Computing Molecular Potential Energy Surface with DIET

Emmanuel Jeannot
LORIA – Université Henri Poincaré, Nancy 1
Vandœuvre-les-Nancy, France
Emmanuel.Jeannot@loria.fr

Gérald Monard
ECBT – Université Henri Poincaré, Nancy 1
Vandœuvre-les-Nancy, France
Gerald.Monard@cbt.uhp-nancy.fr

Abstract

New developments in the field of theoretical chemistry require the computation of numerous Molecular Potential Energy Surfaces (PESs) to generate adequate quantum force field parameters. Because workstations alone cannot fulfill the requirements of these modern chemical advances, we present in this paper how we have tackled this problem using several up-to-date computer science technology such as grid-computing middleware, molecular databases, script interfacing, etc. An example on the optimization of semiempirical parameters for water shows the potential power of our approach and the benefit theoretical chemistry can gain with it.

1 Introduction

Computing the potential energy of a chemical system (a molecule or a set of interacting molecules) is one of the basic operations of modern theoretical chemistry [3]. The application fields are very large. For instance, it enables the design of new molecules in the field of pharmacology or the better understanding of chemical reactions in the field petrochemistry. Given a system, several conformations (*i.e.*, the relative positions of its atoms) have to be studied. A potential energy is given for each conformation. Therefore, for one system, the set of possible conformations defines a (hyper–)surface of potential energy called a PES (Potential Energy Surface). As the Schrödinger equation that defines the quantum behavior of atoms and their components (nuclei and electrons) does not admit an analytical solutions in the general case, approximation schema have to be designed to compute such PES.

Among all the approximation methods available, two methods called *ab initio* and semiempirical methods present complementary advantages. The *ab initio* method, which is derived from the Born-Oppenheimer approximation, is precise but time consuming and is limited to small systems. The semiempirical method uses parameters that need to be evaluated and is suited for large systems. It is faster than the *ab initio* method (by about two orders of magnitude) but less precise. The precision of the semiempirical methods heavily depends on the quality of the parameters. These parameters can be derived from experiments or by fitting semiempirical surfaces on *ab initio* ones. However, experimental parameters are obtained from gas phase systems and hence are not suitable for liquid phase systems that represent the vast majority of the problems studied in theoretical chemistry.

In this paper, we study and define a software architecture that is able to fit the parameters of any semiempirical method based on *ab initio* computations. The problem is the following: given a system, it is required to compute the potential energy of thousands of conformations using an *ab initio* method. Thousands of systems have to be studied which leads to months of computation on a modern PC. However, the computation of the potential energy of each conformation is independent and hence can be easily parallelized. Moreover, each conformation potential energy as well as other information (dipole, charges, etc.), need to be stored for the semiempirical parameter fitting. Finally, due to the nature of the available computing resources, we want our application to work on a large set of distributed and heterogeneous machines (called a grid).

Our contribution is the following. We have used the DIET [1] middleware to dispatch and execute each *ab initio* computation. DIET is working under the GridRPC model [4], while our problem is close to the desktop grid model. Therefore, we show that it is possible to bridge both models into a more general one. We use MySQL database(s) to store conformations to be computed as well as PES results. We provide an integrated framework that is able to fit semiempirical parameters given a set of *ab initio* PESs. Results show that this framework is able to use the full potentiality of the computing environment. On the chemical point of view, we show that the parameter fitting outperforms any previous fitting of the literature. Therefore, this proves that our application can be used in production by chemists.

The paper is organized as follow: in Section 2 the chemical problem and the fitting procedure are introduced. The software architecture (DIET middleware and MySQL database) are described in Section 3. Results and experiments are given in Section 4. In Section 5 we state our concluding remarks.

2 Problem Specifications

2.1 Potential Energy Surface in Chemistry

Theoretical chemistry mainly consists in modeling the properties (*e.g.*, the structure, the reactivity, *etc*) of atoms and molecules. The main equation describing the microscopic behavior of atoms and molecules is the Schrödinger equation (*i.e.*, the famous $H\Psi = E\Psi$ equation) which in the general case does not admit any analytical solution. Along years, theoretical chemists have carefully developed cascading approximations to the Schrödinger equation in order to obtain close analytical solutions. Among them, the Born-Oppenheimer approximation is the main one: it uses the large difference between electron and nuclei masses to suggest a clear separation between the electron and nuclei motions. As a result, the energy of a molecular system can be expressed as the sole function of its nuclei positions. This allows for the definition of *potential energy surface* (PES) which represents the variation of the potential energy of a molecular system as a function of the position of all system nuclei. The latter is called a *conformation* : the association of a set of atoms and their space coordinates.

The Born-Oppenheimer approximation is not the only approximation used by theoretical chemists to compute the energy of a molecular system. Others approximations are further used and lead to diverse computational methods. *Ab initio* and semiempirical methods are among the most popular of these. The former follows directly from the Born-Oppenheimer approximation. They are called *ab initio* methods because they need in their development only the solution of well-defined equations without requiring at any parameter. However, while they are very reliable, they are very CPU time consuming and, therefore, limit studies to very small molecular systems (*e.g.*, less than a couple of dozens of atoms). The latter semiempirical methods can be summarized as simplified *ab initio* methods for which more approximations are made. In this case, many computations are defined as unnecessary and are replaced by the evaluation of much simpler functions using *predefined atomic parameters*. This is why they are called *semiempirical* methods: they solve approximate quantum chemistry equations involving parameters determined from experiments. Overall, semiempirical calculations are far faster than *ab initio* calculations. This implies that larger molecular systems can be tackled by semiempirical systems while *ab initio* methods are confined to smaller molecular systems. However, semiempirical methods suffer from a serious drawback: the atomic empirical parameters they used have most often been determined from experimental data on isolated molecular systems (*i.e.*, from gas phase systems rather than liquid phase systems). Therefore, they are not well suited to describe interactions between molecules which are at the fundamental basis of chemical reactivity in organic and bioorganic chemistry. This leads to the following dilemma: to model interesting molecular systems, theoretical chemists are actually limited to either use less reliable semiempirical methods on large real systems, or to use trustworthy *ab initio* methods on smaller model systems that do not represent reality.

One of the possible solutions to this problem would be to use atomic semiempirical parameters coming from experimental data on interacting systems instead of isolated ones. However, the parameters needed by semiempirical equations cannot be easily derived from these experimental data (in fact, it is not actually possible to treat experimental data on interacting systems to output these parameters). Another solution is to fit semiempirical parameters not on experimental data but on *ab initio* PESs from small (then computable) interacting molecular systems. Using the transferability assumption of chemical properties[1], semiempirical parameters that can well reproduce *ab initio* PESs should be able to reproduce chemical properties of larger molecular system with an *ab initio* precision without requiring *ab initio* computations. This is the solution we present in the rest of the manuscript.

2.2 A Grid Solution

Fitting semiempirical parameters on *ab initio* PESs implies the use of an optimizing procedure. First, considering a set of molecular systems, each represented by a set of conformations, *ab initio* PESs must be generated. It consists in calculating the *ab initio* energy of each conformation of each system. Each energy calculation can be very CPU time consuming, but because *ab initio* PESs will only be used as a reference PES set, these calculations will only be performed once. Second, for a given set of semiempirical parameters, semiempirical PESs are generated. It consists in calculating the semiempirical energy of each conformation of each system with the current set of semiempirical parameters. This step is far less CPU time consuming, as compared to the computation of the *ab initio* reference state, but it will be repeated several times until convergence of the optimizing procedure. Then, a *score* is generated: it reveals the similarity between the *ab initio* PESs and the corresponding current semiempirical PESs. In the scoring

[1] This assumption is the core assumption of many thereotical methods in chemistry, like molecular mechanics for example.

function, we not only include the energy information but also other relevant chemical information data (so called *descriptors*) like the total dipole moment, the energy gradient, *etc*. Given the score for the current iteration, the optimizer can generate a new set of semiempirical parameters that serves for the next optimizing step. This procedure is repeated until convergence of the semiempirical parameters.

The adequate number of conformations by PES can be evaluated at a couple of hundred conformations. The number of PESs needed to perform a good fitting procedure can also be evaluated at a couple of hundred PESs. Therefore, the number of calculations by iteration step in the optimization procedure should range between 10^4 to 10^5 energy evaluations. A single workstation cannot handle this task alone and, because all energy calculations are independent one from another, it is possible to distribute the work on a pool of computers that together will provide enough CPU power.

Many flows of information are transfered during the fitting process. They have four different meanings: *i)* the conformation information defines which systems are involved in the fitting procedure, how many conformations by system are considered, and how atoms are positioned for each conformation; *ii)* the method information defines which solver can be required (*i.e.*, *ab initio* or semiempirical solver, including in the latter case the corresponding parameter information); *iii)* the calculation information associates one conformation and one method together to ask for the corresponding calculation; *iv)* the descriptor information gathers all chemical properties arising from the PESs building, they are used in the scoring function.

Given the necessary computer power needed to fulfill the requirements of evaluating numerous PESs in conjunction with the necessity of an efficient storage of the information, we suggest to build our application in associating modern grid technology to distribute the quantum calculations among independent computers, with a relational database to store all the information generated by the computations.

3 Software Architecture

Each computing and storage element makes up our grid. The software architecture is given in Figure 1.

We have decided to use the DIET[2] (Distributed Interactive Engineering Toolbox) [1] middleware to port our application on our grid. The DIET environment works under the gridRPC [4] model. It features a hierarchy of agents, and a set of clients and servers. Each server provides a service and is registered to an agent of the hierarchy. When a client performs a request, it asks the agent at the top of the hierarchy (called the master agent), to find the best server for its

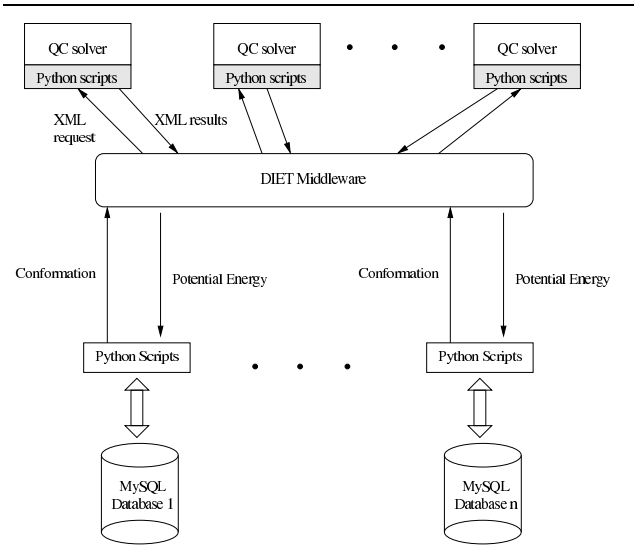

Figure 1. *Software Architecture*

request. Then, the client submits its request in a standard RPC way to the server. One of the most interesting aspects of DIET is the hierarchy of agents. This hierarchy (a logical tree), is designed in order to closely match the topology of the underlying network (each agent is deployed at a key point of the infrastructure) and to improve the scalability of the middleware (a given agent registers only a few servers). Therefore, DIET applications are able to work on very large grid environments.

The application needs to store information about the computations that have to be performed and the results. It is required to access to the information very often and very efficiently. Therefore, we have decided to use MySQL databases for storing the information. As a first approach we think that one MySQL database will provide enough performance for this application. However, if requests to a single database appear to be a bottleneck we will consider switching to a distributed database middleware such as OGSA-DAI[3].

Since the DIET API is in the C/C++ language, some python scripts are used to interface DIET with the databases. These scripts are used to extract requests and to filter and store results in the databases. Requests and results are transferred between the grid components in XML files.

On the computing side, python scripts are used to parse XML request files and call the quantum chemistry (QC) solver with the right arguments. Several solvers can be used to compute *ab initio* or semiempirical energies (*i.e.*, quantum chemical energies, or QCEs). Each time a compute

[2] `http://graal.ens-lyon.fr/DIET`

[3] `http://www.ogsadai.org.uk`

node registers to the DIET environment, it tells DIET which kind of operations it is able to perform.

3.1 MySQL Database

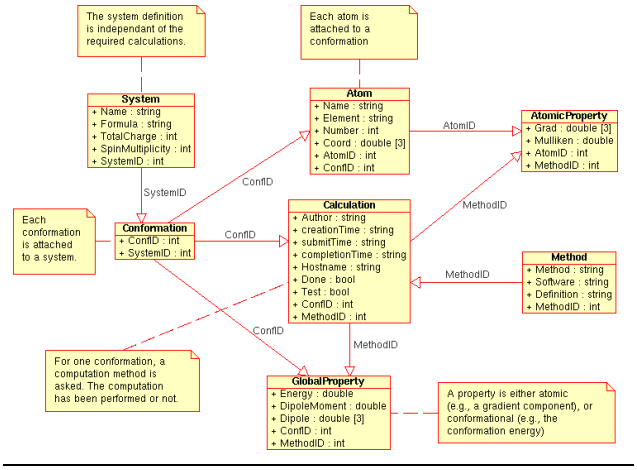

Figure 2. *MySQL Database Description*

Figure 2 shows how the database is organized:

- The conformation information is distributed in three tables: the `System` table includes system invariants like name, chemical formula, total charge, and spin multiplicity; the `Conformation` table makes the correspondence between sets of atoms belonging to the `Atom` table and the `System` table (see definition of a conformation in section 2.1); the `Atom` table stores atom relative information: the name, index and position in space of each considered atom.

- The method information is stored in the `Method` table. It includes the solver required to use the method (the `Software` field) and the parameters used by the method (the `Definition` field).

- The `Calculation` table gathers all information relative to the calculation asked by the user. It associates a conformation with a method. Three time fields are employed to respectively store the time when the user asked for the calculation (the `creationTime` field), the time when the job was submitted to the grid (the `submitTime` field) associated with the hostname of the computer performing the job (the `Hostname` field), and the time when the calculation was completed (the `completionTime` field). Two additional boolean fields are inserted: the `Done` field indicates whether the calculation is terminated **and** all chemical information arising from the calculation have been inserted in the database, and the `Test` field indicates if the calculation is relative to a temporary calculation during the fitting procedure. The latter field enables the periodic cleaning of the database.

- The descriptor information is stored in two tables: the `GlobalProperty` table is relative to all chemical properties that are system general properties like energies, dipole moments, *etc*; the `AtomicProperty` table gathers all chemical properties which are relative to atoms like atomic charges (the `Mulliken` field) or energy gradient.

3.2 DIET Middleware

The DIET middleware features three components: agents, servers, and clients. In order to compute all the PESs, we want to use as many resources as possible. If these resources are not on the same network, DIET agents can be deployed at key points of the infrastructure and, in order to bypass firewalls, a virtual private network (VPN) based on IPSEC can be used

Concerning agents and clients, there are two ways of building the application:

1. In the *push* mode each database plays the role of the client and each QC solver plays the role of a server. The database asks DIET to find a server that is able to compute the QCE of each of its conformations. Servers send back the result that is stored into the database. This is the way, the GridRPC model is designed for.

2. In the *pull* mode each QC solver plays the role of a client. It asks DIET to find a conformation for it to solve. Each database plays the role of a server. It provides conformation to compute to the client/QC solver. It provides a second service that is to store results. This mode is the way desktop-grids work.

For our application, the *push* mode has the following drawbacks. (1) *Asynchronism*: the client (the database) is blocked whenever the server is doing the request. Therefore, no more than one server can be used at a given moment. One solution is to use a non-blocking call. However, this requires extra computations to store the results at the right place in the database, and such non-blocking call is not always provided in grid middleware. (2) *Load balancing*: the middleware has to schedule the requests in order to balance them on the different servers. This requires to keep track of the current requests in the hierarchy of agents and to monitor the resources in order to find the least loaded server. (3) *server overloading*: The client has no knowledge on the number of available servers. Therefore, it can send far many requests (in an asynchronous mode), than the

number of servers. In this case, some servers can become overloaded and may collapse due to lack of memory.

The pull mode solves all these drawbacks. Each client is a computational resource and servers are databases. Therefore, each client computes one and only one QCE at a time. The load is balanced between solvers and there is no possibility to overload a given one.

Hence, we chose to use the pull mode and built our application in a desktop-grid way. We only implement this version and therefore no comparison is provided with the push mode. This shows that the gridRPC model enables to also build desktop-grid applications. We think that this bridges the two models into a more general one and shows that it is possible to design such applications with GridRPC middlewares.

We developed the server code and the client code They use the DIET API to communicate each other. The client sends its request to the database as an XML file. This file is processed by a python script on the server side and a request is generated to the data base. Two kinds of transactions are allowed. One for requesting a conformation, one for storing results in the database. The transmission of data between client and server are done by DIET which rely on CORBA. This is a GRID application as any client and server can run anywhere on any standard network (even a WAN).

Nevertheless, the pull mode requires some problems to be addressed. One is fault tolerance, the other happens when no more conformation have to be computed. The fault tolerance problem happens whenever a client asks for a conformation but fails to send back the result. In this case, the conformation is marked "submitted" and cannot be sent to an other client. This may lead to never compute this conformation. In order to avoid that, we associate to any conformation sent to a client a timeout on the server side. When the timeout is up, the database server is free to send it again to a new client. When a client has finished a conformation, it asks the database server to store the result and requests for a new conformation. If no more conformation needs to be computed, the client suspends itself for a given amount of time. This avoids the server to be flooded by conformation requests it cannot answer.

4 Results

We here present preliminary results we have obtained with our application. We have optimized MNDO semiempirical parameters [2] for the water molecule (H_2O) using *ab initio* MP2/aug-cc-pVDZ [3] calculations as a reference. Our database contains only one system: the water molecule; 300 conformations are considered, each describing the position of the two hydrogens and the oxygen of the water molecule in space. Therefore, the `System`, `Conformation`, and `Atom` tables store respectively 1, 300, and 900 entries.

The database server is represented by a Linux laptop (processor: Intel PIII 700 Mhz with 256 Mo RAM), and the clients are up to 6 Linux desktops (processor: Intel PIV 3Ghz with 1Go RAM); all computers are interconnected through a 10/100 Mbs switch.

4.1 Reference PES

Demand for the calculation of the reference PES is made by associating in the `Calculation` table the 300 conformations of the water molecule with the MP2/aug-cc-pVDZ method. Since the water molecule is very simple, a typical calculation time on a client for 1 water conformation is only 3.5 seconds. Figure 3 shows the speed-up obtained for the building of the reference PES. We see that on 6 machines the efficiency is about 90%. We believe that the scalability of the environment would be even better with more than one database and larger molecular systems.

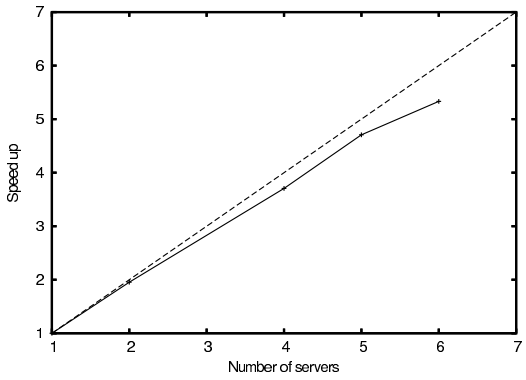

Figure 3. *Speed-up for generating the reference PES*

4.2 Fitting Procedure

The fitting procedure is performed through a simplex optimizer [5]. The initial step is represented by the MNDO semiempirical parameters from Dewar *et al.* [2]. Differences between the reference MP2/aug-cc-pVDZ PES and the MNDO PES can be seen on Figure 4 that represents respective isoenergetic contours plotted from the 300 conformations stored in the database as a function of the two parameters defining the geometry of a water molecule: a common OH distance and the $\widehat{HOH}$ angle.

After 217 iteration steps corresponding to 315 PES evaluations, the simplex optimizer has found a better set of MNDO semiempirical parameters to describe the water molecule (see Table 1). The better agreement of these new parameters with the reference PES can be seen on Figure 5.

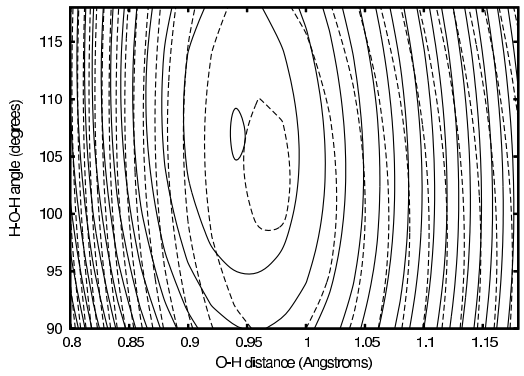

Figure 4. *Isoenergetic contour plot of the water PES as a function of the OH distance and the $\widehat{HOH}$ angle. Initial step of the fitting procedure (plain line: original MNDO parameters, dashed line: reference ab initio method)*

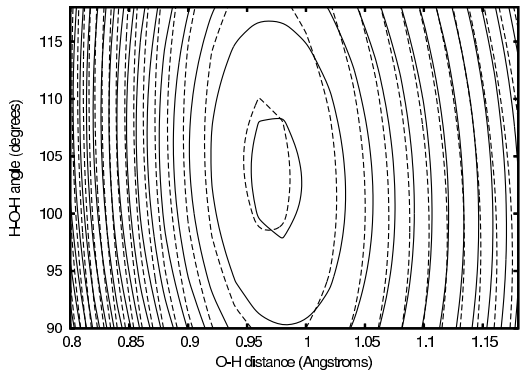

Figure 5. *Isoenergetic contour plot of the water PES as a function of the OH distance and the $\widehat{HOH}$ angle. Final step of the fitting procedure (plain line: optimized MNDO parameters, dashed line: reference ab initio method)*

This small example shows the potentiality of our application to efficiently determine new sets of semiempirical parameters to better describe molecular PESs.

5 Conclusions

Computing the potential energy of a chemical system is one of the most important operation of modern theoretical chemistry. In this paper, we have presented a complete integrated application for computing potential energy surfaces (PESs) of molecular systems in order to determine the parameters of any semiempirical method. The application is ported on a grid environment with the DIET middleware and uses MySQL databases for storing requests and results. Preliminary experiments show that the performance of the environment is good and the found results improve previous parameters of the literature. In our future work, we will port the application to a large-scale grid for computing PESs in a production mode and fitting large sets of parameters.

Atoms	Parameters	Initial Values	Optimized Values
Hydrogen	U_{ss} (eV)	-11.906276	-12.047942
	ζ (au)	1.331967	1.326597
	β (eV)	-6.989064	-7.007905
	α (Å^{-1})	2.544134	2.527682
	ρ_0 (Å)	0.560345	0.558599
Oxygen	U_{ss} (eV)	-99.643090	-102.011717
	U_{pp} (eV)	-77.797472	-77.584175
	ζ (au)	2.699905	2.839812
	β (eV)	-32.688082	-32.443890
	α (Å^{-1})	3.160604	3.155540
	D_1 (Å)	0.282894	0.283812
	D_2 (Å)	0.240043	0.242252
	ρ_0 (Å)	0.466882	0.465616
	ρ_1 (Å)	0.275822	0.276476
	ρ_2 (Å)	0.278628	0.278686
Score		0.155	0.018

Table 1. *Initial and final values of the MNDO parameters for water molecule using a simplex semiempirical parameter fitting procedure. The score line indicates how good is the fit (0 meaning prefect fitting)*

6 Acknowledgments

This work is partially funded by the ACI GRID ASP of the French Ministry of Research.

References

[1] P. Combes, F. Lombard, M. Quinson, and F. Suter. A Scalable Approach to Network Enabled Servers. In *Advances in Computing Science - ASIAN 2002*, volume 2550 of *Lecture Notes in computer Science*.

[2] M. J. S. Dewar and W. Thiel. *J. Am. Chem. Soc.*, 99(15):4899–4907, 1977.

[3] F. Jensen. *Introduction to Computational Chemistry*. John Wiley & Sons, 1999.

[4] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova. A GridRPC Model and API for End-User Applications, Dec. 2003.

[5] J. A. Nelder and R. Mead. *Comput. J.*, 7:308–313, 1965.