Original Paper

Analytic Space Management for Drug Design Application

Takashi Maeno,† Susumu Date,† Yoshiyuki Kido†
and Shinji Shimojo††

Demands on efficient drug design have been increasing with the advancement of computing technology and bioinformatics. A variety of information technologies pertaining to drug design have been proposed recently and such technology mostly contributes to drug design research. Molecular docking simulation is a promising application for drug design, and can be realized with current information technology. However although docking simulation and the related information technology have advanced in recent years, scientists still have difficulty finding a suitable parameter set of docking simulations for accuracy of simulation. The parameter-tuning step takes a long time, and existing computing technology can hardly assist in this step. This is because the parameter-tuning step involves factors that are difficult to automate with computers. In this paper, we propose a new architecture for assisting procedures that require the decisions of scientists, especially when they need to tune parameters in a docking simulation.

1. Introduction

In-silico drug design, or drug design using computers, is becoming an important research topic among life scientists. Generally, creating a new drug is laborious and time-consuming work and thus often takes longer than 10 years. Today, many experts who are engaged in life science want and demand an information technology that helps them effectively, efficiently, and cost-reductively design a drug using computers.

Bioscientists have been fascinated by *in-silico* drug design because of two main factors. The first factor is the advancement and maturity of bioinformatics. This advancement and maturity of bioinformatics has allowed scientists to computationally simulate the behavior of proteins and other chemicals. Scientists are able to investigate their functions and chemical reactions without any *in-vitro* experiments.

The second factor is the advancement in high-performance computing and information technology. There exist many application programs that are executed in parallel in the biological research field. Recent high-performance technology including cluster and Grid computing allows us to execute time-consuming applications within a much shorter time in comparison with traditional single PC-based ways. These facts mean many biological simulations and analysis can be performed in a more efficient way than traditional computational methods.

Today, many researchers have been attempting to apply high-performance computing (HPC) technology to the drug-designing process. The most typical application to which HPC technology is applied is a docking simulation. A docking simulation is an application that checks whether a target protein and a ligand (chemical compound) chemically bind together or not, by simulating the behavior of the protein and the ligand. The effective use of the docking simulation is expected to reduce the total cost for drug design because the docking simulation screens out ligands that cannot be drugs.

In general, drug design using the docking simulation is composed of two steps: parameter tuning and screening. Scientists tend to place emphasis on the second screening step because the main purpose of using docking simulation is to perform a screening of drug candidates. Most research until today has had the tendency to focus on the screening step where hundreds of thousands of ligands have to be simulated to test for binding. The research has tried to reduce the time of the screening step.

In the screening step, scientists run docking simulations to screen drug candidates chosen from some compound databases. Each computation for the docking simulation targeting a compound can be operated independently of other computations. This means that the screening itself can be applied to parallel computing with ease. In practice, many researchers

† Graduate School of Information Science and Technology, Osaka University
†† Cybermedia Center, Osaka University

have succeeded in distributing the computational workload for docking applications using parallel computing technology. Buyya et al.,[1] report on an effective environment for screening step using Grid technology.

From a more practical viewpoint, however, the parameter-tuning step prior to the screening step is more important for efficient drug design because accurate simulation needs an accurate parameter set. In other words, how accurately scientists can describe the protein and compounds used in a docking simulation is the key to success of the simulation. Although the computational time for the screening step is easily reduced using high-performance computing technology by taking advantage of the locality of computation, reducing the time for the parameter-tuning step is difficult because of the complexity explained in this paper.

In this paper, we focus on the parameter-tuning step in using the docking application. We describe the architecture of a middleware that allows scientists to efficiently and effectively perform parameter tuning of docking simulation. Many docking simulation tools have been developed and evaluated, such as DOCK, AutoDock, and GOLD[2),3)]. DOCK[4),5)] is a well-known docking tool that is available free of charge for academic institutions and many results from DOCK are reported. DOCK has been adopted in this research, but we consider to other docking applications can be applied in our system.

The rest of this paper is organized as follows. In Section 2, we describe the target application named DOCK and explain some of the problems that exist in the DOCK application. Section 3 explains our concept and approach to solving these problems. In Section 4, we describe our implementation to realize the concept. In Section 5, we conclude.

2. DOCK and Our Focus

In our research, we use the DOCK application for the docking simulation. Scientists perform docking simulations for the screening of drug candidates. As described in Section 1, drug design using the docking simulation is composed of two steps. The first step is the parameter-tuning step and the second is the screening step. Our research focuses on the parameter-tuning step in using the DOCK application.

In this section, we first describe an overview of the DOCK application. Subsequently, we show an example of the parameter-tuning step. From the example, we derive the problems in the parameter-tuning step.

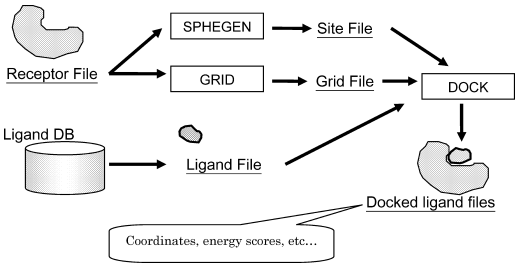


Fig. 1 Overview of DOCK application.

2.1 DOCK Application

"DOCK" is one of the tools used for the docking simulation. The DOCK application provides a suite of tools and programs for performing a docking simulation including GRID, SPHGEN and DOCK. In this paper, we use the term "DOCK application" to refer to these tools and programs including DOCK itself.

Figure 1 shows a simple docking flow using the DOCK application. Roughly, the DOCK application offers us a series of scores that indicate the strength of the binding between a target receptor and ligands. For a docking simulation between a ligand and a receptor, a ligand ("Ligand File" in Fig. 1) selected from "ligand DB" is computationally bound to a receptor ("receptor File"). Next, DOCK calculates scores for the strength of the binding between the ligand and the target receptor. By checking scores derived by the docking simulation between a target receptor and multiple ligands, experts can reduce the number of ligand candidates for the drug to the target receptor.

Before running DOCK, scientists have to prepare two files, a "Site File" and a "Grid File" generated from the target receptor file. A Site file contains a cluster of spheres that are used to determine a point where one of the ligand's atoms is located. A Grid file includes the chemical information at various grid points around active sites. An active site means the possible area where a ligand binds to a receptor. DOCK uses a site file to locate a ligand and then calculates scores using a Grid file. Scientists use two tools named SPHGEN and GRID to generate a Site file and a Grid file respectively.

To perform a docking simulation with the DOCK application, we have to complete two steps. The first step is the parameter-tuning step, and the second step is the screening step.

In the first step, scientists find a suitable parameter set for all tools in the DOCK application. The main purpose of the parameter-tuning step is to feed a suitable parameter set into the docking simulation so that the docking simulation is performed accurately enough to obtain trustworthy results for the screening step.

After the parameter-tuning step has finished, scientists carry on the screening step repeatedly to retrieve ligand files from ligand databases. The total time of the screening step can be reduced by parallel computing technology because each docking simulation can be operated separately. The time required for parameter tuning is, however, difficult to reduce because the parameter-tuning step contains the scientists' trial-and-error processes. We try to resolve these difficulties in the parameter-tuning step.

2.2 Parameter-Tuning Step

The parameter-tuning step has to be finished prior to the screening step. For the accurate screening of drug candidates, scientists have to feed suitable parameters to the DOCK application. **Figure 2** shows the whole flow of the DOCK application. Scientists have to decide all parameters to be fed to each tool. The parameter-tuning step strongly affects the screening step. Thus, whether the parameter-tuning step is correctly completed or not is directly linked to the accuracy of the screening.

To find a suitable parameter set for the docking simulation of a target receptor, scientists need to test the correctness of the docking simulation. Scientists generally test the correctness by using receptor-and-ligand pairs that are known to functionally bind in real experiments. If all parameters are set correctly, the result of the docking simulation is expected to be the same as in the real experiment.

For the parameter-tuning step, scientists' experience and knowledge of tools and target compounds take on an important role. While scientists can reduce the total computational time in the screening step with HPC technology, scientists have difficulty finding a suitable parameter set because of the inexistence of available, established systematic procedures in the parameter-tuning step. In Section 2.3 and Section 2.4, we discuss why systematic procedures do not exist and also discuss issues that need to be resolved.

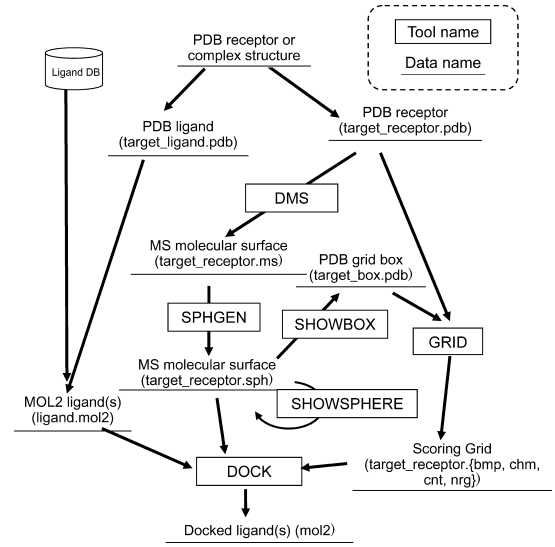


Fig. 2 Flow of DOCK application.

2.3 Example of Parameter Tuning

To discuss parameter-tuning problems, we start with an example of the parameter- tuning step using the DOCK application. Tools in the DOCK application have many parameters. In particular, DOCK has the most parameters among the DOCK application. For example, DOCK version 5.4.0 has 96 parameters. If scientists need to find a parameter set suited for DOCK, they have to consider all parameter values and their theoretical combinations. There are innumerable combinations of parameter values in such cases. Thus, in general, scientists pick a few parameters that they consider important, and then try to tune only those parameters.

Table 1 shows an example of parameter tuning for DOCK. In this example, seven parameters are picked from all 96 parameters of DOCK. These seven parameters are "Minimizing," "Orientation," "Number of Orientation," "Flexible Ligand," "Number of Flexible," "Primary Scoring Function," and "Secondary Scoring Function". The Score and RMSD fields in Table 1 are the results of DOCK. Score indicates the strength of binding between a ligand and a receptor. RMSD is used as an indicator that shows the difference in ligand shape from real experiments' coordination data.

Here we assume that scientists want to fix the values for primary scoring function and secondary scoring function, which control the calculations of docking scores. These two scoring functions correspond to the parameters named

Table 1 An Example of parameter tuning.

Minimizing	Orientation		Flexible		Primary	Secondary	Score	RMSD(Å)
No	No	—	No	—	Contact	Contact	−142	0.00
Yes	No	—	No	—	Contact	Contact	−193	2.67
Yes	Yes	1000	No	—	Contact	Contact	−203	10.29
Yes	No	—	Yes	100	Contact	Contact	−319	8.51
No	No	—	No	—	Contact	Energy	−27.93	0.00
Yes	No	—	No	—	Contact	Energy	−36.52	0.43
Yes	Yes		No	—	Contact	Energy	−36.31	0.44
Yes	No	—	Yes	100	Contact	Energy	−38.02	1.22
No	**No**	—	**No**	—	**Energy**	**Energy**	**−27.93**	**0.00**
Yes	**No**	—	**No**	—	**Energy**	**Energy**	**−36.52**	**0.43**
Yes	**Yes**	**1000**	**No**	—	**Energy**	**Energy**	**−36.31**	**0.44**
Yes	**No**	—	**Yes**	**100**	**Energy**	**Energy**	**−38.09**	**1.09**

"Primary Scoring Function" and "Secondary Scoring Function" in Table 1. The parameter of the scoring function defines the algorism that DOCK uses for calculation scores. We take up these parameters as an example, because these two parameters for scoring function are important for screening ligands. Five other parameters are also picked because they have a strong dependency with the two scoring functions. Scientists, based on their experience and knowledge, pick seven parameters from all parameters of DOCK.

The purpose of the parameter tuning in the example of Table 1 is to fix two parameters on two kinds of score functions. Scientists can fix the values of the two parameters, by checking the Score and RMSD computed for all combinations. One way to fix two parameters is to check that a Score is close to real experiments and RMSD (difference of shape from real experiments) is small. In this scenario, scientists can fix the values of two parameters, "Energy" for primary score function and "Energy" for secondary function.

After two parameters are fixed, scientists subsequently have to fix the remaining 94 parameters of DOCK. Likewise, to fix a certain parameters, scientists pick some parameters and then fix those certain parameters. Scientists have to repeat a similar procedure as shown in Table 1 to find suitable values for all 96 parameters of DOCK.

Moreover, the best parameter set changes if the target receptor used by DOCK changes. Scientists have to find a new, suitable parameter set for DOCK if a target receptor changes. Thus, every time they perform a docking simulation, they have to repeat similar but different parameter-tuning steps based on their experience.

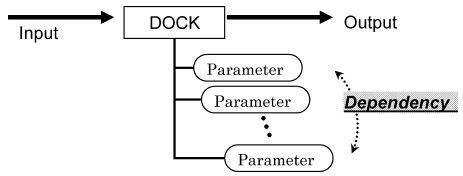
In addition, the parameter-tuning step becomes more complex when the number of tools involved increases. DOCK has other related tools, such as SPHGEN and GRID. As shown in Fig. 2, DOCK takes input files generated by SPHGEN and GRID. Thus, results of SPHGEN and GRID affect DOCK's computation strongly. In other words, the parameter tuning of SPHGEN and GRID affects the entire docking simulation as well as the parameter tuning of DOCK. In the case of the DOCK application, each tool in Fig. 2 has dependency on the other tools. Scientists utilizing DOCK application often have to tune parameters of some tools simultaneously because of dependencies among tools. The fact that scientists have to tune parameters of multiple tools simultaneously makes the parameter-tuning step in the DOCK application more complex. In this paper, we analyze these kinds of problems and then propose a solution to solve these problems.

2.4 Problems

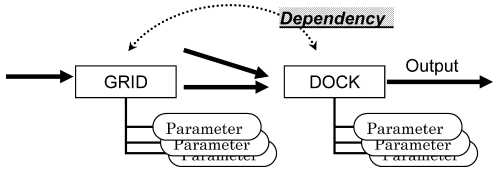
As described in Section 2.3, scientists who use the DOCK application have to perform complex parameter tuning. We believe this complex step leads to an increase of time that drug design takes as a whole. We have reached the idea that a combination of the following two factors makes the situation complex.

(1) Number of parameters

DOCK has many parameters. In fact, DOCK version 5.4.0 has more than 90 parameters. Parameter tuning of a tool with many parameters is difficult because trying all possible cases produced by a combination of values of these parameters within real time is impossible. To reduce the number of combination parameter sets, scientists have to pick parameters which depend on each other and tune them. They check the result of parameter tuning and then pick another set of parameters. In this way, scientists can find the



(a) Dependencies among Parameters



(b) Dependencies among Tools

Fig. 3 Complexity on paramter tuning.

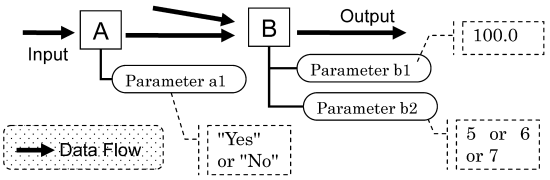


Fig. 4 Simplified model of parameter tuning.

Table 2 Possible cases (analytic space) in Fig. 4.

	Parameter a1	Parameter b1	Parameter b2
Case1	"Yes"	100.0	5
Case2	"Yes"	100.0	6
Case3	"Yes"	100.0	7
Case4	"No"	100.0	5
Case5	"No"	100.0	6
Case6	"No"	100.0	7

most appropriate parameter set while taking account of dependencies among certain parameters (**Fig. 3**-a). The same problem occurs in other cases of using tools in the DOCK application like SPHGEN and GRID.

(2) Number of tools related with DOCK
DOCK has many related tools, such as SPHGEN and GRID. All related tools directly influence the result of "DOCK" because DOCK takes input files from these related tools (as shown in Fig. 2). Scientists need to tune parameters of multiple tools simultaneously. Which tools are used depends on how they relate to each other. Thus, scientists tune parameters of these tools while considering the dependencies among all tools (Fig. 3-b).

Figure 3-b shows the dependency between GRID and DOCK. In the parameter tuning shown in Fig. 3-b, scientists also consider the parameters' dependencies as a factor. The fact that scientists have to consider dependencies of parameters and tools makes the parameter-tuning step complex.

3. Concept and Approach

In this section, we first explain a simplified model of the parameter-tuning procedure, a building block of our solution. Next, we introduce our solution to the problems in Section 2.4.

3.1 Simplified Model of Parameter Tuning

As described in Section 2.4, scientists have to deal with the complexity caused by the relationship between parameters and tools in the parameter-tuning step. To simplify this relationship, we break down the procedure for parameter tuning. **Figure 4** shows a procedure model based on tuning examples frequently seen in the parameter-tuning step. This model indicates the parameter tuning procedure using two tools named "A" and "B". The procedure model shows that tools are executed in order from A to B and that A has 1 input file and 1 output file, while B has 2 input files and 1 output file. This model is applicable to the parameter tuning procedure shown in Fig. 3-b.

Also, this model shows how the parameter tuning is performed. In this figure, tool A has a parameter named "a1," and tool B has 2 parameters named "b1" and "b2". Parameter a1 only takes the value of either "Yes" or "No" and parameter b2 takes the value of 5, 6 or 7. In this example model, if we consider all combinations, there are six possible cases (as shown in **Table 2**). If scientists need to find the best parameter set, they have to run all possible cases (six cases) and then decide which parameter set is the best. In this paper, we refer to this exploration of space consisting of all cases in the parameter-tuning step as "analytic space". In Fig. 4, "analytic space" consists of six cases (as shown in Table 2).

In the example of Fig. 4, there are only three parameters and the value of the parameters is limited. Therefore, finding the best parameter set in this example is not difficult.

However, the analytic space scientists have to explore is almost unlimited because the values of parameters change more finely and have a wide range than the example in Fig. 4. In addition, scientists need to consider parameter tuning for multiple tools.

3.2 Concept

Figure 5 shows our concept for solving the

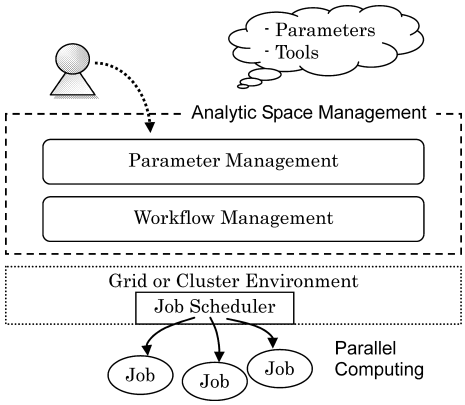


Fig. 5 Our concept.

problems described in Section 2.4. Our concept contains the following two features. We explain the effectiveness and synergy of both features below.

(1) Analytic Space Management: Separation of management for parameters and flow of the tools (also known as *workflow*).
(2) High-Performance Computing: Seamless connection from the analytic space management to existing parallel computing technology, such as Grid computing and cluster computing

First, we propose the separation of the management for parameters from the management of the flow of the tools. With this feature, scientists can concentrate on parameter tuning. In this paper, this management is called "Analytic Space Management".

As described in Section 2.4, in the parameter-tuning step, scientists have to consider the complexity of parameters and workflow simultaneously. We conclude that the complexity of the parameter-tuning steps is caused by this simultaneous management. As a result, we propose to manage parameters and tools separately.

The second feature of our concept shows how analytic space management can utilize parallel computing technology such as Grid computing and cluster computing. Parallel computing technology is a mature and commonly available technology. For example, by using local schedulers such as PBS [6] and Condor [7], scientists can perform intensive computational analysis within a short time on the cluster system composed of multiple computers. Furthermore, Grid computing technology provides us with a more advanced way to use many computers or clusters. The Globus Toolkit [8] developed by Globus Alliance [9] is a solution for the building a Grid environment composed of multiple computers and cluster systems on the Internet.

By using parallel computing technology, we can run many jobs in a short time using multiple computers. In the screening step, scientists can easily utilize such parallel computing technology. All they have to do is to decide which jobs to execute, and then the parallel computing system will run these jobs automatically.

The synergy of these two features provides scientists with a flexible and effective way for trial-and-error analytic space management because this system allows scientists to manage separately the complexity of parameter and workflow and, at the same time, utilize the benefit of HPC technology. With these two features, scientists can concentrate on what they want to perform. For example, scientists who want to perform a parameter tuning of the DOCK application can concentrate only on parameter space management. The system automatically performs workflow management such as job status management and input-output data management after scientists feed workflow data into the system. In addition, by separating the HPC environment from analytic space management, scientists can avoid the management of each tool. This means that once scientists define the analytic space they want to perform, the system can automatically execute necessary tools in the parallel computing environment. In the next section, we explain the implementation for realizing our concept.

4. Implementation

In Section 3.1, we defined the data structure for representing a procedural model. Subsequently, in Section 3.2, we described the system architecture embodying our concept as well as the implementation. In this section, we explain the implementation of our system.

4.1 Data Structure for Analytic Space Management

We use a two-step approach to complete the data structure for the analytic space management. First, we define data structure for workflow management. Second, we add data for parameter management into the workflow data structure.

The first step of data structure design for analytic space is to define workflow information in a database for workflow management. **Figure 6** shows a basic idea to provide data structure for workflows. In Fig. 6, a flow of two tools named

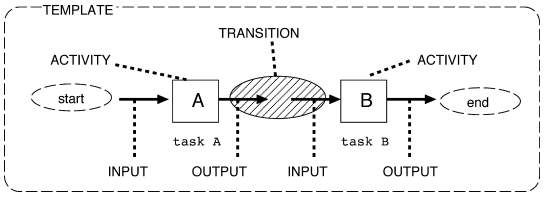


Fig. 6 Workflow template.

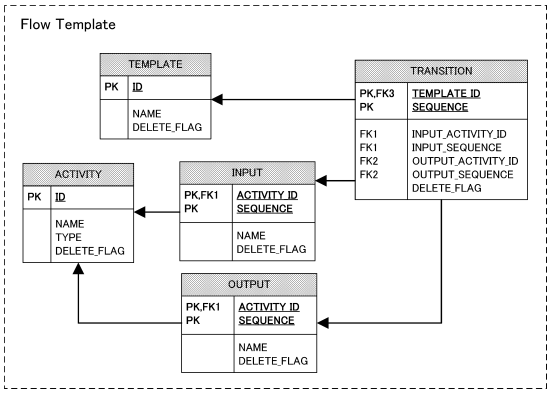


Fig. 7 Data structure for workflow template.

"A" and "B" is shown. A's output is connected to B's input. The words shown with upper case letter in Fig. 6 such as "INPUT", "OUTPUT", "ACTIVITY", "TRANSITION" and "TEMPLATE" are table names in the database. A single tool is defined as ACTIVITY and each ACTIVITY has some INPUTs and OUTPUTs. We define TRANSITION for the connection between INPUT and OUTPUT. In other words, TRANSITION manages the data flow in the workflow.

Figure 7 shows an Entity Relationship Diagram (ERD) to represent workflow templates as shown in Fig. 6. By tracing relations in each table, a system can realize workflow information such as which tools exist, how many input/output data the tool has and how the tools are combined with each other in the workflow. The system can recognize that tool A's output is connected to tool B's input and can see the transition data between A and B. Similarly, the system can realize the information of outputs and inputs to trace the relationship between ACTIVITY data, INPUT data, and OUTPUT data.

In the second step of data structure design for analytic space, we add the parameter management data structure on the workflow template data structure. Parameter management needs the data of parameter variation for each tool, variations such as the range and the fineness necessary for parameter values. The key point to parameter management data structure is utilizing the TRANSITION table, which represents data transition. We equate parameters with input files and define data structure for parameter management associated with TRANSITION. This data structure for parameter management can deal with parameter variation for a certain workflow.

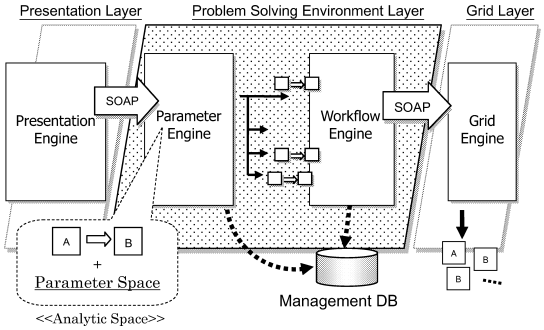


Fig. 8 Proposed system architecture.

In addition to the two steps above, we define the attribute named "DELETE_FLAG" in each table. This attribute is used to delete information from each data table. In Fig. 7, for example, if we need to delete an ACTIVITY, the value of DELETE_FLAG is set to "true". Similarly, we can delete some parameter space by setting the DELETE_FLAG of the target flow instances to "true".

4.2 System Design and Implementation

Based on our concept described in Section 3.2, we implemented our proposed system. **Figure 8** shows our proposed system architecture. The proposed system is composed of three layers. Our main implementation is the Problem Solving Layer, which takes a major role in our concept described in Section 3.2. The "Presentation Layer" is the user interface that system users (scientists) can access directly. The "Grid Layer" plays the role of executing each tool. By separating the analytic space management (Problem Solving Environment Layer) and the execution management (Grid Layer), each layer does not interfere with each other.

Two engines that exist in the Problem Solving Environment layer are the "Parameter Engine" and the "Workflow Engine". The "Parameter Engine" manages parameter tuning and the "Workflow Engine" manages workflow. These two engines correspond to the two managements shown in Fig. 5. These two engines

use a management database that holds the data structure defined in Section 4.1 Thus this database manages the procedural model defined in Section 3.1.

Our main implementation is the Problem Solving Environment layer that is the first feature of our concept described in Section 3.2. When procedural data including analytic space information is inserted, the Parameter Engine interprets the analytic space and then generates flows to be executed. The Workflow Engine receives flows from the Parameter Engine and interprets these flows. Based on flow data, the Workflow Engine decides which tools should be executed and then orders the execution of each tool to the Grid Engine in the Grid Layer. The Workflow Engine only manages the flows themselves and not how the tools are executed in the Grid Layer.

Separating the role of one engine from another is a remarkable characteristic of our implementation. By having engines with specific roles, system users can concentrate on what they want to coordinate. For example, in the parameter-tuning step of the DOCK application, system users can concentrate on the parameter-tuning step after having decided which tools to use.

4.3 Effectiveness of Analytic Space Management

We will explain the system's behavior by using the parameter-tuning example of the DOCK application. In this case, the GRID and DOCK are used for the exploration of analytic space. We assume scientists have to fix three parameters: "Grid Spacing," "Number of Orientation," and "Number of Flexible".

The parameter "Number of Orientation" indicates the maximum number of times the ligand is moved in the docking simulation. "Number of Flexible Shapes (Conformations)" determines how many times a ligand changes shape. The GRID influences the execution of DOCK. In particular, the parameter "Grid Spacing" in GRID is the most critical value for DOCK because this parameter decides the interval of the grid for representing chemical information of the target receptor. Grid points generated by the tool GRID are used to calculate scores in DOCK. If the grid spacing is small enough, DOCK can calculate accurate scores, but requires a much longer time. Because of the dependency among parameters between the two tools, scientists have to perform parameter tuning using both tools simultaneously (as in the example of **Fig. 9**).

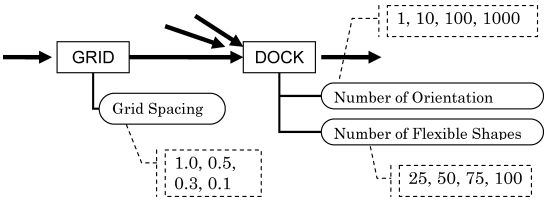


Fig. 9 Procedure model example using DOCK application.

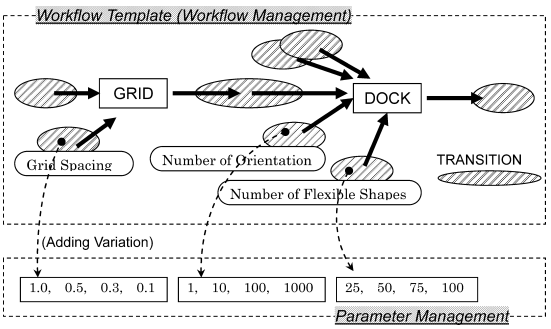


Fig. 10 Data representation for the model in Fig. 9.

In Fig. 9, scientists pick two tools and three parameters. In this example, there exists a total of 64 possible cases if we consider all possible cases. To execute the procedure in Fig. 9, scientists have to first generate the workflow template and then generate parameter variation data for parameter management.

Figure 10 shows data representation for the procedural model shown in Fig. 9. To build workflow data, scientists represent data by connecting the output of GRID to the input of DOCK. This data is stored in the database shown in Fig. 7. After building the workflow template, scientists complete the parameter variation data by linking the corresponding TRANSITION on the workflow template. Scientists can perform their parameter-tuning step by feeding this representation data into the system described in Section 4.2.

Figure 11 shows system behavior after the analytic space data shown in Fig. 10 has been built. First, representation data for analytic space is fed to the Parameter Engine. The Parameter Engine manages the analytic space derived from parameter variation and then interprets the representation data. In the data described in Fig. 10, if all cases have to be performed, the Parameter Engine generates 64 flows based on parameter management data, which contain parameter variation information.

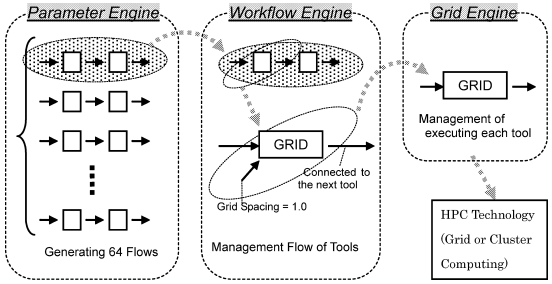


Fig. 11 System behavior.

Each of the 64 flows has a different combination of parameter value.

Second, the Parameter Engine sends 64 flows to the next Workflow Engine. The Workflow Engine manages the flow of tools in the workflow template. In this case, the Workflow Engine manages the order of GRID and DOCK and the data delivery from GRID to DOCK.

Finally, the Workflow Engine judges which tool is to be executed and checks all input data. Next, the Workflow Engine commits the execution order to the next Grid Engine. The Grid Engine then executes the tools using HPC Technology, such as the Grid or Cluster Computing.

Each engine has its own role and does not need to manage the work of other engines. For example, the Workflow Engine manages the order of tools and data transfer, but does not need to be concerned about how the analytic space is made-up (Parameter Engine's role) or how to execute tools in computers (Grid Engine's role). With this architecture, scientists can concentrate on parameter management using the Parameter Engine without being concerned with workflow and execution management.

Our main implementation is the Parameter Engine and the Workflow Engine. For now, we implement the Grid Engine as a stub. We will need to add a mechanism into the Grid Engine to access HPC technology transparently in a way reported by Klous, et al. [10]. Also, our implemented system is command-line based only. We need a user-friendly interface for an effective paramter-tuning step in the Presentation Layer (as shown in Fig. 8).

Furthermore, we designed the data structure with a "delete flag" as described in Section 3.1. By using a delete-flag mechanism, scientists can efficiently reduce the analytic space that they have to explore. The situation in Fig. 9 produces 64 flows in the Parameter Engine. For example, if scientists can judge a parameter value

Table 3 Parameter-tuning step in actual case.

Parameters a	b	c	d	RMSD (Å)	Execution Time (sec)	Number of Workflows Remained	
100	100	250	250	–	–	–	← t0
20	100	250	250	5.11	4,245	3	
50	100	250	250	4.41	4,971	2	
100	100	250	250	4.11	4,202	1	
200	100	250	250	4.11	4,178	0	← t1
100	200	250	250	4.82	14,637	2	
100	50	250	250	4.18	952	1	
100	20	250	250	4.18	298	0	← t2
100	20	100	250	4.18	271	2	
100	20	25	250	3.22	236	1	
100	20	10	250	3.69	204	0	← t3
100	20	10	100	3.72	179	2	
100	20	10	25	3.96	91	1	
100	20	10	10	5.81	90	0	← t4

Parameter a: num_anchor_orients_for_growth
Parameter b: number_confs_for_next_growth
Parameter c: simplex_anchor_max_iterations
Parameter d: simplex_grow_max_iterations
(Other parameters are set to the default values)

that does not need to be performed, scientists can dynamically cut flows including unnecessary parameter values by utilizing the delete-flag. Although we have defined the delete-flag in the data structure, we have not implemented a collaborating system with job managers in Grid or Cluster computing. This mechanism is expected to offer a more efficient way of parameter sweeping in the DOCK application.

4.4 Evaluation of Proposed System

We show an example of parameter tuning for DOCK using an actual scenario. We take up the receptor named SHP-1 [11], which is involved in the cellular signaling pathways for the proliferation and development of hematopoietic cells. We use the crystal structure whose PDB-ID is "1FBR" brought from a real experiment.

Table 3 shows the actual parameter values and results obtained in the process of parameter tuning for four parameters of DOCK in using SHP-1. Parameters "a," "b," "c," and "d" in Table 3 show the actual parameters (shown below the Table). RMSD and Execution Time are the results from DOCK. We used computers with Pentium III Processor (1.4 GHz, dual) and 1 GB memory in a cluster. The operating system is Linux kernel 2.6 and the version of DOCK we used is 5.4.0. The workflow in this experiment consists of only one tool (tool DOCK only) To simplify the explanation, we take up the example of the parameter-tuning step in which only one parameter can be tuned at once..

The criteria for finding a suitable parame-

ter set of DOCK in this case are minimizing the value of RMSD and minimizing the computational time for DOCK. The minimization of RMSD means the improvement of accuracy. If we can make the computational time small, we can perform fast other parameter-tuning step or docking simulation after this paramter-tuning step

The parameter-tuning step was performed in an order from the top to the bottom in Table 3. Using the default values as a starting point, a range of values for each parameter was tested to find the best combination of small RMSD and computational time values. Default values for the four parameters are colored with gray in Table 3.

The points at t1, t2, t3, and t4 indicate the time when we fixed the value of one of the parameters. At time t0, t1, t2, and t3, we defined a new analytic space. For example, at time t1, we fixed the value of parameter "a" to 100 and then we defined another analytic space in which the value of parameter "b" has variation (200, 50, and 20). The reason why we fixed paramter "a" at time t1 is that we assumed parameter "a" does not affect the simulation, and therefore, we chose only the default value of parameter "a". Similarly for parameter "a", we fixed parameters "b," "c," and "d" at time t2, t3, and t4, respectively, with the criteria that maximizes accuracy and minimizes time. Finally, we fixed the values of "a," "b," "c," and "d' to 100, 20, 10, and 25, respectively.

When we perform this paramter-tuning step without our proposed system, for each set of parameters, we have to set parameters and run DOCK to check its results repeatedly. With our proposed system, we only defined an analytic space that contains the workflow and how the parameter values should be moved. Once an analytic space is defined, the system executes the necessary tools to run automatically. In the paramter-tuning step of Table 3, an analytic space we defined contains only one parameter variation. However, we can define more complex analytic space in which multiple parameter values vary.

We could apply our proposed system in the case of the parameter tuning for SHP-1. In the case of Table 3, we had to redefine analytic space at time t1, t2, and t3. We assume this labor redefining process is one of the problems of our system now. We consider one of the solutions for more effective paramter tuning is using the delete-flag mechanism as described in Section 4.1. With the delete-flag mechanism, for example, we first define large analytic space and then we can reduce dynamically the deletion of analytic space (deleting no-need workflows) with our own criteria. We are confident that this delete-flag mechanism will be effective and efficient in the case of the parameter tuning in Table 3.

5. Conclusion

In our research, we focused on the parameter-tuning step using the DOCK application. We realized that the complexity in parameter tuning is caused by a combination of both parameters and tools. We proposed a system based on our concept of "analytic space management" as shown in Fig. 5. Subsequently, we defined the procedural model to be executed in this system. Also, we designed the system architecture and implemented the main part of the system to realize our concept.

Our proposed system allows scientists to perform flexible computational experiments. We verified that our proposed system can be applied with an actual paramter-tuning step. We are confident that this system can and will contribute not only to the field of drug design with computers, but also to other complex computational experiments.

In the future, we plan to implement a function into the system which can utilize the delete-flag mechanism to assist computational experiments more efficiently, such as in the dynamic deletion of workflows and jobs.

Acknowledgments This work was supported in part by the IT-program (Construction of Supercomputer Network) of the Ministry of Education, Culture, Sports, Science and Technology. We would also like to thank visiting students Mr. Marshall Levesque, Ms. Cathy Ao Chang, and Mr. Daniel Goodman from UCSD who were sent to Osaka University a NSF grant "Collaborative Undergraduate Experiences in the Pacific Rim" for their useful comments. The authors also appreciate greatly the comments and reviews from anonymous reviewers.

References

1) Buyya, R., Branson, K., Giddy, J. and Abramson, D.: The Virtual Laboratory: Enabling Molecular Modeling for Drug Design on the World Wide Grid, *Journal of Concurrency and Computation: Practice and Experience*

(*CCPE*), Vol.15, pp.1–25 (2003).
2) Bissantz, C., Folkers, G. and Rognan, D.: Protein-based virtual screening of chemical databases 1. Evaluation of different docking/scoring combinations, *Journal of Medicinal Chemistry*, Vol.43, No.25, pp.4759–4767 (2000).
3) Cummings, M.D., DesJarlais, R.L., Gibbs, A.C., Mohan, V. and Jaeger, E.P.: Comparison of Automated Docking Programs as Virtual Screening Tools, *Journal of Medicinal Chemistry*, Vol.48, pp.962–976 (2005).
4) DOCK. http://dock.compbio.ucsf.edu/
5) Ewing, T.J.A. and Kuntz, I.D.: Critical evaluation of search algorithms for automated molecular docking and database screening, *Journal of Computational Chemistry*, Vol.18, No.9, pp.1175–1189 (1998).
6) OpenPBS. http://www.openpbs.org/
7) Litzkow, M., Livny, M. and Mutka, M.: Condor — A Hunter of Idle Workstations, *Proc. 8th International Conference of Distributed Computing Systems*, pp.104–111 (1988).
8) Foster, I. and Kesselman, C.: Globus: A Toolkit-Based Grid Architecture, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, pp.259–278 (1999).
9) The Globus Alliance. http://www.globus.org/
10) Klous, S., Frey, J., Son, S.-C., Thain, D., Roy, A., Livny, M. and van den Brand, J.: Transparent Access to Grid Resources for User Software, *Concurrency and Computation: Practice and Experience*, Vol.18, Issue 7, pp.787–801 (2006).
11) Yang, J., Cheng, Z., Niu, T., Liang, X., Zhao, Z.J. and Zhou, G.W.: Structural Basis for Substrate Specificity of Protein-tyrosine Phosphatase SHP-1, *Journal of Biological Chemistry*, Vol.275, Issue 6, pp.4066–4071 (2000).

(Received July 30, 2006)
(Accepted September 10, 2006)

(Communicated by *Chiemi Watanabe*)

Takashi Maeno received his B.E. and M.E. degrees from Osaka University in 2002 and 2004, respectively. He is currently a student of Ph.D. course at the Graduate School of Information Science and Technology, Osaka University. Also, he works as a Research Assistant at the Research Center of Socionetwork Strategies, Kansai University. The focus of his research is the application of distributed computing including Grid computing to scientific and business fields, especially life scientific filed.

Susumu Date is an Associate Professor of the Graduate School of Information Science and Technology, Osaka University. He received his B.E., M.E., and Ph.D. from Osaka University in 1997, 2000, and 2002, respectively. He was an Assistant Professor at the Graduate School of Information Science and Technology, Osaka University from 2002 to 2005. In 2005, he also worked as a visiting scholar in the University of California, San Diego. His research field is computer science and his current research interests include application of Grid computing and related information technologies to life sciences. He is a member of IEEE CS and IPSJ.

Yoshiyuki Kido received his B.E. degree from Osaka Sangyo University in 1999. He had engaged in the designing and development of enterprise mission-critical system at Liberty System Ltd. from 1999 to 2002. After that, he joined Mitsui Knowledge Industry Ltd. in 2002 and has been working for the research and development for Grid systems since then. Also, he is a Ph.D. student at the Graduate School of Information Science and Technology, Osaka University from 2005. His concern of research is Grid portal and Data Grid middleware. He is a member of IEEE CS and IPSJ.

Shinji Shimojo received the M.E. and Ph.D. degrees from Osaka University in 1983 and 1986, respectively. He was an Assistant Professor with the Department of Information and Computer Sciences, Faculty of Engineering Science at Osaka University from 1986, and an Associate Professor with Computation Center from 1991 to 1998. During the period, he also worked as a visiting researcher at the University of California, Irvine for a year. He has been a Professor with Cybermedia Center (then Computation Center) at Osaka University since 1998 and currently works as the director of the Center from 2005. His current research work is focusing on a wide variety of multimedia applications, peer-to-peer communication networks, ubiquitous network systems, and Grid technologies. His achievement was awarded Osaka Science Prize in 2005. He is a member of ACM, IEEE and IEICE.