ABCluster Manual

Version 1.0

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 Program Citation .......................... 1
1.3 Contacting the Author .......................... 2
1.4 Copyright and Disclaimer ......................... 2
1.5 Components of ABCluster ......................... 2
1.6 Installation of ABCluster ......................... 2

2 A Minimal Description of Theory ..................... 3

3 The Global Optimization of Atomic Clusters ............... 5
3.1 Introduction .................................. 5
3.2 Example 1: 38 Lennard-Jones Particles ............... 7
3.3 About abcinp ................................ 7
3.4 Example 2: (MgO)20 .......................... 9

4 The Global Optimization of Rigid Molecular Clusters ......... 11
4.1 Introduction .................................. 11
4.2 Example 1: (H2O)6 .......................... 11
4.3 Example 2: Li+, Na+ and Cs+ in (C6H6)6 ............... 14
4.4 Example 3: (CH2=CH2)13 .......................... 15

Bibliography .......................... 19
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 latest 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 four chapters: Chapter 1 gives you an introduction to ABCluster; Chapter 2 gives a minimal description of “artificial bee colony” algorithm; Chapter 3 and 4 tell you how to use ABCluster to treat atomic and rigid molecular clusters, respectively.

1.2 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.
1.3 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 Dr. Jun Zhang by E-mail:

ZhangJunQcc@gmail.com

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)
  
  libLBFGS is the C realization of the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) method. It is Copyright © 2002-2014 Naoaki Okazaki. libLBFGS is distributed under the term of the MIT license.

- **Boost** (see 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.
- **abcinp** The component to generate the input files for atom-optimizer.
- **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! Of course, you can set PATH variable to the directory where ABCluster resides, if you wish. 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 for use, please enjoy it!
Chapter 2

A Minimal Description of Theory

ABCcluster uses the so-called “artificial bee colony” (ABC) algorithm to perform the global optimization. It was proposed by Karaboga in 2005[2] and soon gains a lot of studies and applications in various problems. The ABC algorithm mimics the foraging behaviour of bees. ABCcluster 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 ABCcluster. 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, 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 glimit and the maximum cycle number gmax. Here are some typical choices:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SN</td>
<td>10 to 300 or larger</td>
</tr>
<tr>
<td>gmax</td>
<td>100 to 100000 or larger</td>
</tr>
<tr>
<td>glimit</td>
<td>3 to 10</td>
</tr>
</tbody>
</table>

For details of the algorithms used in ABCcluster, please refer to ABCcluster citation paper[1]. In the following chapters we will give some small systems as examples. In a real scientific research by ABCCluster, 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 gmax and SN are required to get reliable results, e.g. for some clusters with short-ranged Morse potential, gmax = 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.
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 high-level methods, like 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:


\[
U_{\text{CBM}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( \frac{e^2}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} + B_{ij} \exp \left( -\frac{r_{ij}}{\rho_{ij}} \right) \right)
\]

(3.1.1)

2. Lennard-Jones potential[4] \( \text{LJ}_N \). Widely used to describe the dispersion interaction in chemistry.

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

(3.1.2)

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

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

(3.1.3)


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

(3.1.4)

5. Coulomb–Morse–Repulsion potential[6] \( \text{CMR}_N \). This is very useful in the simulation of oxides, silicates, and silica-based glasses.

\[
U_{\text{CMR}} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( q_i q_j \frac{1}{r_{ij}} + D_{ij} \left( \exp \left( -2\alpha_{ij} \left( r_{ij} - \rho_{ij} \right) \right) \right)
\]

\[+2 \exp \left( -\alpha_{ij} \left( r_{ij} - \rho_{ij} \right) \right) + \frac{C_{ij}}{r_{ij}^{12}} \]

(3.1.5)
6. Z potential\[7\] ($Z_N$). This is inspired by an earlier potential proposed by Dzugutov\[8, 9\] 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.

$$U_Z = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( ae^{\alpha r_{ij}} + b \left( \frac{\sigma}{r_{ij}} \right)^n - \frac{V_0}{r_{ij}^2} \right)$$

(3.1.6)

7. Girifalco\[10\] ($G_{FN}$). An effective potential of the fullerene-fullerene interaction.

$$U_{Gi} = \sum_{i=1}^{N} \sum_{i<j}^{N} \left( -\alpha \left( \frac{1}{s_{ij} (s_{ij} - 1)^3} + \frac{1}{s_{ij} (s_{ij} + 1)^3} - \frac{2}{s_{ij}^4} \right) \right)$$

$$+ \beta \left( \frac{1}{s_{ij} (s_{ij} - 1)^9} + \frac{1}{s_{ij} (s_{ij} + 1)^9} - \frac{2}{s_{ij}^{16}} \right)$$

$$s_{ij} = \frac{r_{ij}}{2d}$$

(3.1.7)

$$\alpha = \frac{N^2 A}{12 (2d)^6}$$

(3.1.8)

$$\beta = \frac{N^2 B}{90 (2d)^{12}}$$

(3.1.10)

8. Gupta potential\[11\] ($G_N$). A very important many-body potential for modeling metals.

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

(3.1.11)

where:

$$\rho(r_i) = \sum_{j=1}^{N} c_{ij}^2 \exp \left( -2q_{ij} \left( \frac{r_{ij}}{d_{ij}} - 1 \right) \right)$$

(3.1.12)


$$U_G = \sum_{i=1}^{N} \left( \frac{1}{2} \sum_{j=1}^{N} \epsilon_{ij} \left( \frac{a_{ij}}{r_{ij}} \right)^p - \sqrt{\rho(r_i)} \right)$$

(3.1.13)

where:

$$\rho(r_i) = \sum_{j=1}^{N} c_{ij}^2 \left( \frac{a_{ij}}{r_{ij}} \right)^q$$

(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 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: sigma 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, 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:

```plaintext
1 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:

```bash
abcinp
```
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 “$g_{\text{max}}$” mentioned in Section 2. Recommended value: 100 to 100000 or larger.

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

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

- **natomssymbol1 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.

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

```bash
../../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:

```plaintext
38  # number of atoms
lj38i.xyz  # initial guess file; * - random guess
LennardJones  # force field types
lj38.par  # force field parameters
30  # population size
300  # maximal generations
5  # scout limit
5.00000000  # amplitude
lj38  # 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\([13]\):

\[
\begin{align*}
B(\text{Mg} - \text{O}) &= 821.6; B(\text{Mg} - \text{Mg}) = 0.0; B(\text{O} - \text{O}) = 22764 \\
\rho(\text{Mg} - \text{O}) &= 0.3242; \rho(\text{Mg} - \text{Mg}) = 0.1; \rho(\text{O} - \text{O}) = 0.1490 \\
q_{\text{Mg}} &= +2.0; q_{\text{O}} = -2.0
\end{align*}
\]

If we use \(L = 5.0\), \(SN = 100\), \(g_{\text{max}} = 100\), \(g_{\text{limit}} = 5\), then we can perform the global optimization:

```bash
../../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:

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!

![Diagram](image.png)

**Figure 3.4.1:** The global minimum of (MgO)$_{20}$. 
Chapter 4

The Global Optimization of Rigid Molecular Clusters

4.1 Introduction

Molecular clusters are very important in several fields 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 freedom (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 freedom: 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:

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

(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[14], OPLS[15] and AMBER[16]. Some molecule files suitable for ABCluster are available in the distribution in testfiles/rigidmolecularcluster/charmm36.

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

4.2 Example 1: (H$_2$O)$_6$

We want to know the most stable structure of (H$_2$O)$_6$. 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:
CHAPTER 4. THE GLOBAL OPTIMIZATION OF RIGID MOLECULAR CLUSTERS

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 “g_{max}” 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 “g_{limit}” 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:

```
input.cluster  # cluster file name
20             # population size
20             # maximal generations
3              # scout limit
4.000000000    # amplitude
h2o6           # save optimized configuration
30             # number of LMs to be saved
```

Here:

- Line 1: The number of different components in the cluster. For (H₂O)_6 this is 1. For Na⁺(H₂O)₃(CH₃OH)_₂, this is 3.

- Line 2: The file name (with path!) containing the coordinates and force field parameters, and the number of this 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:

```
1
2
charmm36/tip4p.xyz 6
* 4.0000
```

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. EXAMPLE 1: \((H_2O)_6\)

\[
\begin{array}{cccccc}
4.2 & 3.4 & 3.9 & 1.9 & 0.2 & 3.2 \\
3.6 & 4.7 & 2.9 & 3.6 & 1.7 & 1.7 \\
5.1 & 5.2 & -0.6 & 1.6 & 2.8 & 3.5
\end{array}
\]

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 \(X, Y, Z, \alpha, \beta, \gamma\).

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.
- **h2o6-0PT.xyz** The global minimum in XYZ format. It can be read by e.g. VMD, CYLView.
- **h2o6-0PT.gjf** The global minimum in Gaussian input format. It can be read by e.g. GaussView.
- **h2o6-0PT.cluster** The global minimum in ABCCluster 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 ABCCluster 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 atom-optimizer, thus will not be explained again here. Open **h2o6-0PT.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!

![cage and prism](Figure 4.2.1: The two minima of \((H_2O)_6\).)
4.3 Example 2: Li⁺, Na⁺ and Cs⁺ in (C₆H₆)₆

The section title gives three cation-π 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⁺(C₆H₆)₆. Prepare two files: li-ben.inp and li-ben.cluster:

```
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.

![Li, Na, Cs](image)

**Figure 4.3.1:** The global minima of Li⁺(C₆H₆)₆, Na⁺(C₆H₆)₆ and Cs⁺(C₆H₆)₆.

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:

```
6
ethene
C 0.00000000 0.00000000 0.66542300
H 0.00000000 0.92366200 1.23955100
H 0.00000000 -0.92366200 1.23955100
C 0.00000000 0.00000000 -0.66542300
H 0.00000000 -0.92366200 -1.23955100
H 0.00000000 0.92366200 -1.23955100
```

Step 2: Construct the force field parameters of CH₂=CH₂. Since in this manual we always use CHARMM force field, for consistency we also use this for ethene. You can get the lastest 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 toppar_c36_aug14.tgz and unzip it, fortunately there are parameters for ethene.

In the file toppar/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
IC  C1 C2 H21 H22 1.3370 121.37 -180.00 31.37 1.8845
PATC FIRS NONE LAST NONE
```

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 ϵ and σ parameters, we check the file toppar/par_all36_cgenff.rtf, and find this:

```
CG2D2 0.0 -0.0640 2.0800 ! alkene, yin,adm jr., 12/95
```
-0.0640 and 2.0800 are $\epsilon$ and $\sigma$, respectively. However, since CHARMM and ABCluster use different formulae, we have to transform them! The transformation formulae is:

$$\epsilon_{\text{ABCluster}} = \epsilon_{\text{CHARMM}} \times (-4.184)$$  \hspace{1cm} (4.4.1)

$$\sigma_{\text{ABCluster}} = \sigma_{\text{CHARMM}} \times 2^{\frac{5}{6}}$$  \hspace{1cm} (4.4.2)

Thus its final value is:

$$q = -0.42$$  \hspace{1cm} (4.4.3)

$$\epsilon = -0.0640 \times (-4.184) = 0.2678$$  \hspace{1cm} (4.4.4)

$$\sigma = 2.0800 \times 2^{\frac{5}{6}} = 3.7061$$  \hspace{1cm} (4.4.5)

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

$$q = +0.21$$  \hspace{1cm} (4.4.6)

$$\epsilon = -0.0260 \times (-4.184) = 0.1088$$  \hspace{1cm} (4.4.7)

$$\sigma = 1.2600 \times 2^{\frac{5}{6}} = 2.2451$$  \hspace{1cm} (4.4.8)

Thus the final parameter file c2h4.xyz for \text{CH}_2-\text{CH}_2 is:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ethene</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>4</td>
<td>H</td>
<td>0.00000000</td>
<td>0.92366200</td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>0.00000000</td>
<td>-0.92366200</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>7</td>
<td>H</td>
<td>0.00000000</td>
<td>-0.92366200</td>
</tr>
<tr>
<td>8</td>
<td>H</td>
<td>0.00000000</td>
<td>0.92366200</td>
</tr>
<tr>
<td>9</td>
<td>q</td>
<td>epsilon (kJ/mol)</td>
<td>sigma (Å)</td>
</tr>
<tr>
<td>10</td>
<td>-0.42</td>
<td>0.2678</td>
<td>3.7061</td>
</tr>
<tr>
<td>11</td>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
<tr>
<td>12</td>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
<tr>
<td>13</td>
<td>-0.42</td>
<td>0.2678</td>
<td>3.7061</td>
</tr>
<tr>
<td>14</td>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
<tr>
<td>15</td>
<td>+0.21</td>
<td>0.1088</td>
<td>2.2451</td>
</tr>
</tbody>
</table>

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:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input.cluster</td>
<td># cluster file name</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td># population size</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td># maximal generations</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td># scout limit</td>
</tr>
<tr>
<td>5</td>
<td>10.00000000</td>
<td># amplitude</td>
</tr>
<tr>
<td>6</td>
<td>c2h413</td>
<td># save optimized configuration</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td># number of LMs to be saved</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c2h4.xyz</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>* 10.0000</td>
<td></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 -139.4170519 ...
6  5.85200692 -116.01432862 -140.13516658 ...
7  5.03267295 -124.05305628 -140.13516658 ...
    ...
9  52  4.65923457 -120.62871046 -140.72175322 ...
10  53  3.74502721 -123.42238555 -140.72175322 ...
11  54  4.87904478 -120.29869548 -140.72175322 ...
12  55  4.73791669 -120.80626616 -140.72175322 ...
13  56  4.58123341 -124.32091734 -140.82027375 ...
14  57  5.84959151 -123.90112274 -140.82027371 ...
15  58  7.10242950 -122.13716432 -140.82027374 ...
    ...
16  * Final Global Minimal Energy: -140.82027375
```

At 56th step the final global minimal energy is located. The final global minimum has $C_1$ symmetry (in Figure 4.4.1):

![Figure 4.4.1: The global minimum of (CH$_2$=CH$_2$)$_{13}$.](image)


