ABCluster Manual

Version 1.2

Jun Zhang

Institute for Theoretical Chemistry

University of Cologne

http://www.uni-koeln.de/math-nat-fak/tcchem/mitarbeiter/zhang/zhang/software-abcluster-introduction.html

ABCluster: A software for the global optimization of atomic and molecular clusters by the artificial bee colony algorithm
## Contents

### 1 Introduction to ABCluster
- 1.1 What is ABCluster? ............................................. 1
- 1.2 Contacting the Author ........................................ 1
- 1.3 Program Citation ............................................. 2
- 1.4 Copyright and Disclaimer .................................... 2
- 1.5 Components of ABCluster ..................................... 2
- 1.6 Installation of ABCluster ...................................... 3

### 2 Theory behind ABCluster
- 2.1 Choose Your Cluster and Potential Energy Function ........... 5
- 2.2 The Artificial Bee Colony Algorithm .......................... 6

### 3 The Global Optimization of Atomic Clusters
- 3.1 Introduction .................................................. 9
- 3.2 Example 1: 38 Lennard-Jones Particles ........................ 11
- 3.3 About abcinp ................................................ 11
- 3.4 Example 2: (MgO)20 .......................................... 13
- 3.5 Example 3: Silve and Copper Clusters ......................... 14
  - 3.5.1 Ag38 and Cu38 ........................................... 14
  - 3.5.2 Ag32Cu6 ................................................. 15

### 4 The Global Optimization of Rigid Molecular Clusters
- 4.1 Introduction .................................................. 17
- 4.2 Example 1: (H2O)6 ............................................. 17
- 4.3 Example 2: Li+, Na+ and Cs+ in (C6H6)6 ....................... 19
- 4.4 Example 3: (CH2=CH2)13 .................................... 20

### 5 The Global Optimization with Third-party Programs
- 5.1 Introduction .................................................. 25
- 5.2 ABCluster with Gaussian ..................................... 25
  - 5.2.1 Configuration of Gaussian ............................... 25
  - 5.2.2 Example: Al2O4+ ....................................... 26
  - 5.2.3 More About optfile ................................... 27
  - 5.2.4 Example: Au6 .......................................... 28
- 5.3 ABCluster with Other Programs ............................... 31
  - 5.3.1 How does isomer work? ................................. 31
  - 5.3.2 Example: ABCluster with ORCA on He2Ne2 ............ 32

### Bibliography

i
Chapter 1

Introduction to ABCluster

1.1 What is ABCluster?

Briefly, ABCluster searches the global as well as the local minima of atomic and molecular clusters.

ABCluster is a software developed by Dr. Jun Zhang to perform the global optimization of atomic and molecular clusters. It is an efficient and user-friendly program. It is designed in a way that non-experts as well as experts can perform a global optimization readily without knowing too much about the internal algorithm (being a black-box).

The lastest version of ABCluster, and a lot of resources can be obtained from the web page:

http://www.uni-koeln.de/math-nat-fak/tcchem/mitarbeiter/zhang/zhang/software-abcluster-introduction.html

ABCluster is written in standard C++ with some third-party C codes. The compiled binaries are available for most operating systems, and is fully parallelized with OpenMP.

The name of the program “ABCluster” comes from an algorithm that developed in 2005, i.e. the artificial bee colony algorithm. This algorithm is inspired by the foraging behaviour of bee colonies in nature, by which the bees can find the “best” nectar sources. ABCluster mimics this, trying to find the geometry of a cluster with the lowest energy. Unlike some other ones, controlling the ABC algorithm needs only three parameters. This significantly lowers users’ study curve, enabling them to quickly get the geometrical information without adjusting parameters.

The author knows that many people are very reluctant to read lengthy and tedious manuals, thus the manual is kept as short as possible. There are only five chapters: Chapter 1 gives you an introduction to ABCluster; Chapter 2 gives a minimal description of the theory behind ABCluster; Chapter 3 and 4 tell you how to use ABCluster to treat atomic and rigid molecular clusters, respectively; Chapter 5 tells you how to use ABCluster and third-party programs (e.g. a quantum chemistry program) to perform the global optimization.

1.2 Contacting the Author

If you want to get the source codes, or have any bug reports, comments, suggestions or the possibility of cooperation on ABCluster, please feel free to contact to Dr. Jun Zhang by E-mail:

ZhangJunQcc@gmail.com
1.3 Program Citation

For any published works using ABCluster please include the following reference[1]:


ABCluster is developed at Institute for Theoretical Chemistry, University of Cologne (in German: Institut für Theoretische Chemie, Universität zu Köln). The main author of ABCluster is Dr. Jun Zhang.

1.4 Copyright and Disclaimer

ABCluster is Copyright © 2015 Jun Zhang. ABCluster is free of charge to non-profit academic use. The use of ABCluster in commercial packages is not allowed without a prior written commercial license agreement.

The following codes used by ABCluster have special restrictions:

- **libLBFGS** (see [http://www.chokkan.org/software/liblbfgs](http://www.chokkan.org/software/liblbfgs))
  
  libLBFGS is the C realization of the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method[2]. It is Copyright © 2002-2014 Naoaki Okazaki. libLBFGS is distributed under the term of the MIT license.

- **Boost** (see [http://www.boost.org](http://www.boost.org))

  Boost is a set of C++ libraries. It is Copyright © 2004-2006 Joe Coder. Boost is distributed under the Boost Software License, Version 1.0.

1.5 Components of ABCluster

When you download the ABCluster and unzip it, you can get several files:

- **atom-optimizer** The component to perform the global optimization of atomic cluster.

- **rigidmol-optimizer** The component to perform the global optimization of rigid molecular cluster.

- **isomer** The component to perform the global optimization of clusters in combination with third-party programs.

- **abcinp** The component to generate the input files for atom-optimizer.

- **xyz2gaussian** The component to combine ABCluster with Gaussian.

- **gaussian2xyz** The component to combine ABCluster with Gaussian.

- **bee** Have a lot of fun ...

- **manual.pdf** The manual to ABCluster.

- **testfiles** This directory contains some test input files.
1.6 Installation of ABCluster

ABCluster does not need to be installed or configured! Please note that the current version of ABCluster does not have a graphic user interface (GUI). One must use it in the command line. You can use ABCluster by its absolute path every time, if you wish. You can also set “PATH” variable to the directory where ABCluster resides.

For Windows users: Assume you put ABCluster (atom-optimizer.exe, etc) in the directory D:\ABCluster, then open Start→Control Panel→System or Security→System, then click Advanced System Settings, choose the Environment Variables option. In the System variables list, choose Path, then click Edit. In the editbox, add “;D:\ABCluster” (Don’t forget the semicolon “;” !) at the end of the texts in it, and click OK to confirm it.

For Linux and Mac OS X users: ABCluster (atom-optimizer, etc) in the directory /home/you/bin/abcluster, then open your .bashrc. At the end of the file, add the statement “export PATH=$PATH:/home/you/bin/abcluster”, and run the command source ~/.bashrc.

Moreover, ABCluster is parallelized by OpenMP. In default it will use all the CPU cores of the machine on which it is run. If you want to change this behaviour, please set OMP_NUM_THREADS variable to the number of CPU cores you want to use.

Now ABCluster is ready, please enjoy it!
Chapter 2

Theory behind ABCluster

2.1 Choose Your Cluster and Potential Energy Function

Before using ABCluster to perform the global optimization of clusters, you should know what “type” of clusters and potential energy functions you need.

Given the geometry $X$ of a cluster, the potential energy function $U(X)$ can describe its potential energy. In principle, it can be calculated by quantum chemical methods. However, a global optimization requires a large number of energy calculations. Using quantum chemistry is often too time-consuming. Moreover, except a few cases, many clusters can be described quite well by empirical potentials. These empirical potentials have been designed and optimized for specific purpose, and are much cheaper in computational cost than the first-principle ones. Therefore, we recommend that you use ABCluster and its built-in empirical potentials to do a first exploration for the clusters, then use quantum chemistry to study some important species found by ABCluster.

1. **Ionic clusters.** If you are studying clusters containing ions, like $(\text{Na}^+ \text{Cl}^-)_{30}$, $(\text{Mg}^{2+} \text{O}^{2-})_{20}$, you should use Coulomb-Born-Mayer potentials. See Equation 3.1.1.

2. **Large Metal and alloy clusters.** For the study of something like Pt$_{38}$, Ag$_{90}$Au$_{10}$, Gupta or Sutton–Chen are your choices. The former seems to be better since more parameters are available. See Equation (3.1.11) and (3.1.13). However, for small metallic clusters, like Au$_8$, Cu$_{10}$, the two potentials are very inaccurate. One must use quantum chemistry methods to search their structures.

3. **Fullerene clusters.** The Girifalco potential is the most suitable one. See Equation (3.1.7). In this potential, the fullerene will be described as a point. If its explicit orientation is required, you will have to use non-bonded potential. See Equation (4.1.1).

4. **oxides, silicates, and silica-based glasses.** Use Coulomb–Morse–Repulsion potential. See Equation (3.1.5).

5. **Molecular clusters.** You will always use non-bonded potential. See Equation (4.1.1).

6. **Other model potentials.** ABCluster also supports some model potentials, like Morse potential. See Section (3.1).

7. For systems containing significant many-body effects, like inorganic or small metallic clusters (e.g. Al$_3$O$_7^-$, Au$_8$), it seems that only quantum chemistry can describe
it reasonably. ABCluster can combine with some quantum chemical software, e.g. Gaussian[3], ORCA[4], to compute the energies.

Once you confirm the system you need, you can choose the potentials listed above. The parameters of the potentials have to be searched from literatures, or fitted by yourself. Note that on ABCluster website, there will be a continuous updating of the parameters collected by the author. Thus if you need parameters for ABCluster, please check this website:

http://www.uni-koeln.de/math-nat-fak/tcchem/mitarbeiter/
zhang/zhang/software-abcluster-resources.html

2.2 The Artificial Bee Colony Algorithm

ABCluter uses the so-called “artificial bee colony” (ABC) algorithm to perform the global optimization. It was proposed by Karaboga in 2005[5] and soon gains a lot of studies and applications in various problems. The ABC algorithm mimics the foraging behaviour of bees. ABCluter models this behaviour by three kinds of bees: employed bees (EM), onlooker bees (OL) and scout bees (SC). A colony tries to find the best nectar as food source. To do this, EMs perform a first search; OLs based on the knowledge of all EMs perform a further research; SCs have a memory of the previous search results and decide which nectars are of low quality and then discard them. After several cycles of search the colony can find the best nectar.

Now we interpret this model to the algorithm used in ABCluter. For the global optimization of chemical cluster geometry, a trial solution $X$ is the nectar and the smoothed potential energy $U$ is its quality (a lower numeric value indicates a higher quality). The cluster is characterized by its size $N$, the estimated maximum coordinate value $L$ (see Figure 2.2.1), and of course the potential parameters. The parameters needed for the ABC algorithm are the following: the size of the population of trial solutions $SN$, the scout limit $g_{\text{limit}}$ and the maximum cycle number $g_{\text{max}}$. Here are some typical choices:

<table>
<thead>
<tr>
<th>$SN$</th>
<th>10 to 300 or larger</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g_{\text{max}}$</td>
<td>100 to 10000 or larger</td>
</tr>
<tr>
<td>$g_{\text{limit}}$</td>
<td>3 to 10</td>
</tr>
</tbody>
</table>

Figure 2.2.1: The graphic illustration of $L$.

For details of the algorithms used in ABCluter, please refer to ABCluter citation paper[1]. In the following chapters we will give some small systems as examples. In a real scientific research by ABCluter, you should remember the following tips:

- For large clusters, a better strategy is to perform several optimizations with different initial guesses and compare the final results.
- For clusters with short-ranged or multimodal interactions or with multiple interaction sites (like H$_2$O which has two hydrogen bond acceptor and two donor sites), very large $g_{\text{max}}$ and $SN$ are required to get reliable results, e.g. for some clusters with short-ranged Morse potential, $g_{\text{max}} = 15000$ and $SN = 100$ are required.

- $L$ can be 1.5 to 3 times larger than system size.

- The parameters for atomic clusters can be obtained from literatures. For molecular clusters, one can construct its force field parameters from CHARMM, OPLS or AMBER. Remember the form of equation (4.1.1) and that unit of $\epsilon$ and $\sigma$ is kJ mol$^{-1}$ and Å, respectively! With wrong units you will get absurd results. Of course, one can design his own parameters. Please visit the ABCluster website for more information.

- When you get a lot of local minima, try to pick up the useful ones by chemical intuition and study them by quantum chemistry!
Chapter 3
The Global Optimization of Atomic Clusters

3.1 Introduction

The atoms in an atomic cluster can interact with each by various kinds of potentials. Here “atom” can also mean “particle”, like an ion. ABCluster now support many kinds of potentials, including:


\[
U_{\text{CBM}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{e^2 q_i q_j}{4\pi\varepsilon_0 r_{ij}} + B_{ij} \exp\left(\frac{-r_{ij}}{\rho_{ij}}\right) \right) \tag{3.1.1}
\]

2. Lennard-Jones potential\([7]\) (LJ\(_N\)). Widely used to describe the dispersion interaction in chemistry.

\[
U_{\text{LJ}} = \sum_{i=1}^{N} \sum_{i<j}^{N} 4\epsilon_{ij} \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \tag{3.1.2}
\]

3. Coulomb–Lennard-Jones potential (CLJ\(_N\)). Includes both the Coulomb and dispersion interaction.

\[
U_{\text{LJ}} = \sum_{i=1}^{N} \sum_{i<j}^{N} 4\epsilon_{ij} \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \tag{3.1.3}
\]

4. Morse potential\([8]\) (M\(_N\)). A more advanced form than the harmonic potential for the description of the vibration of molecules.

\[
U_{\text{M}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \epsilon_{ij} \left( \exp\left(\frac{-n\beta_{ij} (r_{ij} - r_{ij}^0)}{r_{ij}}\right) - n \exp\left(-\beta_{ij} (r_{ij} - r_{ij}^0)\right) \right) \tag{3.1.4}
\]

5. Coulomb–Morse–Repulsion potential\([9]\) (CMR\(_N\)). This is very useful in the simulation of oxides, silicates, and silica-based glasses.

\[
U_{\text{CMR}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{q_i q_j}{r_{ij}} + D_{ij} \exp\left(-2\alpha_{ij} (r_{ij} - \rho_{ij})\right) \right) + 2 \exp\left(-\alpha_{ij} (r_{ij} - \rho_{ij})\right) \left( \frac{C_{ij}}{r_{ij}^{12}} \right) \tag{3.1.5}
\]
6. Z potential\[10\] (Z\textsubscript{N}). This is inspired by an earlier potential proposed by Dzugutov\[11, 12\] which was designed to study glass formation. Unlike CBM, LJ and M potentials, they contain minima as well as maxima, which make the close-packing energetically unfavourable and lead to amorphous structures.

\[
U_Z = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( a \frac{e^{r_{ij}}}{r_{ij}^3} \cos (2k_F r_{ij}) + b \left( \frac{\sigma}{r_{ij}} \right)^m + V_0 \right) \theta (r_c - r_{ij}) \tag{3.1.6}
\]

7. Girifalco\[13\] (Gf\textsubscript{N}). An effective potential of the fullerene-fullerene interaction.

\[
U_{Gi} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( -\alpha \left( \frac{1}{s_{ij} (s_{ij} - 1)^3} + \frac{1}{s_{ij} (s_{ij} + 1)^3} - \frac{2}{s_{ij}^4} \right) \right) + \beta \left( \frac{1}{s_{ij} (s_{ij} - 1)^{2g}} + \frac{1}{s_{ij} (s_{ij} + 1)^{2g}} - \frac{2}{s_{ij}^{10}} \right) \tag{3.1.7}
\]

\[
s_{ij} = \frac{r_{ij}}{2d} \tag{3.1.8}
\]

\[
\alpha = \frac{N^2 A}{12 (2d)^6} \tag{3.1.9}
\]

\[
\beta = \frac{N^2 B}{90 (2d)^{12}} \tag{3.1.10}
\]

8. Gupta potential\[14\] (G\textsubscript{N}). A very important many-body potential for modeling metals.

\[
U_G = \sum_{i=1}^{N} \left( \sum_{j=1}^{N} \sum_{j \neq i}^{N} A_{ij} \exp \left( -p_{ij} \left( \frac{r_{ij}}{d_{ij}} - 1 \right) \right) - \sqrt{\rho (r_i)} \right) \tag{3.1.11}
\]

where:

\[
\rho (r_i) = \sum_{j=1}^{N} \sum_{j \neq i}^{N} \epsilon_{ij}^2 \exp \left( -2q_{ij} \left( \frac{r_{ij}}{d_{ij}} - 1 \right) \right) \tag{3.1.12}
\]

9. Sutton–Chen potential\[15\] (SC\textsubscript{N}). Another many-body potential for modeling metals.

\[
U_{SC} = \sum_{i=1}^{N} \left( \frac{1}{2} \sum_{j=1}^{N} \sum_{j \neq i}^{N} \epsilon_{ij} \left( \frac{a_{ij}}{r_{ij}} \right)^p - \sqrt{\rho (r_i)} \right) \tag{3.1.13}
\]

where:

\[
\rho (r_i) = \sum_{j=1}^{N} \sum_{j \neq i}^{N} \epsilon_{ij}^2 \left( \frac{a_{ij}}{r_{ij}} \right)^q \tag{3.1.14}
\]

Now we will see how to perform global optimization on these clusters.
3.2 Example 1: 38 Lennard-Jones Particles

We have 38 particles interacting with Lennard-Jones potential (3.1.2), denoting as \( \text{LJ}_{38} \).

We want to get its global minimum.

Step 1: In the ABCluster distribution, go to the directory `testfiles/atomiccluster`.

Step 2: Use `abcinp` to generate the input files:

```bash
../../abcinp lj38 1 LennardJones 5.0 30 300 5 30 38 C
```

Parameters for atom 0: \( \text{sigma} \) \text{epsilon} > 1.4 1.0

This line will be explained in the next section.

Step 3: Run the global optimization:

```bash
../../atom-optimizer lj38.inp > lj38.out
```

Below is everything you get after optimization:

- `lj38.out` The main output file.
- `lj38.xyz` The global minimum in XYZ format. It can be read by e.g. VMD, CYLView.
- `lj38.gjf` The global minimum in Gaussian input format. It can be read by e.g. GaussView.
- `lj38-LM` Contains the local minima, each one having two files in XYZ and Gaussian input format, respectively. They are sorted by energy-increasing order, e.g. 0.xyz is lower in energy than 13.xyz.
- `abcluster*.xyz/gjf` The file containing the current found global minimum during the running of rigidmol-optimizer. If rigidmol-optimizer crashes before normal termination, one can use this `abcluster*.xyz` to start a new optimization.

Now, you can check the final global minimal energy in `lj38.out`:

```
29 -170.78466518
* Final Global Minimal Energy: -173.92842659
* The Global Minimum is saved as: [ lj38.(gjf/.xyz) ]
```

with the geometry in `lj38.xyz` (See Figure 3.2.1). Also, in the directory `lj38-LM`, there are 30 local minima for you.

3.3 About `abcinp`

Now we want to explain in detail how to generate the input files with `abcinp`. `abcinp` will print help information when run without any arguments:

```
abcinp
```

```bash
abcinp filename kind_of_atoms forcefield amplitude sn gmax glimit nsaves natoms1 symbol1
```

forcefield = LennardJones: \( \text{sigma} \) \text{epsilon} 
forcefield = CoulombLennardJones: \( q \) \text{sigma} \text{epsilon} 
forcefield = Morse: \text{power} \text{epsilon} \beta \rho 
forcefield = CoulombBornMayer: \text{q} \text{B} \rho 
forcefield = CoulombMorseRepulsion: \text{q} \text{D} \rho \text{C} 
forcefield = Girifalco: \text{nC A B d} 
forcefield = Dzugutov: \text{a} \text{alpha} \text{kF b} \text{sigma} \text{m} \text{rC V0} 
forcefield = Gupta: \text{A xi d p q} 
forcefield = SuttonChen: \text{p q epsilon a c} 
```
CHAPTER 3. THE GLOBAL OPTIMIZATION OF ATOMIC CLUSTERS

Figure 3.2.1: The global minimum of LJ$_{38}$.

We will interpret all the arguments:

- **filename** The unique filename. The generated input files will be named as “filename.inp”, “filename.par” and “filename1.xyz”.

- **kind_of_atoms** The number of atomic types. For example, in the case of MgO clusters, kind_of_atoms = 2.

- **forcefield** The name of force field. Available choices are:
  - LennardJones, see (3.1.2).
  - CoulombLennardJones, see (3.1.3).
  - Morse, see (3.1.4).
  - CoulombBornMayer, see (3.1.1).
  - CoulombMorseRepulsion, see (3.1.5).
  - Girifalco, see (3.1.7).
  - Dzugutov, see (3.1.6).
  - Gupta, see (3.1.11) and (3.1.12).
  - SuttonChen, see (3.1.13) and (3.1.14).

- **amplitude** The estimated size of the cluster in Å, corresponding to “$L$” mentioned in Section 2.

- **sn** The size of the population of trial solutions, corresponding to “$SN$” mentioned in Section 2. Recommended value: 10 to 300.

- **gmax** The maximum cycle number, corresponding to “$g_{max}$” mentioned in Section 2. Recommended value: 100 to 100000 or larger.

- **glimit** The scout limit, corresponding to “$g_{limit}$” mentioned in Section 2. Recommended value: 2 to 10.

- **nasves** The number of local minima to be saved.

- **natsm1 symbol1 ...** For all kind_of_atoms atoms, its number and symbol must be given. Here the “symbol” is not necessarily an element symbol like “C” or “Mg”, but can be any strings, like “CT2”.


3.4. Example 2: \((\text{Mg}^Q\text{O}^Q^-)_{20}\)

After this the program will require you to input force field parameters for different pairs of atoms. The parameters are exactly the ones used from (3.1.1) to (3.1.14) in Section 3.1.

Now we give explicit correspondence between the variables in the program input and the potential expression:

<table>
<thead>
<tr>
<th>Potential Type</th>
<th>Variable Correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lennard-Jones, (3.1.2)</td>
<td>(\sigma), (\epsilon)</td>
</tr>
<tr>
<td>Coulomb–Lennard-Jones, (3.1.3)</td>
<td>(q), (\sigma), (\epsilon)</td>
</tr>
<tr>
<td>Morse, (3.1.4)</td>
<td>(n), (\epsilon), (\beta), (r_0), (r^0)</td>
</tr>
<tr>
<td>Coulomb–Born–Mayer, (3.1.1)</td>
<td>(q), (B), (B), (\rho), (\rho)</td>
</tr>
<tr>
<td>Coulomb–Morse–Repulsion, (3.1.5)</td>
<td>(q), (D), (D), (\alpha), (\rho), (C), (C)</td>
</tr>
<tr>
<td>Girifalco, (3.1.7)</td>
<td>(\alpha), (\rho), (\beta), (r_0)</td>
</tr>
<tr>
<td>Dzugutov, (3.1.6)</td>
<td>(a), (\alpha), (k_F), (k_F), (b), (b), (\sigma), (m), (m), (r_C), (r_C), (V_0), (V_0)</td>
</tr>
<tr>
<td>Gupta, (3.1.11)</td>
<td>(A), (A), (\xi), (\xi), (d), (d), (p), (p), (q), (q)</td>
</tr>
<tr>
<td>SuttonChen, (3.1.13)</td>
<td>(p), (p), (q), (q), (\epsilon), (a), (a), (c), (c)</td>
</tr>
</tbody>
</table>

For instance, in the last section, we use the following command:

```
../../abcinp lj38 1 LennardJones 5.0 30 300 5 30 38 C
```

Parameters for atom 0: \(\sigma\), \(\epsilon\) > 1.4 1.0

This means: Perform a global optimization on a LennardJones cluster, with \(L = 5.0\), \(SN = 30\), \(g_{\text{max}} = 300\), \(g_{\text{limit}} = 5\), 30 local minima will be saved. We have 1 kind of particles, its number and symbol is 38 and C, respectively. The generated file will be named as lj38*. 

After this, you will get three files:

- lj38.inp The main input file.
- lj38.par The parameter file.
- lj38i.xyz Initial guess in XYZ format.

Open lj38.inp you will find this:

```
1 38 # number of atoms
1j38i.xyz # initial guess file; * - random guess
LennardJones # force field types
1j38.par # force field parameters
30 # population size
300 # maximal generations
5 # scout limit
5.00000000 # amplitude
1j38 # save optimized configuration to .xyz and .gjf
30 # number of LMs to be saved
```

This file can of course be modified manually. Especially, if you do not have an initial guess file, you can change Line 2 to a star *, then ABCluster will automatically generate an initial guess.

3.4 Example 2: \((\text{Mg}^Q\text{O}^Q^-)_{20}\)

In this section we will find that the parameters can change the global minima significantly. We consider \((\text{MgO})_{20}\). It can be described by Coulomb–Born–Mayer potential 3.1.1. A set of parameters is[16]:

```
38
1j38i.xyz
LennardJones
1j38.par
30
5
1j38
30
```
If we use $L = 5.0$, $SN = 100$, $g_{\text{max}} = 100$, $g_{\text{limit}} = 5$, then we can perform the global optimization:

```plaintext
../../abcinp mg20o20-q2 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: $q > +2$
Parameters for atom 1: $q > -2$
Parameters for atom-pair 0-0: $B\rho > 0.0$ 0.1
Parameters for atom-pair 0-1: $B\rho > 821.6$ 0.3242
Parameters for atom-pair 1-1: $B\rho > 22746$ 0.1490

Since we have 2 kinds of atoms, "Mg" and "O", we have to write 20 Mg 20 O to indicate that we have 20 Mg and 20 O atoms. Now perform the optimization:

```plaintext
atom-optimizer mg20o20-q2.inp > mg20o20-q2.out
```

Before looking at its geometry, what will it be if the formal charge $Q$ on Mg and O are not 2.0, but other numbers, e.g. 1.5 and 1.0? Let’s try it. Generate the input files for Mg$^{1.5+}$O$^{1.5–}$,

```plaintext
../../abcinp mg20o20-q1.5 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: $q > +1.5$
Parameters for atom 1: $q > -1.5$
Parameters for atom-pair 0-0: $B\rho > 0.0$ 0.1
Parameters for atom-pair 0-1: $B\rho > 821.6$ 0.3242
Parameters for atom-pair 1-1: $B\rho > 22746$ 0.1490

and for Mg$^{1.0+}$O$^{1.0–}$,

```plaintext
../../abcinp mg20o20-q1 2 CoulombBornMayer 5.0 100 100 5 30 20 Mg 20 O
```

Parameters for atom 0: $q > +1.0$
Parameters for atom 1: $q > -1.0$
Parameters for atom-pair 0-0: $B\rho > 0.0$ 0.1
Parameters for atom-pair 0-1: $B\rho > 821.6$ 0.3242
Parameters for atom-pair 1-1: $B\rho > 22746$ 0.1490

Run them!

```plaintext
atom-optimizer mg20o20-q1.5.inp > mg20o20-q1.5.out
atom-optimizer mg20o20-q1.inp > mg20o20-q1.out
```

Now check mg20o20-q2.xyz, mg20o20-q1.5.xyz and mg20o20-q1.xyz, you will observe that the global minimum is a sphere, tube and cuboid, respectively!

### 3.5 Example 3: Silver and Copper Clusters

#### 3.5.1 Ag\textsubscript{38} and Cu\textsubscript{38}

Assuming you get a silver and copper cluster. By mass spectroscopy you know that their formulae are Ag\textsubscript{38} and Cu\textsubscript{38}, respectively. What are their most stable structures?

The best potential for metal clusters are Gupta potential (3.1.11). From the literature[17], one can find their Gupta potential parameters:
3.5. EXAMPLE 3: SILVER AND COPPER CLUSTERS

Figure 3.4.1: The global minimum of \((\text{MgO})_{20}\).

<table>
<thead>
<tr>
<th></th>
<th>(A)</th>
<th>(\xi)</th>
<th>(d)</th>
<th>(p)</th>
<th>(q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ag-Ag</td>
<td>0.1028</td>
<td>1.1780</td>
<td>2.8885</td>
<td>10.928</td>
<td>3.1390</td>
</tr>
<tr>
<td>Cu-Cu</td>
<td>0.0855</td>
<td>1.2240</td>
<td>2.5562</td>
<td>10.960</td>
<td>2.2780</td>
</tr>
<tr>
<td>Cu-Ag</td>
<td>0.0980</td>
<td>1.2274</td>
<td>2.7224</td>
<td>10.700</td>
<td>2.8050</td>
</tr>
</tbody>
</table>

OK, now we can perform the global optimization. Let’s see \(\text{Ag}_{38}\) first.

Step 1: Use abcinp to generate the input files:

```
abcinp Ag38 1 Gupta 10 100 100 5 20 38 Ag
```

Parameters for atom-pair 0-0: \(A \xi d p q > 0.1028 1.1780 2.8885 10.928 3.1390\)

Step 2: Run the optimization:

```
nohup atom-optimizer Ag38.inp > Ag38.out 
```

Step 3: Now we examine the structure by opening the file \(\text{Ag38.xyz}\). See Figure 3.5.1.

Now we do the same for \(\text{Cu}_{38}\).

Step 1: Use abcinp to generate the input files:

```
abcinp Cu38 1 Gupta 10 100 100 5 20 38 Cu
```

Parameters for atom-pair 0-0: \(A \xi d p q > 0.0855 1.2240 2.5562 10.960 2.2780\)

Step 2: Run the optimization:

```
nohup atom-optimizer Cu38.inp > Cu38.out 
```

Step 3: Now we examine the structure by opening the file \(\text{Ag38.xyz}\). See Figure 3.5.1.

Interestingly, both are a truncated octahedron.

### 3.5.2 \(\text{Ag}_{32}\text{Cu}_6\)

However, you may get a “nanoalloy”, a mixture of silver and copper atoms. For example, the alloy is \(\text{Ag}_{32}\text{Cu}_6\). What shape will it be? Still truncated octahedron? Let’s see.

Step 1: Use abcinp to generate the input files:

```
abcinp Ag32Cu6 2 Gupta 10 3000 5000 5 20 32 Ag 6 Cu
```

Parameters for atom-pair 0-0: \(A \xi d p q > 0.1028 1.1780 2.8885 10.928 3.1390\)
Parameters for atom-pair 0-1: \(A \xi d p q > 0.0980 1.2274 2.7224 10.700 2.8050\)
Parameters for atom-pair 1-1: \(A \xi d p q > 0.0855 1.2240 2.5562 10.960 2.2780\)
Note that, for mixed alloys, the number of steps required to find the global minimum may be very large. Thus here we set very large $SN$ and $g_{\text{max}}$. During the running you can examine the current global minimum by examining the temporal file $\text{abcluster}\ast .\text{xyz}$.

Step 2: Run the optimization:

```
nohup atom-optimizer Ag32Cu6.inp > Ag32Cu6.out &
```

Step 3: Now we examine the structure by opening the file $\text{Ag32Cu6.\text{xyz}}$. See Figure 3.5.2. Amazingly, it is not a “pancake”, with the six copper atoms forming a hexagonal ring!

```
Figure 3.5.1: The global minimum of $\text{Ag}_{38}$ and $\text{Cu}_{38}$.
```

```
Figure 3.5.2: The global minimum of $\text{Ag}_{32}\text{Cu}_{6}$.
```
Chapter 4

The Global Optimization of Rigid Molecular Clusters

4.1 Introduction

Molecular clusters are very important in several field of chemistry, biology and physics. In the current version of ABCluster, the molecules in cluster are assumed to be rigid, meaning that the internal degrees of freedoms (bond lengths, bond angles, dihedral angles, etc.) are kept unchanged during the optimization. For small molecules this is a very good approximation. Thus each molecule in the cluster can be described by six external degrees of freedoms: the coordinates of its geometrical center \( R \equiv \{X,Y,Z\} \) and three Euler angles \( \Omega \equiv \{\alpha,\beta,\gamma\} \) relative to its pre-defined body-fixed coordinate system.

The potential energy of the rigid molecular cluster is assumed to be of the following form:

\[
U(Q) = \sum_{I=1}^{N} \sum_{J=1}^{N} \sum_{i \in I} \sum_{j \in J} \left( \frac{e^2 q_i q_j}{4\pi \epsilon_0 r_{ij}} + 4\epsilon_{ij} \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right)
\]  

(4.1.1)

Obviously, (4.1.1) describes the intermolecular Coulomb and Lennard-Jones interactions. This simple form enables one to compute large systems fast and quickly gain chemical insights. One can get a lot of local minima by ABCluster first, and then use high-level theory, like quantum chemistry, to further study these isomers.

The unit of \( \epsilon \) and \( \sigma \) in (4.1.1) is kJ mol\(^{-1}\) and Å, respectively. These parameters can be obtained from modern force fields, like CHARMM[18], OPLS[19] and AMBER[20]. Some molecule files suitable for ABCluster are available in the distribution in

- testfiles/rigidmolecularcluster/charmm36
- testfiles/rigidmolecularcluster/oplsaa

Also, the latest force field parameters will be available on the ABCluster website.

4.2 Example 1: \((H_2O)_6\)

We want to know the most stable structure of \((H_2O)_6\). We describe water by TIP4P model. This can be done in the following steps.
Step 1: In the ABCluster distribution, go to the directory `testfiles/rigidmolecularcluster`. Note that there is a directory `charmm36`.

Step 2: Prepare an input file named `h2o6.inp` with the following content:

```plaintext
input.cluster # cluster file name
20 # population size
20 # maximal generations
3 # scout limit
4.00000000 # amplitude
h2o6 # save optimized configuration
30 # number of LMs to be saved
```

This file is self-explained:

- Line 1: The name of the file that contains the cluster components and geometry, it will be explained in the next step.
- Line 2: The size of the population of trial solutions, corresponding to “SN” mentioned in Section 2. The larger the cluster, the larger SN is required.
- Line 3: The maximum cycle number, corresponding to “gmax” mentioned in Section 2. For such small systems 20 is sufficient. For larger ones, perhaps 100000 is required!
- Line 4: The scout limit, corresponding to “glimit” mentioned in Section 2. Usually 3 to 10 is OK.
- Line 5: The estimated size of the cluster in Å, corresponding to “L” mentioned in Section 2. For an efficient optimization of large systems, this value should be 1.5 to 3 times of the system size.
- Line 6: The name to save the results.
- Line 7: The number of local minima to be saved.

Step 3: Prepare the cluster file named `input.cluster` with the following content:

```plaintext
1
charmm36/tip4p.xyz 6
* 4.0000
```

Here:

- Line 1: The number of different components in the cluster. For \((\text{H}_2\text{O})_6\) this is 1. For \(\text{Na}^+(\text{H}_2\text{O})_3(\text{CH}_3\text{OH})_2\), this is 3.
- Line 2: The file name (with path!) containing the coordinates and force field parameters, and the number of these molecules in the cluster. If there are more than one components in the cluster, then each component should be described by a line like this.
- Line 3: Here, “* 4.0000” means generating a random initial guess with \(L = 4.0000\)

It can also have another form:
4.3. EXAMPLE 2: Li$^+$, Na$^+$ AND Cs$^+$ IN (C$_6$H$_6$)$_6$

The section title gives three cation-$\pi$ system. Let’s see what its global minimum will be.

Step 1: In the ABCluster distribution, go to the directory testfiles/rigidmolecularcluster. Note that there is a directory charmm36.

Step 2: Optimize Li$^+$(C$_6$H$_6$)$_6$. Prepare two files: li-ben.inp and li-ben.cluster:

```
1 2
charm36/tip4p.xyz 6

Water hexamer
3.5 7.0 -0.2 1.3 2.9 0.6
4.0 0.9 3.1 4.3 3.2 3.8
-1.8 4.4 1.0 0.8 0.2 0.6
4.2 3.4 3.9 1.9 0.2 3.2
3.6 4.7 2.9 3.6 1.7 1.7
5.1 5.2 -0.6 1.6 2.8 3.5
```

In ABCCluster, as long as the first line after cluster components is not of the form like "*5.000" but anything else (e.g. a title), then in the following one has to give an explicit initial guess in the form X, Y, Z, $\alpha$, $\beta$ and $\gamma$.

Step 4: Now we have h2o6.inp and input.cluster, we can run the global optimization:

```
../rigidmol-optimizer h2o6.inp > h2o6.out
```

When the optimization is finished, we get the following files.

- h2o6.out The main output file.
- h2o6-OPT.xyz The global minimum in XYZ format. It can be read by e.g. VMD, CYLView.
- h2o6-OPT.gjf The global minimum in Gaussian input format. It can be read by e.g. GaussView.
- h2o6-OPT.cluster The global minimum in ABCluster format. It can only be read by rigidmol-optimizer. Also, it can be used as the initial guess of a new optimization.
- h2o6-LM Contains the local minima, each one having three files in XYZ, Gaussian input, and ABCluster format. They are sorted by energy-increasing order, e.g. 0.xyz is lower in energy than 13.xyz.
- abcluster*.xyz/gjf/cluster The file containing the current found global minimum during the running of rigidmol-optimizer. If rigidmol-optimizer crashes before normal termination, one can use this *.cluster to start a new optimization.

Now let us see the result. The h2o6.out is similar to that generated by atomoptimizer, thus will not be explained again here. Open h2o6-OPT.xyz, we found that the global minimum is the cage isomer. Open h2o6-LM/2.xyz, we found a local minimum: the prism isomer. See Figure 4.2.1. To get a reliable conclusion on which one is more stable, please use high-level methods like quantum chemistry for further study. Of course, we can play with all 30 local minima!
Figure 4.2.1: The two minima of (H₂O)₆.

```
li-ben.cluster  # cluster file name
20               # population size
20               # maximal generations
3                # scout limit
10.00000000     # amplitude
li-ben           # save optimized configuration
30               # number of LMs to be saved
```

Then run the optimization:

```
../../rigidmol-optimizer li-ben.inp > li-ben.out
```

Step 3: Do the similar thing for Na⁺(C₆H₆)₆ and Cs⁺(C₆H₆)₆. Remember to change the file name from li* to na* or cs*, and in *.cluster, change charmm36/li.xyz to charmm36/na.xyz or charmm36/cs.xyz.

Now let’s open li-ben-OPT.xyz, na-ben-OPT.xyz and cs-ben-OPT.xyz to see their global minima, shown in Figure 4.3.1.

One can see that in the first solvation shell, Li⁺ has only 2 C₆H₆, but Na⁺ and Cs⁺ have 5. Moreover, the remaining C₆H₆ is more closer to Cs⁺ than that to Na⁺. This is obviously related to their charge density. Maybe you want to see what will happen for K⁺ or SO₄²⁻. Just try it! ABCluster can do many amazing things!

### 4.4 Example 3: (CH₂=CH₂)₁₃

Ethene is of fundamental importance in chemical industrial. Now we want to know the global minimum of (CH₂=CH₂)₁₃. However, it seems that there is no ethene in testfiles/rigidmolecularcluster/charmm36. Therefore we have to construct it by ourselves.

Step 1: Construct the geometry of CH₂=CH₂. An optimization at B3LYP/6-31G(d) level is sufficient for our purpose. The final geometry is stored in a file named c2h4.xyz in XYZ format:
4.4. EXAMPLE 3: \((\text{CH}_2=\text{CH}_2)_{13}\)

The global minima of \(\text{Li}^+(\text{C}_6\text{H}_6)_6\), \(\text{Na}^+(\text{C}_6\text{H}_6)_6\) and \(\text{Cs}^+(\text{C}_6\text{H}_6)_6\).

![Diagram of Li, Na, Cs](image)

Figure 4.3.1: The global minima of \(\text{Li}^+(\text{C}_6\text{H}_6)_6\), \(\text{Na}^+(\text{C}_6\text{H}_6)_6\) and \(\text{Cs}^+(\text{C}_6\text{H}_6)_6\).

Step 2: Construct the force field parameters of \(\text{CH}_2=\text{CH}_2\). Since in this manual we always use CHARMM force field, for consistency we also use this for ethene. You can get the latest CHARMM force field files from:

http://mackerell.umaryland.edu/charmm_ff.shtml

At the time this manual is written, the latest version is CHARMM36. After you download topar_c36_aug14.tgz and unzip it, fortunately there are parameters for ethene.

In the file topar/top_all36_cgenff.rtf you find the following words:

```
RESI ETHE 0.00 ! C2H4 ethylene, yin/adm jr.
GROUP
ATOM C1 CG2D2 -0.42 !
ATOM H11 HGA5 0.21 ! H11 H21
ATOM H12 HGA5 0.21 ! \ / 
GROUP ! C1=C2
ATOM C2 CG2D2 -0.42 ! / \ 
ATOM H21 HGA5 0.21 ! H12 H22
ATOM H22 HGA5 0.21 !
BOND C1 H11 C1 H12
DOUBLE C1 C2
BOND C2 H21 C2 H22
IC H11 C1 C2 H21 1.1036 121.37 180.00 121.37 1.1036
IC H12 C2 *C1 H11 1.1036 121.37 180.00 121.37 1.1036
IC H22 C1 *C2 H21 1.1036 121.37 180.00 121.37 1.1036
```
Take the first line ATOM C1 CG2D2 -0.42 as example, C1 indicates the corresponding atom; CG2D2 is the atomic type; -0.42 is its q parameter in (4.1.1).

To find its $\epsilon$ and $\sigma$ parameters, we check the file toppar/par_all36_cgenff.rtf, and find this:

\begin{verbatim}
CG2D2 0.0 -0.0640 2.0800 ! alkene, yin, adm jr., 12/95
\end{verbatim}

-0.0640 and 2.0800 are $\epsilon$ and $\sigma$, respectively. However, since CHARMM and ABCluster use different formulæ, we have to transform them! The transformation formulæ is:

\begin{align}
\epsilon_{\text{ABCluster}} &= \epsilon_{\text{CHARMM}} \times (-4.184) \\
\sigma_{\text{ABCluster}} &= \sigma_{\text{CHARMM}} \times 2^{\frac{5}{6}}
\end{align}

Thus its final value is:

\begin{align}
q &= -0.42 \\
\epsilon &= -0.0640 \times (-4.184) = 0.2678 \\
\sigma &= 2.0800 \times 2^{\frac{5}{6}} = 3.7061
\end{align}

The parameters of H can be found in a similar way:

\begin{align}
q &= +0.21 \\
\epsilon &= -0.0260 \times (-4.184) = 0.1088 \\
\sigma &= 1.2600 \times 2^{\frac{5}{6}} = 2.2451
\end{align}

Thus the final parameter file c2h4.xyz for CH$_2$=CH$_2$ is:

\begin{verbatim}
ethene

q epsilon (kJ/mol) sigma (Å)

-0.42 0.2678 3.7061 # CG2D2
+0.21 0.1088 2.2451 # HGA5
+0.21 0.1088 2.2451 # HGA5
-0.42 0.2678 3.7061 # CG2D2
+0.21 0.1088 2.2451 # HGA5
+0.21 0.1088 2.2451 # HGA5
\end{verbatim}

Note that there is a comment line between the geometry and parameter section. The words after # are comments.

Step 3: Now prepare the files c2h413.inp and input.cluster:

\begin{verbatim}
input.cluster # cluster file name
20 # population size
100 # maximal generations
3 # scout limit
\end{verbatim}
4.4. EXAMPLE 3: \((CH_2=CH_2)_{13}\)

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.00000000 # amplitude</td>
</tr>
<tr>
<td>2</td>
<td>c2h413 # save optimized configuration</td>
</tr>
<tr>
<td>3</td>
<td>30 # number of LMs to be saved</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>c2h4.xyz 13</td>
</tr>
<tr>
<td>6</td>
<td>* 10.0000</td>
</tr>
</tbody>
</table>

Please pay attention that in this case, \(g_{\text{max}}\) should be larger than 100 to converge.

Step 4: Run the global optimization:

```
../../rigidmol-optimizer c2h413.inp > c2h413.out
```

Now open `c2h413.out`, you can find something like this:

```
  1  2.04470195 -126.16552671 -133.24995529 ...
  2  2.92567497 -126.24112753 -138.25196792 ...
  3  3.50028571 -126.84189306 -138.36356518 ...
  4  3.41023240 -128.06552411 -138.36356515 ...
  5  3.86775258 -126.57913695 -139.41470519 ...
  6  5.85200692 -116.01432862 -140.13516658 ...
  7  5.03267295 -124.05305628 -140.13516658 ...
      ...
  52  4.65923457 -120.62871046 -140.72175322 ...
  53  3.74052721 -123.42238555 -140.72175322 ...
  54  4.87904478 -120.29869548 -140.72175322 ...
  55  4.73791669 -120.80626616 -140.72175322 ...
  56  4.58123341 -124.32091734 -140.82027375 ...
  57  5.84959151 -123.90112274 -140.82027374 ...
  58  7.10242950 -122.13716432 -140.82027374 ...
      ...
* Final Global Minimal Energy: -140.82027375
```

At 56th step the final global minimal energy is located. The final global minimum has \(C_i\) symmetry (in Figure 4.4.1):

![Figure 4.4.1: The global minimum of \((CH_2=CH_2)_{13}\).](image)
Chapter 5

The Global Optimization with Third-party Programs

5.1 Introduction

In the last two chapters, we discussed several kinds of force fields which is supported natively by ABCluster. However, for species which exhibits strong many-body effects, like inorganic clusters or small metal clusters, to the best of the author’s knowledge, no force fields can describe them reasonably. In these cases, one must use quantum chemistry instead of additive empirical force field potentials to calculate their energies. The component isomer of ABCluster is designed for this purpose. The program itself provides some simple interfaces. By using these interfaces one can perform the global optimization with energy calculated by any third-party programs, e.g. Gaussian, Gamess, VASP. Since these calculations are much more expensive than those of force fields, the ABC algorithm is simplified to increase the efficiency. We will now show how to combine the power of ABCluster and other programs.

5.2 ABCluster with Gaussian

Gaussian is one of the most popular quantum chemistry software, thus we have provided official support for it. In the distribution of ABCluster you can find two files: xyz2gaussian and gaussian2xyz. They will be used by ABCluster to compute the energy with Gaussian.

5.2.1 Configuration of Gaussian

One must do some preparations before using Gaussian, e.g., setting up some environmental variables. If you have done this you can skip this section.

For Windows users: Assume you use Gaussian09 and put it in the directory D:\G09W, then open Start→Control Panel→System or Security→System, then click Advanced System Settings, choose the Environment Variables option. In the User variables for you list, click New. In the Variable name editbox, write “GAUSS_EXEDIR”, and in the Variable value editbox, write “D:\G09W”. Then click OK to confirm it.

For Linux and Mac OS X users: Assume you use Gaussian09 and put it in the directory /home/you/g09 and you set the scratch directory for the computation as /scratch/you. then open your .bashrc. At the end of the file, add the statements:

```bash
export g09root="/home/you/g09"
export GAUSS_SCRDIR=/scratch/you
```
source $g09root/g09/bsd/g09.profile

5.2.2 Example: Al$_3$O$_4^+$

The first example is, what is the global minimum of the cluster Al$_3$O$_4^+$? We decide to use a classical method: B3LYP/6-31g(d) to calculate energy. Now we will do this global optimization.

Step 1: In the ABCluster distribution, go to the directory testfiles/qc. Create a file called mol.inp. For Windows users, the content should be:

```
1 mol          # Result file name
2 Al 3 O 4    # Symbols
3 cube 3 3 3  # Structure types
4 30          # Maximal number of calculations
5 >>>>
6 ../../../xyz2gaussian optfile $inp$ > $xxx$.gjf
7 "D:\G09W\g09.exe" $xxx$.gjf
8 ../../../gaussian2xyz $xxx$.out > $out$
9 >>>>
```

For Linux and Mac OS X users,

```
1 mol          # Result file name
2 Al 3 O 4    # Symbols
3 cube 3 3 3  # Structure types
4 30          # Maximal number of calculations
5 >>>>
6 ../../../xyz2gaussian optfile $inp$ > $xxx$.gjf
7 g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
8 ../../../gaussian2xyz $xxx$.log > $out$
9 >>>>
```

The meaning of each line is:

- Line 1: The name of the directory that stores the found local minima (LM). After optimization, all the found LMs will be stored in mol-LM.

- Line 2: The cluster components. For Al$_3$O$_4^+$, just write Al 3 O 4. For Au$_{13}$Pt$_1$, write Au 13 Pt 1.

- Line 3: For this line, one can give the type of initial guesses. Available choices are:
  - line Generate a linear initial structure.
  - ring Generate a ring initial structure.
  - sphere Generate a sphere initial structure.
  - plane 2 3 Generate a 2D initial structure from a $2 \times 3$ lattice.
  - cube 3 4 3 Generate a 3D initial structure from a $3 \times 4 \times 3$ lattice.

Note that this does not mean that the program will only give linear structures when you use line. It is just a “preference”.

- Line 4: The maximal number of calculations you want to do. Here it means isomer will do 30 calculations.
5.2. ABCLUSTER WITH GAUSSIAN

- The lines between two >>>> are the commands to call third-party programs to calculate the energy. This will be explained latter.

Step 2: Create a file named `optfile` with the following content:

```plaintext
%nproc=20
%mem=300 GB
#N B3LYP/6-31g(d) scf(xqc,novaracc) opt(MaxCycles=100)
b opt
+1 1
>>>>
>>>>
```

Note that you should modify `nproc` and `mem` for your machine.

Step 3: Run the global optimization:

```bash
nohup ../../isomer mol.inp > mol.out &
```

Note that this optimization is very expensive. During the optimization, there will be a directory named `mol-LM`, in which one can check the found local minima so far. Also, one can check the output file `mol.out`, where one can see the current energies. An example output is:

```
Will perform at most 30 calculations.
==========================================================================
#     Energy      Time   State
==========================================================================
  0   -1028.04693646  397   Succeed
  1    -899.91003266   51   Succeed
  2   -1028.05070398  380   Succeed
  3   -1027.95122573  173   Succeed
  4   -1027.98390199  377   Succeed
  5   -1027.94655101  357   Succeed
  6   -1027.84124312  260   Succeed
  7   -1027.84124312  260   Succeed
  8   -1027.84124312  260   Succeed
  9   -1027.84124312  260   Succeed
```

This means, the program is performing the 10th calculations. You can check the structure with low energy $-1028.08693656$ by the file `mol-LM/5.xyz`. See Figure 5.2.1.

Step 4: After the optimization, you can check all the structures. There 27th structure has the lowest energy $-1028.18359791$ and is the global minimum. You can check it from `27.xyz`. The structures are show in Figure 5.2.1.

5.2.3 More About `optfile`

Now we describe what `optfile` is. It is used by `xyz2gaussian` to convert a XYZ molecule into a Gaussian input file. Here is an example `optfile`. `xyz2gaussian` will put all the lines before the first `>>>>` in front of the coordinates, and all the lines between the two `>>>>` at the end of the coordinates, to generate a Gaussian input file.

In practise, you have to modify the charge, spin multiplicity, method, basis set, and other options to satisfy your requirement. Here `scf(xqc, novaracc)` is recommended
CHAPTER 5. THE GLOBAL OPTIMIZATION WITH THIRD-PARTY PROGRAMS

\[ -1028.08693656 \quad -1028.18359791 \]

Figure 5.2.1: One local and global minimum of \( \text{Al}_3\text{O}_4^+ \).

since the initial guess can be very distorted, and this option can guarantee the convergence of the SCF calculations. For their meanings, please refer to the Gaussian manual.

5.2.4 Example: \( \text{Au}_6 \)

Now we want to know the global minimum of \( \text{Au}_6 \). Maybe you are thinking about the Gupta or Sutton–Chen potential. Unfortunately, for small metallic clusters, both potentials are very inaccurate\[21\]. Thus quantum chemistry must be used. Also, gold “enjoys” very strong relativistic effects. Thus one should use a suitable pseudopotential to describe gold. A good choice is the ECP60MDF. This pseudopotential can be obtained from http://www.tc.uni-koeln.de/PP/clickpse.en.html.

Step 1: In the ABCluster distribution, go to the directory testfiles/qc. Create a file called mol.inp. For Windows users, the content should be:

```
1 mol  # Result file name
2 Au 6  # Symbols
3 plane 4 3  # Structure types
4 30  # Maximal number of calculations
5 >>>>
6 ../../../xyz2gaussian optfile $inp$ > $xxx$.gjf
7 "D:\G09W\g09.exe" $xxx$.gjf
8 ../../../gaussian2xyz $xxx$.out > $out$
9 >>>>
```

For Linux and Mac OS X users,

```
1 mol  # Result file name
2 Au 6  # Symbols
3 plane 4 3  # Structure types
4 30  # Maximal number of calculations
5 >>>>
6 ../../../xyz2gaussian optfile $inp$ > $xxx$.gjf
7 g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
8 ../../../gaussian2xyz $xxx$.log > $out$
9 >>>>
```

Since I know that the global minimum of small gold clusters should be plane, I use plane 4 3 for you to save time. In practice, however, when you do not know what the global minimum should be, you’d better use cube to search more kinds of structures.
Step 2: Create a file named `optfile` with the following content:

```plaintext
%nproc=20
%mem=400GB
#N TPSSTPSS/GENECP scf(xqc,novaracc) opt(Cartesian,
    MaxCycles=100)
b opt
0 1
>>>>
Au 0
S 8 1.00
  0.38000800E+02  0.20009000E-01
  0.23972500E+02 -0.15167900E+00
  0.15218200E+02  0.36396000E+00
  0.55399900E+01 -0.82132600E+00
  0.13855100E+01  0.93664100E+00
  0.64246100E+00  0.42352700E+00
  0.15649600E+00  0.16250000E-01
  0.54910000E-01 -0.97800000E-03
S 8 1.00
  0.38000800E+02 -0.53040000E-02
  0.23972500E+02  0.46318000E-01
  0.15218200E+02 -0.11994000E+00
  0.55399900E+01  0.30406200E+00
  0.13855100E+01 -0.49450100E+00
  0.64246100E+00 -0.25516000E+00
  0.15649600E+00  0.60916300E+00
  0.54910000E-01  0.59776700E+00
S 8 1.00
  0.38000800E+02  0.32340000E-01
  0.23972500E+02 -0.17176200E+00
  0.15218200E+02  0.74617900E+00
  0.55399900E+01 -0.19174980E+01
  0.13855100E+01  0.13662840E+01
  0.64246100E+00 -0.11299310E+01
  0.15649600E+00 -0.14423910E+01
S 1 1.00
  0.54910000E-01  0.10000000E+01
P 7 1.00
  0.10309200E+02  0.12821700E+00
  0.66276500E+01 -0.35379000E+00
  0.16744700E+01  0.56621100E+00
  0.30111600E+00  0.49317100E+00
  0.34687900E+00  0.11377500E+00
  0.12270100E+00  0.25340000E-02
  0.42428000E-01  0.67900000E-03
P 7 1.00
  0.10309200E+02 -0.35885000E-01
  0.66276500E+01  0.10289000E+00
```
CHAPTER 5. THE GLOBAL OPTIMIZATION WITH THIRD-PARTY PROGRAMS

0.16744700E+01 -0.2098800E+00
0.8011600E+00 -0.20104500E+00
0.34687900E+00 0.11654900E+00
0.12270100E+00 0.59496900E+00
0.42428000E-01 0.45552000E+00
P 7 1.00
0.10309200E+02 -0.7724200E-01
0.66276500E+00 0.22261100E+00
0.16744700E+01 -0.49535300E+00
0.8011600E+00 -0.36074700E+00
0.34687900E+00 0.71000000E+00
0.12270100E+00 0.56883900E+00
0.42428000E+00 0.11930000E-02
P 1 1.00
0.42428000E-01 0.10000000E+01
D 6 1.00
0.11002700E+02 0.16467000E-01
0.68916600E+01 -0.68013000E-01
0.18080800E+01 0.29949200E+00
0.82105100E+00 0.45430000E+00
0.34416100E+00 0.34422400E+00
0.12974300E+00 0.12125600E+00
D 6 1.00
0.11002700E+02 -0.23628000E-01
0.68916600E+01 0.95672000E-01
0.18080800E+01 -0.54129400E+00
0.82105100E+00 -0.39553300E+00
0.34416100E+00 0.61916500E+00
0.12974300E+00 0.46827600E+00
D 1 1.00
0.12974300E+00 0.10000000E+01
F 1 1.00
0.89190000E+00 0.10000000E+01
****

Au 0
ECP60MDF 5 60
H-Komponente

1
2 1.000000 0.000000
S-H
2
2 13.523218 426.641867
2 6.264384 36.800668
P-H
4
2 11.413867 87.002091
2 10.329215 174.004370
2 5.707424 8.870610
2 4.828165 17.902438
D-H
5.3. ABCLUSTER WITH OTHER PROGRAMS

ABCluster can in principle work with any programs. Now we will see how to do this.

5.3.1 How does isomer work?

When isomer is running, it will generate a geometry with a long name, like x-34. You do not need to know its exact name, in ABCluster it can be represented by $xxx$. This geometry will be saved in standard XYZ format too a file named x-34.xxxxxyz, represented by $inp$. Then, commands between the two >>> will be run one by one. Finally, the optimized geometry and energy should be written in XYZ format to a file named x-34.ooooout (The title line is the energy!), represented by $out$. Then isomer

Note that here we use TPSS/ECP60MDF method.

Step 3: Run the global optimization:

    nohup ..../isomer mol.inp > mol.out &

Step 4: After the optimization, you can check the structures. For my run, the first structure (mol-LM/0.xyz) happens to be global minimum, which is shown in Figure 5.2.2.

![Figure 5.2.2: The global minimum of Au6.](image)

Figure 5.2.2: The global minimum of Au$_6$.
will read optimized geometry and energy from this file and store it. Then \texttt{isomer} will start next search. This mechanism can be shown in Figure 5.3.1.

Now let’s see how we let ABCluster work with Gaussian. For the following commands:

\begin{verbatim}
>>>>
../..xyz2gaussian optfile $inp$ > $xxx$.gjf
g09 < $xxx$.gjf > $xxx$.log 2>/dev/null
../..gaussian2xyz $xxx$.log > $out$
>>>>
\end{verbatim}

In the first line, \texttt{xyz2gaussian} use the structure ($\inp$) generated by \texttt{isomer} and \texttt{optfile} to generate a Gaussian input file named $\xxx$.gjf. The second line calls Gaussian to perform the local optimization, the output of which is in $\xxx$.log. The third line extracts the final optimized structure and corresponding energy and saves it in $\out$. This is exactly what Figure 5.3.1 instructs you to do.

In summary, your main task is to write a program or script to translate the XYZ format geometry into the format that the third-party program accepts, and perform the calculation, then extract energy and optimized geometry and save them in another XYZ file.

5.3.2 Example: ABCluster with ORCA on He$_2$Ne$_2$

Assuming you work in Linux and want to use ORCA to perform the quantum chemistry calculation. You want to know the global minimum of a very weakly bound cluster He$_2$Ne$_2$.

Step 1: In the ABCluster distribution, go to the directory \texttt{testfiles/qc}.

Step 2: In Linux, one can write a shell script to generate the input file, do the calculation, and extract the energy and geometry information. One possible script is:
5.3. ABCLUSTER WITH OTHER PROGRAMS

```bash
#!/bin/bash

# Create input file
fn=$1-orca
xyzfn=$2
outfn=$3
echo "!' B3LYP D3BJ TZVPP TZVPP/JK RIJCOSX TightSCF OPT
  * xyz 0 1"
  > ${fn}.inp
awk 'NR>2{ print }' ${xyzfn} >> ${fn}.inp
echo "*
  >> ${fn}.inp

# Run the calculation
/orca/3.0.3/linux_x86-64/orca ${fn}.inp
  > ${fn}.out

# Transform the output file
energy=`grep "Total Energy" ${fn}.out | tail -n 1 | awk '{
  print $4}'`
sed "2s/^.* $/${energy}/" ${fn}.xyz > ${outfn}
rm -rf ${fn}.engrad ${fn}.gbw ${fn}.opt ${fn}.prop ${fn}.trj

If you are familiar with shell programming and ORCA input/output file format, this
file is self-explaining. We call it as oa. Once create it, you must give an “executable”
permission.

chmod +x oa

Step 2: Prepare a file called mol.inp:

```
mol  # Result file name
He 2 Ne 2  # Symbols
cube 2 2 2  # Structure types
10  # Maximal number of calculations
>>>>
./oa $xxx$ $inp$ $out$
>>>>
```

Step 3: Run the optimization:

nohup ../../isomer mol.inp > mol.out &

Step 4: Check the output file mol.out, it seems that structure 3 is the most stable
one, shown in Figure 5.3.2.

```
# Energy Time State
---------------------------------
0 -263.68047485 233 Succeed
1 -263.68100419 277 Succeed
2 -263.68070979 287 Succeed
3 -263.68123254 242 Succeed
4 -263.68029763 307 Succeed
5 -263.68116986 274 Succeed
```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-263.68101136</td>
<td>293</td>
<td>Succeed</td>
</tr>
<tr>
<td>7</td>
<td>-263.68053291</td>
<td>248</td>
<td>Succeed</td>
</tr>
<tr>
<td>8</td>
<td>-263.68026220</td>
<td>119</td>
<td>Succeed</td>
</tr>
<tr>
<td>9</td>
<td>-263.68019605</td>
<td>260</td>
<td>Succeed</td>
</tr>
</tbody>
</table>

Figure 5.3.2: The global minimum of $\text{He}_2\text{Ne}_2$. 


Bibliography


