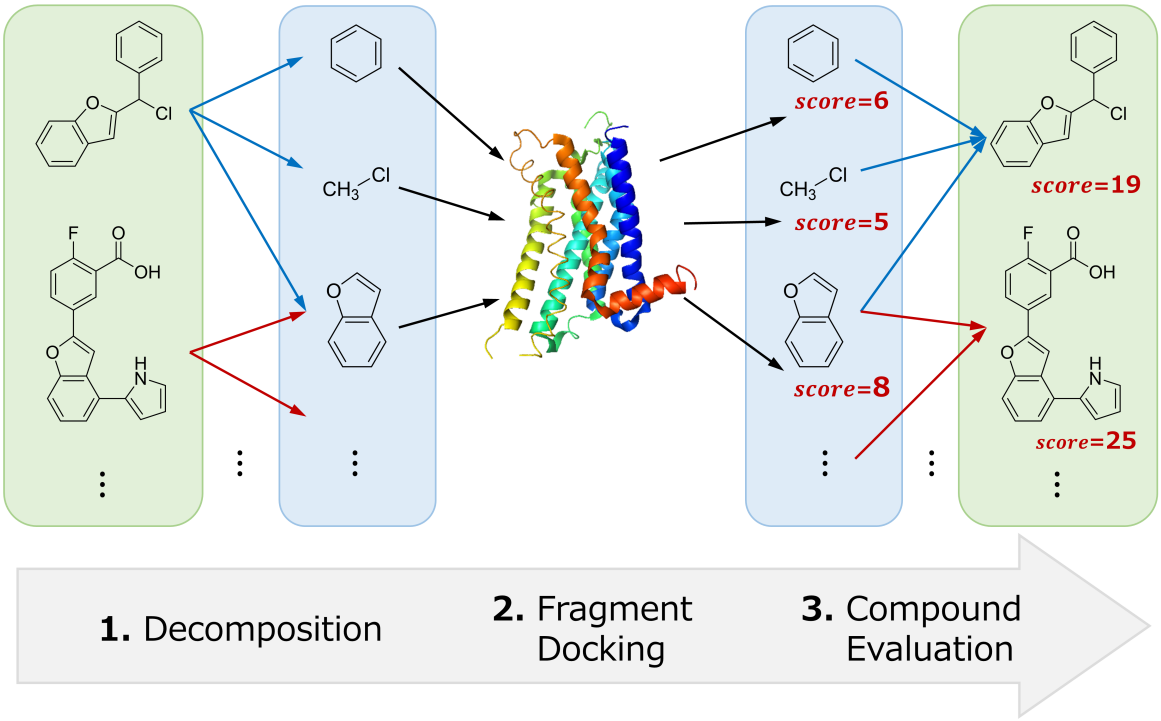


Figure 3.1: Spresso flowchart.

Compound Decomposition

We utilize the decomposition rule that is already mentioned in Chapter 2. The decomposition method make fragments with no internal degrees of freedom, which enable docking calculation faster.

Fragment docking

After decomposition, all rigid fragments are docked to the best location regardless of the other fragments. This means that all fragments are independently docked to the location in the protein cavity where they fit best. Fragments having the same substructures as those from different compounds can be scored identically, thereby significantly decreasing the number of fragments needing to be docked. The best score from the docking results for each fragment is recorded. For this procedure, any docking tool capable of outputting a score can be used, including AutoDock Vina [40], Glide [37], or GOLD [57].

Fragment-based evaluation of each compound

Compounds are evaluated after fragment docking is completed. Given that only fragments are docked into the target protein, we cannot obtain docking scores for entire compounds. Therefore, the screening evaluation score for each compound must be calculated based on the docking scores of fragments decomposed from the original compound. There are two strategies for compound evaluation: 1) choosing combinations of fragment conformations that avoid contradictions, and 2) choosing the best conformation without consideration of fragment collisions. The former strategy is more precise as compared to the latter strategy, but searching for conformation combinations can also be computationally expensive. Given our goal of creating a computationally faster "pre-screening" method, we chose the latter strategy for compound evaluation.

We can consider many formulae for calculating compound-evaluation scores from fragment-docking scores ($score_f$). In this study, we evaluated seven calculation formulae.

(I) Summation of fragment-docking scores (SUM)

$$\mathrm{SUM} = \sum_f score_f \tag{3.1}$$

Summation is one of the simplest evaluation methods, where SUM reflects the approximate rough upper bound of the compound-docking score. Generally, the SUM value is larger when a compound is divided into more fragments.

(II) Best value of fragment-docking scores (MAX)

$$\mathrm{MAX} = \max_f(score_f) \tag{3.2}$$

Utilizing the best value is also a simple evaluation method, where MAX reflects the estimated rough lower bound of compound-docking scores. In most cases, the MAX value will be less than the compound-docking score; however, a docking score associated with a single fragment may exceed the compound-docking score in specific cases (i.e., a compound too large for a protein cavity).

(III) Arithmetic Mean (AM)

$$\mathrm{AM} = \frac{\mathrm{SUM} + \mathrm{MAX}}{2} \tag{3.3}$$

(IV) Geological Mean (GM)

$$\mathrm{GM} = \sqrt{\mathrm{MAX} \cdot \mathrm{SUM}} \tag{3.4}$$

(V) Harmonic Mean (HM)

$$\mathrm{HM} = \frac{2 \cdot \mathrm{SUM} \cdot \mathrm{MAX}}{\mathrm{SUM} + \mathrm{MAX}} \tag{3.5}$$

(VI) Generalized Sum (GS)

$$\mathrm{GS}_x = \sqrt[x]{\sum_f (score_f)^x} \tag{3.6}$$

GS_1 is equal to SUM, while $\lim_{x\to\infty} \mathrm{GS}_x$ is equal to MAX; therefore, GS can express the mixture of SUM and MAX values continuously. In this study, we chose GS_3 from the $\mathrm{GS}_2 \sim \mathrm{GS}_{10}$ evaluation results (Fig. 3.2).

(VII) Energetic Sum (ENE)

$$\mathrm{ENE}_x = x \log \left(\sum_f \exp\left(score_f / x\right) \right) \tag{3.7}$$

This calculation is an analogy of sum of energies, and it is called "logsum-exp" in some fields such as Deep Learning because of the structure of formula. $\lim_{x\to 0} \mathrm{ENE}_x$ is also equal to MAX, however, there is no equivalent expression of SUM. In this study, we chose $\mathrm{ENE}_{1.8}$ from the $\mathrm{ENE}_1 \sim \mathrm{ENE}_{2.5}$ evaluation results (Fig. 3.3).

GS requires non-negative values as input, while the fragment-docking score is almost always a negative value because the score was fitted to experimental ΔG. Therefore, the fragment-docking scores are inverted, and positive docking scores (which are inverted to negative values) are treated as zero.

The best pre-screening accuracy was achieved when (VII) $\mathrm{ENE}_{1.8}$ was used (detailed description provided in section 3.3.2; thus, $\mathrm{ENE}_{1.8}$ was adopted as the default formula in Spresso.

3.2.1 Datasets

The Directory of Useful Decoys, Enhanced (DUD-E) [86] was used to evaluate the performance of pre-screening during the virtual screening process. The DUD-E dataset is widely used and consists of 102 diverse sets of protein targets, as well as active and

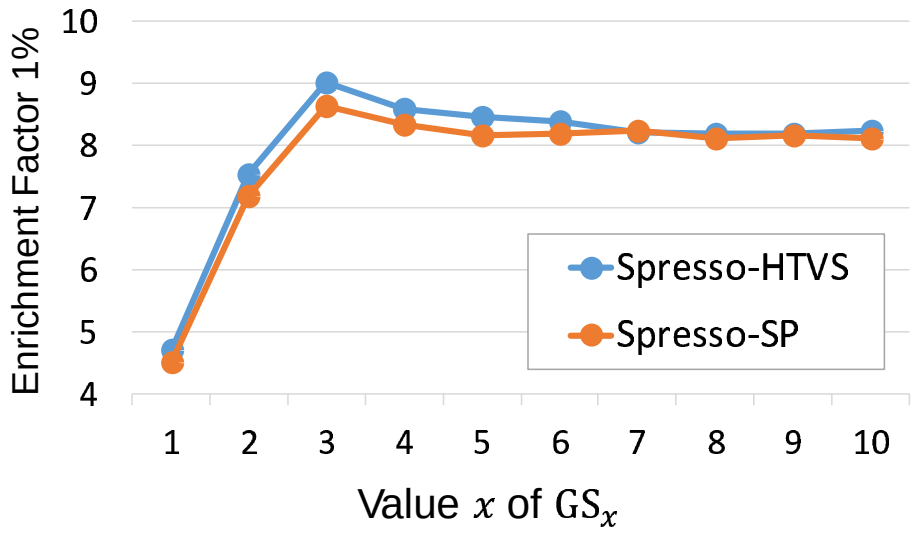


Figure 3.2: Results of the averaged prediction accuracy for 102 DUD-E targets among GS_x formulae. This figure shows 2%–1% results that represent the $EF_{1\%}$ when 2% of all compounds were pre-screened, and suggests that GS_3 is the best parameter.

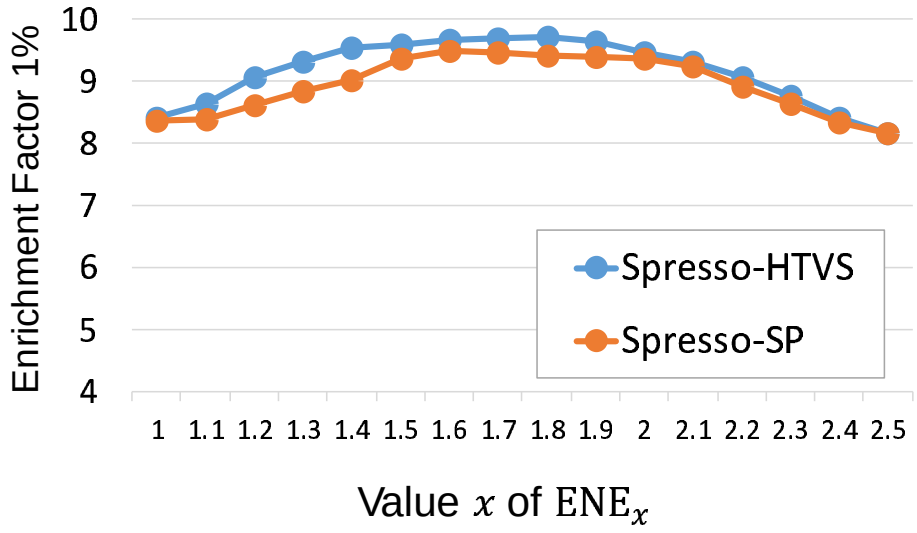


Figure 3.3: Results of the averaged prediction accuracy for 102 DUD-E targets among ENE_x formulae. This figure shows 2%–1% results that represent the $EF_{1\%}$ when 2% of all compounds were pre-screened, and suggests that $ENE_{1.8}$ is the best parameter.

decoy compounds. The ZINC database [48] was also used to measure calculation time, since the number of active compounds and decoys in each set is insufficient as compared

to those used in actual virtual screening. The library consists of 28,629,602 compounds.

3.2.2 Implementation and Settings

The code of compound-evaluation score calculations were written in Python. Spresso code (including decomposition code) is freely available at `http://www.bi.cs.titech.ac.jp/spresso/` under the GPL version 3 license. The URL has been accessed from not only Japan, but also Belgium, Chile, China, France, Poland, Portugal, South Korea, Turkey, and United States.

For the docking tool required in fragment docking step in Spresso, we used Glide SP mode and Glide HTVS mode. It is noted that fragment-based docking tools, such as eHiTS and FlexX, are more similar to the idea of Spresso; however, we did not utilize them since eHiTS is no longer available (as of Feb. 8, 2019) and FlexX have been reported the accuracy is worse than Glide [42, 43]. Glide HTVS was also used to dock compounds for comparison of pre-screening. The version of Glide is 2018-3.

3.2.3 Computing environment

All calculations were conducted on the TSUBAME 3.0 supercomputing system, Tokyo Institute of Technology, Japan. Each node has two Intel Xeon E5-2680 V4 CPUs (14 cores per CPU) and 256 GiB of RAM. Because Glide software is a single-thread program, all docking simulations were performed in parallel using 25 CPU cores. It should be noted that Glide is a proprietary software, and thus it cannot be optimized for specific computing environments.

3.2.4 Metrics

Two computational experiments were conducted: 1) evaluation of calculation speed, and 2) evaluation of virtual screening accuracy. Since one license will allow us to use only one CPU core, we used CPU time to evaluate calculation speed. Accuracy was measured by performance efficiency according to enrichment factors (EFs) [87].

$$\mathrm{EF}_{x\%} = \frac{\mathrm{Pos}_{x\%}/\mathrm{All}_{x\%}}{\mathrm{Pos}_{100\%}/\mathrm{All}_{100\%}} \tag{3.8}$$

In (3.8), $\mathrm{Pos}_{x\%}$, $\mathrm{All}_{x\%}$, $\mathrm{Pos}_{100\%}$, and $\mathrm{All}_{100\%}$ are the number of active compounds in the top $x\%$ of screened compounds, the number of compounds in the top $x\%$ of screened compounds, the total number of screened active compounds, and the total number of

screened compounds, respectively. In virtual screening, it is pragmatically meaningless to assess differences between lower ranked compounds because wet-lab experiments can be executed up to only a few thousand compounds even though computational methods can deal with more than 1 million compounds. Therefore, $EF_{1\%}$ and $EF_{2\%}$ were calculated to evaluate accuracy.

3.2.5 Assessment of prediction accuracy

As previously mentioned, Spresso is not intended for independent use. Therefore, an evaluation must involve not only Spresso but also a following compound docking calculation. The procedure used for evaluation of accuracy was as follows: 1) with each pre-screening method, 2%, 5%, or 10% of the number of all target compounds were selected; 2) pre-screened candidates were docked using Glide SP to obtain a docking score; and 3) the top 1% and 2% of compounds were used to calculate $EF_{1\%}$ and $EF_{2\%}$. We calculated five combinations for each pre-screening method.

3.3 Results

To evaluate the usefulness of Spresso with regard to speed and prediction accuracy, two experiments were performed. In all experiments, Glide HTVS, which is a conventional pre-screening method, was also evaluated for comparison.

3.3.1 Comparison of docking calculation speed

Table 3.1 shows the calculation times for docking of all 28,629,602 ZINC compounds into three target proteins from the DUD-E dataset. Spresso using Glide SP-mode fragment docking (Spresso-SP) required < 3 CPU days, and Spresso with Glide HTVS-mode fragment docking (Spresso-HTVS) required < 1 CPU day, while whole-compound docking using Glide HTVS mode required > 8 CPU months. These results suggest that Spresso is 300-fold faster than compound docking with conventional Glide HTVS pre-screening.

3.3.2 Prediction accuracy in DUD-E benchmarking

Table 3.2 shows the average EF values associated with each DUD-E target. The formulae are listed as (I)-(VII) in section 3.2. Seven score calculations were evaluated,

Table 3.1: Computation times for docking of all 28,629,602 ZINC compounds into DUD-E diverse subset targets. The calculation times for Spresso were measured with all fragments decomposed from all ZINC compounds. On the other hand, the times for Glide HTVS and Glide SP were measured with 100,000 and 10,000 compounds randomly sampled from ZINC compounds 5 times respectively. The shown values for Glides are estimated from the measured results. For the results of Glide HTVS, the shown values expresses averages and standard deviations among 5 times calculation.

Target	Calculation time [CPU core days]			
	Spresso-SP	Spresso-HTVS	Glide HTVS (estimated)	Glide SP (estimated)
AKT1	2.81	0.82	136.1 ± 0.5	2726.6 ± 83.4
AMPC	2.70	0.82	195.2 ± 0.9	3444.2 ± 58.6
CP3A4	2.84	0.95	427.0 ± 1.9	7185.0 ± 163.0
CXCR4	2.74	0.86	320.3 ± 12.2	5581.7 ± 114.1
GCR	2.87	0.88	270.0 ± 2.4	4600.8 ± 93.2
HIVPR	2.81	0.91	342.7 ± 15.2	6015.8 ± 122.9
HIVRT	2.84	0.87	159.7 ± 9.0	2989.5 ± 60.4
KIF11	2.69	0.83	258.7 ± 2.1	4373.6 ± 73.1
mean	2.79	0.87	263.7	4614.6
median	2.81	0.87	264.3	4487.2

revealing that the combination of Spresso-HTVS and $ENE_{1.8}$ was the best. Interestingly, the results of Spresso-SP were slightly less accurate results as compared to Spresso-HTVS. Our results indicate that Spresso was less accurate when compared with conventional method. On the other hand, Pearson's correlation coefficients with Glide SP score is almost same between Spresso-HTVS ($ENE_{1.8}$, $R = 0.49$, Fig. 3.4) and Glide HTVS ($R = 0.50$, Fig. 3.5) for CP3A4, one of the DUD-E target.

In order to reveal how many compounds selected by Glide SP are included in the compounds selected by pre-screening methods Glide HTVS or Spresso, the overlap in selected compounds identified with each method was calculated for DUD-E Diverse Subset (8 targets). Venn diagrams are shown in Fig. 3.6. These diagrams indicate that the compounds identified with Spresso have less intersection with those from Glide SP than Glide HTVS.

3.3.3 Trade-off of accuracy and speediness

As we mentioned in Sections 3.3.1 and 3.3.2, it was revealed that Spresso was 300-fold faster than Glide HTVS while its accuracy was slightly less accurate. The relation between the accuracy and the speediness is a trade-off relation generally, thus we newly

Figure 3.4: A scatter plot of the Glide SP score and the Spresso-HTVS score for DUD-E CP3A4 target. Each dot represents a compound in DUD-E CP3A4 dataset. The Pearson correlation coefficient is $R = 0.49$.

Figure 3.5: A scatter plot of the Glide SP and Glide HTVS scores for the DUD-E CP3A4 target. Each dot represents a compound in the DUD-E CP3A4 dataset. The Pearson correlation coefficient is $R = 0.50$.

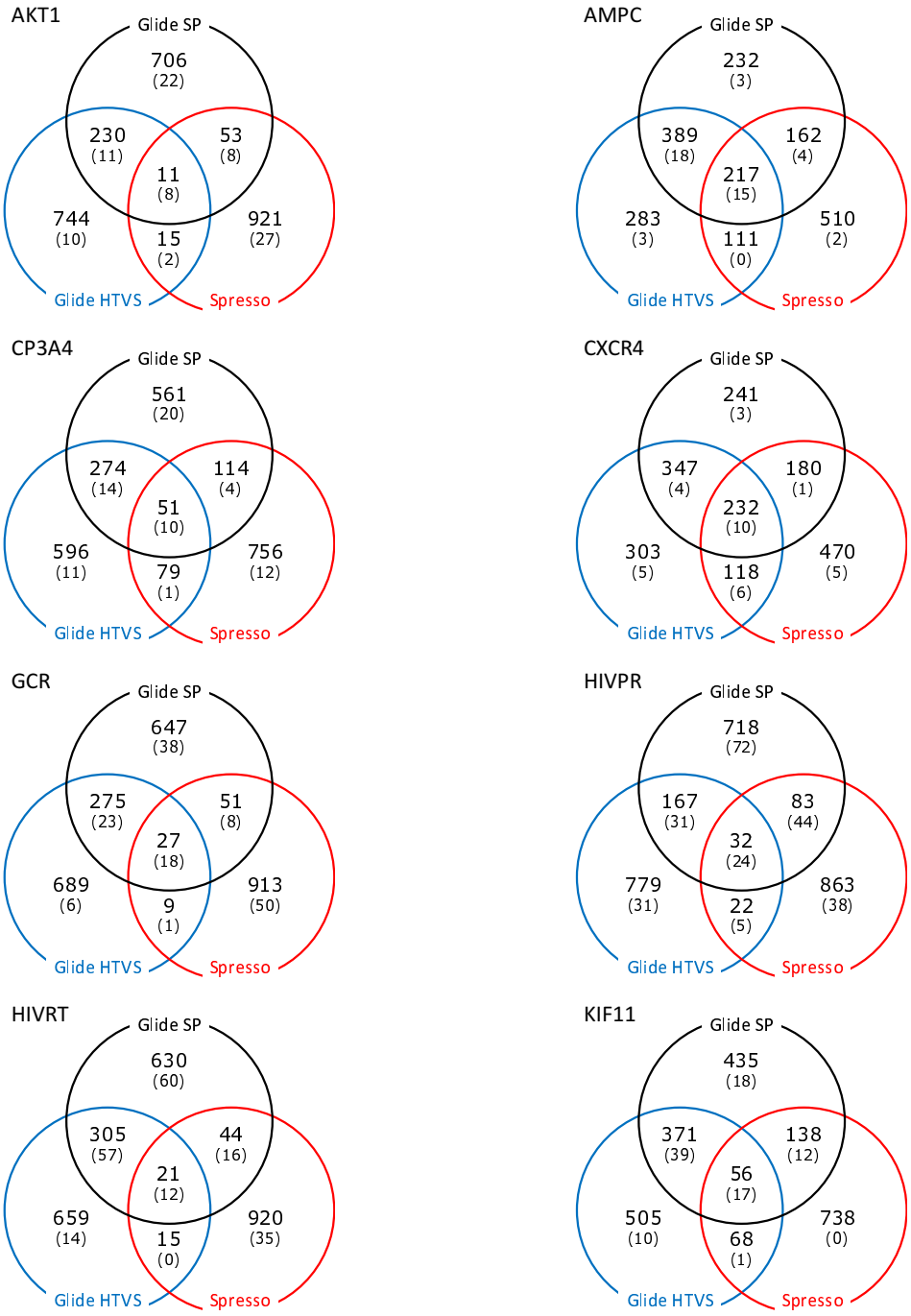


Figure 3.6: Venn diagrams of selected compounds identified by pre-screening for DUD-E diverse subset targets. The top 1,000 compounds identified by Glide SP, Glide HTVS, and Spresso-HTVS are shown. The number of compounds for each method is shown, and numbers of true positives are in parentheses.

Table 3.2: The results of averaged prediction accuracy for 102 DUD-E targets. Note that all enrichment factors represent the average of 102 EFs from DUD-E protein targets. a%–b% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds. Best EF values among Spressos are written in bold.

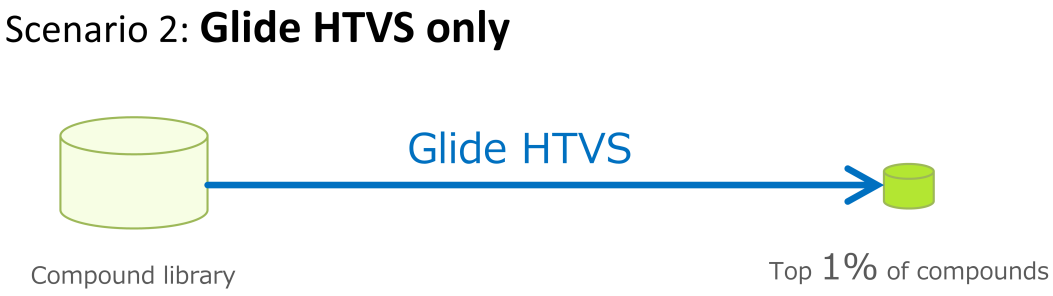
Target		Enrichment Factor				
		2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
Spresso-SP	SUM	4.51	6.35	7.76	3.91	4.98
	MAX	8.10	9.29	11.10	6.30	7.43
	AM	4.96	6.67	8.05	4.25	5.21
	GM	6.15	8.02	9.65	5.26	6.36
	HM	6.41	8.06	10.37	5.60	6.87
	GS_3	8.63	11.06	13.43	7.34	8.75
	$ENE_{1.8}$	9.42	11.86	13.63	**7.83**	8.89
Spresso-HTVS	SUM	4.71	6.41	8.12	4.03	5.26
	MAX	8.43	9.74	11.91	6.24	7.72
	AM	5.22	6.89	8.48	4.38	5.51
	GM	6.76	8.64	10.59	5.48	6.69
	HM	6.83	9.19	11.65	5.93	7.46
	GS_3	9.02	11.64	14.20	7.64	9.05
	$ENE_{1.8}$	**9.70**	**11.95**	**14.53**	7.64	**9.26**
Glide HTVS		17.29	18.04	18.55	11.82	12.20

defined the accuracy divided by the calculation time as the screening efficiency, then single step screenings by Glide SP, Glide HTVS, or Spresso (scenarios 1–3, Figs. 3.7–3.9), and pre-screening by Glide HTVS or Spresso (scenarios 4, 5, Figs. 3.10, 3.11) were evaluated in terms of the screening accuracies (EF), the total docking calculation time (time [days]), and the screening efficiencies. Additionally, three step screening scenario was also evaluated (scenario 6, Fig. 3.12) because Spresso was ultrafast compared to Glide HTVS. It is noted that the docking time was calculated under the condition that 10 licenses (equivalent to 10 CPU cores) are utilized since that of docking tools is expensive (e.g. more than 50 thousand dollars per Glide academic license per year).

The period available for the virtual screening in drug discovery is not enough, from a few days to a month usually. Furthermore, multiple protein structures are sometimes used for docking to consider the conformational change. Because of the reasons, scenarios with Spress, scenarios 5 and 6, are more suitable in terms of their speediness and its effectiveness (Fig. 3.13). Interestingly, scenario 6: three step screening marked the best efficiency of the three scenarios because the EF value was considerably improved with small amount of additional computational cost.

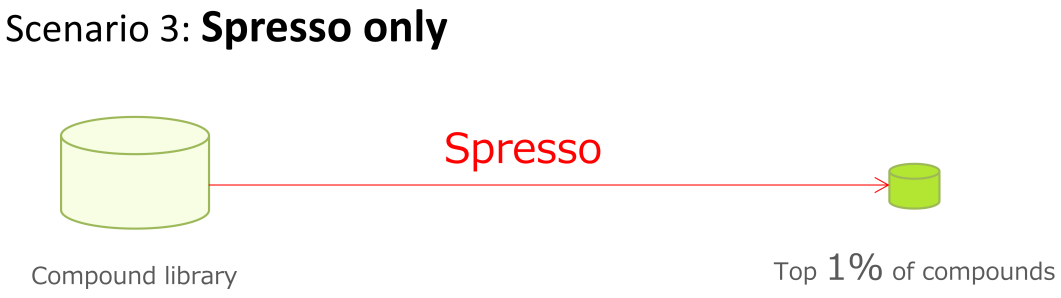
$EF_{1\%}$	20.4
Time [days]	461
$EF_{1\%}$ / time	0.04

Figure 3.7: The 1% enrichment factor, the expected time consumption, and the screening efficiency for scenario 1: Glide SP only



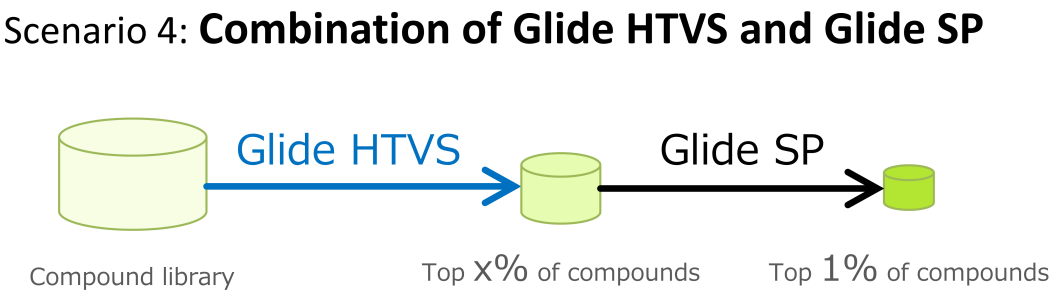
$EF_{1\%}$	16.3
Time [days]	26.4
$EF_{1\%}$ / time	0.62

Figure 3.8: The 1% enrichment factor, the expected time consumption, and the screening efficiency for scenario 2: Glide HTVS only



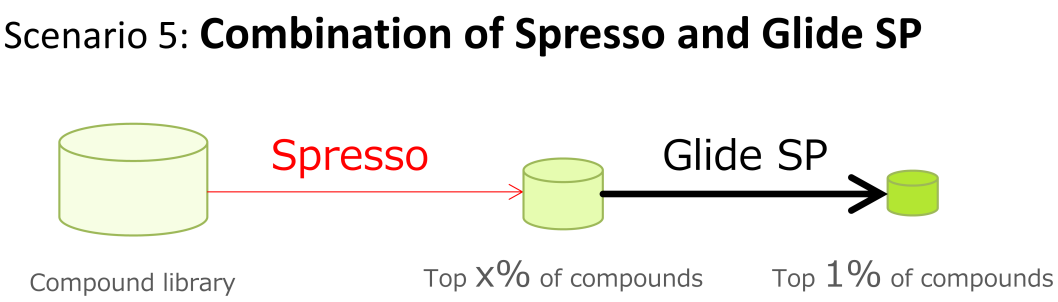
$EF_{1\%}$	8.0
Time [days]	0.1
$EF_{1\%}$ / time	88.8

Figure 3.9: The 1% enrichment factor, the expected time consumption, and the screening efficiency for scenario 3: Spresso only



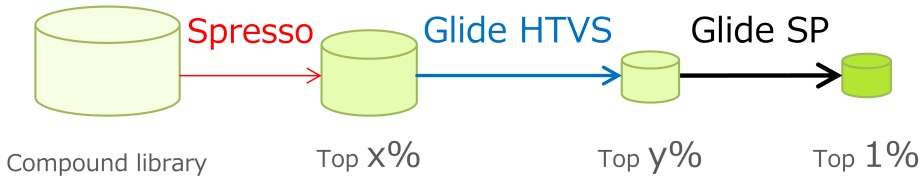
	screening x%-1%		
	(a) 2%-1%	(b) 5%-1%	(c) 10%-1%
$EF_{1\%}$	17.3	18.0	18.6
Time [days]	35.6	49.4	72.5
$EF_{1\%}$ / time	0.49	0.37	0.26

Figure 3.10: The 1% enrichment factors, the expected time consumptions, and the screening efficiencies for scenario 4: Combination of Glide HTVS and Glide SP



	screening x%-1%		
	(a) 2%-1%	(b) 5%-1%	(c) 10%-1%
$EF_{1\%}$	9.4	11.8	14.5
Time [days]	9.3	23.2	46.2
$EF_{1\%}$ / time	1.01	0.51	0.31

Figure 3.11: The 1% enrichment factors, the expected time consumptions, and the screening efficiencies for scenario 5: Combination of Spresso and Glide SP



	screening x%-y%-1%		
	(a) 5%-2%-1%	(b) 10%-2%-1%	(c) 10%-5%-1%
$EF_{1\%}$	11.1	13.0	14.1
Time [days]	10.6	12.0	25.8
$EF_{1\%}$ / time	1.05	1.08	0.55

Figure 3.12: The 1% enrichment factors, the expected time consumptions, and the screening efficiencies for scenario 6: Three step screening

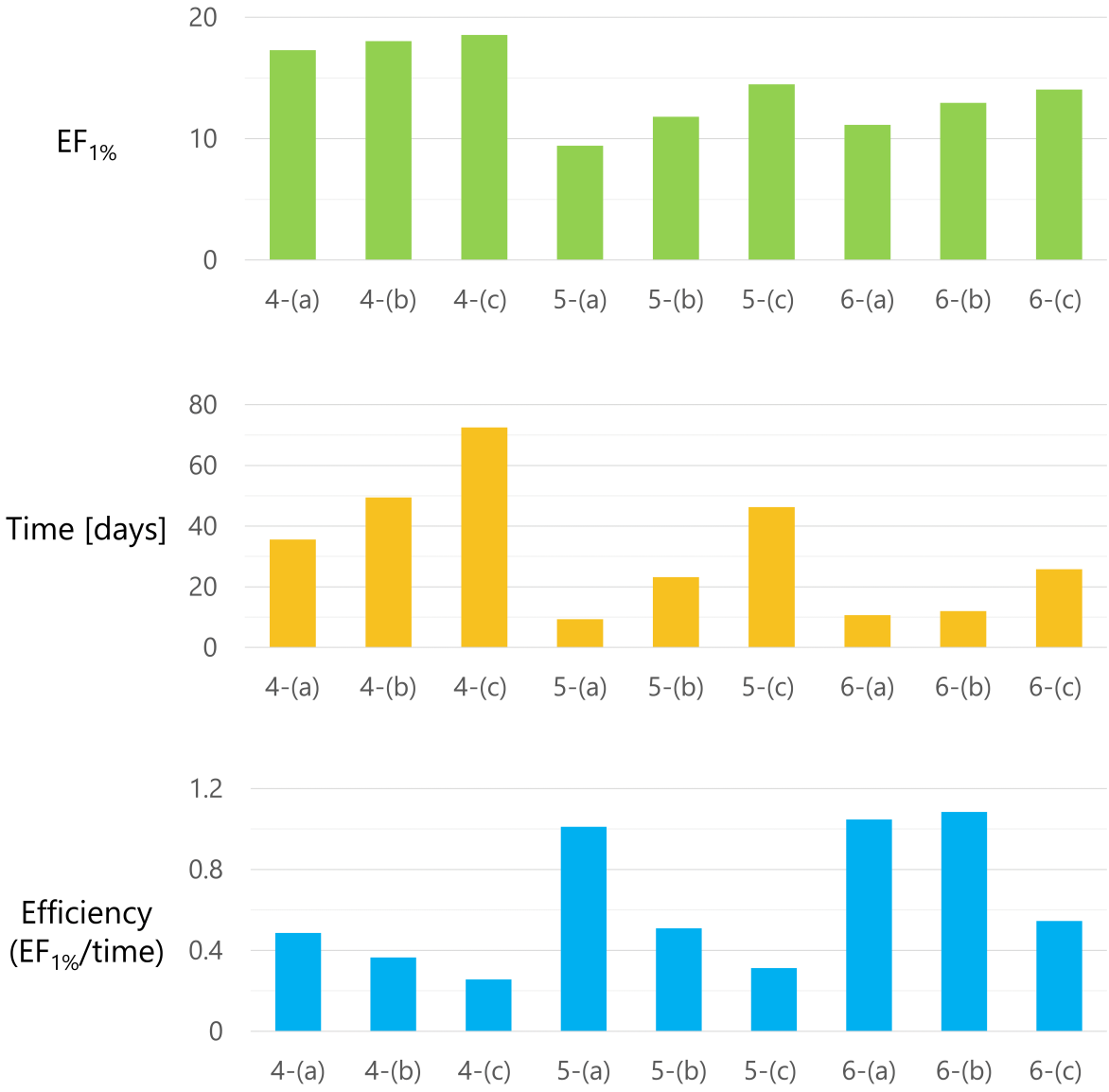


Figure 3.13: The comparison between pre-screened scenarios. The EF, time, efficiency values are reproduced from Figs. 3.10–3.12.

3.4 Discussion

3.4.1 Superiority of $\mathrm{ENE}_{1.8}$

The computational experiment in section 3.2 revealed the (VII) $\mathrm{ENE}_{1.8}$ formula as the best of the seven possible methods for calculating compound-evaluation score. Method (I) SUM, utilizing all fragment scores equally, was the worst of the seven, while $\mathrm{ENE}_{1.8}$ returned acceptable results. The exponent of $\mathrm{ENE}_{1.8}$ acts as a weight coefficient, which implies that the result indicates that higher-scoring fragments should be more weighted. However, $\mathrm{ENE}_{1.8}$ returned more accurate results relative to method (II) MAX, given that considering the top few fragment scores is more informative than considering only the top fragment score.

3.4.2 Score fitting to Glide SP

Linear least squares fitting is often applied to experimental results or precise estimates in fragment-based, compound property estimation methods [69, 70]. In the compound property estimation methods, common explanatory variables include the fragment type, number of cleaved bonds, and number of rings, amongst others; however, it is inappropriate to determine the contribution of each fragment in docking simulations since docking scores differ based on the target protein, and thus fragment-docking scores are used with equal contribution. Additionally, the number of cleaved bonds must affect the sum of fragment score. Because of above reasons, we generated a linear regression model with two factors, $score_{\mathrm{SUM}}$ and the number of cleaved bonds, performed fittings with the Glide SP compound docking score as a target using the DUD-E HIVPR dataset, and then calculated the DUD-E CP3A4 dataset compounds' $score_{\mathrm{fitting}}$ with the fitted parameter. The data utilized for this pre-screening is detailed in Table 3.3. The correlation coefficient between $score_{\mathrm{fitting}}$ and Glide SP of CP3A4 was $R = 0.49$ (Fig. 3.14), which is lower than that between $\mathrm{ENE}_{1.8}$ and Glide SP ($R = 0.55$, Fig. 3.4), thus the linear regression fitting did not work well and explanatory valiables should be more considered.

3.4.3 Can Spresso conserve compound diversity?

Drwal *et al.* showed that structure-based methods are likely to maintain the diversity of compound structures as compared with ligand-based methods [26]. While this is one reason to use structure-based methods, it does not guarantee that the diversity

Table 3.3: Information used to generate the linear regression model and estimation.

Model creation	Protein	DUD-E HIVPR (PDBID: 1XL2)
	Compound	DUD-E HIVPR (36,286 compounds)
	Target score	The score of Glide SP compound docking
	Base model	$score_{\text{fitting}} = a \cdot score_{\text{SUM}} + b \cdot \lvert \#cleaved\ bonds \rvert$
Score estimation	Protein	DUD-E CP3A4 (PDBID: 3NXU)
	Compound	DUD-E CP3A4 (11,970 compounds)
Calculation Library		Python StatsModels

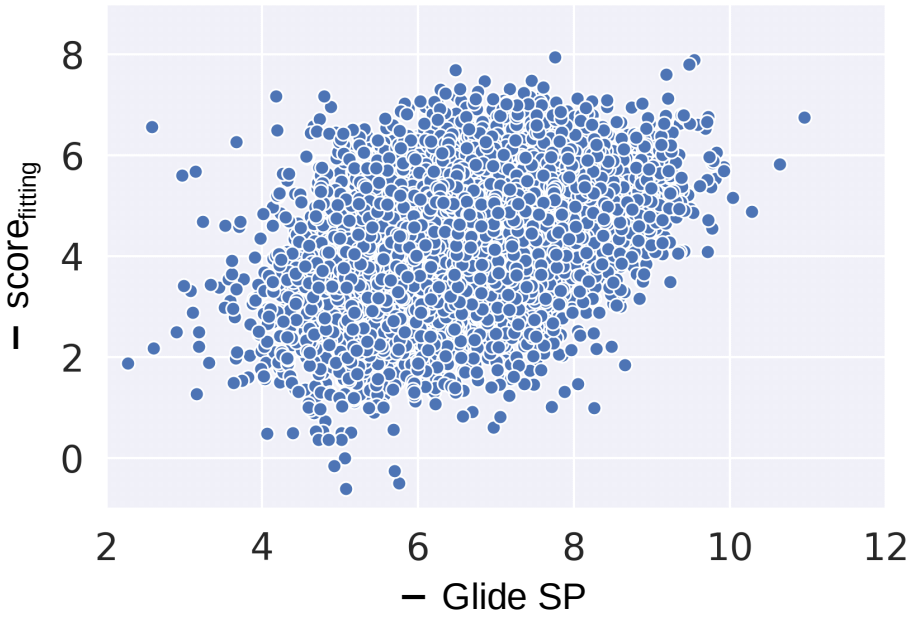


Figure 3.14: A scatter plot of the Glide SP and fitted scores for the DUD-E CP3A4 target. Each dot represents a compound in the DUD-E CP3A4 dataset. The correlation coefficient is $R = 0.40$.

Table 3.4: Machine-learning settings on the support vector machine (SVM). Active compounds and decoys were obtained from the DUD-E dataset. We performed parameter tuning by adopting the optimum-cost parameter C and the RBF-kernel parameter γ from grid-search results during three-fold cross validation.

DUD-E Target name	Training data		Best parameter	
	#Active compounds	#Inactive compounds	C: Cost parameter	γ: RBF-kernel parameter
ACES	1,635	487	2^3	2^{-5}
EGFR	1,620	407	2^3	2^{-7}
PGH1	543	1,070	2^3	2^{-5}

of compounds selected by Spresso will be maintained. We analyzed the diversity of compounds selected by Spresso according to two characteristics: physicochemical features and structural diversity. We focused on three DUD-E targets (PGH1, ACES, and EGFR) and screened ZINC compounds using Spresso-SP, Glide HTVS, and a ligand-based method. As for the ligand-based method, a support vector machine (SVM) with RBF kernel was adopted because it is one of the most popular machine learning methods for ligand-based screening. ECFP4 fingerprint [88] was used for input feature vectors of SVM. The details associated with the SVM are shown in Table 3.4. The logP and the molecular weight of the top 0.1% of compounds were calculated in order to assess the bias of physicochemical features. Additionally, the maximum Tanimoto coefficient value between each known active compound was also calculated based on ECFP4 fingerprint in order to assess structural diversity. A high Tanimoto coefficient between two compounds indicates that the two structures share structural similarity.

LogP-MW scatter plots of the ACES, EGFR, and PGH1 targets are shown in Figs. 3.15, 3.16, and 3.17, respectively. Fig. 3.15 shows that Spresso is likely to assign higher scores to large compounds. This is expected in some cases, because larger compounds are more likely to obtain higher scores in docking simulations [89]; however, compounds that are too large to enter protein cavities must be omitted. Structural diversity assessment results of ACES, EGFR, and PGH1 (Figs. 3.18, 3.19, and 3.20) show that Spresso conserved structural diversity on the same scale as that observed with Glide HTVS, while bias toward known active compounds was observed in results from the ligand-based method (SVM).

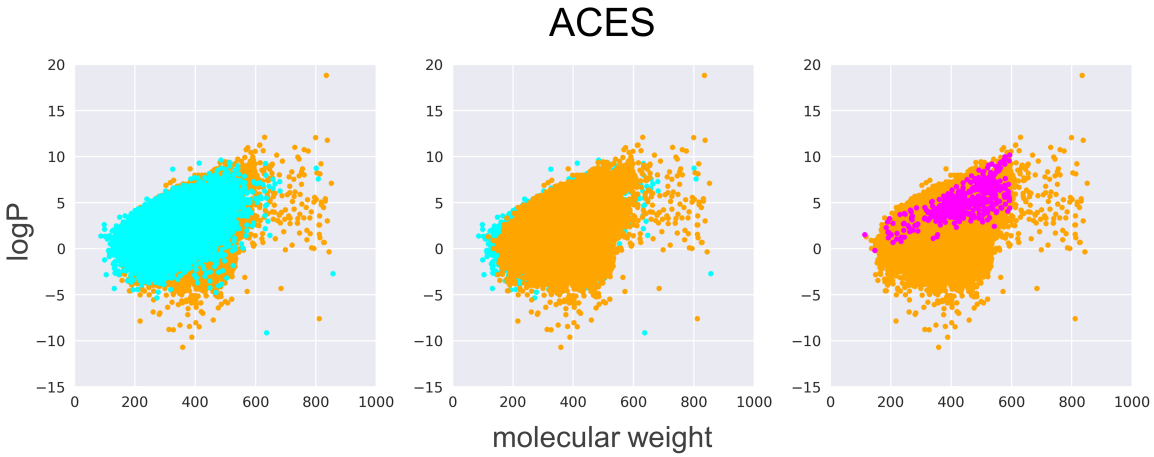


Figure 3.15: Scatter plot of physicochemical features based on pre-screening for ACES, a DUD-E protein target. Each dot represents a compound: cyan dots represent 0.1% of the compounds from the ZINC database; orange dots represent the top 0.1% of Spresso-HTVS compounds calculated using the method (VII) $ENE_{1.8}$ formula; and magenta dots represent active compounds for ACES from the DUD-E dataset.

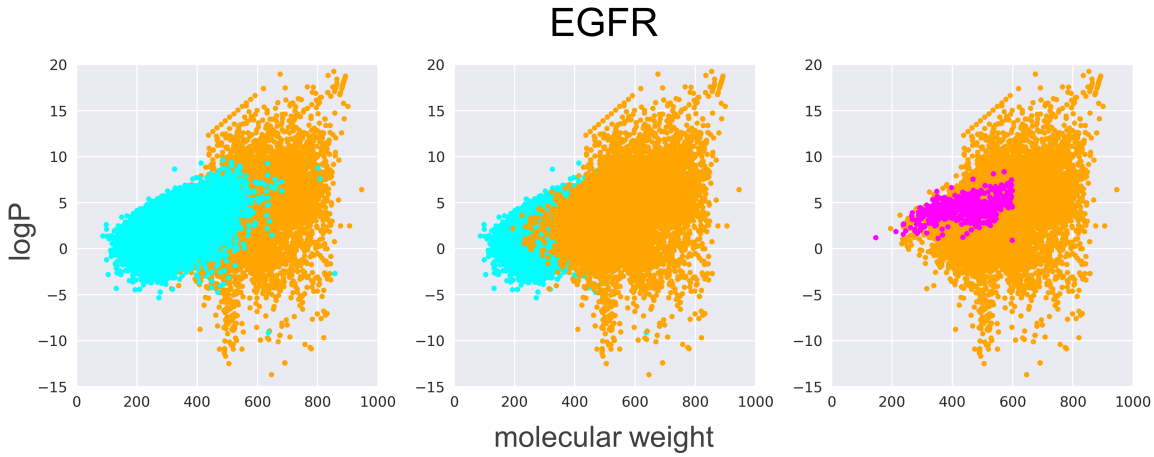


Figure 3.16: The scatter plot of physicochemical features based on pre-screening for EGFR, a DUD-E target. Each dot represents a compound: cyan dots represent 0.1% of compounds from the ZINC database, orange dots represent the top 0.1% of Spresso-HTVS results using the method (VII) $ENE_{1.8}$ formula, and magenta dots represent active compounds for EGFR from the DUD-E dataset.

3.4.4 The top-screened compounds by Spresso

The highest scoring compound in DUD-E for the target protein EGFR is shown in Fig. 3.21. The top compound screened by Spresso was ZINC16956948 (Fig. 3.21A), with a molecular weight of 370.8 Da and a logP of 2.79. These physicochemical features indicate a likely drug compound according to Lipinski's rule of five [60]. The decomposition and fragment-docking results are shown in Fig. 3.21B and Fig. 3.21C. Since Spresso did not consider collisions and connectivities between fragments in order to keep computation time low, some fragments appear to have collided or the connection is totally broken (Fig. 3.21C). Interestingly, the best compound still exhibited a reasonable molecular weight according to Lipinski's rule despite the problem.

3.5 Conclusion

In this study, we described Spresso, a docking-based pre-screening method for database-wide screening. In order to evaluate all compounds from large databases within a practical amount of time, Spresso uses compound decomposition into fragments, resulting in reuse of fragment scores, followed by fragment-docking results to estimate screening values without structure reconstruction. Our results showed that Spresso achieved up to ∼200-fold faster calculation using ∼29 million compounds as compared to compound docking by Glide HTVS. This acceleration rate is positively correlated to the number of compounds in a target database. Consequently, this method is capable of screening over tens of millions of compounds with limited computational resources.

For compound evaluation, the $ENE_{1.8}$ formula was adopted; however, according to the physicochemical assessment, Spresso-preferred compounds are likely to be large, despite the need to filter compounds too large for a given target protein cavity.

The computational efficiency of Spresso enables the screening of large compound databases within realistic times. In order to manage chemical compound libraries that continue to increase in size, corresponding increases in computational speed are necessary for virtual screening.

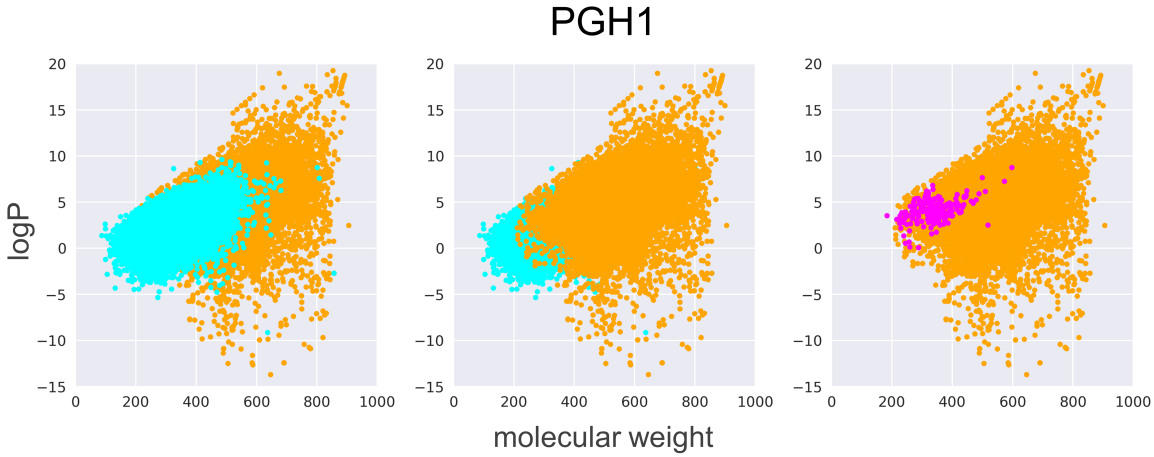


Figure 3.17: The scatter plot of physicochemical features based on pre-screening for PGH1, a DUD-E target. Each dot represents a compound: cyan dots represent 0.1% of compounds from the ZINC database, orange dots represent the top 0.1% of Spresso-HTVS results using the method (VII) $ENE_{1.8}$ formula, and magenta dots represent active compounds for PGH1 from the DUD-E dataset.

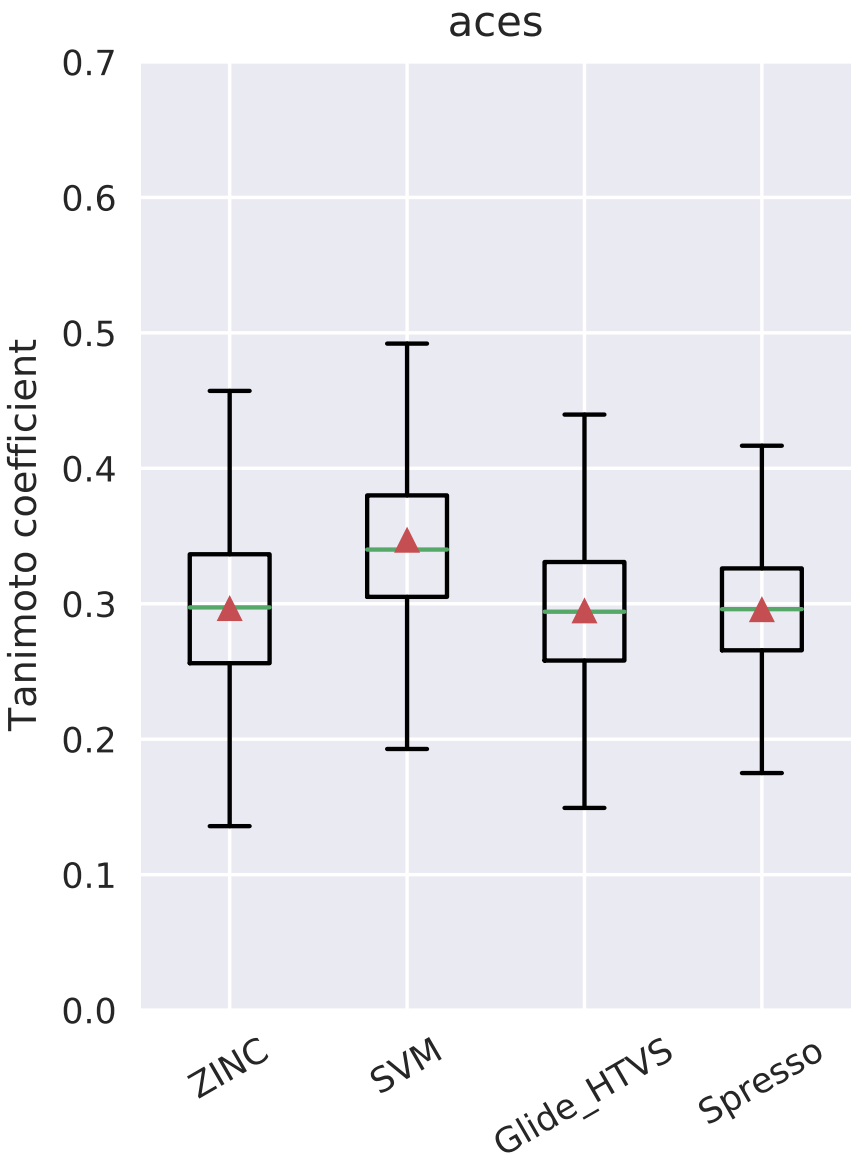


Figure 3.18: Boxplot representation and average (triangles) of the ECFP4 maximum Tanimoto coefficient between active compounds of target ACES. The data indicate structural diversity. ZINC, SVM, Glide HTVS, and Spresso represent 0.1% of randomly selected compounds from the ZINC database, the top 0.1% of compounds resulting from SVM prediction, the top 0.1% of compounds resulting from Glide HTVS scoring, and the top 0.1% of compounds returned from Spresso-HTVS results using method (VII) $ENE_{1.8}$ scoring, respectively.

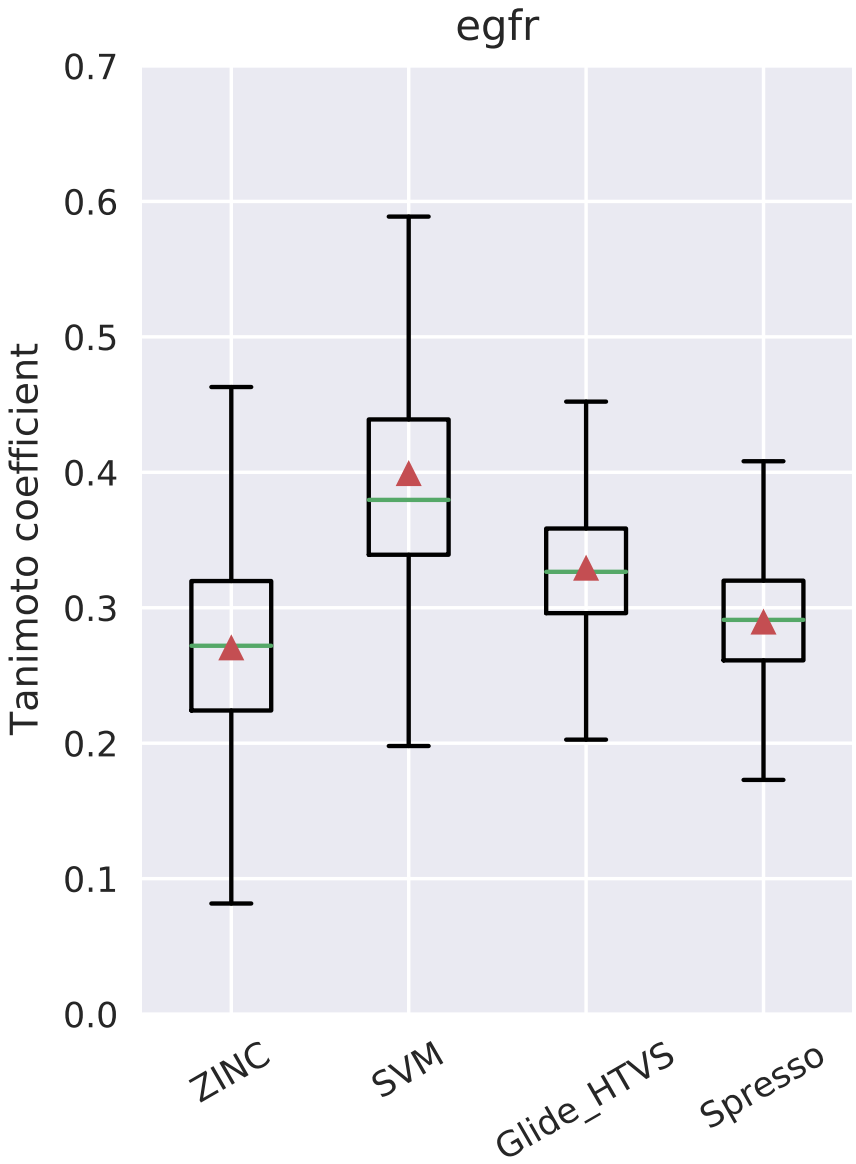


Figure 3.19: Boxplot representation and average (triangles) of the ECFP4 maximum Tanimoto coefficient between active compounds of target EGFR. The data indicate structural diversity. ZINC, SVM, Glide HTVS, and Spresso represent 0.1% of randomly selected compounds from the ZINC database, the top 0.1% of compounds resulting from SVM prediction, the top 0.1% of compounds resulting from Glide HTVS scoring, and the top 0.1% of compounds returned from Spresso-HTVS results using method (VII) $ENE_{1.8}$ scoring, respectively.

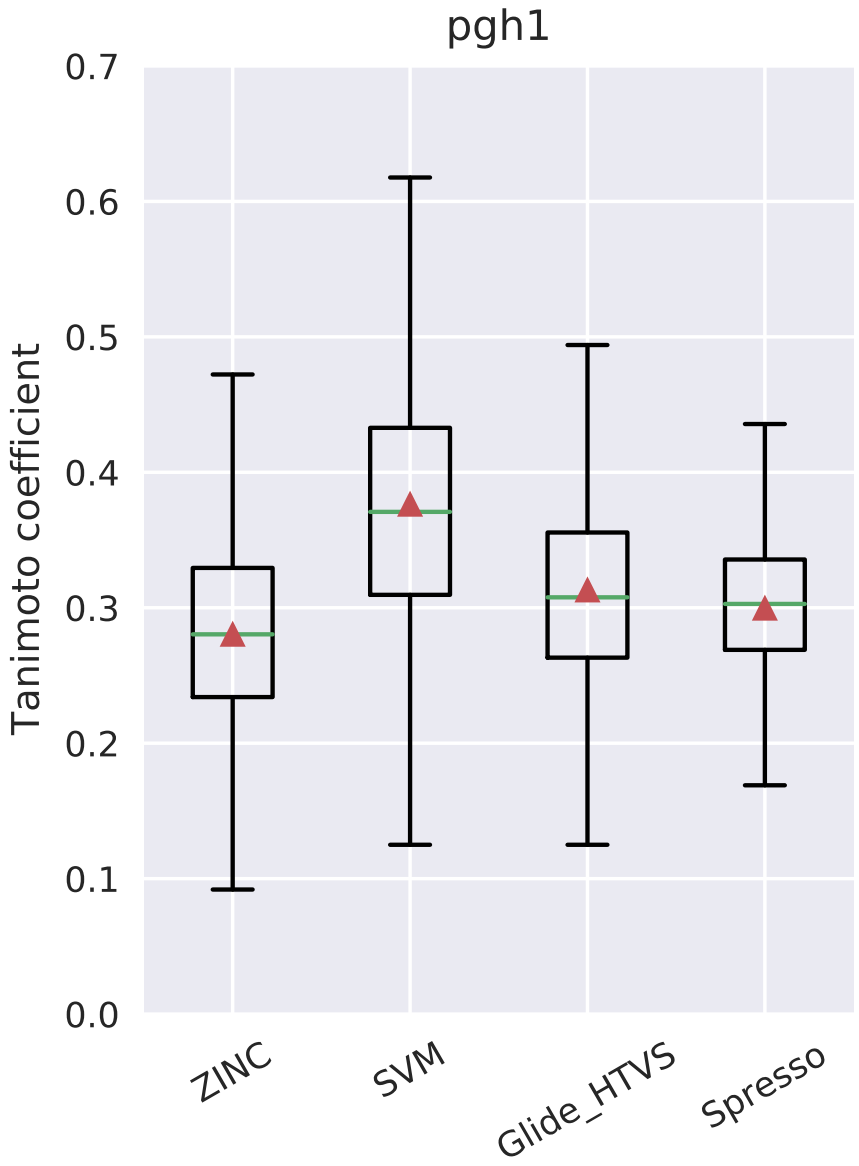


Figure 3.20: Boxplot representation and average (triangles) of the ECFP4 maximum Tanimoto coefficient between active compounds of target PGH1. The data indicate structural diversity. ZINC, SVM, Glide HTVS, and Spresso represent 0.1% of randomly selected compounds from the ZINC database, the top 0.1% of compounds resulting from SVM prediction, the top 0.1% of compounds resulting from Glide HTVS scoring, and the top 0.1% of compounds returned from Spresso-HTVS results using method (VII) $ENE_{1.8}$ scoring, respectively.

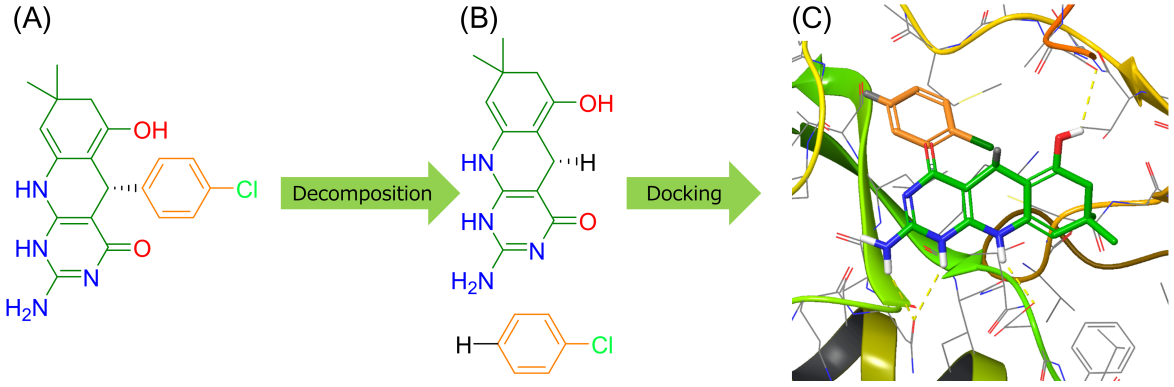


Figure 3.21: An example of Spresso procedure. (A) Structure of ZINC16956948, the highest scoring compound for the protein target EGFR. (B) Result of ZINC16956948 decomposition. (C) Results of fragment docking. The color of the structure mimics those of the structures shown in (A) and (B).

Part IV

Protein-ligand Docking calculation with reuse of fragment

Chapter 4

Toward development of protein–ligand docking tool with substructure commonality

4.1 Introduction

Spresso is a fragment-based compound pre-screening method by storing only the best docking score of each fragment. The method achieved a calculation speed of approximately 200 times that of the Glide [37] HTVS mode although its accuracy is lower because it does not consider the crush or connectivity between fragments. On the other hand, some docking tools, such as eHiTS [38], dock compounds by first dividing them into fragments (partial structures). After that, candidate conformations and corresponding docking scores for each individual fragment are calculated and memorized (Fig. 4.1), followed by reconstruction of the conformation of the whole compound by considering the crushes and connectivities by using the stored partial docking results. Therefore, when two or more compounds contain the same fragment, it is possible to reuse the docking result of that fragment. Furthermore, fragment extension-based methods, such as FlexX [54], can reuse the intermediate results of fragment extension with an appropriate arrangement of the order in which compounds are evaluated (Fig. 4.2). In particular, eHiTS is one of the most accurate docking tools [90] and has a functionality that enables the fragment docking results to be saved in an SQL-based database, thereby achieving 2–4 times speed-up for docking with several hundred compounds to several thousand compounds. However, for about 10,000 compounds, it was reported that the speed-up ratio tends to level out [38]. The factor responsible for

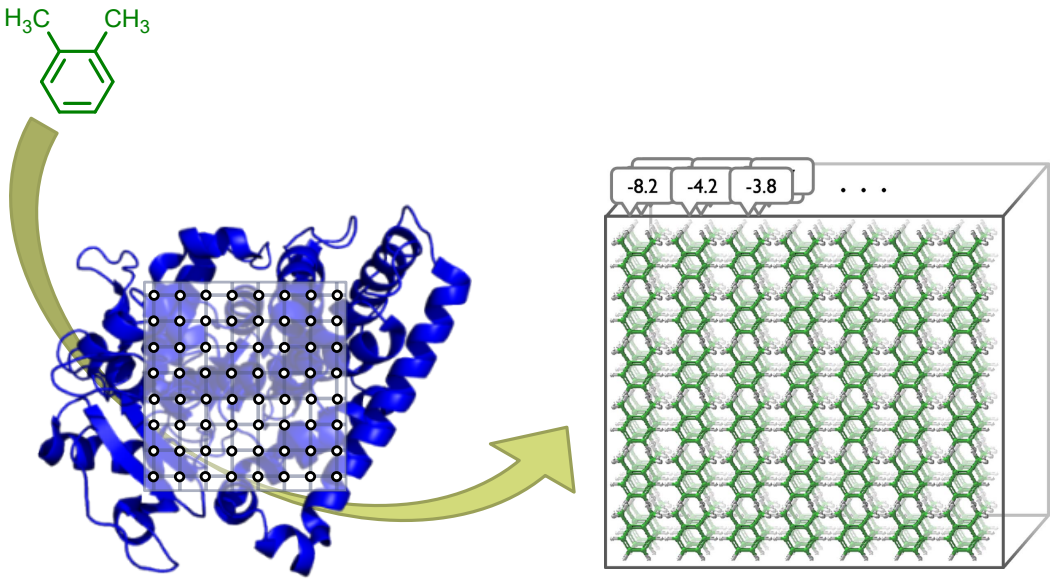


Figure 4.1: Image of a fragment docking result. The square containing the dots on the protein structure show the docking boundary and docking points, respectively. The box on the right is the fragment docking result of ortho-xylene (green fragment) with the corresponding docking scores.

decreasing the speed is disk access. This cost arises frequently since the SQL database is accessed every time to confirm the existence of data and to load them into the main memory before performing a fragment docking calculation. Further speed-up would require storing the fragment docking results in the main memory (rather than on disk) as much as possible, as a matter of importance, to enable the existence of data to be verified and to utilize data faster.

In this chapter, we explain the docking procedure which reuse intermediate results of fragments, and the pros and cons of the procedure. Additionally, we revealed that the caching algorithm can solve the weakness of the procedure.

4.2 Overview of proposed docking procedure

Whole procedure is shown in Fig. 4.3. The procedure is based on anchor and grow algorithm, and is separated into four sub-procedures. In this section, we firstly mentioned about the utilized docking algorithm, followed by explanation of each sub-procedure.

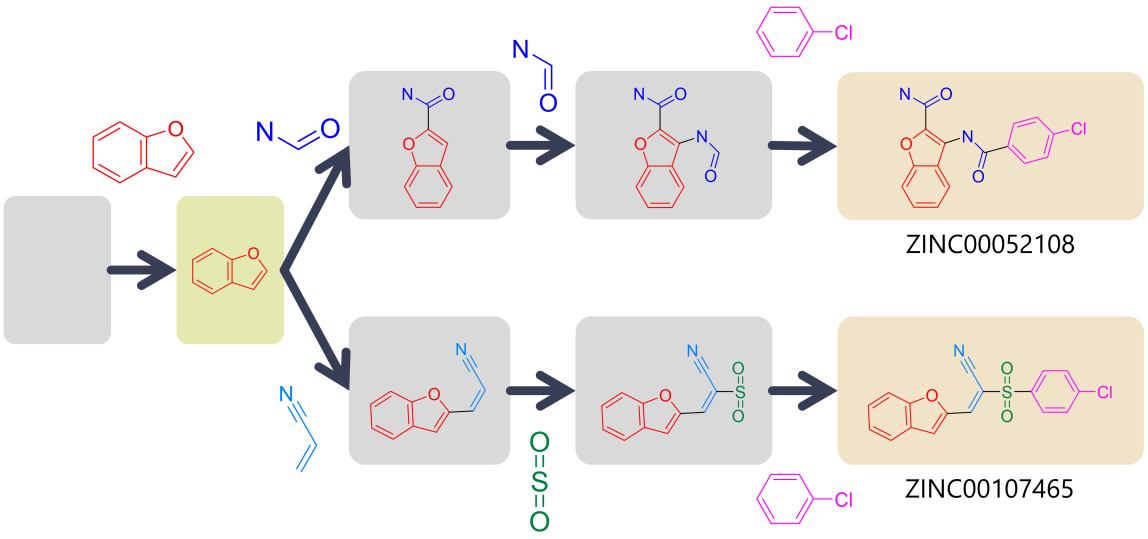


Figure 4.2: Reuse of fragment extension results. The red fragment is first included in the two compounds; therefore, it is possible to reuse the intermediate results for fragment extension.

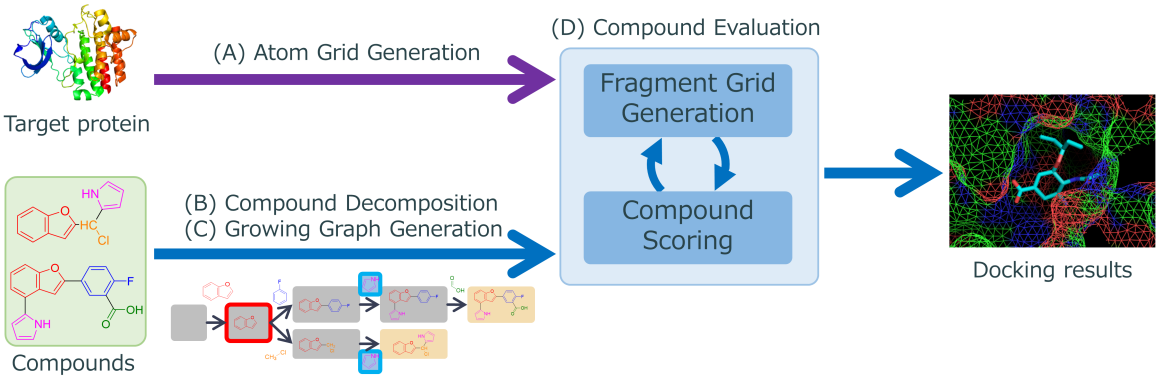


Figure 4.3: Docking procedure

4.2.1 docking algorithm

As we mentioned in Chapter 1, several algorithms for fragment-based docking have been proposed (e.g. consistent allocation of fragments by maximum clique finding (eHiTS), anchor and grow, or compound allocation by fragment extension (DOCK, FlexX)). We proposed a docking procedure based on anchor and grow algorithm because of its possibility of speediness. Since maximum clique finding is NP-complete

problem, thus it will be computationally expensive. Furthermore, intermediate results of anchor and grow algorithm can be reused as shown in Fig. 4.2.

4.2.2 (A) Atom grid generation

In the first step, atom grid for the target protein is generated. Atom grid, or merely grid, is a cubic lattice, which contains the interaction scores beetween an atom of ligand and all atoms of protein. It is commonly generated and used for protein-ligand docking to accelerate the calculation [91].

4.2.3 (B) Compound decomposition

Secondly, compounds are decomposed into fragments, with the decomposition method mentioned in Chapter 2. The decomposition method make fragments with no internal degrees of freedom.

In addition, growing orders, or fragment placement orders of compounds are decided. The decision of a base fragment, or a fragment which place into the cavity firstly, affects the docking result, as the article of FlexX mentioned "The docking algorithm is quite sensitive to the selection of the base fragment" [54]. Thus the procedure tests all fragments in a compound as the base. After that, largest fragment adjacented to placed substructure is selected as "next" fragment.

4.2.4 (C) Growing graph generation

In next step, the docking order of compounds, named growing graph, is generated. It is noted that docking order of compounds and growing order of each compound must be decided to generate the graph. The growing orders are sorted in lexicographical order of SMILES of fragments to maximize the reuse of intermediate results of anchor and grow algorithm. After that, common growing orders are unified, resulting in generation of growing graph.

The procedure is graphically shown in Fig. 4.4.

4.2.5 (D) Compound evaluation

Finally, compounds are evaluated with atom grids and the growing graph. The evaluation process is done with two sub-steps: fragment grid generation and compound scoring.

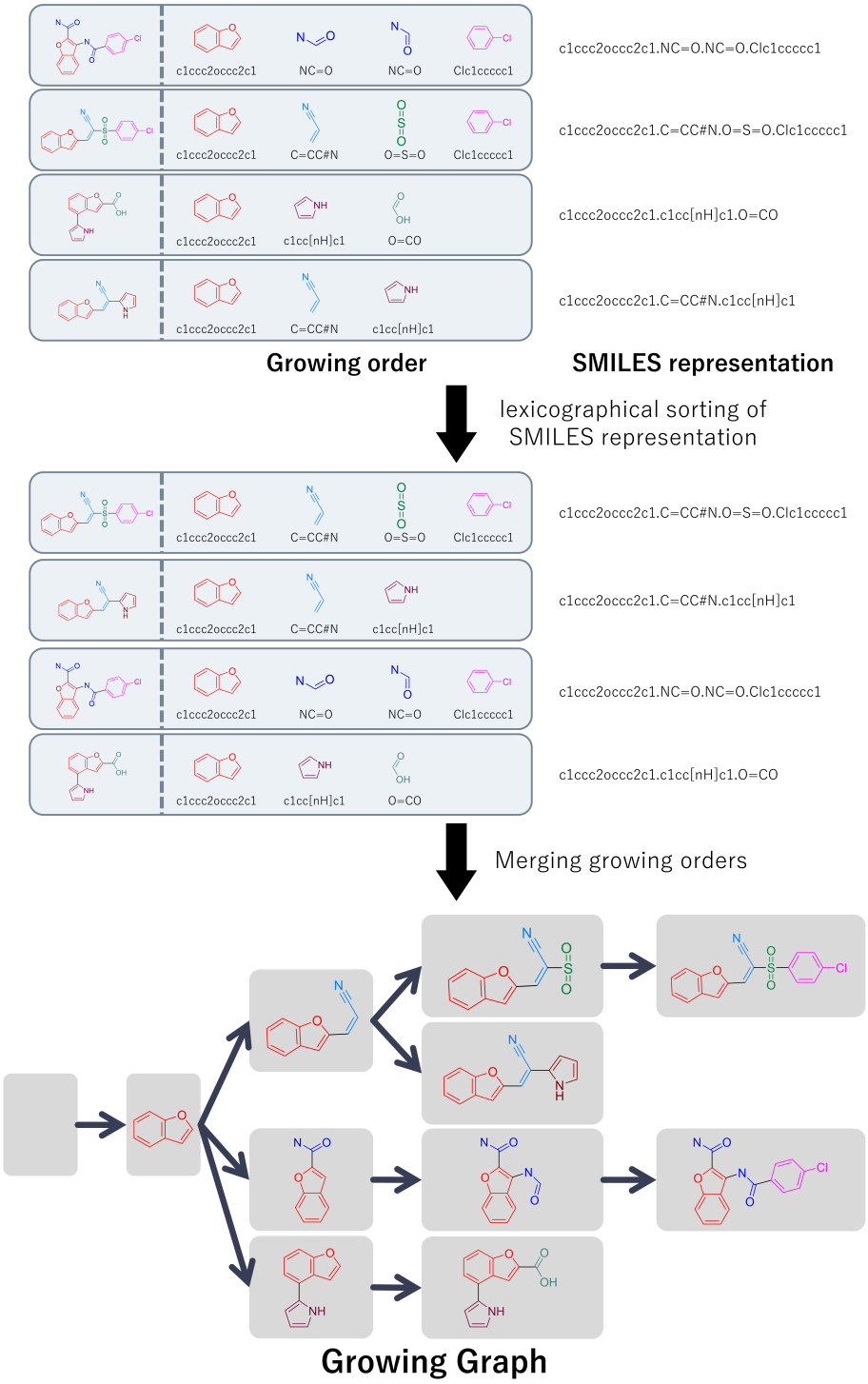


Figure 4.4: The procedure of growing graph generation

Fragment grid generation

Fragment grid is a natural extension of atom grid for fragments. The difference between the atom grid and the fragment grid is rotation. Atoms are regarded as sphere, thus they have rotational invariance. On the other hand, fragments are not sphere, thus each lattice point in the grid must have multiple scores for rotated structures. The fragment grid can be also reused in multiple growing orders, thus further acceleration will be realized.

Compound docking and scoring

Compounds are placed with fragment grids. The placement procedure consists of base placement step and complex construction steps. Base placement phase is to place base fragment into the cavity without any structural restriction while complex construction steps are to extend the substructure already placed in the cavity with consideration of connectivity of the cleaved bond.

4.3 Strengths and weakness of proposed procedure

4.3.1 Strengths

The proposed procedure can reuse two types of intermediate results, fragment grids and intermediate states of growing, as we mentioned. The acceleration of docking calculation without degradation of accuracy is highly expected.

4.3.2 Weakness: memory usage problem

The procedure generates several fragment grids (e.g. 263,319 fragments for ZINC library as we showed in Chapter 2), while the grids spend much memory because of the rotation. The memory consumption per fragment is, for instance, more than 100 MB if the variation of rotation set to 60 (The number is same as eHiTS [38]). It results in more than 25 TB memory consumption, which is thus far practically impossible to store at a same time.

4.4 Caching algorithm: A solution of the memory usage problem

Since all fragment grids cannot be stored at a same time, it is important to effectively use limited memory space. This problem can be considered as a cache problem in the field of computer architecture.

The accession to the main memory is done in hierarchical way: obtaining data from the cache space if there is (called "cache hit"), otherwise access to access main memory space, where is quite farer and larger than the cache space (called "cache miss"). The average memory access time (AMAT) is calculated with below formula:

$$\text{AMAT} = t_{\text{cache}} + \text{MR} \cdot t_{\text{MEM}} \tag{4.1}$$

where $t_{\text{cache}}, t_{\text{MEM}}, \text{MR}$ are average access time to cache, average access time to main memory, cache miss ratio, respectively.

The generation cost of fragment grid depends on the number of atoms of a fragment, thus the formula is slightly different from the original AMAT.

$$\text{Fragment grid obtaining time (FGOT)} = \begin{cases} t_{\text{MEM}}, & \textbf{when} \text{ cache hit} \\ \text{HAC} \cdot t_{\text{calc}}, & \textbf{when} \text{ cache miss} \end{cases} \tag{4.2}$$

$$\text{Total FGOT} = \left(\sum_{f \in F} t_{\text{MEM}} \right) + \left(\sum_{f \in F/cache} \text{HAC}_f \right) \cdot t_{\text{calc}} \tag{4.3}$$

where $\text{HAC}, t_{\text{calc}}, F$ are the heavy atom count (HAC) of a fragment f, the average calculation time to generate a single-atom fragment grid, series of needed fragments among calculation, respectively.

If the memory space is enough to store all fragment grids, cache miss will happen only one time per fragment type. Thus Total FGOT_{Ideal} is:

$$\text{Total FGOT}_{Ideal} = \left(\sum_{f \in F} t_{\text{MEM}} \right) + \left(\sum_{f \in F_{uniq}} \text{HAC}_f \right) \cdot t_{\text{calc}} \tag{4.4}$$

On the other hand, the computation time without reuse can be calculated as cache

miss ratio is equal to 1, thus Total $\mathrm{FGOT}_{\mathrm{w/o}}$ is:

$$\text{Total FGOT}_{w/o} = \left(\sum_{f \in F} t_{\mathrm{MEM}}\right) + \left(\sum_{f \in F} \mathrm{HAC}_f\right) \cdot t_{\mathrm{calc}} \tag{4.5}$$

$\sum_{f \in F_{uniq}} \mathrm{HAC}_f$ is equal to the total atoms among fragments while $\sum_{f \in F} \mathrm{HAC}_f$ is equal to the total atoms among all growing orders. It is important that the selection only affects the summation of HAC in the second term of (4.3) when the same compound set are calculated, thus the reduction of calculation cost can be derived:

$$\text{Total FGOT} - \text{Total FGOT}_{w/o} = \left(\sum_{f \in F} \mathrm{HAC}_f - \sum_{f \in F/cache} \mathrm{HAC}_f\right) \cdot t_{\mathrm{calc}} \tag{4.6}$$

Storing all fragment grids at a same time is obviously impossible in the real situation, thus selection of fragment grid storage is needed. The simplest algorithm is storing the most frequent fragment grids from beginning to end while it must be better if the optimum can be derived. Tables 4.1–4.3 show the difference of the summation of HAC between *Ideal*, *w/o*, *Simplest*, and *Optimum* under the assumption of 100,000 compounds calculation with 100 grids of memory space. Simplest method can reduce 30–85% of ideal reduction, while optimization can reduce additional a few hours approximately if it is possible within realistic time consumption. Especially, the optimization is more effective with limited memory space.

In the next Chapter, we detailed the memory usage problem and reduced it as a polynomial-time problem. Additionally, we proposed more efficient optimization algorithm.

Table 4.1: Estimated reduction of fragment grid generation time with memory space of 10 fragment grids. Percentages in the parentheses are the relative calculation time reduction compared to the *Ideal*.

		$\left(\sum_{f\in F}\mathrm{HAC}_f - \sum_{f\in F/cache}\mathrm{HAC}_f\right)\cdot t_{\mathrm{calc}}$	
without reuse	(w/o)	0 core sec	(0.0%)
most frequent	$(Simplest)$	25,253 core sec	(29.1%)
optimized	$(Optimum)$	36,201 core sec	(41.8%)
infinite memory	$(Ideal)$	86,650 core sec	(100.0%)

Note: the experimental conditions are follows. compounds: 100,000 compounds randomly sampled from ZINC all purchasable and all boutique subsets, t_{calc}: 10 milliseconds per HAC per fragment grid.

Table 4.2: Estimated reduction of fragment grid generation time with memory space of 100 fragment grids. Percentages in the parentheses are the relative calculation time reduction compared to the *Ideal*.

		$\left(\sum_{f\in F}\mathrm{HAC}_f - \sum_{f\in F/cache}\mathrm{HAC}_f\right)\cdot t_{\mathrm{calc}}$	
without reuse	(w/o)	0 core sec	(0.0%)
most frequent	$(Simplest)$	56,711 core sec	(65.4%)
optimized	$(Optimum)$	61,807 core sec	(71.3%)
infinite memory	$(Ideal)$	86,650 core sec	(100.0%)

Note: the experimental conditions are follows. compounds: 100,000 compounds randomly sampled from ZINC all purchasable and all boutique subsets, t_{calc}: 10 milliseconds per HAC per fragment grid.

Table 4.3: Estimated reduction of fragment grid generation time with memory space of 1,000 fragment grids. Percentages in the parentheses are the relative calculation time reduction compared to the *Ideal*.

		$\left(\sum_{f\in F}\mathrm{HAC}_f - \sum_{f\in F/cache}\mathrm{HAC}_f\right)\cdot t_{\mathrm{calc}}$	
without reuse	(w/o)	0 core sec	(0.0%)
most frequent	$(Simplest)$	74,858 core sec	(86.4%)
optimized	$(Optimum)$	77,914 core sec	(89.9%)
infinite memory	$(Ideal)$	86,650 core sec	(100.0%)

Note: the experimental conditions are follows. compounds: 100,000 compounds randomly sampled from ZINC all purchasable and all boutique subsets, t_{calc}: 10 milliseconds per HAC per fragment grid.

Chapter 5

Optimization of memory use of fragment extension-based protein-ligand docking with an original fast minimum cost flow algorithm

5.1 Introduction

In previous Chapter we described protein-ligand docking procedure with reuse of intermediate results, and mentioned the memory usage problem. To use main memory more efficiently, the optimization of construction, storing, and destruction of fragment grids is inevitable, and thus we also mentioned caching algorithm.

In this Chapter, we first show the formulation and the reduction of the sequence of requested fragment docking results to a graph of the minimum cost flow problem via the weighted offline cache problem. Second, the characteristics of the obtained graph are described, followed by the proposal of a faster algorithm based on these characteristics. The whole structure of our method is illustrated in Fig. 5.1.

5.2 Method

In our docking procedure, compounds are firstly sorted based on growing orders, thus the requiring order of fragment grids is decided before the calculation. The optimization

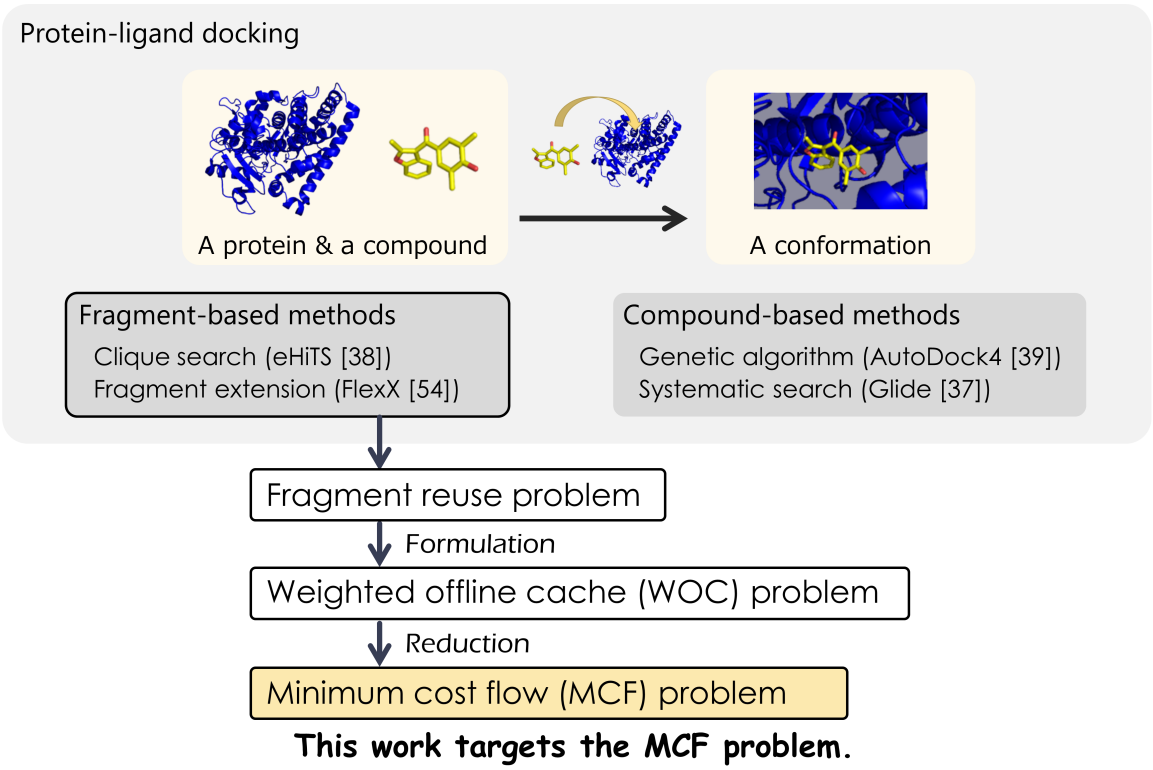


Figure 5.1: Overview of this work

of memory usage with known requiring order can be considered as the weighted offline cache problem [92], which is related to cache algorithms that have been widely studied in the field of computer science. This makes it possible to optimize the choice of data to be stored in the main memory.

The weighted offline cache problem involves minimizing the cost with the following constraints:

- The cache can have a limited number of pages (denoted as k).
- The input consists of a sequence of page requests, and its ordering is perfectly observable in advance (i.e., "offline").
- The cost of caching varies from page to page.

The cache and a page correspond to the main memory and a fragment docking result, respectively. Lopez-Ortiz *et al.* showed that the weighted offline cache problem can be reduced to the minimum cost flow problem [93]. The latter problem has been

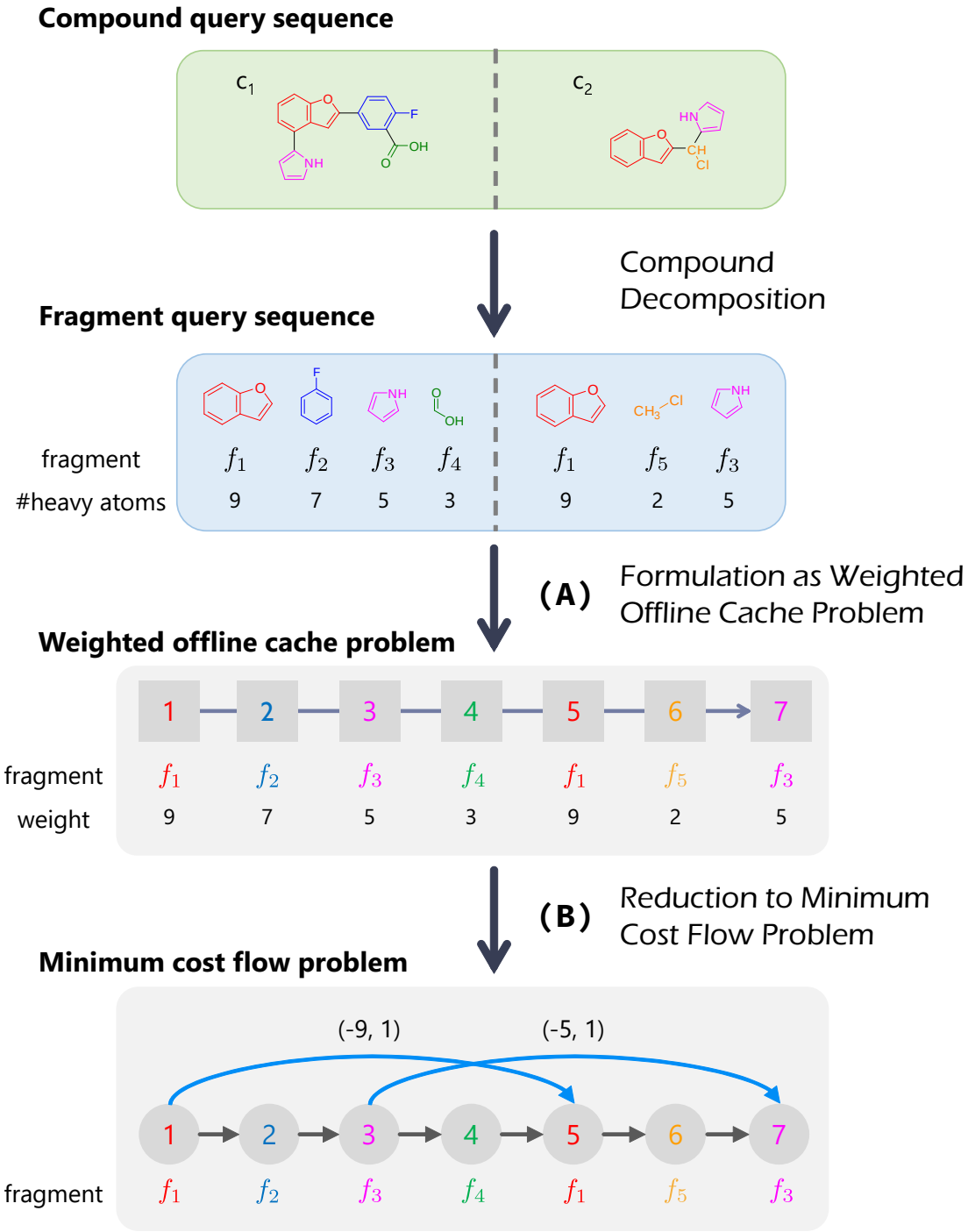


Figure 5.2: The flow of formulation and reduction. it illustrates the reduction of the results of the access order of fragment docking to a weighted offline cache problem, and finally to a minimum cost flow problem.

extensively studied since the proposal of the Successive Shortest Path algorithm in 1958, which searches for the shortest path iteratively until the amount of flow is equal to k [94, 95, 96, 97]. The Cost-Scaling algorithm is a generalization of the push-relabel algorithm and the maximum value of the edge cost influences the computational complexity [98]. The Network-Simplex algorithm uses a simplex method for efficiency improvement [99, 100, 101]. A fast algorithm specified for directed acyclic graphs has also been proposed by Pirsiavash *et al.* [102].

The graph in a minimum cost flow problem generated from a weighted offline cache problem has specific characteristics that facilitate acceleration.

5.2.1 Formulation of the memory usage problem as the weighted offline cache problem (Fig. 5.2–(A))

The memory usage problem can be considered as cache problem, as we mentioned in Section 4.4. Additionally, it is possible to know the precise order in which fragments are accessed before the fragment docking calculation is started because compound evaluation order is determined beforehand to maximize the reuse of fragment extension results (Fig. 4.4). Therefore, the problem is offline, which is easier problem than online problem. On the other hand, the generation cost of a fragment grid is not uniform but almost proportional to the heavy atom count of the fragment, which means this problem has a kind of weight. Because of the facts, the memory usage problem can be considered as weighted offline cache problem.

A fragment docking result is represented as a page in a weighted offline cache problem, where the size of the cache represents the size of the main memory. Further, the heavy atom count of the each fragment represents as the weight of each fragment in the weighted offline cache problem.

5.2.2 Reduction to the minimum cost flow problem (Fig. 5.2–(B))

The reduction from the weighted offline cache problem to the minimum cost flow problem was shown [93]. In this part, we briefly show how to make a minimum cost flow problem. Performing the following operations on the page (fragment docking result) request sequence in a weighted offline cache problem enables the problem to be reduced to a minimum cost flow problem that can derive the optimal sequential order in which to store the required pages in the cache (main memory).

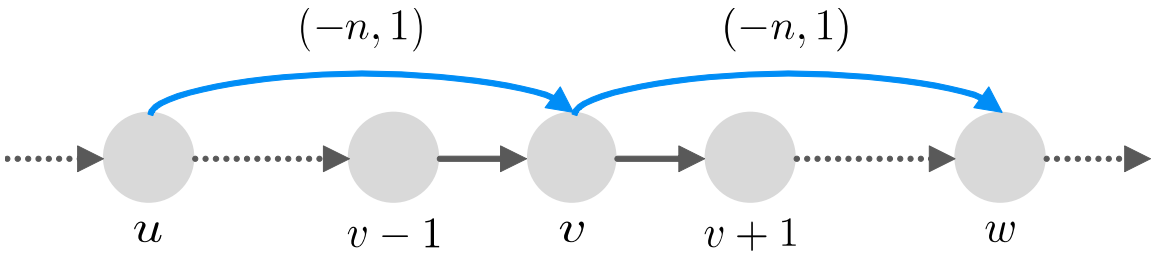


Figure 5.3: A node in the reduced graph, where u and w are the immediate previous/next nodes originated from the same fragment request as v. (These previous/next nodes may not exist.) n is the generation cost of the fragment grid v, which is equal to the number of heavy atoms of the fragment. It is noticed that every node has a maximum of four edges (0–2 reuse edges and 2 non-reuse edges).

1. All page requests are represented as nodes. The nodes are sorted uniquely in relation to the page request order (circles in Fig. 5.3).

2. For each node, a "reuse" edge connected to the next appearing node, which shares the same fragment request, is generated. The edge is given the properties of $(cost, capacity) = (\text{a negative of the page request cost}, 1)$ (blue arrows in Fig. 5.3).

3. For each node, a "non-reuse" edge to the next node is generated. The edge is $(cost, capacity) = (0, \infty)$ (black arrows in Fig. 5.3).

4. The total amount of flow is equal to k, the number of pages the cache can store.

5.2.3 Characteristics of the reduced graph

The graph $G(V, E)$ of a minimum cost flow problem generated from a weighted offline cache problem can arrange nodes V in one row, with one node v having no more than four edges, as shown in Fig. 5.3. Therefore, the reduced graph has the following characteristics:

- It is a directed acyclic graph.
- Unique topological sorting is possible.
- The graph is sparse ($O(|E|) = O(|V|)$)
- The capacity of an edge is 1 or ∞.

In consideration of the special characteristics of the graph, we propose a faster algorithm in the next section.

5.2.4 Proposed method: an exact algorithm for the minimum cost flow problem generated from the weighted offline cache problem

The proposed method is shown in Algorithm 5.1 and 5.2, which is based on the Successive Shortest Path (SSP) algorithm. In the SSP algorithm, the ShortestPath function repeatedly determines the minimum cost path from a source to a sink and updates the flow and the graph until the path becomes equal to k. The CalcCosts function searches the minimum cost path from the source to the sink based on a dynamic programming method. An example showing how the dynamic programming table is updated is presented in Fig. 5.4.

Algorithm 5.1 Successive shortest path (SSP) algorithm

```
1: function SuccessiveShortestPath(G, d)
                                          ▷ G, d: graph, total amount of flow
2:     f_0 ← 0                                  ▷ f:current amount of flow
3:     G_{f_0} ← G                          ▷ G_f: graph when the flows are f
4:     i ← 0
5:     while f_i ≠ d do
6:         i ← i + 1
7:         P_i ← ShortestPath(G_{f_{i-1}})
8:         f_i, G_{f_i}, cost_i ← UpdateStates(f_{i-1}, G_{f_{i-1}}, P_i, cost_{i-1})
                                  ▷ A function to update the graph, flow, and cost
9:     end while
10: end function

11: function ShortestPath(G)                  ▷ source and sink are v_1 and v_n
12:     prev ← CalcCosts(G)
13:     path ← GeneratePath(prev)
                        ▷ A function generating the path from the source to the sink
14:     return path
15: end function
```

Algorithm 5.2 Proposed Algorithm

```
1: function CalcCosts(G)
2:     cost ← {cost(i) = 0|1 ≤ i ≤ n}
3:     prev ← {prev(i) = i − 1|1 ≤ i ≤ n}
4:     now ← 1
5:     while now < n do
6:         updated ← φ
7:         for all {e|e ∈ G.E, e.from = now} do
                                          ▷ e.from : a start point of the edge e
                                            ▷ e.to : an end point of the edge e
                                              ▷ e.cost : a cost of the edge e
8:             if cost(e.to) > cost(now) + e.cost then
9:                 cost(e.to) ← cost(now) + e.cost
10:                prev(e.to) ← now
11:                updated.add(e.to)
12:            end if
13:        end for
14:        now ← min(updated, now + 1)
15:    end while
16:    return prev
17: end function
```

5.2.5 Proving the optimality of the proposed algorithm

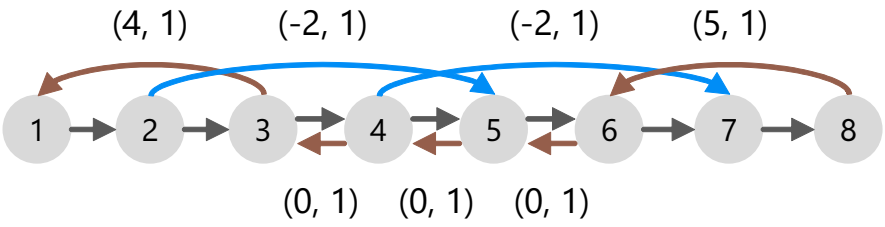
In the framework of the SSP algorithm, it suffices to prove the optimality of the ShortestPath function to prove the optimality of the minimum cost flow algorithm.

Lemma 5.1. CalcCosts *function terminates in finite steps.*

Proof. The function will terminate if the number of while loops is finite. In the while loop (lines 5–15 of Algorithm 5.2), value *now* will increase except for the backward updates. On the other hand, the values of *cost* are no less than the costs of the true shortest path which are finite numbers. It means the number of updates is finite, therefore, the number of while loop is also finite. The fact reveals that the CalcCosts function terminates in finite steps. □

Lemma 5.2. *In the* CalcCosts *function, the number of steps the while loop required to finish the operation of the last* $now = i$ *defines* t_i *(According to the Lemma 5.1,* t_i *is finite number), and the value of* $cost(v)$ *at step* t *defines* $cost[t][v]$*. If there is an edge* $(i, j) \in E$*, the following inequality holds:*

$$cost[t][j] \leq cost[t][i] + (i, j).\text{cost} \quad \textbf{if} \quad t \geq t_i$$



1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	**-2**	0	0	0
0	0	0	0	-2	0	0	0
0	0	0	0	-2	0	**-2**	0
0	0	0	**-2**	-2	**-2**	-2	0
0	0	**-2**	-2	-2	-2	**-4**	0
0	0	-2	-2	-2	-2	-4	0
0	0	-2	-2	-2	-2	-4	0
0	0	-2	-2	-2	-2	-4	0
0	0	-2	-2	-2	-2	-4	0
0	0	-2	-2	-2	-2	-4	**-4**

Figure 5.4: Example of the execution of the CalcCosts function. The $(cost, capacity)$ labels are shown for valid edges. Edges without labels have $(0, \infty)$. The cell at row i and column j of the table shows the $cost(j)$ at step i. The cells updated in each step are highlighted.

Proof. According to lines 7–13 of Algorithm 5.2, all nodes connected from node v_i are updated every time at $now = i$ thus we obtain $cost[t_i][j] \leq cost[t_i][i] + (i, j).\text{cost}$. After time t_i (define the time as $t \geq t_i$), $cost[t][i] = cost[t_i][i], cost[t][j] \leq cost[t_i][j]$ are satisfied; thus, the Lemma 5.2 is satisfied. □

Lemma 5.3. *The* CalcCosts *function can find the exact minimum cost path.*

Proof. The proof is by contradiction. Suppose that $cost[t_n][n] > \hat{d}_n$ ($\hat{d}_n$ represents the cost of the true shortest path from source v_1 to sink v_n).

If $C(i) \equiv cost[t_n][i] - \hat{d}_i$, it is true that $C(n) > 0, C(1) = 0$. Therefore, there is at least one edge $(p, q) \in E, C(p) \leq 0, C(q) > 0$ that is contained in the true shortest

path from v_1 to v_n, and the following inequality is satisfied for such an edge (p, q):

$$cost[t_n][q] > cost[t_n][p] + \hat{d}_q - \hat{d}_p = cost[t_n][p] + (p, q).\text{cost}$$

However, there is a contradiction to Lemma 5.2.

The inequality $\hat{d}_n \leq cost[t_n][n]$ is an immediate consequence of the optimality of $\hat{d}_n$, resulting in $cost[t_n][n] = \hat{d}_n$. □

5.3 Experiments

5.3.1 Dataset

In this study, the existing and proposed methods are evaluated with two types of datasets.

Real data: graphs generated from fragment decomposition results

Assuming fragment data reuse in a docking calculation based on fragment extension with tree search such as FlexX, we generated realistic request sequences of the fragment docking results with compounds in the ZINC database [48]. The protocol of generation is shown below:

1. $\{10000, 100000, 1000000\}$ compounds were randomly obtained from the ZINC Purchasable subset (version 2014-11-28; 22,724,825 compounds).

2. Fragments were generated by decomposing sampled compounds at rotatable bonds. The decomposition method described in Chapter 2 was utilized.

3. Assuming that the tree search is reused, the compounds were sorted based on their partial structures, followed by the partial deletion of the fragment sequence identical to the compound immediately previous to that one (Fig. 4.2).

4. Fragment sequences of compounds are concatenated to form a request sequence.

Random sampling was performed three times for each of the number of compounds. Since the number of fragments varies depending on the compound, the lengths of request sequences are different. Thus, when comparing the calculation results, the lengths of request sequences must be specified.

Artificial data: randomly generated graphs

The weighted offline cache problem has three elements: the number of requests, the weight of pages, and the number of types of pages. However, as the real data did not include a sufficient variety of patterns, further assessment of the performance of the methods required data to be randomly generated. Data were randomly generated by changing the parameters as follows:

- The number of page requests are $\{10000, 100000, 1000000\}$.
- The maximum weight of pages is 100, where each weight is set uniformly by assigning a random integer value.
- The number of different kinds of pages are $\{1000, 10000, \ldots, \frac{\#\text{requests}}{10}\}$.

5.3.2 Related methods

This section introduces existing exact algorithms for the minimum cost flow problem. At first we compare general-purpose existing algorithms implemented in LEMON [103, 104], and the best algorithm is used as a baseline method in the comparison made in Section 5.4.

LEMON ([103, 104])

LEMON (Library for Efficient Modeling and Optimization in Networks) provides for the implementation of data structures and algorithms related to graphs and network structures written in C++. With regard to the minimum cost flow problem, there are implementations of Cost-Scaling, Capacity-Scaling, Network-Simplex, and Cycle-Canceling as exact algorithms. Kovacs showed that the Network-Simplex and Cost-Scaling algorithms implemented in LEMON were the most efficient and robust in general among several algorithms and several implementations [105]. Interestingly, a comparison of these implementations using our real data (Table 5.1) revealed that Cost-Scaling was the fastest of the algorithms implemented in LEMON. Therefore, we chose Cost-Scaling as the baseline method used in Section 5.4.

An algorithm proposed by Pirsiavash *et al.* [102]

Pirsiavash *et al.* proposed an algorithm specifically for directed acyclic graphs. This algorithm has a shorter calculation time by sacrificing generality. Their algorithm is

Table 5.1: Calculation time of four existing methods implemented in LEMON for real data. The total amount of flow is $k = 100$. Each experiment was conducted within the 3-hour time limit. The fastest times are written in bold.

#compounds	#pages (n)	Cost-Scaling	Capacity-Scaling
	272,851	**25,263** ms	67,219 ms
10,000	273,095	**26,853** ms	67,251 ms
	274,127	**26,650** ms	65,219 ms
	2,327,638	**455,921** ms	1,741,264 ms
100,000	2,332,426	**452,442** ms	1,727,584 ms
	2,343,338	**445,881** ms	1,991,428 ms
	18,663,564	> 3 h	> 3 h
1,000,000	18,663,905	> 3 h	> 3 h
	18,698,105	> 3 h	> 3 h
#compounds	#pages (n)	Network-Simplex	Cycle-Canceling
	272,851	404,497 ms	3,047,511 ms
10,000	273,095	402,680 ms	3,118,584 ms
	274,127	404,778 ms	2,823,840 ms
	2,327,638	> 3 h	> 3 h
100,000	2,332,426	> 3 h	> 3 h
	2,343,338	> 3 h	> 3 h
	18,663,564	> 3 h	> 3 h
1,000,000	18,663,905	> 3 h	> 3 h
	18,698,105	> 3 h	> 3 h

also based on the SSP framework, which involves finding the shortest path in each step with the Dijkstra method.

5.3.3 Computer environment

A thin node consisting of a TSUBAME 2.5 was used for the experiments at the Tokyo Institute of Technology, Japan. Each thin node has two sockets of Intel Xeon X5670 (6 cores per socket), and 54 GB main memory. Because none of the algorithms are parallelized, all time measurements were carried out with the use of a single CPU core. All results are the median values of three measurements.

5.4 Results

The results for the total amount of flow $k = 10, 100, 1000$ are shown in Tables 5.2, 5.3, and 5.4 for real data, and Tables 5.5, 5.6, and 5.7 for artificial data, respectively.

Note that each fragment docking result consumes up to a few hundred megabytes, and thus $k = 1000$ means main memory of hundreds of gigabytes. Each experiment was conducted within the 3-hour time limit for real data and memory consumption did not exceed a gigabyte for all the experiments.

5.4.1 Comparison of the proposed method and existing methods with real data

The results of the comparison of the proposed method, Pirsiavash's method, and the Cost-Scaling method for real data are provided in Tables 5.2–5.4. The results show that our proposed method is 7.2–9.2 times faster than Pirsiavash's method, and more than 200 times faster than the Cost-Scaling method for the largest amount of data. It should be noted that the acceleration rate is higher when the total amount of flow k is larger.

5.4.2 Comparison of methods using artificial data

We assessed the superiority of our proposed method with a wide spectrum of problem characteristics. The calculation times required for artificial data generated with several parameters are presented in Tables 5.5–5.7. The results show that our proposed method is also faster than existing methods. Interestingly, the acceleration rate is lower when the total amount of flow k is larger, unless for the real data.

5.5 Discussion

5.5.1 Speed-up ratio of the proposed method: difference between real data and artificial data

According to the Tables 5.2–5.7, the speed-up ratio with real data differs from that obtained with artificial data. As we already mentioned, the effects of the total amount of flow k is totally different. The cause of this difference is considered to be the different frequencies of occurrence of the same fragment requests. In reality, the frequency of fragments is not uniform, namely, the frequencies of a few popular fragments are very dominant, whereas the frequencies of most fragments are very low. Storing popular fragments over a long period greatly reduces the cost, thus the edges of popular fragments are likely to be used. Since the edges of frequent fragments are shorter than

Table 5.2: Calculation time for real data. The total amount of flow is $k = 10$. Each experiment was conducted within the 3-hour time limit.

#compounds	#pages (n)	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
	272,851	74 ms	594 ms	27,524 ms
10,000	273,095	74 ms	600 ms	27,640 ms
	274,127	75 ms	599 ms	27,415 ms
	2,327,638	753 ms	5,557 ms	1,978,609 ms
100,000	2,332,426	747 ms	5,574 ms	2,166,534 ms
	2,343,338	751 ms	5,570 ms	2,006,954 ms
	18,663,564	6,256 ms	45,412 ms	$>$ 3 h
1,000,000	18,663,905	6,250 ms	45,185 ms	$>$ 3 h
	18,698,105	6,221 ms	45,320 ms	$>$ 3 h

Table 5.3: Calculation time for real data. The total amount of flow is $k = 100$. Each experiment was conducted within the 3-hour time limit.

#compounds	#pages (n)	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
	272,851	723 ms	5,761 ms	25,263 ms
10,000	273,095	681 ms	5,731 ms	26,853 ms
	274,127	688 ms	5,755 ms	26,650 ms
	2,327,638	6,702 ms	52,281 ms	455,921 ms
100,000	2,332,426	6,670 ms	51,314 ms	452,442 ms
	2,343,338	6,732 ms	52,685 ms	445,881 ms
	18,663,564	55,354 ms	432,360 ms	$>$ 3 h
1,000,000	18,663,905	55,151 ms	427,696 ms	$>$ 3 h
	18,698,105	62,283 ms	466,169 ms	$>$ 3 h

Table 5.4: Calculation time for real data. The total amount of flow is $k = 1{,}000$. Each experiment was conducted within the 3-hour time limit.

#compounds	#pages (n)	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
	272,851	6,413 ms	55,073 ms	102,048 ms
10,000	273,095	6,372 ms	55,459 ms	112,462 ms
	274,127	6,386 ms	55,020 ms	108,839 ms
	2,327,638	62,084 ms	519,230 ms	$>$ 3 h
100,000	2,332,426	62,035 ms	521,809 ms	$>$ 3 h
	2,343,338	63,105 ms	524,776 ms	$>$ 3 h
	18,663,564	500,488 ms	4,607,737 ms	$>$ 3 h
1,000,000	18,663,905	500,011 ms	4,357,040 ms	$>$ 3 h
	18,698,105	497,837 ms	4,322,122 ms	$>$ 3 h

Table 5.5: Calculation time for artificial data. The total amount of flow is $k = 10$.

#pages (n)	#kinds of pages	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
10,000	1,000	2 ms	19 ms	221 ms
100,000	1,000	22 ms	231 ms	8,068 ms
	10,000	22 ms	270 ms	4,594 ms
1,000,000	1,000	253 ms	2,552 ms	1,055,883 ms
	10,000	272 ms	3,197 ms	349,771 ms
	100,000	339 ms	3,664 ms	123,676 ms

Table 5.6: Calculation time for artificial data. The total amount of flow is $k = 100$.

#pages (n)	#kinds of pages	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
10,000	1,000	60 ms	178 ms	361 ms
100,000	1,000	632 ms	2,467 ms	6,576 ms
	10,000	462 ms	2,804 ms	9,195 ms
1,000,000	1,000	6,521 ms	27,645 ms	293,918 ms
	10,000	4,913 ms	34,767 ms	167,089 ms
	100,000	5,640 ms	37,763 ms	178,613 ms

Table 5.7: Calculation time for artificial data. The total amount of flow is $k = 1{,}000$. Each experiment was conducted within the 3-hour time limit.

#pages (n)	#kinds of pages	Proposed	Pirsiavash *et al.* [102]	Cost-Scaling
10,000	1,000	532 ms	1,327 ms	1,915 ms
100,000	1,000	6,902 ms	20,024 ms	25,199 ms
	10,000	8,561 ms	23,563 ms	23,814 ms
1,000,000	1,000	71,658 ms	239,408 ms	$>$ 3 h
	10,000	91,733 ms	300,225 ms	$>$ 3 h
	100,000	74,356 ms	342,660 ms	$>$ 3 h

those of rare fragments, the algorithm is unlikely to select edges that return to a very distant node. On the other hand, the frequency of fragments is assumed to be uniform in artificial data hence the algorithm is more likely to return to previous nodes. According to Algorithm 5.2, the total number of return movements affects the loop count at lines 5–15. Therefore, different frequencies influence the calculation time.

5.5.2 The effectiveness for fragment grid generation cost

The generation time of fragment grids will be decreased approximately 1 hour with optimization for 100,000 compounds with $k = 1{,}000$, we already mentioned in section 4.4. On the other hand, the optimization costs of the condition is 60 seconds for

Table 5.8: Worst computational complexity of related methods. n, C, and U denote the number of nodes, the max cost of edges, and the maximum supply value, respectively. Since the number of edges of the reduced graph do not exceed a constant multiple of the number of nodes, the computational complexities are written under the assumption of $O(n) = O(|V|) = O(|E|)$.

Algorithm name	Acceptable graph	Worst complexity
Cost-Scaling	any graph	$O(n^3 \log(nC))$
Capacity-Scaling	any graph	$O(n^2 \log U \cdot \log n)$
Network-Simplex	any graph	$O(n^3 CU)$
Cycle-Canceling	any graph	$O(n^3 CU)$
Pirsiavash *et al.* [102]	directed acyclic graph	$O(Un \log n)$
Proposed	topologically sortable, sparse graph	*Unsolved*, $O(Un \log n)$?

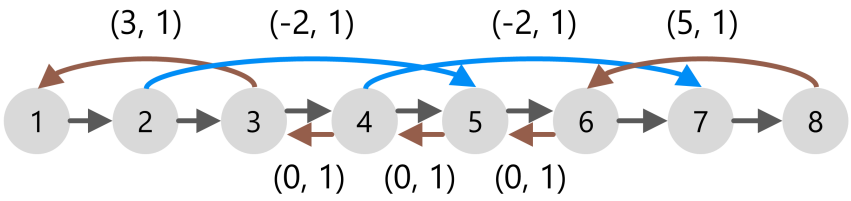
proposed algorithm, thus the optimization cost is small enough. Pirsiavash's algorithm spent 520 seconds in the same condition; however, the generation of fragment grids can be easily parallelized, while the optimization is difficult to be.

5.5.3 Estimation of calculation complexity

The calculation time is considered to depend on the total number of return movements regarding the CalcCosts function. The total number of return movements must depend on the total number of flows k and the number of page requests n. We have not yet succeeded in theoretically deriving the worst computational complexity of the proposed method because our previous attempts to determine the worst case for solving the minimum cost flow problem were unsuccessful. The worst computational complexity of the proposed and existing methods are listed in Table 5.8. According to the results in Tables 5.2–5.4, the speed-up ratios of the proposed method are up to 9 times faster for real data regardless of the amount of data; thus, the order of the complexity of the proposed method seems to be similar to that of Pirsiavash.

5.5.4 Avoiding unnecessary update calculations

According to line 14 of Algorithm 5.2 and Fig. 5.4, Node $(i+1)$ will be targeted after Node i regardless of whether the value of $cost(i+1)$ was updated since the last time Node $(i+1)$ was targeted. This is sometimes a wasteful calculation. In this regard, a priority queue is useful to store nodes that are updated to avoid wasteful calculation (updating the smaller indexed nodes often affects the *cost* values of nodes with larger



1	2	3	4	5	6	7	8	updated
0	0	0	0	0	0	0	0	[ ]
0	0	0	0	0	0	0	0	[**2**]
0	0	0	0	**-2**	0	0	0	[**3**,**5**]
0	0	0	0	-2	0	0	0	[**4**,5]
0	0	0	0	-2	0	**-2**	0	[5,**7**]
0	0	0	**-2**	-2	**-2**	-2	0	[**4**,**6**,7]
0	0	**-2**	-2	-2	-2	**-4**	0	[**3**,6,7]
0	0	-2	-2	-2	-2	-4	0	[6,7]
0	0	-2	-2	-2	-2	-4	0	[7]
0	0	-2	-2	-2	-2	-4	**-4**	[**8**]

Figure 5.5: An example of execution with a priority queue ("updated"). The $(cost, capacity)$ labels are shown for valid edges. Edges without labels have $(0, \infty)$. The cell at row i and column j of the table shows the $cost(j)$ at step i. The cells updated in each step are highlighted.

indices). The example of execution with a priority queue is shown in Fig. 5.5. However, the priority queue needs $O(\log n)$ calculation time for enqueue and dequeue operations, resulting in an increase in the total amount of calculation, thus we did not employ the technique.

5.6 Conclusion

In this Chapter, two things were mentioned; (1) we first revealed the optimization problem can be reduced to the minimum cost flow problem, and (2) we proposed a faster, exact algorithm for the problem. Our experiments showed that the proposed algorithm is up to nine times faster than existing algorithms.

Part V

Concluding Remarks

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we document two procedures based on compound decomposition, Spresso and a protein-ligand docking method. Spresso reuses the results of fragments, resulting in up to 300-fold acceleration with a docking tool, Glide, in our experiment cases. We also proposed protein-ligand docking procedure, and showed the optimization of fragment reuse is possible within polynomial computation complexity. Below, we describe the contributions provided by this work.

6.1.1 Contributions

- In Chapter 2, we proposed a decomposition method, that generates fragments having no rotatable bond. The number of kinds of fragments of ZINC library was 263,319, which is less than one-hundredth of the number of compounds. Furthermore, we assessed the relationship between the ratio and the number of compounds and revealed that the ratio is smaller when the number of compounds is larger. We also checked what kind of fragments are emerged fruently and rarely.

- In Chapter 3, we proposed ultrafast pre-screening method named Spresso. A compound is evaluated by the docking scores of fragments. The calculation speed is up to 300-fold faster than Glide HTVS. We also showed some virtual screening scenarios and revealed Spresso is a only way to evaluate huge compound library.

- In Chapter 4, we described fragment extension-based docking procedure with reuse and sharing of two types of intermediate results. The growing orders or compound extension orders are sorted to maximize the reuse of intermediate

results of growing. We also found the fragment grid is too memory consuming, resulting in the lack of memory space.

- In Chapter 5, we focused on the memory consumption problem mentioned in Chapter 4. Firstly the problem was formulated as the weighted offline cache problem, which can be reduced to the minimum cost flow problem. Secondly, we pointed out the characteristics of the reduced graph of the minimum cost flow problem, and we proposed novel algorithm specified to the graph. Our method achieved up to 9 times faster than existing algorithms for the real data.

6.2 Future Works

6.2.1 Importance assignment to each fragment

Our methods, Spresso and docking procedure are aimed to accelerate the calculation by reuse of fragments. On the other hand, ignoring unimportant fragments may make the methods faster with subtle difference in evaluation. According to the Fig. 2.5, there are tiny fragments (e.g. > 3 atoms), which hardly make huge difference in terms of Spresso's results. Additionally, the score functions of docking highly pay attention to specific interactions such as hydrogen bonding and aromatic interaction, thus it may possible to assign importance before docking. The easiest criteria are (1) The number of atoms, (2) the type of atoms. The importance is also useful for docking procedure, to improve anchor fragment selection, or re-ordering of fragment extension.

6.2.2 Improvement of pre-screening accuracy

The pre-screening method Spresso has great advantage about calculation cost, while its accuracy is compromised. Thus, improvement of accuracy is highly demanded.

Compound score fitting with more complicated machine learning methods

We already tried to evaluate compounds by linear regression model in Section 3.4.2. It is possible to improve accuracy with more complicated model because the vast number of training data can be generated by docking lots of compounds. Graph convolutional neural network is one of ways, by considering compound as a tree structure of fragments. It can consider the connectivity between fragments.

Consideration of collision and connectivity of fragments in Spresso

As we mentioned in Section 3.4.4, Spresso will generate unreasonable conformation since the method does not consider collisions nor connectivities between fragments. It may be a reason of its accuracy. To improve the accuracy of Spresso, a possible way is generating several fragment conformations, followed by selection of best combination. It must be mentioned that the combination optimization is time-consuming, thus the balance of speediness and accuracy have to be considered.

6.2.3 Development of protein-ligand docking tool

We described docking calculation procedure, and we proposed the optimization method for memory usage problem. The implementation of docking tool, however, is not covered by this thesis. To proof its superiority, it is needed to develop the docking tool.

6.2.4 Simultaneous optimization of compound evaluation order and memory usage

In Chapter 5, we assumed that evaluation order of compounds are decided firstly; however, a growing graph in Fig. 4.4 indicates the order can be partially changed without any loss of reuse of intermediate results. The simultaneous optimization of compound evaluation order and memory usage makes the docking procedure accelerated while the optimization may not be solved in polynomial order.

Part VI

Appendix

Appendix A

Previous results of Spresso

In this thesis, we re-experienced most of evaluations of Spresso because of newer version of the Glide and newer supercomputing system TSUBAME 3.0. This appendix shows previous results already shown in our publication [71].

A.1 Computing environment

All calculations were conducted on the TSUBAME 2.5 supercomputing system, Tokyo Institute of Technology, Japan. We used its thin nodes in all experiments, with each node having two Intel Xeon X5670 CPUs (six cores per CPU) and 54 GB of RAM. Glide (Schrödinger suite 2014-4) was used as the docking tool. Because Glide software is a single-thread program, all docking simulations were performed in parallel using 12 CPU cores. It should be noted that Glide is a proprietary software, and thus it cannot be optimized for specific computing environments.

A.2 Experiments

A.2.1 Calculation speed

Table A.1 shows the calculation speed with Glide version 2014-4 for three targets, ACES, EGFR, and PGH1.

A.2.2 Prediction accuracy

Table A.2 shows the pre-screening accuracy with Glide version 2014-4.

Table A.1: The results of docking times for docking of all 28,629,602 ZINC compounds into three DUD-E protein targets. Values in parentheses indicate the fold increase in speed exhibited by Spresso relative to Glide HTVS.

Target	Calculation time [CPU core hours]		
	Spresso-SP	Spresso-HTVS	Glide HTVS
ACES	42.6 (× 76.8)	22.8 (× 143.1)	3268.8
EGFR	38.9 (× 126.4)	21.5 (× 229.3)	4925.1
PGH1	41.8 (× 88.0)	20.9 (× 175.4)	3674.5

Table A.2: The results of averaged prediction accuracy for 102 DUD-E targets. Note that all enrichment factors represent the average of 102 EFs from DUD-E protein targets. *a*%–*b*% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds. Best EF values among Spressos are written in bold.

Methods		Enrichment factors				
		2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
Spresso-SP	SUM	4.58	6.78	8.92	4.00	5.53
	MAX	9.28	11.01	11.94	7.51	8.31
	GS3	**9.73**	**12.79**	**15.03**	**8.01**	**9.94**
Spresso-HTVS	SUM	4.60	6.78	8.93	4.20	5.46
	MAX	9.29	9.93	12.41	6.38	8.29
	GS3	9.00	12.18	14.49	7.39	9.24
Glide HTVS		17.85	18.97	19.60	12.50	12.92

Appendix B

Detailed information of DUD-E

DUD-E (Directory of Useful Decoys, Enhanced) is a benchmarking set for protein-ligand docking proposed by Mysinger *et al.* [86]. They selected 102 proteins which are widely targeted of many drug design campaigns with consideration of target diversity. Active ligand sets were also obtained from ChEMBL, and decoy compound sets were generated.

1. Active ligands

   (a) Obtain experimented compounds for each target protein from ChEMBL[85]

   (b) Select compounds which $IC_{50} \leq 1\mu M$ as known actives

   (c) Make clusters based on Bemis-Murcko scaffold [106] and select IC_{50} best N actives from each cluster in order to generate active sets which contain 100 actives

2. Decoy compounds
   Decoy compounds consist of two types:

   - Experimentally inactive compounds (confirmed as inactive, $IC_{50} \geq 30\mu$M)
   - Expected inactive compounds generated with each active ligand (not experimented, but considered as inactive)

   Expected inactive compounds were generated with below procedure.

   (a) For each active ligand, calculate physico-chemical properties such as MW, logP, followed by obtaining several thousand compounds from ZINC [48], which has similar properties

(b) Filter out compounds which have high fingeprint similarity to active ligands

(c) Select 50 compounds randomly without duplication

3. Representative 3D structure of the target

(a) Obtain all 3D structures labeled as *H. sapiens* for each target from PDB [27]

(b) Dock active ligands and inactive compounds using DOCK [41]

(c) Choose higher resolution, higher accuracy 3D structure as the representative

The detailed information of each target including the average of fragments are shown in Tables B.1–B.4.

B.1 Single accuracies of Glide SP, Glide HTVS, and Spresso (scenarios 1–3)

Enrichment factors for each target are shown in Tables B.5–B.8. It is noted that the average EF values of all DUD-E targets are shown in Figs. 3.7–3.9.

B.2 Pre-screening accuracies of Glide HTVS and Spresso (scenarios 4, 5)

Enrichment factors for each target are shown in Tables B.9–B.16. It is noted that the average EF values of all DUD-E targets are shown in Figs. 3.10, 3.11.

B.3 Three-step screening accuracies with Spresso and Glide (scenario 6)

Enrichment factors for each target are shown in Tables B.17–B.20. It is noted that the average EF values of all DUD-E targets are shown in Fig. 3.12.

Table B.1: Detailed information of DUD-E targets (1)

Target	PDBID	Description	# compounds		Ave. # fragments	
			Pos.	Neg.	Pos.	Neg.
aa2ar	3EML	Adenosine A2a receptor	482	31,498	6.26	7.04
abl1	2HZI	Tyrosine-protein kinase ABL	182	10,746	7.08	7.33
ace	3BKL	Angiotensin-converting enzyme	282	16,860	9.90	7.52
aces	1E66	Acetylcholinesterase	453	26,234	9.48	8.07
ada	2E1W	Adenosine deaminase	93	5,449	8.16	8.01
ada17	2OI0	ADAM17	532	35,810	8.81	8.32
adrb1	2VT4	Beta-1 adrenergic receptor	247	15,842	10.05	9.62
adrb2	3NY8	Beta-2 adrenergic receptor	231	14,993	10.35	9.72
akt1	3CQW	Serine / threonine-protein kinase AKT	293	16,426	6.94	8.00
akt2	3D0E	Serine / threonine-protein kinase AKT2	117	6,893	6.61	7.39
aldr	2HV5	Aldose reductase	159	8,995	4.81	5.34
ampc	1L2S	Beta-lactamase	48	2,847	4.94	4.95
andr	2AM9	Androgen Receptor	269	14,343	3.91	4.67
aofb	1S3B	Monoamine oxidase B	122	6,900	4.37	4.39
bace1	3L5D	Beta-secretase 1	283	18,080	10.72	8.79
braf	3D4Q	Serine / threonine-protein kinase B-raf	152	9,944	7.27	7.24
cah2	1BCD	Carbonic anhydrase II	492	31,134	7.14	7.10
casp3	2CNK	Caspase-3	199	10,692	10.47	8.72
cdk2	1H00	Cyclin-dependent kinase 2	474	27,831	6.25	6.70
comt	3BWM	Catechol O-methyltransferase	41	3,848	4.85	5.15
cp2c9	1R9O	Cytochrome P450 2C9	120	7,446	7.18	6.78
cp3a4	3NXU	Cytochrome P450 3A4	170	11,796	7.71	7.42
csf1r	3KRJ	Macrophage colony stimulating factor receptor	166	12,144	6.86	6.92
cxcr4	3ODU	C-X-C chemokine receptor type 4	40	3,406	6.75	7.30
def	1LRU	Peptide deformylase	102	5,696	9.78	7.77
dhi1	3FRJ	11-beta-hydroxysteroid dehydrogenase 1	330	19,341	5.89	5.54
dpp4	2I78	Dipeptidyl peptidase IV	533	40,916	6.52	6.44

Table B.2: Detailed information of DUD-E targets (2)

Target	PDBID	Description	# compounds		Ave. # fragments	
			Pos.	Neg.	Pos.	Neg.
drd3	3PBL	Dopamine D3 receptor	480	34,022	7.58	7.30
dyr	3NXO	Dihydrofolate reductase	231	17,170	6.68	7.10
egfr	2RGP	Epidermal growth factor receptor erbB1	542	35,021	7.27	7.74
esr1	1SJ0	Estrogen receptor alpha	383	20,663	5.55	6.72
esr2	2FSZ	Estrogen receptor beta	367	20,182	5.21	6.56
fa10	3KL6	Coagulation factor X	537	28,247	9.01	8.18
fa7	1W7X	Coagulation factor VII	114	6,245	10.29	8.26
fabp4	2NNQ	Fatty acid binding protein adipocyte	47	2,749	6.68	6.53
fak1	3BZ3	Focal adhesion kinase 1	100	5,350	8.44	8.12
fgfr1	3C4F	Fibroblast growth factor receptor 1	139	8,691	7.33	7.55
fkb1a	1J4H	FK506-binding protein 1A	111	5,800	9.75	8.42
fnta	3E37	Protein farnesyltransferase / geranylgeranyltransferase type I alpha subunit	592	51,430	8.20	7.65
fpps	1ZW5	Farnesyl diphosphate synthase	85	8,823	7.08	7.01
gcr	3BQD	Glucocorticoid receptor	258	14,987	5.33	5.94
glcm	2V3F	Beta-glucocerebrosidase	54	3,799	8.57	7.93
gria2	3KGC	Glutamate receptor ionotropic, AMPA 2	158	11,832	6.47	6.52
grik1	1VSO	Glutamate receptor ionotropic kainate 1	101	6,547	5.83	6.32
hdac2	3MAX	Histone deacetylase 2	185	10,300	10.02	8.29
hdac8	3F07	Histone deacetylase 8	170	10,448	9.52	7.95
hivint	3NF7	Human immunodeficiency virus type 1 integrase	100	6,645	6.42	6.35
hivpr	1XL2	Human immunodeficiency virus type 1 protease	536	35,688	11.16	8.75
hivrt	3LAN	Human immunodeficiency virus type 1 reverse transcriptase	338	18,879	4.98	5.65
hmdh	3CCW	HMG-CoA reductase	170	8,743	9.79	8.56

Table B.3: Detailed information of DUD-E targets (3)

Target	PDBID	Description	# compounds		Ave. # fragments	
			Pos.	Neg.	Pos.	Neg.
hs90a	1UYG	Heat shock protein HSP 90-alpha	88	4,848	5.56	7.20
hxk4	3F9M	Hexokinase type IV	92	4,698	6.78	6.97
igf1r	2OJ9	Insulin-like growth factor I receptor	148	9,291	7.95	8.27
inha	2H7L	Enoyl-[acyl-carrier-protein] reductase	43	2,300	7.14	6.17
ital	2ICA	Leukocyte adhesion glycoprotein LFA-1 alpha	138	8,487	7.72	7.62
jak2	3LPB	Tyrosine-protein kinase JAK2	107	6,495	6.23	6.63
kif11	3CJO	Kinesin-like protein 1	116	6,849	6.78	6.18
kit	3G0E	Stem cell growth factor receptor	166	10,447	7.80	7.57
kith	2B8T	Thymidine kinase	57	2,850	6.70	7.60
kpcb	2I0E	Protein kinase C beta	135	8,692	5.67	6.85
lck	2OF2	Tyrosine-protein kinase LCK	420	27,375	7.20	7.32
lkha4	3CHP	Leukotriene A4 hydrolase	171	9,449	7.70	7.59
mapk2	3M2W	MAP kinase-activated protein kinase 2	101	6,147	4.67	5.58
mcr	2AA2	Mineralocorticoid receptor	94	5,146	5.43	5.91
met	3LQ8	Hepatocyte growth factor receptor	166	11,240	7.45	7.48
mk01	2OJG	MAP kinase ERK2	79	4,548	7.25	6.84
mk10	2ZDT	c-Jun N-terminal kinase 3	104	6,599	6.63	6.71
mk14	2QD9	MAP kinase p38 alpha	578	35,812	7.31	7.03
mmp13	830C	Matrix metalloproteinase 13	572	37,127	8.98	8.29
mp2k1	3EQH	Dual specificity mitogen-activated protein kinase kinase 1	121	8,147	6.92	8.02
nos1	1QW6	Nitric-oxide synthase, brain	100	8,050	4.86	5.50
nram	1B9V	Neuraminidase	98	6,199	8.39	7.01
pa2ga	1KVO	Phospholipase A2 group IIA	99	5,147	10.12	9.63
parp1	3L3M	Poly [ADP-ribose] polymerase-1	508	30,035	5.07	5.57
pde5a	1UDT	Phosphodiesterase 5A	398	27,521	6.95	7.63
pgh1	2OYU	Cyclooxygenase-1	195	10,797	4.66	4.72
pgh2	3LN1	Cyclooxygenase-2	435	23,136	5.14	5.28

Table B.4: Detailed information of DUD-E targets (4)

Target	PDBID	Description	# compounds		Ave. # fragments	
			Pos.	Neg.	Pos.	Neg.
plk1	2OWB	Serine / threonine-protein kinase PLK1	107	6,797	7.16	7.95
pnph	3BGS	Purine nucleoside phosphorylase	103	6,950	3.93	6.08
ppara	2P54	Peroxisome proliferator-activated receptor alpha	373	19,358	9.75	8.98
ppard	2ZNP	Peroxisome proliferator-activated receptor delta	240	12,224	8.95	8.69
pparg	2GTK	Peroxisome proliferator-activated receptor gamma	484	25,256	9.19	8.61
prgr	3KBA	Progesterone receptor	293	15,643	3.95	4.65
ptn1	2AZR	Protein-tyrosine phosphatase 1B	130	7,243	8.98	7.62
pur2	1NJS	GAR transformylase	50	2,694	12.18	8.54
pygm	1C8K	Muscle glycogen phosphorylase	77	3,940	6.74	6.19
pyrd	1D3G	Dihydroorotate dehydrogenase	111	6,446	5.66	5.64
reni	3G6Z	Renin	104	6,955	14.59	11.68
rock1	2ETR	Rho-associated protein kinase 1	100	6,297	5.54	6.30
rxra	1MV9	Retinoid X receptor alpha	131	6,935	5.95	5.79
sahh	1LI4	Adenosylhomocysteinase	63	3,450	3.46	5.67
src	3EL8	Tyrosine-protein kinase SRC	524	34,455	7.18	7.61
tgfr1	3HMM	TGF-beta receptor type I	133	8,498	5.02	5.66
thb	1Q4X	Thyroid hormone receptor beta-1	103	7,442	6.52	7.19
thrb	1YPE	Thrombin	461	26,948	11.09	8.75
try1	2AYW	Trypsin I	449	25,915	10.42	8.34
tryb1	2ZEC	Tryptase beta-1	148	7,643	11.55	9.19
tysy	1SYN	Thymidylate synthase	109	6,738	8.57	7.14
urok	1SQT	Urokinase-type plasminogen activator	162	9,841	7.29	6.88
vgfr2	2P2I	Vascular endothelial growth factor receptor 2	409	24,928	7.23	7.37
wee1	3BIZ	Serine / threonine-protein kinase WEE1	102	6,149	5.87	7.57
xiap	3HL5	Inhibitor of apoptosis protein 3	100	5,145	11.89	8.87

Table B.5: The results of single prediction accuracies for 102 DUD-E targets (1)

Target	Glide SP		Glide HTVS		Spresso	
	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$
aa2ar	16.79	12.96	10.16	7.05	4.15	3.21
abl1	16.38	9.32	9.28	6.85	6.00	4.11
ace	8.48	7.09	7.42	5.14	3.18	4.61
aces	3.97	4.30	2.43	2.10	29.34	19.08
ada	17.03	12.88	13.83	9.13	3.19	2.15
ada17	33.78	20.11	19.71	13.25	1.31	1.88
adrb1	7.69	7.48	9.71	7.48	2.43	2.02
adrb2	24.12	15.77	15.51	9.94	8.18	6.27
akt1	10.53	6.13	4.08	3.07	3.40	3.41
akt2	9.28	7.65	5.06	4.25	4.22	2.97
aldr	24.41	16.27	15.02	8.76	10.01	7.51
ampc	6.24	7.28	2.08	4.16	0.00	0.00
andr	11.82	9.08	12.56	8.16	15.52	9.64
aofb	8.92	6.53	8.92	6.53	2.43	2.04
bace1	14.11	10.40	5.99	4.41	15.87	12.34
braf	27.62	16.77	26.96	15.13	3.29	3.62
cah2	1.01	0.61	1.82	1.12	4.06	2.94
casp3	15.06	9.04	7.53	6.78	10.04	7.78
cdk2	10.51	8.95	13.04	8.32	2.10	2.42
comt	0.00	1.22	4.86	4.86	4.86	2.43
cp2c9	0.83	0.83	0.83	0.41	1.66	1.66
cp3a4	8.80	6.16	5.28	3.23	4.11	2.64
csf1r	11.96	9.01	10.76	8.41	2.99	1.80
cxcr4	4.92	2.50	0.00	2.50	7.38	4.99
def	32.34	23.03	23.52	15.68	2.94	2.45
dhi1	12.10	7.72	5.14	5.30	0.30	0.15
dpp4	23.98	17.26	19.86	14.63	2.62	2.25

Table B.6: The results of single prediction accuracies for 102 DUD-E targets (2)

Target	Glide SP		Glide HTVS		Spresso	
	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$
drd3	4.78	3.85	4.78	4.26	3.74	2.50
dyr	11.19	9.50	7.32	6.26	21.52	15.11
egfr	20.46	11.80	10.14	6.64	3.32	3.59
esr1	47.14	28.98	30.21	18.27	20.57	14.75
esr2	46.21	31.20	36.42	22.07	13.59	8.99
fa10	38.53	25.03	24.94	16.66	3.16	2.98
fa7	36.61	22.66	34.86	23.53	12.20	9.59
fabp4	19.12	13.81	14.87	13.81	2.12	4.25
fak1	19.82	12.00	14.86	10.50	6.94	7.00
fgfr1	4.28	3.95	0.71	1.79	0.00	1.08
fkb1a	44.38	24.16	23.08	16.56	2.66	1.79
fnta	7.76	6.16	6.24	4.98	8.60	5.91
fpps	1.16	1.17	5.82	4.10	0.00	0.00
gcr	16.61	10.46	11.97	7.36	15.83	11.24
glcm	10.98	9.15	9.15	7.32	12.81	9.15
gria2	30.99	18.34	30.99	17.71	24.03	15.49
grik1	15.72	10.89	28.49	20.29	12.77	7.42
hdac2	10.80	9.72	3.78	2.70	1.08	1.08
hdac8	12.84	11.73	11.67	12.90	0.00	0.00
hivint	4.96	6.00	5.95	4.00	3.97	3.50
hivpr	20.11	13.80	9.31	7.36	12.66	8.39
hivrt	22.39	14.32	13.55	8.86	5.60	3.84
hmdh	37.86	24.60	25.05	16.69	2.33	2.05

Table B.7: The results of single prediction accuracies for 102 DUD-E targets (3)

Target	Glide SP		Glide HTVS		Spresso	
	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$
hs90a	1.12	1.70	4.49	3.40	0.00	0.00
hxk4	15.19	7.59	11.93	7.59	1.08	0.54
igf1r	13.43	9.11	9.40	5.74	10.74	9.79
inha	4.54	2.32	6.81	3.48	0.00	3.48
ital	2.87	2.89	2.16	1.81	1.44	2.17
jak2	25.79	19.48	21.18	14.38	0.92	0.93
kif11	42.03	24.88	22.30	14.58	6.00	5.58
kit	3.59	2.40	4.78	3.00	8.37	5.70
kith	27.20	17.29	15.30	9.51	6.80	7.78
kpcb	49.22	28.07	36.73	20.32	19.84	12.19
lck	16.43	12.50	15.95	10.12	3.33	3.81
lkha4	21.46	19.24	22.04	15.16	0.00	0.00
mapk2	21.60	17.32	24.55	18.81	10.80	6.93
mcr	9.47	4.78	5.26	2.65	10.52	6.90
met	48.39	28.20	31.66	17.40	10.16	10.50
mk01	16.20	13.23	3.74	4.41	1.25	0.63
mk10	8.53	4.77	8.53	6.21	1.90	1.91
mk14	9.34	5.45	7.09	4.76	5.02	3.20
mmp13	18.71	12.50	12.94	7.78	4.37	3.06
mp2k1	4.94	3.29	4.94	4.53	11.53	8.23
nos1	1.99	1.50	2.98	3.50	7.95	7.50
nram	33.66	22.95	22.44	14.28	13.26	11.22
pa2ga	30.99	19.18	17.00	11.10	5.00	3.53
parp1	48.73	35.52	44.01	27.85	22.40	13.97
pde5a	15.78	12.17	11.27	8.28	8.52	6.15
pgh1	11.79	7.94	13.32	7.94	2.05	1.79
pgh2	26.63	16.53	21.35	12.17	0.69	1.38

Table B.8: The results of single prediction accuracies for 102 DUD-E targets (4)

Target	Glide SP		Glide HTVS		Spresso	
	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$	$EF_{1\%}$	$EF_{2\%}$
plk1	30.42	19.50	19.36	14.39	0.92	0.46
pnph	38.58	26.52	49.19	26.52	44.36	28.45
ppara	9.35	8.70	5.61	6.16	3.74	3.21
ppard	3.32	3.95	4.15	3.53	1.25	1.04
pparg	6.60	5.37	5.77	4.75	4.33	3.20
prgr	25.15	17.39	19.04	12.96	13.26	8.52
ptn1	18.39	14.56	19.93	12.65	10.73	8.43
pur2	54.88	47.90	54.88	34.92	29.40	22.95
pygm	2.54	1.29	1.27	3.86	0.00	0.64
pyrd	41.17	23.72	30.43	17.45	3.58	4.92
reni	22.94	14.82	11.47	6.21	2.87	5.26
rock1	18.99	12.49	16.99	11.99	10.00	8.50
rxra	48.62	34.95	41.02	25.83	0.00	0.00
sahh	55.76	40.84	55.76	36.91	1.55	3.93
src	7.44	5.63	7.44	5.15	9.35	6.39
tgfr1	11.93	9.38	20.89	13.50	0.00	0.38
thb	27.95	17.95	16.39	13.10	0.96	1.46
thrb	29.19	18.52	21.84	14.73	21.84	16.79
try1	29.58	18.24	26.24	16.46	11.34	8.67
tryb1	14.85	11.14	15.52	10.12	2.70	3.71
tysy	35.50	24.76	24.58	15.59	29.13	21.09
urok	36.68	20.58	30.57	16.59	6.72	5.84
vgfr2	17.56	10.87	15.36	9.16	15.36	10.14
wee1	61.28	46.21	61.28	43.77	61.28	44.75
xiap	52.45	40.96	52.45	33.47	10.89	7.99

Table B.9: The results of prediction accuracies for 102 DUD-E targets with Glide HTVS and Glide SP (1). a%–b% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds.

Target	Glide HTVS - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
aa2ar	12.03	13.89	16.38	9.33	11.51
abl1	12.55	12.01	12.55	7.13	8.23
ace	7.42	7.42	6.71	5.85	4.96
aces	2.43	3.31	3.75	2.43	2.65
ada	12.77	13.83	15.96	9.66	11.27
ada17	24.21	27.40	28.90	14.94	16.35
adrb1	10.52	8.09	8.09	6.47	6.27
adrb2	18.09	16.37	15.51	11.02	11.02
akt1	4.42	5.09	6.79	2.90	3.75
akt2	5.91	6.75	8.44	5.52	4.67
aldr	15.64	17.52	18.15	9.07	10.33
ampc	4.16	8.32	8.32	8.32	8.32
andr	13.67	13.67	12.19	9.27	9.64
aofb	11.35	8.11	9.73	7.75	7.35
bace1	7.05	8.11	8.82	6.52	5.82
braf	25.65	27.62	26.96	15.78	16.44
cah2	1.82	1.22	1.01	1.12	0.61
casp3	8.54	12.05	13.05	8.54	8.28
cdk2	12.41	13.88	14.09	8.64	9.48
comt	0.00	0.00	0.00	3.65	0.00
cp2c9	0.00	0.83	0.83	0.83	0.83
cp3a4	4.69	7.04	9.39	5.28	6.45
csf1r	11.96	14.35	14.95	8.41	9.31
cxcr4	2.46	2.46	4.92	1.25	2.50
def	21.56	23.52	25.48	17.15	16.66
dhi1	6.35	8.17	10.59	6.05	7.11
dpp4	23.23	24.73	24.73	17.26	17.26

Table B.10: The results of prediction accuracies for 102 DUD-E targets with Glide HTVS and Glide SP (2). a%–b% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds.

Target	Glide HTVS - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
drd3	4.57	4.78	5.19	4.47	4.06
dyr	11.62	10.76	9.47	6.91	6.48
egfr	11.98	13.64	15.30	7.93	8.75
esr1	33.07	33.60	35.68	18.53	19.84
esr2	39.14	39.41	39.96	23.30	23.84
fa10	27.92	31.45	33.31	19.36	20.75
fa7	39.22	39.22	38.35	27.45	25.71
fabp4	17.00	19.12	21.25	14.87	12.75
fak1	17.84	14.86	12.88	11.00	12.00
fgfr1	3.57	2.14	3.57	2.15	2.87
fkb1a	27.51	29.29	31.95	16.11	17.45
fnta	7.25	7.59	8.43	6.25	6.67
fpps	1.16	2.33	2.33	2.93	2.93
gcr	13.52	13.13	13.52	7.56	8.33
glcm	14.64	16.47	16.47	8.23	8.23
gria2	31.62	29.09	29.09	18.97	18.34
grik1	23.58	22.60	17.68	14.85	13.36
hdac2	5.40	6.48	7.02	4.05	5.67
hdac8	18.10	15.76	15.76	15.83	15.83
hivint	6.94	7.94	8.93	6.50	6.50
hivpr	13.40	14.34	17.69	10.53	12.40
hivrt	15.02	15.91	15.91	10.93	11.52
hmdh	29.71	36.70	37.86	21.97	23.43

Table B.11: The results of prediction accuracies for 102 DUD-E targets with Glide HTVS and Glide SP (3). a%–b% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds.

Target	Glide HTVS - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
hs90a	5.61	3.37	0.00	4.53	3.40
hxk4	11.93	13.02	13.02	8.14	6.51
igf1r	7.38	7.38	8.06	6.07	6.07
inha	6.81	6.81	4.54	4.64	4.64
ital	2.87	2.16	2.87	1.81	3.25
jak2	21.18	23.02	22.10	14.38	15.77
kif11	21.44	27.45	32.59	15.87	18.44
kit	4.78	4.18	3.59	2.70	2.70
kith	15.30	15.30	15.30	7.78	7.78
kpcb	40.41	41.88	42.61	22.53	23.27
lck	16.66	17.14	17.62	12.26	12.97
lkha4	24.36	26.68	27.84	17.78	18.36
mapk2	29.46	29.46	26.51	19.30	20.29
mcr	5.26	6.31	6.31	3.19	3.19
met	32.26	38.83	42.42	20.70	22.20
mk01	6.23	7.48	8.72	4.41	4.41
mk10	4.74	4.74	5.69	4.30	5.73
mk14	5.02	5.36	5.53	3.29	3.63
mmp13	13.64	16.61	16.61	10.23	10.93
mp2k1	7.41	5.76	4.12	4.53	4.94
nos1	2.98	2.98	2.98	1.50	1.50
nram	25.50	29.58	31.62	17.34	17.34
pa2ga	18.00	18.00	17.00	10.60	11.10
parp1	47.16	45.78	47.16	30.60	32.18
pde5a	13.28	14.53	16.03	10.04	10.79
pgh1	13.32	11.79	11.27	7.43	7.69
pgh2	21.12	21.35	22.04	12.63	13.09

Table B.12: The results of prediction accuracies for 102 DUD-E targets with Glide HTVS and Glide SP (4). a%–b% indicates the $EF_{b\%}$ when compounds were pre-screened using a% of all compounds.

Target	Glide HTVS - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
plk1	23.97	26.73	29.50	16.25	19.03
pnph	44.36	46.29	46.29	28.45	27.49
ppara	8.28	9.08	8.82	6.96	7.23
ppard	3.32	4.57	4.99	3.32	3.74
pparg	6.18	6.60	7.01	5.06	5.06
prgr	22.10	22.10	24.14	14.83	15.86
ptn1	18.39	17.63	17.63	12.65	13.03
pur2	54.88	54.88	54.88	39.91	42.91
pygm	2.54	2.54	2.54	4.51	2.58
pyrd	32.22	34.01	34.91	19.69	20.59
reni	11.47	18.16	20.08	10.04	11.95
rock1	15.99	11.99	16.99	11.49	11.99
rxra	45.58	47.10	49.38	28.49	29.63
sahh	55.76	55.76	54.21	37.70	37.70
src	7.63	8.39	7.44	6.10	6.58
tgfr1	20.14	14.92	14.17	12.00	12.38
thb	23.13	24.10	24.10	16.49	16.49
thrb	21.62	23.13	25.30	14.73	15.70
try1	24.24	25.80	26.91	14.68	15.23
tryb1	18.90	18.22	15.52	10.46	9.79
tysy	28.22	30.04	29.13	18.34	16.97
urok	28.12	28.73	30.57	15.97	16.90
vgfr2	14.39	15.85	15.61	9.16	8.92
wee1	61.28	61.28	61.28	44.26	44.75
xiap	52.45	52.45	52.45	34.47	34.97

Table B.13: The results of prediction accuracies for 102 DUD-E targets with Spresso and Glide SP (1). a%–b% means the $EF_{b\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide SP docking calculation for a% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
aa2ar	6.43	9.12	10.99	4.87	6.22
abl1	7.64	7.10	8.19	7.40	6.31
ace	5.65	6.36	8.13	4.79	4.79
aces	15.89	11.69	10.59	7.94	7.28
ada	4.26	6.38	12.77	3.76	8.59
ada17	3.57	11.26	19.89	6.01	10.90
adrb1	3.64	5.66	6.88	3.03	5.26
adrb2	11.63	13.78	20.68	8.21	12.32
akt1	5.43	6.79	6.45	4.94	3.75
akt2	5.91	5.91	7.59	4.67	5.95
aldr	12.52	16.90	23.78	10.01	13.77
ampc	0.00	6.24	10.40	3.12	6.24
andr	11.82	14.41	17.74	7.23	9.08
aofb	1.62	5.67	7.29	3.67	5.31
bace1	22.22	26.80	25.04	16.93	15.52
braf	5.92	6.58	10.52	5.92	6.91
cah2	0.00	0.00	0.20	0.00	0.10
casp3	14.56	19.58	21.59	11.55	14.56
cdk2	4.00	5.05	7.57	3.37	4.84
comt	2.43	2.43	2.43	1.22	1.22
cp2c9	0.00	0.00	2.49	0.00	1.66
cp3a4	5.28	7.63	6.45	5.57	4.99
csf1r	2.39	4.19	5.38	2.40	3.00
cxcr4	9.85	4.92	0.00	4.99	1.25
def	4.90	7.84	8.82	4.90	5.39
dhi1	0.30	0.30	1.51	0.45	0.91
dpp4	4.50	10.87	14.24	5.82	8.72

Table B.14: The results of prediction accuracies for 102 DUD-E targets with Spresso and Glide SP (2). a%–b% means the $EF_{b\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide SP docking calculation for a% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
drd3	4.15	6.65	7.06	4.16	4.89
dyr	24.11	19.37	12.91	14.68	12.95
egfr	6.27	10.69	13.82	6.08	7.93
esr1	23.18	30.73	35.94	17.23	19.32
esr2	16.31	22.56	29.90	12.26	17.03
fa10	5.58	8.75	10.79	5.21	6.79
fa7	19.17	23.53	29.63	16.12	17.87
fabp4	8.50	12.75	21.25	6.37	11.69
fak1	8.92	7.93	5.95	7.00	7.50
fgfr1	0.71	0.71	2.86	0.36	1.79
fkb1a	3.55	4.44	5.33	2.68	4.03
fnta	10.96	14.50	14.00	8.78	9.88
fpps	0.00	0.00	0.00	0.00	0.00
gcr	8.88	10.04	11.20	5.62	6.97
glcm	14.64	16.47	14.64	10.98	10.06
gria2	20.87	22.77	30.35	12.65	15.49
grik1	8.84	12.77	20.63	8.41	15.84
hdac2	1.62	4.86	7.56	2.43	3.78
hdac8	0.00	2.33	3.50	1.17	2.05
hivint	5.95	5.95	9.92	5.00	6.50
hivpr	15.45	21.97	23.46	12.77	15.57
hivrt	6.48	12.08	14.43	7.68	9.75
hmdh	2.33	8.74	16.31	4.98	8.49

Table B.15: The results of prediction accuracies for 102 DUD-E targets with Spresso and Glide SP (3). a%–b% means the $EF_{b\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide SP docking calculation for a% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
hs90a	0.00	0.00	0.00	0.00	0.00
hxk4	1.08	0.00	2.17	1.08	3.80
igf1r	16.11	16.11	14.10	11.47	11.47
inha	4.54	0.00	0.00	2.32	0.00
ital	2.16	2.16	1.44	1.08	1.81
jak2	1.84	6.45	13.81	4.64	7.42
kif11	11.15	12.01	16.30	6.00	9.01
kit	4.78	3.59	1.20	4.50	1.80
kith	6.80	8.50	20.40	6.05	13.83
kpcb	20.57	27.92	32.33	15.52	17.73
lck	6.90	8.81	10.47	5.83	6.67
lkha4	0.00	0.00	1.74	0.00	0.87
mapk2	7.86	15.71	20.62	8.91	10.89
mcr	7.36	6.31	7.36	3.19	3.72
met	19.72	22.70	25.09	12.60	14.10
mk01	0.00	0.00	3.74	0.00	2.52
mk10	2.84	1.90	2.84	2.39	1.91
mk14	2.77	1.90	2.77	1.99	2.08
mmp13	5.94	9.61	12.24	5.33	7.17
mp2k1	8.23	4.12	3.29	5.35	2.88
nos1	9.94	4.97	3.98	5.50	4.50
nram	17.34	31.62	37.74	17.85	21.93
pa2ga	6.00	6.00	11.00	5.05	7.07
parp1	24.95	33.21	40.87	17.61	23.12
pde5a	10.27	8.02	8.27	6.15	6.53
pgh1	3.07	1.54	1.54	0.77	1.28
pgh2	1.38	1.84	2.53	0.92	1.38

Table B.16: The results of prediction accuracies for 102 DUD-E targets with Spresso and Glide SP (4). a%–b% means the $EF_{b\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide SP docking calculation for a% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide SP				
	2%–1%	5%–1%	10%–1%	5%–2%	10%–2%
plk1	0.00	0.92	4.61	0.46	2.32
pnph	50.15	52.08	49.19	30.86	30.86
ppara	6.14	8.82	10.42	4.82	6.16
ppard	1.66	2.91	2.91	2.49	3.74
pparg	5.98	10.10	9.48	6.09	7.74
prgr	9.52	11.22	13.60	5.80	7.33
ptn1	12.26	17.63	22.23	10.73	12.26
pur2	45.08	50.96	52.92	25.94	26.94
pygm	0.00	5.09	6.36	2.58	5.15
pyrd	8.95	17.90	20.59	8.95	11.19
reni	9.56	19.12	19.12	12.43	13.86
rock1	13.99	22.99	22.99	14.99	14.49
rxra	0.00	5.32	25.83	2.66	12.91
sahh	0.00	15.49	51.12	7.85	31.42
src	9.35	6.68	6.68	5.91	6.01
tgfr1	0.75	2.98	3.73	1.88	3.00
thb	0.00	0.00	4.82	0.00	2.43
thrb	30.27	31.57	30.27	20.90	21.12
try1	15.79	15.79	18.68	11.90	12.90
tryb1	6.75	10.12	11.47	7.09	8.10
tysy	39.15	42.79	42.79	30.26	29.80
urok	8.56	14.67	15.90	8.91	10.75
vgfr2	17.56	13.90	16.10	11.85	11.00
wee1	61.28	61.28	61.28	46.69	47.67
xiap	15.83	24.74	36.62	12.99	18.48

Table B.17: The results of prediction accuracies for 102 DUD-E targets with three-step screening (1). a%–b%–c% means the $EF_{c\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide HTVS pre-screening from a% subset of compounds to b% subset of compounds, (c) Glide SP docking calculation for b% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide HTVS - Glide SP		
	5%–2%–1%	10%–2%–1%	10%–5%–1%
aa2ar	8.71	9.54	10.99
abl1	8.73	7.10	8.19
ace	6.36	6.01	6.71
aces	11.69	5.30	10.59
ada	7.45	13.83	14.90
ada17	9.57	16.14	17.83
adrb1	5.66	5.26	6.88
adrb2	13.78	20.68	20.68
akt1	6.79	6.45	6.45
akt2	5.91	5.91	7.59
aldr	15.02	16.27	21.90
ampc	6.24	12.48	10.40
andr	14.41	17.74	17.74
aofb	5.67	7.29	7.29
bace1	22.57	17.98	22.92
braf	8.55	9.21	11.18
cah2	0.00	0.20	0.20
casp3	19.08	21.09	20.08
cdk2	5.26	7.57	7.57
comt	2.43	0.00	2.43
cp2c9	0.00	2.49	2.49
cp3a4	8.80	8.21	7.63
csf1r	4.19	4.78	5.38
cxcr4	9.85	4.92	2.46
def	6.86	5.88	5.88
dhi1	0.30	1.21	1.51
dpp4	10.87	14.80	14.05

Table B.18: The results of prediction accuracies for 102 DUD-E targets with three-step screening (2). a%–b%–c% means the $EF_{c\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide HTVS pre-screening from a% subset of compounds to b% subset of compounds, (c) Glide SP docking calculation for b% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide HTVS - Glide SP		
	5%–2%–1%	10%–2%–1%	10%–5%–1%
drd3	6.65	5.40	7.06
dyr	13.77	12.91	12.48
egfr	9.40	10.87	13.09
esr1	22.92	22.66	26.56
esr2	16.31	21.47	29.90
fa10	8.56	10.24	10.61
fa7	28.76	30.51	32.25
fabp4	12.75	17.00	21.25
fak1	9.91	9.91	8.92
fgfr1	0.71	2.86	2.86
fkb1a	3.55	7.10	7.10
fnta	16.02	17.20	16.70
fpps	0.00	0.00	0.00
gcr	10.04	11.20	11.20
glcm	16.47	14.64	14.64
gria2	20.24	25.93	27.82
grik1	12.77	23.58	23.58
hdac2	4.86	7.56	7.56
hdac8	2.33	2.92	3.50
hivint	7.94	13.89	10.91
hivpr	17.13	18.99	21.78
hivrt	12.08	14.43	14.43
hmdh	6.99	11.07	14.56

Table B.19: The results of prediction accuracies for 102 DUD-E targets with three-step screening (3). a%–b%–c% means the $EF_{c\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide HTVS pre-screening from a% subset of compounds to b% subset of compounds, (c) Glide SP docking calculation for b% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide HTVS - Glide SP		
	5%–2%–1%	10%–2%–1%	10%–5%–1%
hs90a	0.00	0.00	0.00
hxk4	0.00	1.08	2.17
igf1r	14.10	15.44	13.43
inha	0.00	0.00	0.00
ital	0.72	0.72	1.44
jak2	5.53	6.45	10.13
kif11	10.29	12.87	14.58
kit	6.57	3.59	1.79
kith	1.70	6.80	18.70
kpcb	21.31	26.45	27.18
lck	8.57	10.95	9.76
lkha4	0.00	1.74	1.74
mapk2	14.73	16.69	18.66
mcr	6.31	7.36	7.36
met	21.51	21.51	25.69
mk01	0.00	4.98	3.74
mk10	4.74	3.79	2.84
mk14	2.42	2.77	2.77
mmp13	7.17	9.09	10.49
mp2k1	8.23	4.94	3.29
nos1	7.95	4.97	4.97
nram	23.46	27.54	30.60
pa2ga	2.00	4.00	7.00
parp1	29.67	35.96	38.12
pde5a	8.77	10.27	9.77
pgh1	1.54	1.54	1.54
pgh2	1.84	2.53	2.53

Table B.20: The results of prediction accuracies for 102 DUD-E targets with three-step screening (4). a%–b%–c% means the $EF_{c\%}$ value with below procedure: (a) Spresso pre-screening from compound library to a% subset of compounds, (b) Glide HTVS pre-screening from a% subset of compounds to b% subset of compounds, (c) Glide SP docking calculation for b% subset of compounds. The score function of Spresso was ENE1.8 and it utilized Glide HTVS fragment docking.

Target	Spresso - Glide HTVS - Glide SP		
	5%–2%–1%	10%–2%–1%	10%–5%–1%
plk1	0.92	5.53	4.61
pnph	51.12	52.08	50.15
ppara	8.82	10.42	10.42
ppard	2.91	4.99	2.91
pparg	8.25	9.89	9.48
prgr	11.22	13.60	13.60
ptn1	13.03	16.09	22.23
pur2	47.04	43.12	49.00
pygm	2.54	2.54	2.54
pyrd	17.90	15.22	20.59
reni	17.21	14.34	21.03
rock1	21.99	19.99	25.99
rxra	5.32	25.83	25.83
sahh	15.49	51.12	51.12
src	7.82	6.68	7.06
tgfr1	1.49	2.24	2.98
thb	0.00	4.82	4.82
thrb	31.78	30.92	31.35
try1	16.68	20.02	18.68
tryb1	6.75	8.77	7.42
tysy	40.97	36.42	42.79
urok	13.45	13.45	14.06
vgfr2	13.90	14.14	16.10
wee1	61.28	61.28	61.28
xiap	24.74	33.65	35.63

References

[1] Paul S.M. *et al.*: "How to improve R&D productivity: the pharmaceutical industry's grand challenge", *Nat. Rev. Drug Discov.* 9, 203–214, 2010.

[2] DiMasi J.A. *et al.*: "Innovation in the pharmaceutical industry: New estimates of R&D costs", *J. Health Econ.* 47, 20–33, 2016.

[3] Cheng T. *et al.*: "Structure-based virtual screening for drug discovery: a problem-centric review", *AAPS J.* 14, 133–141, 2012.

[4] Lavecchia A.: "Machine-learning approaches in drug discovery: methods and applications", *Drug Discov. Today* 20, 318–331, 2015.

[5] Cereto-Massagué A. *et al.*: "Molecular fingerprint similarity search in virtual screening", *Methods* 71, 58–63, 2015.

[6] Cerqueira N.M.F.S.A. *et al.*: "Receptor-based virtual screening protocol for drug discovery", *Arch. Biochem. Biophys.* 582, 56–67, 2015.

[7] Lionta E. *et al.*: "Structure-Based Virtual Screening for Drug Discovery: Principles, Applications and Recent Advances", *Curr. Top. Med. Chem.* 14, 1923–1938, 2014.

[8] Wang Y. *et al.*: "*In silico* ADME/T modelling for rational drug design", *Q. Rev. Biophys.* 48, 488–515, 2015.

[9] Xu X. *et al.*: "Docking-based inverse virtual screening: methods, applications, and challenges", *Biophys. Rep.* 4, 1–16, 2018.

[10] Klon A.E. *et al.*: "Finding more needles in the haystack: a simple and efficient method for improving high-throughput docking results", *J. Med. Chem.* 47, 2743–2749, 2004.

[11] Sliwoski G. *et al.*: "Computational methods in drug discovery", *Pharmacol. Rev.* 66, 334–395, 2014.

[12] Yuriev E. *et al.*: "Improvements, trends, and new ideas in molecular docking: 2012-2013 in review", *J Mol. Recognit.* 28, 581–604, 2015.

[13] Śledź P & Caflisch A.: "Protein structure-based drug design: from docking to molecular dynamics", *Curr. Opin. Struct. Biol.* 48, 93–102, 2018.

[14] Zhao H. & Caflisch A.: "Molecular dynamics in drug design", *Eur. J. Med. Chem.* 91, 4–14, 2015.

[15] Kurczab R. & Bojarski A.J.: "New strategy for receptor-based pharmacophore query construction: a case study for 5-HT7 receptor ligands", *J. Chem. Inf. Model.* 53, 3233–3243, 2013.

[16] Yang S.Y.: "Pharmacophore modeling and applications in drug discovery: challenges and recent advances", *Drug Discov. Today* 15, 444–450, 2010.

[17] Hansch C. & Fujita T.: "ρ-σ-π Analysis. A Method for the Correlation of Biological Activity and Chemical Structure", *J. Am. Chem. Soc.* 86, 1616–1626, 1964.

[18] Damale M.G. *et al.*: "Recent advances in multidimensional QSAR (4D-6D): a critical review", *J. Mol. Recognit.* 14, 35–55, 2014.

[19] Ivanciuc O.: "Applications of support vector machines in chemistry", *Rev. Comput. Chem.* 23, 291–400, 2007.

[20] Zoete V. *et al.*: "SwissSimilarity: A Web Tool for Low to Ultra High Throughput Ligand-Based Virtual Screening", *J. Chem. Inf. Model.* 56, 1399–1404, 2016.

[21] Jasial S. *et al.*: "Activity-relevant similarity values for fingerprints and implications for similarity searching", *F1000Res.* 5, Chem. Inf. Sci.-591, 2016.

[22] Wolber G. *et al.*: "Molecule-pharmacophore superpositioning and pattern matching in computational drug design", *Drug Discov. Today* 13, 23–29, 2008.

[23] Brown J.B. & Okuno Y.: "Systems Biology and Systems Chemistry: New Directions for Drug Discovery", *Chem. Biol.* 19, 23–28, 2012.

[24] Hamanaka M. *et al.*: "CGBVS- DNN: Prediction of Compound- protein Interactions Based on Deep Learning", *Mol. Inform.* 36, 1600045, 2017.

[25] Liu Y. *et al.*: "Neighborhood regularized logistic matrix factorization for drug-target interaction prediction", *PLoS Comput. Biol.* 12, e1004760, 2016.

[26] Drwal M.N. & Griffith R.: "Combination of ligand- and structure-based methods in virtual screening", *Drug Discov. Today Technol.* 10, e395–e401, 2013.

[27] Rose P.W. *et al.*: "The RCSB Protein Data Bank: views of structural biology for basic and applied research and education", *Nucleic Acids Res.* 43, D345–D356, 2015.

[28] Ferreira L.G. *et al.*: "Molecular docking and structure-based drug design strategies", *Molecules* 20, 13384-13421, 2015.

[29] Shin W.H. *et al.*: "PL-PatchSurfer2: improved local surface matching-based virtual screening method that is tolerant to target and ligand structure variation", *J. Chem. Inf. Model.* 56, 1676–1691, 2016.

[30] Henzler A.M. *et al.*: "An integrated approach to knowledge-driven structure-based virtual screening", *J. Comput.-Aided. Mol. Des.* 28, 927–939, 2014.

[31] Gowthaman R. *et al.*: "DARC 2.0: improved docking and virtual screening at protein interaction sites", *PLoS one* 10, e0131612, 2015.

[32] Meng X. *et al.*: "Molecular docking: a powerful approach for structure-based drug discovery", *Curr. Comput. Aided Drug Des.* 7, 146–157, 2011.

[33] Halgren T.A. *et al.*: "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 2. Enrichment Factors in Database Screening", *J. Med. Chem.* 47, 1750–1759, 2004.

[34] Deng Z. *et al.*: "Structural Interaction Fingerprint (SIFt): A Novel Method for Analyzing Three-Dimensional Protein-Ligand Binding Interactions", *J. Med. Chem.* 47, 37–344, 2004.

[35] Da C. & Kireev D.: "Structural ProteinLigand Interaction Fingerprints (SPLIF) for Structure-Based Virtual Screening: Method and Benchmark Study", *J. Chem. Inf. Model.* 54, 2555–2561, 2014.

[36] Krishna S. *et al.*: "Identification of potent inhibitors of DNA methyltransferase 1 (DNMT1) through a pharmacophore-based virtual screening approach", *J. Mol. Graph. Model.* 75, 174–188, 2017.

[37] Friesner R.A. *et al.*: "Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy", *J. Med. Chem.* 47, 1739–1749, 2004.

[38] Zsoldos Z. *et al.*: "eHiTS: A new fast, exhaustive flexible ligand docking system", *J. Mol. Graph. Model.* 26, 198–212, 2007.

[39] Morris G.M. *et al.*: "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility", *J. Comput. Chem.* 30, 2785–2791, 2009.

[40] Trott O. & Olson A.: "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multi-threading", *J. Comput. Chem.* 31, 455–461, 2010.

[41] Ewing T.J. *et al.*: "DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases", *J. Comput.-Aided Mol. Des.* 15, 411–428, 2001.

[42] Krüger D.M. & Evers A.: "Comparison of structure- and ligand-based virtual screening protocols considering hit list complementarity and enrichment factors", *ChemMedChem* 5, 148–158, 2010.

[43] Chaput L. *et al.*: "Benchmark of four popular virtual screening programs: construction of the active/decoy dataset remains a major determinant of measured performance", *J. Cheminform.* 8, 56, 2016.

[44] Harder E. *et al.*: "OPLS3: A Force Field Providing Broad Coverage of Drug-like Small Molecules and Proteins", *J. Chem. Theory Comput.* 12, 281–296, 2016.

[45] Yuriev E. & Ramsland P.A.: "Latest developments in molecular docking: 2010–2011 in review", *J Mol. Recognit.* 26, 215–239, 2013.

[46] Kannan S. & Ganji R.: "Porting Autodock to CUDA", In *Proc. of CEC2010* 1–8, 2010.

[47] McIntosh-Smith S. *et al.*: "High performance in silico virtual drug screening on many-core processors", *Int. J. High Perform. Comput. Appl.* 29, 119–134, 2015.

[48] Irwin J.J. *et al.*: "ZINC: a free tool to discover chemistry for biology", *J. Chem. Inf. Model.* 52, 1757–1768, 2012.

[49] Kim S. *et al.*: "PubChem substance and compound databases", *Nucleic Acids Res.* 44, D1202-D1213, 2016.

[50] Sterling T. & Irwin J.J.: "ZINC 15 – Ligand Discovery for Everyone", *J. Chem. Inf. Model.* 55, 2324–2337, 2015.

[51] Ruddigkeit L. *et al.*: "Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17", *J. Chem. Inf. Model.* 52, 2864–2875, 2012.

[52] Edlridge M.D. *et al.*: "Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes", *J. Comput. Aided Mol. Des.* 11, 425–445, 1997.

[53] Kumar A. & Zhang K.: "Hierarchical virtual screening approaches in small-molecule drug discovery", *Methods* 71, 26–37, 2015.

[54] Rarey M. *et al.*: "A Fast Flexible Docking Method using an Incremental Construction Algorithm", *J. Mol. Biol.* 261, 470–489, 1996.

[55] McGann M.: "FRED pose prediction and virtual screening accuracy", *J. Chem. Inf. Model.* 51, 578–596, 2011.

[56] Jones G. *et al.*: "Development and validation of a genetic algorithm for flexible docking", *J. Mol. Biol.* 267, 727–748, 1997.

[57] Verdonk M.L. *et al.*: "Improved protein-ligand docking using GOLD", *Proteins* 52, 609–623, 2003.

[58] Abagyan R. *et al.*: "ICM—a new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation", *J. Comput. Chem.* 15, 488–506, 1994.

[59] Ruiz-Carmona S. *et al.*: "rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids", *PLoS Comput. Biol.* 10, e1003571, 2014.

[60] Lipinski C.A. *et al.*: "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings", *Adv. Drug Deliv. Rev.* 23, 3–25, 1997.

[61] Bickerton G.R. *et al.*: "Quantifying the chemical beauty of drugs", *Nat. Chem.* 4, 90–98, 2012.

[62] Nilakantan R. *et al.*: "New method for rapid characterization of molecular shapes: applications in drug design", *J. Chem. Inf. Comput. Sci.* 33, 79–85, 1993.

[63] Parenti M.D. *et al.*: "Docking and Database Screening Reveal New Classes of Plasmodium falciparum Dihydrofolate Reductase Inhibitors", *J. Med. Chem.* 46, 2834–2845, 2003.

[64] Fujimoto T. *et al.*: "*In silico* multi-filter screening approaches for developing novel beta-secretase inhibitors", *Bioorg. Med. Chem. Lett.* 18, 2771–2775, 2008.

[65] Grover A. *et al.*: "A leishmaniasis study: Structure-based screening and molecular dynamics mechanistic analysis for discovering potent inhibitors of spermidine synthase", *Biochim. Biophys. Acta* 1824, 1476–1483, 2012.

[66] Muralidharan A.R. *et al.*: "Structure-Based Virtual Screening and Biological Evaluation of a Calpain Inhibitor for Prevention of Selenite-Induced Cataractogenesis in an in Vitro System", *J. Chem. Inf. Model.* 55, 1686–1697, 2015.

[67] Mirza S.B. *et al.*: "Virtual Screening of Eighteen Million Compounds against Dengue Virus: Combined Molecular Docking and Molecular Dynamics Simulations Study", *J. Mol. Graph. Model.* 66, 99–107, 2016.

[68] Niinivehmas S.P. *et al.*: "Ultrafast protein structure-based virtual screening with Panther", *J. Comput. Aided Mol. Des.* 29, 989–1006, 2015.

[69] Ertl P. *et al.*: "Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties", *J. Med. Chem.* 43, 3714–3717, 2000.

[70] Zhao Y.H. *et al.*: "Fast calculation of van der Waals volume as a sum of atomic and bond contributions and its application to drug compounds", *J. Org. Chem.* 68, 7368–7373, 2003.

[71] Yanagisawa K. *et al.*: "Spresso: An ultrafast compound pre-screening method based on compound decomposition", *Bioinformatics* 33, 3836–3843, 2017.

[72] Yanagisawa K. *et al.*: "ESPRESSO: An ultrafast compound pre-screening method based on compound decomposition", *IPSJ SIG Tech. Rep.* 2016-BIO-46(18), 1–7, 2016.

[73] Yanagisawa K. *et al.*: "An exact algorithm for the weighted offline cache problem in protein-ligand docking based on fragment extension", *IPSJ SIG Tech. Rep.* 2017-BIO-50(38), 1–8, 2017.

[74] Yanagisawa K. *et al.*: "Optimization of memory use of fragment extension-based protein-ligand docking with an original fast minimum cost flow algorithm", *Comput. Biol. Chem.* 74, 399–406, 2018.

[75] Verdonk M.L. & Rees D.C.: "Group Efficiency: A Guideline for Hits-to-Leads Chemistry", *ChemMedChem* 3, 1179–1180, 2008.

[76] Möbitz H. *et al.*: "Discovery of Potent, Selective, and Structurally Novel Dot1L Inhibitors by a Fragment Linking Approach", *ACS Med. Chem. Lett.* 2017, 338–343, 2017.

[77] Hall D.R. *et al.*: "Hot Spot Analysis for Driving the Development of Hits into Leads in Fragment-Based Drug Discovery", *J. Chem. Inf. Model.* 52, 199–209, 2012.

[78] Barker J.J. *et al.*: "Discovery of a Novel Hsp90 Inhibitor by Fragment Linking", *ChemMedChem* 5, 1697–1700, 2010.

[79] Howard N. *et al.*: "Application of Fragment Screening and Fragment Linking to the Discovery of Novel Thrombin Inhibitors", *J. Med. Chem.* 49, 1346–1355, 2006.

[80] Lewell X.Q. *et al.*: "RECAP—Retrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry", *J. Chem. Inf. Comput. Sci.* 38, 511–522, 1998.

[81] Degen J. *et al.*: "On the art of compiling and using 'drug-like' chemical fragment spaces", *ChemMedChem* 3, 1503–1507, 2008.

[82] Congreve M. *et al.*: "A 'Rule of Three' for fragment-based lead discovery?", *Drug Discov. Today* 8, 876–877, 2003.

[83] O'Boyle N. M. *et al.*: "Open Babel: An open chemical toolbox", *J. Cheminform.* 3, 33, 2011.

[84] RDKit: Open-source cheminformatics. `https://www.rdkit.org/` (accessed at 2018-11-19)

[85] Bento A.P. *et al.*: "The ChEMBL bioactivity database: an update", *Nucleic Acids Res.* 42, 1083–1090, 2014.

[86] Mysinger M.M. *et al.*: "Directory of Useful Decoys, Enhanced (DUD-E): better ligands and decoys for better benchmarking", *J. Med. Chem.* 55, 6582–6594, 2012.

[87] Hamza A. *et al.*: "Ligand-based virtual screening approach using a new scoring function", *J. Chem. Inf. Model.* 52, 963–974, 2012.

[88] Rogers D. & Hahn M.: "Extended-connectivity fingerprints", *J. Chem. Inf. Model.* 50, 742–754, 2010.

[89] Verdonk M.L. *et al.*: "Virtual screening using protein-ligand docking: avoiding artificial enrichment", *J. Chem. Inf. Comput. Sci.* 44, 793–806, 2004.

[90] Plewczynski D. *et al.*: "Can We Trust Docking Results? Evaluation of Seven Commonly Used Programs on PDBbind Database", *J. Comput. Chem.* 32, 742–755, 2011.

[91] Meng E.C. *et al.*: "Automated Docking with Grid-Based Energy Evaluation", *J. Comput. Chem.* 13, 505–524, 1992.

[92] Buchbinder N. & Naor J.S.: "The design of competitive online algorithms via a primal dual approach", *Found. Trends Theor. Comput. Sci.* 3, 93-263, 2009.

[93] Lopez-Ortiz A. & Salinger A.: "Minimizing Cache Usage in Paging", In *Proc. of WAOA2012* 145–158, 2013.

[94] Jewell W.S.: "Optimal flow through networks" *Interim Tech. Rep. No.8*, Operations Research Center, MIT, Cambridge, MA, 1958.

[95] Jewell W.S.: "Optimal flow through networks with gains", *Oper. Res.* 10, 476–499, 1962.

[96] Busacker R.G. & Gowen P.: "A procedure for determining a family of minimum-cost network flow patterns" *Interim Tech. Rep. No.15*, Operations Research Center, The Johns Hopkins University, Bethesda, MD, 1960.

[97] Iri M.: "A new method for solving transportation-network problems", *J. Oper. Res. Soc. Jpn.* 3, 27–87, 1960.

[98] Gabow H.N. & Tarjan R.E.: "Faster Scaling Algorithms for Network", *SIAM J.Comput.* 18, 1013–1036, 1989.

[99] Dantzig G.B.: "Linear Programming and Extensions" Princeton Univ. Press, West Sussex, England, 1963.

[100] Kelly D.J. & O'Neill G.M.: "The minimum cost flow problem and the Network-Simplex method", Master's thesis, University College, Dublin, Ireland, 1991.

[101] Orlin J.B.: "A polynomical time primal Network-Simplex algorithm for minimum cost flows", *Math. Program.* 78, 109–129, 1997.

[102] Pirsiavash H. *et al.*: "Globally-optimal greedy algorithms for tracking a variable number of objects", In *Proc. of 2011 IEEE Conf. CVPR* 1201–1208, 2011.

[103] Dezso B. *et al.*: "LEMON - an Open Source C++ Graph Template Library", *Electron. Notes Theor. Comput. Sci.* 264, 23–45, 2011.

[104] Kiraly Z. & Kovacs P.: "Efficient implementations of minimum-cost flow algorithms", *Acta Univ. Sapientiae Informatica* 4, 67–118, 2012.

[105] Kovacs P.: "Minimum-cost flow algorithms: an experimental evaluation", *Optim. Methods Softw.* 30, 94–127, 2015.

[106] Bemis G.W. & Murcko M.A.: "The properties of known drugs. 1. Molecular frameworks", *J. Med. Chem.* 39, 2887–2893, 1996.

List of Publications

Journal Papers

1. Masahiro Mochizuki, Shogo D. Suzuki, **Keisuke Yanagisawa**, Masahito Ohue, Yutaka Akiyama. "QEX: Target-specific druglikeness filter enhances ligand-based virtual screening", *Molecular Diversity* 23, 11–18, 2019.

2. Takashi Tajimi, Naoki Wakui, **Keisuke Yanagisawa**, Yasushi Yoshikawa, Masahito Ohue, Yutaka Akiyama. "Computational prediction of plasma protein binding of cyclic peptides from small molecule experimental data using sparse modeling techniques", *BMC Bioinformatics* 19, 527, 2018.

3. **Keisuke Yanagisawa**, Shunta Komine, Rikuto Kubota, Masahito Ohue, Yutaka Akiyama. "Optimization of memory use of fragment extension-based protein-ligand docking with an original fast minimum cost flow algorithm", *Computational Biology and Chemistry* 74, 399–406, 2018.

4. Takanori Hayashi, Yuri Matsuzaki, **Keisuke Yanagisawa**, Masahito Ohue, Yutaka Akiyama. "MEGADOCK-Web: an integrated database of high-throughput structure-based protein-protein interaction predictions", *BMC Bioinformatics* 19, 62, 2018.

5. **Keisuke Yanagisawa**, Shunta Komine, Shogo D. Suzuki, Masahito Ohue, Takashi Ishida, Yutaka Akiyama. "Spresso: An ultrafast compound pre-screening method based on compound decomposition", *Bioinformatics* 33, 3836–3843, 2017.

6. Shuntaro Chiba, Takashi Ishida, Kazuyoshi Ikeda, Masahiro Mochizuki, Reiji Teramoto, Y-h. Taguchi, Mitsuo Iwadate, Hideaki Umeyama, Chandrasekaran Ramakrishnan, A. Mary Thangakani, D. Velmurugan, M. Michael Gromiha, Tatsuya Okuno, Koya Kato, Shintaro Minami, George Chikenji, Shogo D. Suzuki, **Keisuke Yanagisawa**, Woong-Hee Shin, Daisuke Kihara, Kazuki Z. Yamamoto, Yoshitaka Moriwaki, Nobuaki Yasuo, Ryunosuke Yoshino, Sergey Zozulya, Petro Borysko, Roman Stavniichuk, Teruki Honma, Takatsugu Hirokawa, Yutaka Akiyama, Masakazu Sekijima. "An iterative compound screening contest method

for identifying target protein inhibitors using the tyrosine-protein kinase Yes", *Scientific Reports* 7, 12038, 2017.

7. Shuntaro Chiba, Kazuyoshi Ikeda, Takashi Ishida, M. Michael Gromiha, Y-h. Taguchi, Mitsuo Iwadate, Hideaki Umeyama, Kun-Yi Hsin, Hiroaki Kitano, Kazuki Yamamoto, Nobuyoshi Sugaya, Koya Kato, Tatsuya Okuno, George Chikenji, Masahiro Mochizuki, Nobuaki Yasuo, Ryunosuke Yoshino, **Keisuke Yanagisawa**, Tomohiro Ban, Reiji Teramoto, Chandrasekaran Ramakrishnan, A. Mary Thangakani, D. Velmurugan, Philip Prathipati, Junichi Ito, Yuko Tsuchiya, Kenji Mizuguchi, Teruki Honma, Takatsugu Hirokawa, Yutaka Akiyama, Masakazu Sekijima. "Identification of potential inhibitors based on compound proposal contest: Tyrosine-protein kinase Yes as a target", *Scientific Reports* 5, 17209, 2015.

8. **Keisuke Yanagisawa**, Takashi Ishida, Yutaka Akiyama. "Drug clearance pathway prediction based on semi-supervised learning", *IPSJ Transactions on Bioinformatics* 8, 21–27, 2015.

Peer-reviewed International Conferences

1. Takashi Tajimi, Naoki Wakui, **Keisuke Yanagisawa**, Yasushi Yoshikawa, Masahito Ohue, Yutaka Akiyama. "Computational prediction of plasma protein binding of cyclic peptides from small molecule experimental data using sparse modeling techniques", *The 29th International Conference on Genome Informatics (GIW 2018)*, 3rd–5th, Dec., 2018, Yunnan, China.

2. **Keisuke Yanagisawa**, Shunta Komine, Rikuto Kubota, Masahito Ohue, Yutaka Akiyama. "Optimization of memory use of fragment extension-based protein-ligand docking with an original fast minimum cost flow algorithm", *The 16th Asia Pacific Bioinformatics Conference (APBC2018)*, 15th–17th, Jan., 2018, Yokohama, Japan.

3. Takanori Hayashi, Yuri Matsuzaki, **Keisuke Yanagisawa**, Masahito Ohue, Yutaka Akiyama. "MEGADOCK-Web: an integrated database of high-throughput structure-based protein-protein interaction predictions", *The 16th Asia Pacific Bioinformatics Conference (APBC2018)*, 15th–17th, Jan., 2018, Yokohama, Japan.

4. **Keisuke Yanagisawa**, Shunta Komine, Shogo D. Suzuki, Masahito Ohue, Takashi Ishida, Yutaka Akiyama. "ESPRESSO: An ultrafast compound prescreening method based on compound decomposition", *The 27th International Conference on Genome Informatics (GIW 2016)*, 3rd–5th, Oct., 2016, Shanghai, China.

Honors and Awards

1. 4th Computer-Aided Drug Discovery Contest Grand Prize (Schrödinger K.K. Prize) (2017)
2. JSPS Research Fellow (DC2) (2017–2019)
3. Grant-in-Aid for JSPS Fellows (17J06897) (2017–2019)
4. 2nd Computer-Aided Drug Discovery Contest Student Encouragement Prize (2015)
5. 2014 IPSJ SIGBIO Best Student Presentation Award (2015)
6. 1st Computer-Aided Drug Discovery Contest Student Encouragement Prize (2014)
7. 2013 Tokyo Institute of Technology Academic Excellence Awards (2014)
8. 2009 High School Chemistry Grand Prix Bronze Prize (2009)