A Framework for Protein Structure Prediction on the Grid

Eduardo HUEDO[1], Ugo BASTOLLA[1], Rubén S. MONTERO[2] and Ignacio M. LLORENTE[2,1]

[1] *Centro de Astrobiología (CSIC-INTA). 28850 Torrejón de Ardoz, Spain.*

[2] *Dpto. de Arquitectura de Computadores y Automática. Universidad Complutense, 28040 Madrid, Spain.*

{huedoce,bastollau}@inta.es, {rubensm,llorente}@dacya.ucm.es

Received 15 November 2003

Abstract

The large number of protein sequences, provided by genomic projects at an increasing pace, constitutes a challenge for large scale computational studies of protein structure and thermodynamics. Grid technology is very suitable to face this challenge, since it provides a way to access the resources needed in compute and data intensive applications. In this paper, we show the procedure to adapt to the Grid an algorithm for the prediction of protein thermodynamics, using the Grid*W*ay tool. Grid*W*ay allows the resolution of large computational experiments by reacting to events dynamically generated by both the Grid and the application.

Keywords Bioinformatics, Grid Technology, Adaptive Scheduling and Execution.

§1 Introduction

Bioinformatics, which has to do with the management and analysis of huge amounts of biological data, could enormously benefit from the suitability of the Grid to execute high-throughput applications. It is foreseeable that the Grid will be soon adopted, because biological data is growing very fast, due

to the proliferation of automated high-throughput experimental techniques and organizations dedicated to Biotechnology. Therefore, the resources required to manage and analyze this data will be only accessible through the Grid.

One of the main challenges in Computational Biology concerns with the analysis of the huge amount of protein sequences provided by genomic projects at an ever increasing pace. The structure of a protein is coded in its amino acid sequence, but deciphering it has turned out to be a very difficult problem, which is still waiting for a complete solution. Nevertheless, in several cases, particularly when homologous proteins are known, computational methods can be quite reliable. At an higher level of complexity, a very significant effort is being dedicated to mapping the protein interactions, which ultimately determine many of the response properties of the cell. Also for this task, intensive computational methods are needed to complement the different experimental approaches, and analyze their results.

The aim of this paper is to present some experiences obtained on applying Grid technology to Bioinformatics. In particular, we will consider an algorithm to predict the structure and thermodynamic properties of proteins, which could be applied to several kinds of large scale studies, to demonstrate the usefulness of the Grid to build sequence-structure alignments for a large set of sequences. The main characteristics of the structure prediction algorithm are briefly described in Section 2. Then, in Section 3, we present the Grid*W*ay framework to deal with the complexity of the Grid, and we enumerate the steps needed to adapt the application to take advantage of the Grid*W*ay features. In Section 4, we show the biological problems for which experimental computational results are presented. Finally, we give some conclusions in Section 5.

§2 Prediction of Protein Structure and Thermodynamics

In the past decades, a great effort has been dedicated to the prediction of the native structure of proteins from the knowledge of their amino acid sequence. Despite promising recent progress, the accepted principle that the native state is the thermodynamic state of minimal free energy of the protein plus solvent system is still unable to allow the prediction of protein structure on purely physical grounds. The most successful methods are based on the biological principle that protein structure is very conserved during evolution. Inspired by this principle, homology modelling aims at detecting an evolutionary relationship between the

target sequence and the sequence of a protein with known structure, in order to infer the target structure by analogy. A third class, known as threading methods, combines both the evolutionary and the physical approach. Based on the observation that protein sequences can diverge through evolution to the point that their similarity is undetectable, while conserving roughly the same structure, these methods try to fit all known protein structures to the target sequences, scoring the match in terms of both sequence similarity and some simplified free energy function. Methods of this class can in principle identify even distant homologous proteins sharing the same fold as the query protein. Here we use such methods to obtain estimates of protein thermodynamics functions.

In this work we will consider an effective free energy function able to assign to the experimentally known native structure lowest energy of the whole set of candidate structures obtained aligning without gaps the target sequence with structures in the Protein Data Bank (PDB)[5, 2]. This procedure for generating candidate structures is called gapless threading. In this way, the correct structure is recognized for most of the sequences in the PDB. Exceptions are proteins with large cofactors (i.e. non-proteic molecules needed for the functioning of the protein, like the heme group in hemoglobin), which are not included in the effective energy function, small fragments, and multimeric proteins with strong inter-chain interactions. The effective energy function is able to estimate to a satisfactory accuracy the folding free energy (difference in free energy between the native state and the almost random unfolded state) of proteins whose structure is known.

We have applied the effective energy function to estimate the normalized energy gap[12, 4], a parameter involved in folding efficiency, for sets of orthologous proteins performing the same function in different organisms. This study showed that proteins of intracellular bacteria have smaller folding efficiency than the corresponding proteins of free living bacteria[20]. This result was expected from the argument that intracellular bacteria live in small populations, and natural selection is less effective in maintaining the properties of their macromolecules.

In order to use the effective energy function described above for protein structure prediction, we have to apply it to *gapped* alignments between the query sequences and the candidate structures. Gaps in the alignment represent residues that are deleted either from the sequence or from the structure in order to fit them together. This is motivated by the fact that during evolution amino acids are inserted in or deleted from protein sequences, thus spoiling the perfect

gapless alignment that two sequences had when they originated from a common ancestor. Introducing gaps increases enormously the space of candidate structures for protein structure prediction. In order to eliminate spurious matches obtained by placing a large number of gaps, one has to penalize the introduction of gaps. We therefore score an alignment $ali(A, B)$ from each residue in the protein A to the corresponding residue in the protein B with the expression:

$$Energy(Seq(A), Str(B), ali(A, B)) + G0 \cdot N_{gap} + G1 \cdot L_{gap},$$

where N_{gap} and L_{gap} are respectively the number and total length of gaps, and $G0$ and $G1$ are two parameters that have to be set by trial and error. Details on the implementation of the scoring function and its optimization will be given elsewhere.

For each structure in the PDB, our algorithm builds the gapped alignment between the target sequence and the structure which maximizes the above score. The method has been tested in the 5th round of *Critical Assessment of techniques for protein Structure Prediction* (CASP5)[*1] [1]. Although it is less efficient than homology based methods in recognizing distantly related proteins, when a close relative of the target structure is present in the PDB, even with very low sequence similarity, the algorithm recognizes it and produces a good alignment between sequence and structure. In such cases, the algorithm can be used to estimate thermodynamic parameters of the target sequence, such as the folding free energy and the normalized energy gap, and as such it has been used to confirm our previous results on the folding efficiency of proteins of different bacteria[3].

§3 The Grid*W*ay Framework

The Globus toolkit has become a *de facto* standard in Grid computing[11]. Globus services allow secure and transparent access to resources across multiple administrative domains, and serve as building blocks to implement the stages of Grid scheduling[19]: resource discovery and selection, and job preparation, submission, monitoring, migration and termination. However, the user is responsible for manually performing all the scheduling steps in order to achieve any functionality. Moreover, the Globus toolkit does not provide support for adaptive execution, required in dynamic Grid environments. In fact, one of the most challenging problems that the Grid computing community has to deal

[*1] http://PredictionCenter.llnl.gov/casp5/

with is the fact that Grids present a high fault rate and unpredictable changing conditions (dynamic resource availability, load and cost).

To overcome these limitations, we have recently developed the Grid*W*ay experimental framework[*2]. The core of the Grid*W*ay framework[15] is a personal *submission agent* that performs all scheduling stages and watches over the correct and efficient execution of jobs. Adaptation to changing conditions is achieved by dynamic rescheduling of jobs, which can lead to a job migration if it is considered feasible and worthwhile[17], when one of the following events is detected:

- A "better" resource is discovered (opportunistic migration)[17].
- The remote resource or its network connection fails.
- The submitted job is cancelled or suspended.
- Performance degradation is detected.
- The resource demands of the application change (self-migration).

The architecture of the Grid*W*ay framework is depicted in Figure 1. The user interacts with the framework through a programming or command line interface, which forwards client requests (*submit*, *kill*, *stop*, *resume...*) to the *dispatch manager*. The *dispatch manager* periodically wakes up at each *scheduling interval*, and tries to submit pending and rescheduled jobs to Grid resources. Once a job is allocated to a resource, a *submission manager* and a *performance monitor* are started to watch over its correct and efficient execution[15].

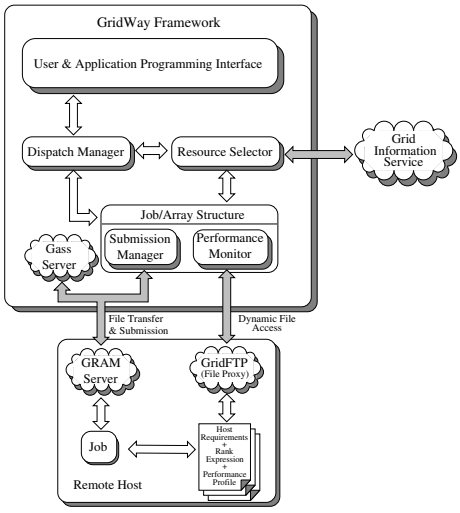

Fig. 1 Architecture of the Grid*W*ay framework.

[*2] http://asds.dacya.ucm.es

The framework has been designed to be modular, thus allowing extensibility, adaptation and improvement of its capabilities. The following modules can be set on a per job basis:

- *resource selector*, which searches for candidate resources following the application demands.
- *performance evaluator*, which evaluates the application performance.
- *prologue*, which prepares the remote system and stages input files.
- *wrapper*, which executes the actual job and returns its exit code.
- *epilogue*, which stages output files and cleans up the remote system.

The *submission agent* also provides the application with the fault tolerance capabilities needed in such a faulty environment. When an unrecoverable failure is detected, the *submission agent* retries the submission of *prologue*, *wrapper* or *epilogue* a number of times specified by the user and, when no more retries are left, it performs an action chosen by the user among two possibilities: stop the job for manually resuming it later, or automatically reschedule it.

We have developed both an API and a command line interface to interact with the *submission agent*. They allow scientists and engineers to express their computational problems in a Grid environment. The capture of the job exit code allow users to define complex jobs, where each depends on the output and exit code from the previous job. They may even involve branching, looping and spawning of subtasks, allowing the exploitation of the parallelism on the work flow of certain type of applications.

Our framework is not bounded to a specific class of applications, does not require new services, and does not necessarily require source code changes. We would like to remark that the Grid*W*ay framework does not require new system software to be installed in the Grid resources. The framework is currently functional on any Grid testbed based on Globus. We believe that is an important advantage because of socio-political issues: cooperation between different research centers, administrators, and users can be very difficult.

The management of jobs within the same department is addressed by many research and commercial systems[9]: Condor, Load Sharing Facility, Sun Grid Engine, Portable Batch System, LoadLeveler... Some of these tools, such as Sun Grid Engine Enterprise Edition[13], also allow the interconnection of multiple departments within the same administrative domain. Other tools, such as Condor Flocking[10], even allow the interconnection of multiple domains, as long as they run the same distributed resource management software. However, they

are unsuitable in computational Grids where resources are scattered across several administrative domains, each with its own security policies and distributed resource management systems.

The AppLeS project[6] has previously dealt with the concept of adaptive scheduling on Grids. AppLeS is currently focused on defining templates for characteristic applications, like APST for parameter sweep and AMWAT for master/worker applications. Also, Nimrod/G[7] dynamically optimizes the schedule to meet the user-defined deadline and budget constraints. On the other hand, the need for a nomadic migration approach for adaptive execution on Grids has been previously discussed in the context of the GrADS project[16].

3.1 Changes to Make the Application Grid-Aware

Due to the high fault rate and the dynamic rescheduling, the application must generate `restart files` in order to restart the execution from a given point in the case of automatic job migration. If these files are not provided, the job is restarted from the beginning. User-level checkpointing managed by the programmer must be implemented because system-level checkpointing is not currently possible among heterogeneous resources. The application has been modified to periodically generate an architecture independent `restart file` that stores the best candidate proteins found to that moment and the next protein in the PDB to analyze.

In order to detect performance slowdown, the application is advised to keep a `performance profile` with its performance activity in terms of application intrinsic metrics. We have modified the application to provide a `performance profile` that stores the time spent on each iteration of the algorithm, where an iteration consists of the analysis of a given number of sequences.

In order to adapt the execution of a job to its dynamic demands, the application must specify its host requirements through a `requirement expression`. The target application does not impose any requirement to the resources.

Also, in order to prioritize the resources that fulfill the requirements according to its runtime needs, the application must specify its hosts preferences through a `ranking expression`. The `ranking expression` uses a performance model to estimate the job turnaround time as the sum of the estimated execution and transfer times, derived from the performance and proximity of the candidate resources[14].

§4 Experiences

4.1 Biological Problem

In Figure 2, we compare the results of the thermodynamic calculations of the folding free energy and the normalized energy gap obtained with the present method and with a simpler one previously used[20]. In the former method, whose results are shown on the x axis, the alignment between the target sequence and the representative structure is obtained from their sequence alignment, in the spirit of homology modelling, where the alignment between query sequence and candidate structure is built maximizing their sequence similarity. In the method described here (y axis), the alignment between sequence and structure is optimized as described in Section 2.

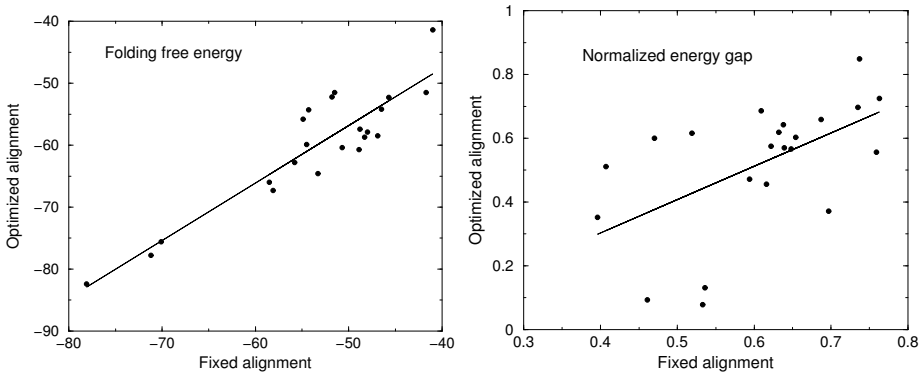

Fig. 2 Comparison of the folding free energy (left-hand chart) and the normalized energy gap (right-hand chart) estimated by using a fixed alignment (previous algorithm), and by optimizing the sequence-structure alignment (current algorithm).

The structures and the thermodynamic properties predicted with the two methods are very similar. In fact, all predicted structures are TIM barrels, and the folding free energies and normalized energy gaps predicted with the two methods correlate nicely. Nevertheless, the second method is able to estimate lower values of the folding free energy, and more reliable normalized energy gaps, due to its improved ability to explore alternative states with low energy. The three points in the right-hand chart of Figure 2, whose normalized energy gap appears significantly lower with the second method, are in fact proteins of intracellular bacteria, whose folding efficiency is expected, on evolutionary grounds, to be lower than for free living bacteria[20].

The method described above can be applied in two kinds of large scale studies. In both cases it will be necessary to build sequence-structure alignments for a large set of sequences. One study consists in predicting structure and thermodynamic properties of the proteins in a whole genome. This application still requires some improvements in the method, which fails if no structure closely related to the target is present in the PDB. We are currently working at improving the energy function and enlarging the space of candidate structures evaluated. Grid technology will make possible to apply our methods of thermodynamic predictions on a genome scale.

In the other study, we want to apply the structure prediction algorithm to a large number of families of orthologous proteins (i.e. proteins with common origin which perform the same function in different organisms), extending the study described previously. This comparative study has shown that folding efficiency is lower in proteins of intracellular bacteria than in their free-living relatives[20]. If a representative structure of one protein of the set is known, we expect from our previous experience that the algorithm recognizes it as the best structural model for each sequence, and allows to estimate its thermodynamics properties. The test presented in Section 4.2 is an example of this application, where we have applied the structure prediction algorithm to 88 sequences of the *Triose Phosphate Isomerase* enzyme expressed in different organisms. The protein assumes the TIM barrel fold, which is the most common fold in the whole space of protein structures[18].

The results of the comparative study of protein folding for this and other proteins have shown that there is a correlation between genomic features, such as the genome size and the base content of DNA, and protein folding thermodynamics[3].

4.2 Computational Results

We have performed the experiments in the UCM-CAB research testbed, depicted in Table 1. The testbed is highly heterogeneous and dynamic, and consists of three *virtual organizations*, two of them connected through a campus area network, and both connected to the other one through the RedIRIS Spanish academic network.

The experiment files consist of the executable (provided for all the resource architectures in the testbed), the PDB files (preprocessed and compressed, to reduce the transfer time), some parameter files, and the file with the sequence

Table 1 The UCM-CAB research testbed.

Host	Processors	Speed	Mem.	OS	DRMS
ursa.dacya.ucm.es	1× US-IIe	500MHz	256MB	Solaris	fork
draco.dacya.ucm.es	1× US-I	167MHz	128MB	Solaris	fork
pegasus.dacya.ucm.es	1× P4	2.4GHz	1GB	Linux	fork
solea.quim.ucm.es	2× US-II	296MHz	256MB	Solaris	fork
babieca.cab.inta.es	5× EV6	466MHz	256MB	Linux	PBS

to be analyzed. The final name of the executable file is obtained by resolving the variable `GW_ARCH` at runtime for the selected host, and the final name of the file holding the sequence to be analyzed, with the variable `GW_TASK_ID` for the current job. Input files can be local or remote (specified as a GASS, Global Access to Secondary Storage, or GridFTP URL), and both can be compressed (to be decompressed on the selected host) and declared as shared (to be stored in the GASS cache and shared by all the jobs submitted to the same resource). Most of the time spent on file transfers is due to the transferring of the PDB files, with a total file size of 40MB (12MB when compressed), but once the files are transferred to the remote resource, they are highly reused.

All the information needed to schedule and execute the job is included in the `job template` file. The whole experiment is submitted as an array job, consisting of 88 tasks where each task analyzes one sequence. Figure 3 shows the behavior when all machines in the testbed were up. Total experiment time was 7.15 hours, and the mean throughput was 12.30 jobs/hour, which supposes a mean job turnaround time of 4.88 minutes. Figure 4 shows the average transfer and execution times for each resource in the testbed for this scenario.

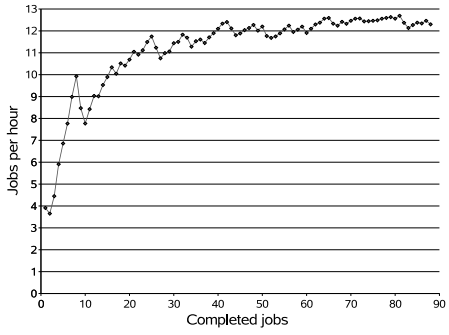

Fig. 3 Throughput when all machines in the testbed were up.

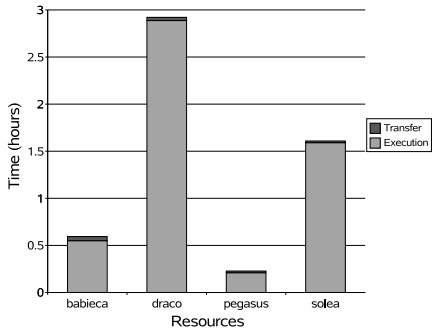

Fig. 4 Average transfer and execution times for each resource in the testbed.

Figure 5 (left-hand chart) shows the attained throughput when babieca was temporarily shutdown for maintenance. As a consequence, the *resource selector* removes babieca from the candidate resource list. Moreover, jobs already allocated to babieca were dynamically rescheduled to other Grid hosts. Total experiment time was 7.31 hours (only 9.6 minutes more than the previous experiment), and the mean throughput was 12.04 jobs/hour, which supposes a mean job turnaround time of 4.98 minutes. The mean throughput dropped from 12.71 to 10.61 jobs/hour during the period when babieca was down, but when it recovered, the throughput increased up to 12.04 jobs/hour.

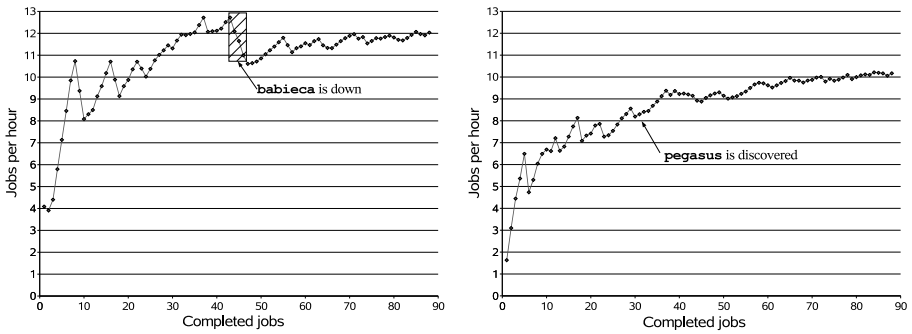

Fig. 5 Throughput when babieca was temporarily down (left-hand chart) and when pegasus was discovered in the middle of the experiment (right-hand chart).

Figure 5 (right-hand chart) shows the attained throughput when pegasus was discovered in the middle of the experiment, because it was turned on in that moment. The *dispatch manager*, wisely chose to send pending jobs to the new resource discovered rather than to migrate a running one. Total experiment time was 8.65 hours, and the mean throughput was 10.17 jobs/hour, which supposes a

mean job turnaround time of 5.9 minutes. Before discovering pegasus, the mean throughput was only 8.31 jobs/hour, and after that, it increased to 10.17.

We will next evaluate the schedule performed by the Grid*W*ay framework in the above scenarios compared to the optimum *static* schedule. This comparison can only be seen as a reference, whose main goals are to establish an upper performance bound and to highlight the relevance of adaptive scheduling in a Grid environment. The optimum Grid schedule will minimize the makespan of the application[8]:

$$\textbf{minimize}$$
$$\max\{N_i\overline{T}_i\}\ \forall i \in GR$$
$$\textbf{subject to}$$
$$\sum_{i\in GR} N_i - 88 = 0;$$
$$0 \leq N_i \leq 88\ \forall i \in GR, \qquad (1)$$

where N_i is the number of jobs executed on host i, $\overline{T}_i$ is the average job turnaround time on host i (as shown on Figure 4), and GR is the set of all Grid resources ($GR = \{draco,\ pegasus,\ solea \times 2,\ babieca \times 5\}$). Figure 6 shows the comparison. The theoretical experiment time for the optimum schedule would be 6.46 hours, which is nearly 10% better than the one obtained by Grid*W*ay when all machines were up. This is a very good result, since the worse time obtained by Grid*W*ay is mainly due to the overhead of gathering the dynamic information in order to adapt the schedule to the changing Grid conditions.

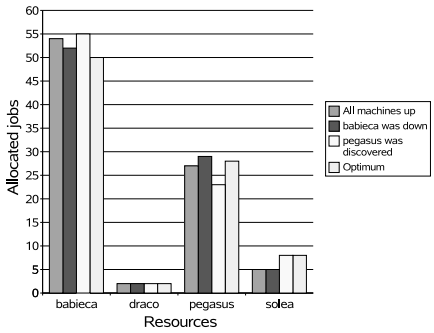

Fig. 6 Schedule performed by Grid*W*ay in different scenarios and optimum schedule statically calculated.

§5 Conclusions and Future Work

In this paper, we have presented some experiences obtained on applying Grid technology to Bioinformatics. We have shown a procedure to adapt an existing Bioinformatics application to the dynamism of the Grid, with the help of the Grid*W*ay framework. Moreover we have demonstrated the benefits of adaptive scheduling and execution to provide both performance improvement and fault tolerance in a dynamic and faulty Grid environment.

In the scope of the target application, this promising experiment show the potentiality of the Grid for the study of large numbers of protein sequences, and they suggest the possible application of this methods to the whole set of proteins in a complete microbial genome. However, this problem still requires some improvements in the structure prediction algorithm itself.

Acknowledgment

This research was supported by Ministerio de Ciencia y Tecnología (research grant TIC 2003-01321), Instituto Nacional de Técnica Aeroespacial "Esteban Terradas", and Consejo Superior de Investigaciones Científicas (I3P Programme, financed through the European Social Fund).

References

1) U. Bastolla. Sequence-Structure Alignments with the Protfinder Algorithm. In *Abstracts of the 5th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction*, December 2002. Available at http://predictioncenter.llnl.gov/casp5/doc/Abstr.doc.

2) U. Bastolla, J. Farwer, E. W. Knapp, and M. Vendruscolo. How to Guarantee Optimal Stability for Most Protein Native Structures in the Protein Data Bank. *Proteins: Structure, Function, and Genetics*, 44:79–96, 2001.

3) U. Bastolla, A. Moya, E. Viguera, and R. van Ham. Genomic Determinants of Protein Folding Thermodynamics. prepring, 2004.

4) U. Bastolla, H. E. Roman, and M. Vendruscolo. Neutral Evolution of Model Proteins: Diffusion in Sequence Space and Overdispersion. *Journal of Theoretical Biology*, 200:49–64, 1999.

5) U. Bastolla, M. Vendruscolo, and E. W. Knapp. A Statistical Mechanical Method to Optimize Energy Parameters for Protein Folding. *Proceedings of the National Academy of Sciences (PNAS) of USA*, 97:3977–3981, 2000.

6) F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):369–382, 2003.

7) R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computation Grid. In *Proceedings of the 4th IEEE International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia)*, 2000.

8) H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Proceedings of the 9th Heterogeneous Computing Workshop*, pages 349–363. IEEE Computer Society Press, 2000.

9) T. El-Ghazawi, K. Gaj, N. Alexandinis, and B. Schott. Conceptual Comparative Study of Job Management Systems. Technical report, George Mason University, February 2001. Available at http://ece.gmu.edu/lucite/reports/.

10) D. H. J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A Worldwide Flock of Condors: Load Sharing among Workstation Clusters. *Future Generation Computer Systems*, 12(1):53–65, 1996.

11) I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

12) A. M. Gutin, V. I. Abkevich, and E. I. Shakhnovich. Evolution-Like Selection of Fast-Folding Model Proteins. *Proceedings of National Academy of Sciences (PNAS) of USA*, 92:1282–1286, 1995.

13) How Sun Grid Engine, Enterprise Edition 5.3 Works. Technical report, Sun Microsystems, 2002. Available at http://www.sun.com/software/gridware/sgeee53/wp-sgeee.

14) E. Huedo, R. S. Montero, and I. M. Llorente. Experiences on Grid Resource Selection Considering Resource Proximity. In *Proceedings of the 1st European Across Grids Conference*, volume 2970 of *Lecture Notes in Computer Science*, pages 1–8. Springer-Verlag, February 2003.

15) E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Journal of Software – Practice and Experience*, 34(7):631–651, 2004.

16) G. Lanfermann, G. Allen, T. Radke, and E. Seidel. Nomadic Migration: A New Tool for Dynamic Grid Computing. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC'01)*, pages 429–430. IEEE Computer Society Press, 2001.

17) R. S. Montero, E. Huedo, and I. M. Llorente. Grid Resource Selection for Opportunistic Job Migration. In *Proceedings of the 9th International Conference on Parallel and Distributed Computing (Euro-Par 2003)*, volume 2790 of *Lecture Notes in Computer Science*, pages 366–373. Springer-Verlag, August 2003.

18) N. Nagano, C. A. Orengo, and J. M. Thornton. One Fold with Many Functions: The Evolutionary Relationships Between TIM Barrel Families Based on their Sequences, Structures and Functions. *Journal of Molecular Biology*, 321:741–765, 2002.

19) J. M. Schopf. Ten Actions when Superscheduling. Technical Report GFD-I.4, Scheduling Working Group – The Global Grid Forum, 2001.

20) R. van Ham, J. Kamerbeek, C. Palacios, C. Rausell, F. Abascal, U. Bastolla, J. M. Fernandez, L. Jimenez, M. Postigo, F.J. Silva, J. Tamames, E. Viguera,

A. Latorre, A. Valencia, F. Morán, and A. Moya. Reductive Genome Evolution in Buchnera Aphidicola. *Proceedings of National Academy of Sciences (PNAS) of USA*, 100:581–586, 2003.