

Development of Multifunctional Object-Oriented Program Library for Molecular Simulation and Structure Analysis

Toshiyuki MEGURO, Tadashi ANDO and Ichiro YAMATO*

Department of Biological Science and Technology, Tokyo University of Science
2641 Yamazaki, Noda-shi, Chiba 278-8510, Japan

*e-mail: iyamato@rs.noda.tus.ac.jp

(Received: May 20, 2002; Accepted for publication: January 15, 2003; Published on Web: March 6, 2003)

We developed the object-oriented class library “YLO” to study molecular structures. YLO is written in C++ and composed of a set of classes, which are a kind of program units, resembling subroutines in Fortran. Each class performs a certain job independently in a whole program. YLO deals with non-organic, organic molecules and biological macromolecules. The coordinate file of a molecule is read / written in PDB format. Using the YLO library, one can make various types of application programs efficiently, such as Monte Carlo simulator, Genetic Algorithm optimizer and so on.

Keywords: Object-oriented programming, C++, Molecular simulation, Inorganic compounds, Organic compounds, Biological macromolecules

1 Introduction

Genome analysis provides us with an enormous number of primary sequences of DNA, RNA and proteins. Organic chemistry produces more and more new compounds. Researchers are trying to determine the three dimensional (3D) structures of such molecules, non-organic, organic and bio-molecules. Development of computer hardware enables us to deal with such 3D structures. There are many program systems available for molecular simulation and structure analysis, such as TINKER, AMBER, PEACH and so on [1–3].

However, when one wants to develop and apply new algorithms, one has to make new programs for oneself. In such situations, the program development would be more efficient if multifunctional class library is available. The object-oriented programming (OOP) technique is now used in various fields of computation. OOP technique can provide us with much easier maintainability and much higher flexibility of a program system.

Therefore, we developed a new object-oriented class library, named “YLO”, in order to construct flexible and multifunctional software systems for molecular simulation and structure analysis. YLO stands for “Yamazaki (address of our laboratory) Library for Object-oriented programming”.

Using YLO, we developed several application programs for molecular structure studies, such as Monte Carlo simulator, Genetic Algorithm optimizer, crystal

structure builder and so on. In this article, we describe the outline of YLO library and a few examples of derived programs.

2 Hardware and software

YLO was developed on IBM/AT compatible (DOS/V) machines under the operating system of Red Hat Linux 6.1. The library is written in C++ and is currently running on a Duron 800MHz PC system in our laboratory.

Most classes of YLO are composed of machine independent commands. Thus, the application programs derived from YLO library run on various operating systems (Red Hat Linux 6.1, windows95, windows98, DEC digital Unix 4.0B and so on) as long as an appropriate C++ compiler (such as visual C++ and so on) is installed.

3 Description of YLO library and application software

3.1 YLO library

YLO consists of many functional classes. Figure 1 shows a flow chart (diagram) indicating how to use such individual classes of YLO library. Assembling these classes, one can make various programs efficiently.

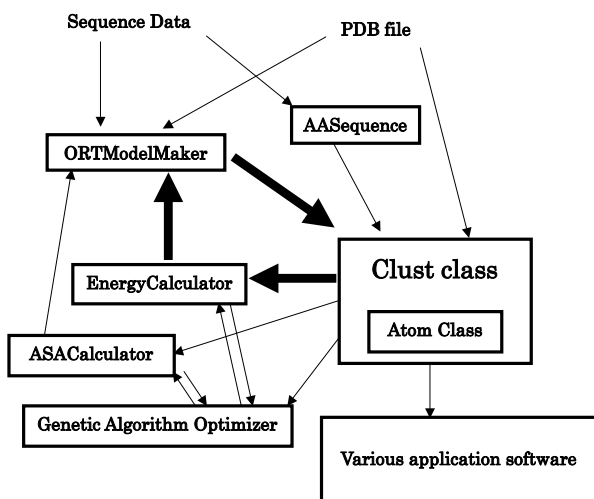


Figure 1. Flow Chart of the YLO library. Thick arrows indicate the calculation procedure of Monte Carlo simulation.

```

//-----Variables-----
double OldEnergy , NewEnergy ;
//-----Preparation-----
ORTModelMakerCls  ORTModelMaker ("H_ALA GLY ALA_0 " ) ;
ClustCls          Clust = ORTModelMaker.GetCluster() ;
EnergyCalculatorCls  EnergyCalculator( Clust ) ;
//-----Simulation-----
for ( int i = 1 ; i = 100 ; i++ ) {
    OldEnergy = EnergyCalculator.CalculateEnergy ( Clust ) ;
    Clust     = ORTModelMaker. RandomTransform ( Clust ) ;
    NewEnergy = EnergyCalculator.CalculateEnergy ( Clust ) ;
    if(JudgeMetro(OldEnergy,NewEnergy)==0) {
        Clust = ORTModelMaker.Restore ( Clust ) ;
    }
}
EnergyCalculator.ReportEnergy ( Clust ) ;

```

Figure 3-A. Source code of an include file for Monte Carlo simulator.

The usage of these classes is the same as that of general C++ classes: First, a user selects the appropriate classes according to his purpose and copies them to an appropriate directory. Second, the statements that declare the usage of the YLO classes are written in C++ source codes, shown in Figure 2 and Figure 3 as examples.

Descriptions of major classes are given below.

3.2 Clust class

Clust class is used to deal with the cluster of atoms and/or molecules. The class name “Clust” is an abbrevi-

```

(A)
Clust CluA (3);
CluA.SetAtomPosition (1, 1.0, 0.0, 0.0);
CluA.SetAtomPosition (2, 0.0, 1.0, 0.0);
CluA.SetAtomPosition (3, 0.0, 0.0, 1.0);
CluA.Report ( );

```

```

(B)
Clust CluB ("PDB coordinate file name");
CluB.Report ( );

```

```

(C)
ORTModelMakerCls ORTModelMaker ("H_ALA GLY ALA_0");
ClustCls CluC = ORTModelMaker.GetCluster();
CluC.Report ( );

```

```

(C-1)
ORTModelMakerCls ORTModelMaker ("H_ALA GLY ALA_0", "helix");

```

```

(C-2)
ORTModelMakerCls ORTModelMaker ("H_ALA GLY ALA_0", "strand");

```

Figure 2. How to use Clust class. Just writing a simple include file is enough for a programmer to use the Clust class. Various patterns are available. In this figure, three different patterns are shown as samples.

Pattern (A) is used to build non-organic simple molecules. In this example, the molecule that consists of 3 atoms is built. The coordinates of atoms 1,2 and 3 are (1.0,0.0,0.0), (0.0,1.0,0.0) and (0.0,0.0,1.0), respectively. Pattern (B) is recommended when peptide or protein molecules are dealt. In this pattern, appropriate PDB file is read and its parameter values are registered.

Pattern (C) is used to build peptide molecules from arbitrary amino acid sequence. In first line, ORTModelMaker Class builds straight-chain-type structure from amino acid sequence. In second line, Clust Class gets structural parameters from ORTModelMaker Class. In the case where one wants to build an alpha-helix-type structure or a beta-strand-type structure, the first line in (C) is substituted by the line in (C-1) or (C-2), respectively.

ation of “cluster”. Clust class builds molecular structures and records general parameters. This class also includes Atom class and registers atomic parameter values in it. Clust class is the most frequently used in this library.

The molecular parameters can be modified through member functions of Clust class. Clust class has also member functions that read / write coordinate files in PDB (Protein Data Bank) format [4]. Using these functions, a user can easily produce arbitrary molecular structures in his application programs.

```

//-----Preparation-----
int Population          = 30;
int TotalNumberOfGenerationSteps = 60;
double MutationProbability = 0.3;
double DesiredValueOfDeltaE = 50.0;
double MutationRate     = 0.1;
GeneticAlgorithmForForceFieldParameterCls GA_Optimizer;
IndividualCls Indiv;
//-----Set Gene-----
Indiv.AddGene("Cali", -0.0018); Indiv.AddGene("Caro", 0.0210);
Indiv.AddGene("Ohyd", -0.0208); Indiv.AddGene("Nami", -0.0181);
Indiv.AddGene("Ccar", 0.1912); Indiv.AddGene("Ocar", 0.0027);
Indiv.AddGene("Ssul", 0.0163);
//-----Add Individuals into GA_Optimizer-----
for(int i=1; i<=Population; i++){
    GA_Optimizer.AddIndividual(Indiv);
}
//-----Add protein structures into GA_Optimizer-----
GA_Optimizer.AddNativeStructure( ClustCls("Native_1.pdb") );
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_1.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_2.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_3.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_4.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_5.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_6.pdb"));
GA_Optimizer.AddNonNativeStructure(ClustCls("Non_7.pdb"));
GA_Optimizer.DesiredValueOfDeltaE(DesiredValueOfDeltaE);
//-----Execute genetic algorithm-----
GA_Optimizer.SetMutationRate(MutationRate);
GA_Optimizer.Evolve(TotalNumberOfGenerationSteps);
//-----Show results-----
GA_Optimizer.Report();

```

Figure 3-B. Source code of an include file for genetic algorithm.

Usage samples of Clust class are shown in Figure 2. In YLO library, non-peptide molecular structures are recorded in PDB format, although other formats for coordinate file are also available. The major functions of Clust class are listed in Table 1-A.

3.3 ORT Model Maker class

One of the major purposes for constructing YLO library is to make the program development easier for analysis of structures of protein molecules. In order to deal with such bio-molecules, ORT Model Maker class was developed; the first version of ORT Monte Carlo simulator for peptides [5] was divided into several pieces and reconstructed.

The proteins are biopolymers in which a lot of amino acids are connected in a chain by peptide bonds. There are 20 different amino acids and a protein has its unique amino acid sequence.

ORT Model Maker class reads "Residue.data" file in which structural data of 20 amino acids are written beforehand. Therefore, one does not have to describe topological information of protein molecules in detail to build their 3D structures. ORT Model Maker class reads an amino acid sequence file and builds corresponding 3D structure models. In the current version of YLO, alpha helix, beta strand, straight chain and random coil are available as model structures. The alpha helix and the beta strand are the basic secondary structure units that are found in protein molecules frequently. The straight chain and the random coil are rather unnatural structures that are mainly used as initial structures of folding simulation. This class also transforms the 3D structures of peptide chains with various methods. The main functions of ORT Model Maker class are listed in Table 1-B.

Table 1-A. The major functions of Clust class

Function (argument)	Description
ReadPdb (filename)	Read coordinate file in PDB format
WritePdb (filename)	Write coordinate file in PDB format
SetAtomPosition(ID number of atom, vector)	Set position of an atom into Clust
GetAtomPosition (ID number of atom)	Get position vector of an atom from Clust
SetAtomVelocity(ID number of atom, vector)	Set velocity of an atom into Clust
GetAtomVelocity (ID number of atom)	Get velocity of an atom from Clust
SetAtomForce (ID number of atom, vector)	Set force of an atom into Clust
GetAtomForce (ID number of atom)	Get force of an atom from Clust
GetCovalentBondAB (ID number of bond)	Get atom numbers of 1-2 interaction
GetAngleInteractionABC(ID number of angle)	Get atom numbers of 1-3 interaction
GetDihedralInteractionABCD(ID number of dihedral)	Get atom numbers of 1-4 interaction
Report ()	Report parameters of cluster.

Table 1-B. The major functions of ORTModelMaker class.

Function (argument)	Description
Constructor (sequence, structure type)	Build structure of peptide chain
WritePdb (filename)	Write coordinate file in PDB format
GetCluster ()	Get Cluster from ORT model maker
RotateTransform (Clust, position, angle)	Transform model using rotation method
ParallelTransform (Clust, position, distance)	Transform model using PT method [5]
RandomTransform (Clust)	Transform molecular model randomly
Restore(Clust)	Restore the former structure
Report ()	Report parameters of model maker

Table 1-C. The major functions of EnergyCalculator class.

Function (argument)	Description
SetResidueCutoff (Clust)	Make pair list with residue based cutoff
CalculateBondEnergy (Clust)	Calculate 1-2 energy
CalculateAngleEnergy (Clust)	Calculate 1-3 energy
CalculateDihedralEnergy (Clust)	Calculate 1-4 energy
CalculateVdwEnergy (Clust)	Calculate 1-5 (vdw) energy
CalculateElectrostaticEnergy (Clust)	Calculate 1-5 (Electrostatic) energy
CalculateEnergy (Clust)	Calculate total energy
ReportEnergy (Clust)	Report energy values
CalculateBondForce (Clust)	Calculate force derived from 1-2 interaction
CalculateAngleForce (Clust)	Calculate force derived from 1-3 interaction
CalculateDihedralForce (Clust)	Calculate force derived from 1-4 interaction
CalculateVdwForce (Clust)	Calculate vdw force
CalculateElectrostaticForce (Clust)	Calculate electrostatic force
CalculateTotalForce (Clust)	Calculate total Force
ReportForce (Clust)	Report force vectors

Table 1-D. The major functions of ASACalculator class.

Function (argument)	Description
CalculateASA (clust, atom number)	Calculate Accessible Surface Area
CalculateEnergyOfASA(clust, atom number)	Estimate hydration energy from ASA (Ooi Oobatake method)

3.4 Energy Calculator class

Energy Calculator class registers the force field parameters and calculates structural energies of molecules. Parm94 force field parameters of AMBER [6] are used in this class. This class produces molecular 1-2, 1-3, 1-4 and 1-5 energies separately according to the demand of the user. Combining this class with Clust and ORT Model Maker classes, a user can make Monte Carlo simulator easily. The main functions of this class are listed in Table 1-C.

3.5 ASA (Accessible Surface Area) Calculator class

Solvent molecules are often omitted in simulations of macromolecules. In such cases, ASA is used to estimate

solute-solvent interaction energy. In YLO library, ASA can be easily obtained by using ASA Calculator class. The algorithm to estimate ASA is constructed according to Ref. [7]. This class also calculates hydration free energies of peptide chains using the Ooi-Oobatake method [8]. The main functions of this class are listed in Table 1-D.

3.6 Other classes

In YLO library, many other classes are available, such as AASequence class (it is used for analysis of amino acid sequence), SDMethod class (it is used for energy minimization) and so on.

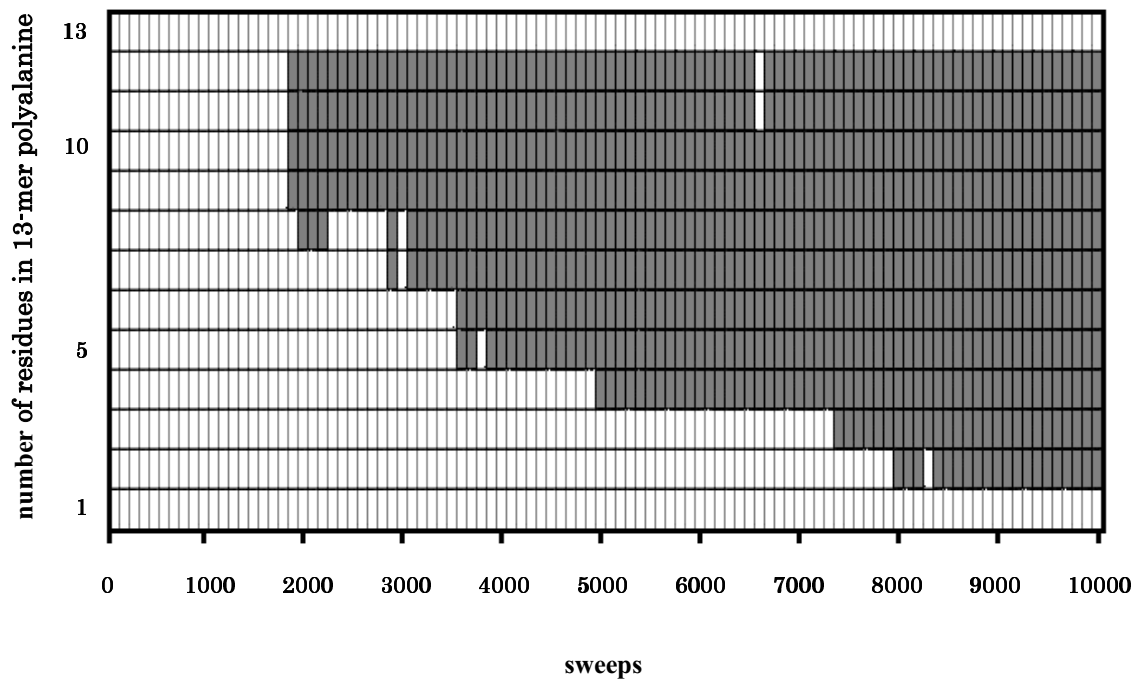


Figure 4-A. Evolution of helical conformations in the MC simulation of 13-mer polyaniline. Gray boxes in this graph indicate that the residues which correspond to these boxes formed helical conformation.

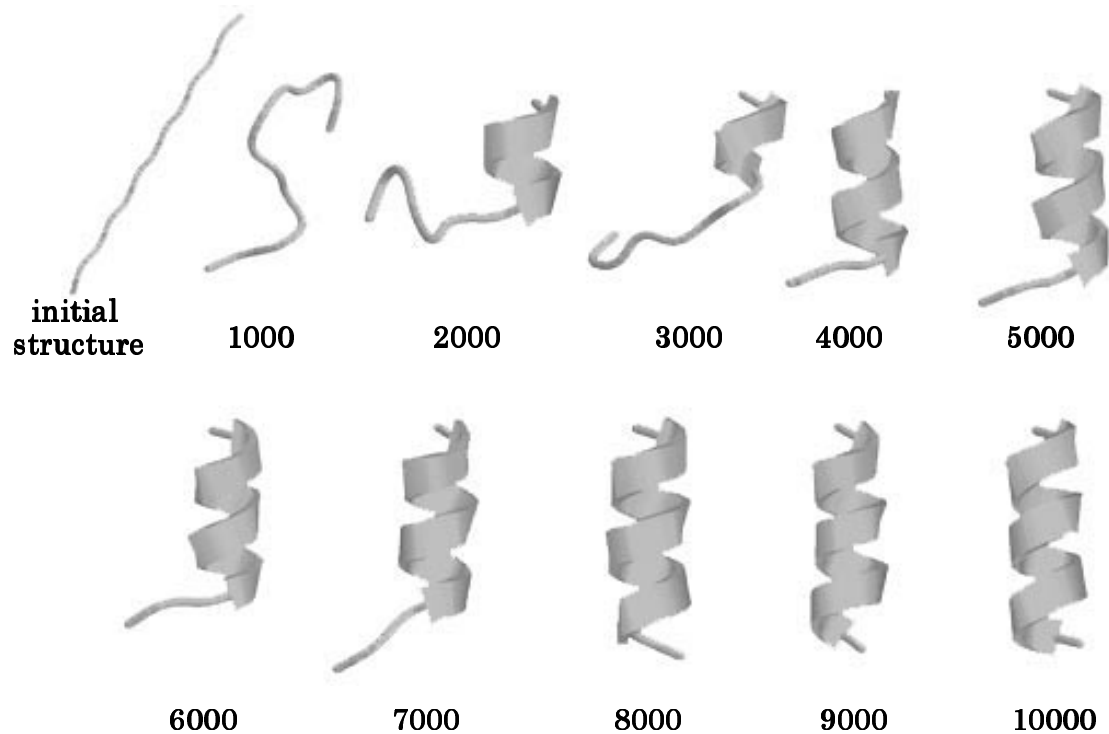


Figure 4-B. Snapshot structures of the 13-mer polyaniline at various sweeps in MC simulation. The numbers under respective structures show the sweep numbers. The figures were created with RasMol [10].

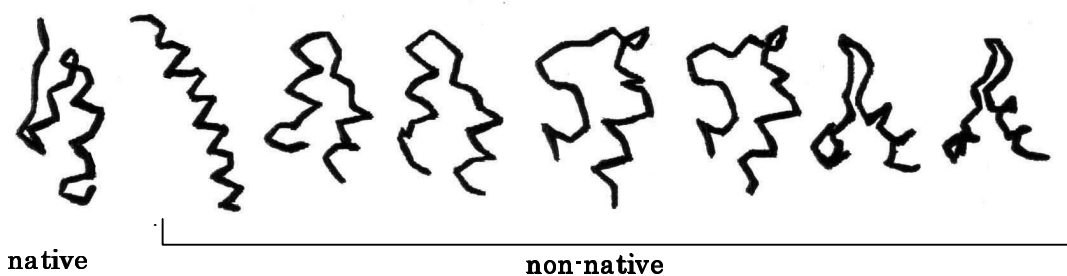


Figure 5. Native and non-native structures of the 28-mer polypeptide. Only the backbones are shown. These structures were used as sample structures in GA optimizer. The figures were created with RasMol [10].

3.7 Application software

Using YLO library, several application programs were produced easily, such as Monte Carlo simulator (MC simulator) for analysis of peptide chain folding, Genetic Algorithm optimizer (GA optimizer) for optimizing the parameters of ASA for use in peptide chain folding, whole crystal structure builder from an asymmetric structure and so on.

3.7.1 MC simulator

To make a new application programs one writes only one include file. Figure 3-A shows a sample include file for Monte Carlo simulation. This file is divided into two parts, Preparation part with four declare statements and Simulation part with 9 lines. Most of the fundamental functions in this program (Figure 2) are supplied from the YLO library. Therefore, just writing the outline of calculation algorithm is enough for a user to make a new piece of software.

We checked the performance of the MC simulator derived from YLO library by calculating the simulations of 13-residue polyalanine. The MC simulator used parm94 force field parameters to calculate structural energies. Each dihedral angle was changed randomly and the new dihedral angle was accepted according to the Metropolis criterion. All dihedral angles were changed in sequence in one sweep. Figure 4-A shows the evolution of helical conformations. Representative polyalanine structures at various sweeps are shown in Figure 4-B. Residues 9-12 formed a short helix at 1900 sweep. Then the helix grew gradually and finally residues 2-12 formed a long helix.

The result described above indicates that this MC simulator simulated nucleation and propagation of a helix.

3.7.2 GA optimizer

We describe an example of GA optimization here. The structural energy of a macromolecule is sometimes estimated from the accessible surface area (ASA) in MC simulation. We call this energy 'ASA energy'.

ASA energy is calculated using equation 1.

$$\begin{aligned} \text{ASA energy} = & k_{Cali} * \text{ASA of aliphatic C atoms} \\ & + k_{Caro} * \text{ASA of aromatic C atoms} \\ & + k_{Ohyd} * \text{ASA of hydroxyl O atoms} \\ & + k_{Nami} * \text{ASA of amide N Atoms} \\ & + k_{Ccar} * \text{ASA of carbonyl C atoms} \\ & + k_{Ocar} * \text{ASA of carbonyl O atoms} \\ & + k_{Ssul} * \text{ASA of sulfide S atoms} \\ & \text{--- (equation 1)} \end{aligned}$$

,where k_{Cali} , k_{Caro} , k_{Ohyd} , k_{Nami} , k_{Ccar} , k_{Ocar} and k_{Ssul} stand for parameters to estimate solvation free energies according to ASA's of aliphatic C atoms, aromatic C atoms, hydroxyl O atoms, amide N atoms, carbonyl C atoms, carbonyl O atoms and sulfide S atoms, respectively. The values of these parameters must be optimized correctly in order to get reliable ASA energy values. If the parameters are appropriate, the ASA energies of native structures should be lower than those of non-native structures.

We optimized these parameters using a genetic algorithm program derived from the YLO library. Several structures of 28-mer polypeptide [9] were used in this optimization (Figure 5). A purpose of this GA procedure was to get sets of parameters that make the ASA energies of native structures lower than those of other structures.

We executed the general GA optimization procedure (Figure 3-B). Parameter sets that consisted of k_{Cali} , k_{Caro} , k_{Ohyd} , k_{Nami} , k_{Ccar} and k_{Ocar} were dealt with as individuals in the GA procedure. The parameter k_{Ssul} was ignored because there was no sulfide atom in 28-mer. 30 parameter sets were used in each generation. The 30 sets in the first generation were made from one original parameter set. Parameter values in every set were slightly changed by mutation. The original values of k_{Cali} , k_{Caro} , k_{Ohyd} , k_{Nami} , k_{Ccar} and k_{Ocar} were decided according to ref [8].

In each generation, the scores of each set were calculated using the evaluation function (equation 2).

$$\text{score} = \frac{\{ \sum \min((E_{non} - E_{native}), D) \}}{N}$$

--- (equation 2)

E_{non} and E_{native} represent ASA energies of non-native and native structures, respectively. N is the total number of non-native structures. D is the desired value of $(E_{non} - E_{native})$. The score reaches maximum value when E_{native} is smaller than any E_{non} by at least D . If the value of D is too small, the score of equation 2 reaches maximum value easily with any parameter set and we cannot select a reasonable parameter set. On the other hand, if the value of D is too large, some parameters become extraordinarily

large in many parameter sets. Though it is difficult to estimate optimal value of D , we set the value to 50 kcal/mol tentatively, because generally the value of the free energy difference between native and non-native structure of a protein is several times larger than that of hydrogen bond energy (here the hydrogen bond energy is about 15-20 kcal/mol).

After the score calculation procedure, the whole group of parameter sets was divided into three small groups, A, B and C. A is high score group, B is middle score group and C is low score group. Each group consisted of 10 parameter sets.

Table 2. This table shows desirable parameter sets that were calculated using GA optimizer (ref. Figure 6). The score of each parameter set reached the maximum value, 50. These sets were obtained at 60th generation.

No.	Score	k_{Cali}	k_{Caro}	k_{Ohyd}	k_{Nami}	k_{Ccar}	k_{Ocar}
1	50	0.18	0.55	0.13	0.13	0.09	-0.38
2	50	-0.02	0.92	0.26	0.13	-0.01	-0.38
3	50	-0.02	1.70	-0.17	0.13	0.07	-0.38
4	50	0.18	0.55	0.13	0.00	-0.01	-0.38
5	50	-0.02	1.70	0.05	-0.06	-0.01	-0.41
6	50	0.12	0.55	0.26	0.13	0.00	-0.38
7	50	0.18	0.55	-0.24	0.13	0.02	-0.38
8	50	0.18	0.55	-0.24	0.13	0.01	-0.38
9	50	-0.02	1.70	0.13	0.13	-0.01	-0.38
10	50	0.18	0.55	-0.22	0.26	0.00	-0.38

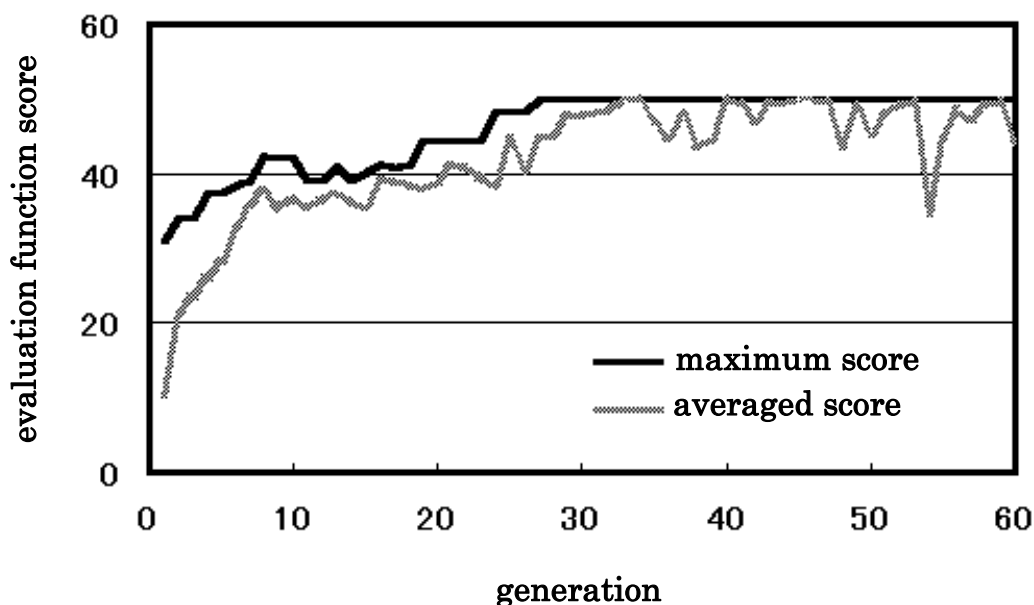


Figure 6. Evolution of evaluation function score of parameter sets during GA optimization. The total number of parameter sets is 30 in each generation. The gray line shows averaged score over 30 sets. The black line shows the maximum score in 30 sets. The upper limit of the score for a parameter set is 50.

A parent group that consisted of 30 parameter sets was created by merging 3 small groups (A, A' and B') in order to yield next generation parameter sets. Group A' and B' were made from group A and B by mutation, respectively. Each parameter in the parameter sets was changed with mutation rate 0.1. When a parameter was mutated, the mutated value was produced according to equation 3.

$$P_n = P_o \cdot R \quad \text{--- (equation 3)}$$

P_n and P_o are "new parameter value" and "old parameter value", respectively. R is a uniform random number ($0.5 < R < 1.5$). Altogether, one or two parameter values were randomly changed in any one parameter set.

Thirty new next generation parameter sets were generated. Each new set was made by hybridization of two parent sets selected from the parent set group by chance.

Figure 6 shows the result of the GA optimization. Scores for many parameter sets were 50 after 27th generation: Score 50 is the upper limit of the evaluation function. Table 2 shows optimized parameter values at 60th generation. The values of k_{Caro} were positive and those of k_{Ocar} were negative in all parameter sets. This result indicates that aromatic carbon atoms tend to avoid contact with water and that carboxyl oxygen atoms tend to contact with water.

4 Discussion

The class pattern of the Clust class we used in our YLO library has two basic classes, Atom class and Clust class; Atom class is directly included in Clust class. Therefore, every substance, such as non-organic or organic molecule, peptide chain, solvent cluster, ligand-enzyme complex and so on, is similarly dealt with as a Clust class. Furthermore, a Clust class can be combined with another Clust class by a "+" operator, creating a new Clust class; it is easy to build a large molecule system from smaller ones. By virtue of these advantages, one can deal with various compounds easily in a unified fashion.

There are other philosophies for constructing class patterns to create easy-to-use class library. Generally, in object-oriented programs, elements of the real world are often represented by the corresponding classes to make it easy to get the over-view of the program structure. We tried a class pattern having atom, residue, molecule and oligomer classes in a hierarchical manner. Although this hierarchical class pattern gave us bio-molecule images easily, we were faced with serious difficulties: (1) Non-organic molecules had to be dealt with quite differently from peptide molecules. (2) Atom class position was too deep in the hierarchical structure to obtain information about the Atom class. Our YLO library overcomes these

difficulties and makes it easy and simple to describe various kinds of molecules.

5 Conclusion

YLO library provides us with multifunctional classes for molecular simulation and structure analysis. Then the development of application programs becomes much easier owing to this library. Atom coordinate data of peptide and non-peptide molecules are read easily, as long as their structural data are written in PDB file format.

We developed several application programs for molecular structure studies, such as Monte Carlo simulator, Genetic Algorithm optimizer, crystal structure builder and so on. YLO library enables us to generate many novel and useful application programs; further construction of this library for functional expansion is in progress.

This work was supported by a grant-in-aid (to I.Y.) for Scientific Research on Priority Area (C) Genome Information Science from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

6 Availability

YLO library is distributed through a web site. The URL is:

http://www.rs.noda.tus.ac.jp/~biost/OPFU/yama/public_html/freesoft/index_e.htm

The mirror site URL is:

http://www63.tok2.com/home2/meguro/index_e.htm

When you have any questions and/or requests for YLO library, please mail to the following addresses.

tmeguro@rs.noda.tus.ac.jp or
iyamato@rs.noda.tus.ac.jp

Abbreviations

OOP,	Object-Oriented Programming
PDB,	Protein Data Bank
ASA,	Accessible Surface Area
PT method,	Parallel Transform method

References

- [1] Ponder, J. W. and Richards, F. M., *J. Comput. Chem.*, **8**, 1016-1024 (1987).
- [2] Weiner, P. K. and Kollman, P. A., *J. Comput. Chem.*, **2**, 287-303 (1981).
- [3] Komeiji, Y., Uebayasi, M., Takata, R., Shimizu, A., Itsukashi, K. and Taiji, M., *J. Comput. Chem.*, **18**, 1546-1563 (1997).
- [4] Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. and Tasumi, M., *J. Mol. Biol.*, **112**, 535-542 (1977).
- [5] Meguro, T. and Yamato, I., *J. Comput. Chem. Jpn.*, **1**, 9-22 (2002).
- [6] Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M. Jr., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W. and Kollman, P. A., *J. Am. Chem. Soc.*, **117**, 5179-5197 (1995).
- [7] Chothia, C., *Nature*, **254**, 304- 308 (1975).
- [8] Ooi, T., Oobatake, M., Nemethy, G. and Scheraga, H. A., *Proc. Natl. Acad. Sci. USA*, **84**, 3086- 3090 (1987).
- [9] Dahiyat, B. I., Sarisky, C. A. and Mayo, S. L., *J. Mol. Biol.*, **273**, 789-796 (1997).
- [10] Sayle, R.A. and Milner-White, E. J., *Trends Biochem. Sci.*, **20**, 373-375 (1995).

分子構造を立体的に解析するための 多機能オブジェクト指向クラスライブラリの開発

目黒 俊幸, 安藤 格士, 山登 一郎*

東京理科大学基礎工学部生物工学科, 〒 278-8510 千葉県野田市山崎 2641

*e-mail: iyamoto@rs.noda.tus.ac.jp

私たちは分子の構造を研究するため、様々な場面で活用できるオブジェクト指向クラスライブラリー、YLOを開発した。本ライブラリーは無機分子から生体高分子までの幅広い対象を扱えるように設計されている。

YLOはC++で書かれており、多くのクラス(プログラムユニットの一種であり、Fortranのサブルーチンに似ている)がその中に含まれている。全体的なクラス構成の様子はFigure 1に示される。各々のクラスごとに、分子の構造情報を操作するために必要な様々な機能が割り当てられている (Table 1)。

Figures 2, 3に示したように、ユーザーは複数のクラスを組み合わせることで、分子構造の解析やシミュレーションなどを行う様々なソフトウェア(“モンテカルロシミュレーター”や“遺伝的アルゴリズムを用いたパラメータ最適化プログラム”等)を作成することができる。

キーワード: オブジェクト指向プログラミング, C++, 分子シミュレーション, 無機化合物, 有機化合物, 生体高分子