



---

**MAGUS:**  
**Machine learning And Graph theory assisted**  
**Universal structure Searcher**

高豪, 王俊杰, 韩瑜, Dc, 孙建

**使用手册**

Version 1.0.2, 2021 年 12 月 14 日.

<https://git.nju.edu.cn/gaaooh/magus>

# 目录

<b>1 安装</b>	<b>3</b>
1.1 依赖	3
1.2 准备	3
1.2.1 设置 ase 的 vasp 接口	3
1.2.2 机器学习包安装	4
1.3 pip 安装	4
1.4 source 安装	5
1.4.1 代码下载	5
1.4.2 编译 GenerateNew 模块	5
1.4.3 编译 lrpot 模块	6
1.4.4 设置入口	6
1.4.5 设置环境变量	7
1.5 检查安装	7
1.6 设置自动补全 (可选)	7
<b>2 输入文件</b>	<b>8</b>
2.1 input.yaml 参数	8
2.1.1 基本参数	8
2.1.2 种群相关	8
2.1.3 结构产生	8
2.1.4 结构演化	9
2.1.5 煮计算器	9
2.1.6 机器学习	10
2.1.7 代理计算器	10
2.2 inputFold	11

2.2.1	Vasp . . . . .	11
2.2.2	Gulp . . . . .	12
2.2.3	Lammps . . . . .	13
2.2.4	ASE 系列 (EMT, LJ, XTB...) . . . . .	14
2.2.5	MTP . . . . .	15
2.3	Seeds . . . . .	16
<b>3</b>	<b>程序指令</b>	<b>18</b>
3.1	search . . . . .	18
3.2	summary . . . . .	18
3.3	calc . . . . .	20
<b>4</b>	<b>输出文件</b>	<b>21</b>
4.1	calcFold . . . . .	21
4.2	mlFold . . . . .	22
4.3	results . . . . .	22
<b>5</b>	<b>例子</b>	<b>24</b>
5.1	$NH_4NO_3$ 分子晶体产生 . . . . .	24
5.2	MTP 大批量优化随机结构 . . . . .	25
5.3	$TiO_2$ 定组分结构搜索 . . . . .	28
5.4	$Zn_x(OH)_y$ 变组分结构搜索 . . . . .	30
5.5	$MgSiO_3$ 机器学习结构搜索 . . . . .	33
5.6	$CH_4$ 分子晶体搜索 . . . . .	34
<b>6</b>	<b>常见问题</b>	<b>37</b>

# 1 安装

## 1.1 依赖

Python  $\geq 3.6$

NumPy

SciPy

Scikit-learn

PyYAML

Ase  $\geq 3.19$

Networkx  $= 2.1$

Spglib

Pandas

可选项:

Pytest  $\geq 3.6.1$ : unittest

Xtb  $= 6.3$ : XTBCalculator

Mlip: MTPCalculator

## 1.2 准备

### 1.2.1 设置 ase 的 vasp 接口

- 1) 新建一个 run\_vasp.py:

```
import subprocess  
exitcode = subprocess.call("mpiexec.hydra  
/your/path/to/vasp", shell=True)
```

- 2) 建立 mypps 目录存放 vasp 脚本，可以用软连接:

mypps/

```
|— potpaw  
|— potpaw_GGA  
|— potpaw_PBE
```

```
$ ln -s /your/path/PBE-5.4 mypps/potpaw_PBE
```

三个子目录分别对应 LDA, PW91, PBE 也可以加入其他赝势库。

3) 设置环境变量:

```
$ export VASP_SCRIPT=/your/path/run_vasp.py  
$ export VASP_PP_PATH=/your/path/mypps
```

更多信息见:<https://wiki.fysik.dtu.dk/ase/ase/calculators/vasp.html#module-ase.calculators.vasp>

注意: run\_vasp.py 和 mypps 不要放在 magus 目录下

### 1.2.2 机器学习包安装

TwoShareMTPCalculator 使用了修改过的 mtp 代码, 官方版本不支持此功能. 如需使用可进入 magus/mtp-api 目录, 按其中教程安装. 详见:<https://git.nju.edu.cn/bigd4/mtp-api>

## 1.3 pip 安装

magus 安装需要 boost\_python, boost\_numpy. 若所使用的 python 环境 (如 anaconda-3/5.0.1) 的 lib 中有会自动探测路径, 否则需要在环境变量中给出相应路径 (无需写入 bashrc)

```
$ export MAGUS_INCLUDE_PATH=your_path_to_include_dir:  
your_path_to_py_include_dir  
$ export MAGUS_LD_LIBRARY_PATH=your_ld_library_path
```

如:

```
$ CONDA_PATH=/fs00/software/anaconda/3-5.0.1  
$ export MAGUS_INCLUDE_PATH=$CONDA_PATH/include:  
$CONDA_PATH/include/python3.6m  
$ export MAGUS_LD_LIBRARY_PATH=$CONDA_PATH/lib
```

配置好后执行:

```
$ pip install -i  
https://repo.nju.edu.cn/repository/pypi-nju/simple  
magus-test --upgrade
```

若直接在集群安装需加入--user参数

## 1.4 source 安装

### 1.4.1 代码下载

克隆库到本地并初始化子项目:

```
$ git clone git@git.nju.edu.cn:gaaoh/magus.git  
$ git submodule init  
$ git submodule update
```

或直接下载压缩包.

### 1.4.2 编译 GenerateNew 模块

进入 magus/generatenew 中, 编译 generatenew.so 文件, 并将其放入 magus/magus 文件夹下. 如使用集群 anaconda/3-5.0.1, 命令为:

```
$ g++ -std=c++11 -I/fs00/software/anaconda/3-5.0.1/include  
-I/fs00/software/anaconda/3-5.0.1/include/python3.6m  
-L/fs00/software/anaconda/3-5.0.1/lib -lboost_python  
-lboost_numpy -lpython3.6m main.cpp -o GenerateNew.so  
-shared -fPIC
```

具体可见: <https://git.nju.edu.cn/HanYu/generatenew>

### 1.4.3 编译 lrpot 模块

进入 magus/lrpot 中, 编译 lrpot.so 文件, 并将其放入 magus/magus 文件夹下. 如使用集群 anaconda/3-5.0.1, 命令为:

```
$ g++ -std=c++11 -I/fs00/software/anaconda/3-5.0.1/include  
-I/fs00/software/anaconda/3-5.0.1/include/python3.6m  
-L/fs00/software/anaconda/3-5.0.1/lib -lboost_python  
-lboost_numpy -lpython3.6m lrpot.cpp -o lrpot.so -shared  
-fPIC
```

具体可见: <https://git.nju.edu.cn/bigd4/lrpot>

### 1.4.4 设置入口

新建 magus 文件:

```
import re  
import sys  
from magus.entrypoints.main import main  
if __name__ == '__main__':  
    sys.argv[0] = re.sub(r'(-script|.pyw|\.exe)?$', '',  
        sys.argv[0])  
    sys.exit(main())
```

保存后执行 `chmod +x magus` 设置为可执行文件。

#### 1.4.5 设置环境变量

在 `bashrc` 中加入

```
$ export PYTHONPATH=$PYTHONPATH:/your/path/magus  
$ export PATH=$PATH:/your/path/magus
```

### 1.5 检查安装

```
$ magus -v  
-----  
1.0.2
```

### 1.6 设置自动补全（可选）

在 `bashrc` 中加入

```
$ source your_path_to_magus/auto_complete.sh
```



## 2 输入文件

一个典型的结构搜索任务一般包含以下文件：

- `input.yaml`, 给出任务的主要参数. 详见 2.1
- `inputFold`, 补充 `input.yaml` 中定义的 Calculator 所需要的一些额外的信息, 如 Vasp 的 `INCAR`, Lammps 的 `in.lammps` 等. 详见 2.2
- `Seeds`, 给出种子结构, 可指定在哪一代加入. 详见 2.3

### 2.1 `input.yaml` 参数

#### 2.1.1 基本参数

- `formulaType` : 计算类型, 可用值: `fix` (定组分), `var` (变组分)
- `pressure` : 压强 (GPa)
- `molMode` : 是否使用分子晶体模式产生结构
- `symprec` : 默认值: 0.1, 判断空间群时容忍误差

#### 2.1.2 种群相关

- `initSize` : 初代种群大小
- `popSize` : 种群大小
- `numGen` : 迭代次数
- `saveGood` : 保留到下一代的优秀结构数

#### 2.1.3 结构产生

- `spacegroup` : 随机结构的空間群, 例: `[1,2,20-30]`
- `minNAtoms` : 最小原子数

- maxNAtoms : 最大原子数
- symbols : 元素类型例: ['Ti', 'O'], 外层是方括号, 每个元素用引号括起来,
- formula : 元素比例, 例: [1, 2] (定组分), [[1,0],[0,1]] (变组分)
- eleSize : 变组分搜索时, 产生的不包含所有元素的结构数, 如: eleSize=2, 则每代产生 (He, Al, O, HexAl, AlxOy, HexOy) 各两个
- volRatio : 随机产生结构时的体积参数, 为结构体积除以原子球体积之和
- dRatio : 若存在两原子距离除以共价半径之和小于此值, 则认为过近, 排除此结构
- distanceMatrix : 原子间的最小距离矩阵, 与 dRatio 同时出现时使用此数值, 例: [[1, 1.5, 2.0], [1.5, 1.8, 1.9], [2.0, 1.9, 2.5]]

#### 2.1.4 结构演化

- randFrac : 随机结构比例
- molDetector : 结构演化时判断分子片段的方法可用值: 0(不判断分子局域结构, 默认值) 1(自动判断分子局域结构) 2(使用 Girvan-Newman 算法划分局域结构)
- addSym : 产生结构之前是否为父代加入对称性

#### 2.1.5 煮计算器

主计算器定义了 Magus 搜索时使用的 MainCalculator, 以下所有参数都定义于 MainCalculator 条目下, 需要缩进:

- jobPrefix : 计算器所需附属文件在 inputFold 文件夹中的名称, 如 EMT, OvO 等如需使用多个计算器串接则给出对应列表, 如: ['Gulp', 'Vasp1', 'Vasp2']
- calculator : 程序种类, 如未给出则按照 jobPrefix 文件名判断, 可用值: vasp, gulp, lammps, emt, xtb, lj, quip
- mode : 运行方式, 可用值: serial (串行), parallel (并行)
- xc : (VASP) 交换关联类型, 可用值: PBE, LDA, PW-91

- ppLabel : (VASP) VASP 赝势的后缀, 与 symbols 顺序一致, 若无后缀则填入", 如: ['\_sv', "].
- exeCmd : (gulp, lammps) 运行结构优化程序的命令, 如 `gulp < input > output`, `mpirun -np 4 lmp_mpi -in in.lammps`

以下选项为并行模式下的队列控制选项, 同样适用于代理计算器

- numParallel : 并行优化结构的数目
- numCore : 结构优化使用的核数
- queueName : 结构优化任务的队列
- verbose : log 中是否显示详细队列信息
- waitTime : 检查任务的时间间隔 (s)
- killTime : 杀死任务的时间间隔 (s)
- preProcessing : 提交脚本时的预处理

### 2.1.6 机器学习

- poolSize : 用于初始化势场时随机产生的结构数
- initTimes : 初始化势场时的迭代次数, 如果已有训练过的力场可设为 0. 默认值: 2
- DFTRelax : 是否使用 DFT 演算

### 2.1.7 代理计算器

代理计算器定义了 MLMagus 搜索时使用的 MLCalculator, 以下所有参数都定义于 MLCalculator 条目下, 需要缩进:

- jobPrefix: 同2.1.5
- calculator: 程序种类, 如未给出则按照 jobPrefix 文件名判断, 可用值: mtp, mtp-lammps

- connect: 多个 mtp 的连接方式, 可用值: [naive, share-trainset]
- force\_tolerance: 结构优化力收敛判据, 默认值: 0.05
- stress\_tolerance: 结构优化应力收敛判据, 默认值: 1.
- weights: 训练时能量、力、应力权重, 默认值: [1., 0.01, 0.001]
- scaled\_by\_force: 训练时给予较小力的额外权重, 默认值: 0.
- min\_dist: 优化时最小距离, 默认值: 0.5
- n\_epoch: 训练代数, 默认值: 200

## 2.2 inputFold

inputFold 中为不同 calculator 所需要的补充文件, 放在对应的 jobPrefix 文件夹中。以下为各 calculator 所需文件的示例, 内容可根据需要修改 (名字不行):

### 2.2.1 Vasp

INCAR

就是 INCAR

```
PREC = Accurate
EDIFF = 1e-4
EDIFFG = 1e-3
IBRION = 2
ISIF = 3
NSW = 40
ISMEAR = 0
SIGMA = 0.050
POTIM = 0.250
ISTART = 0
LCHARG = FALSE
LWAVE = FALSE
KSPACING = 0.314
```

```
NCORE= 4
```

注意：INCAR 中不需要给出 pstress

### 2.2.2 Gulp

```
goption.relax
```

gulp 优化参数配置

```
opti conjugate nosymmetry conp
```

```
goption.scf
```

gulp 自洽参数配置

```
nosymmetry conp gradients
```

```
goption.scf
```

gulp 势函数文件

```
space
```

```
1
```

```
species
```

```
Mg 2.0
```

```
Al 3.0
```

```
O -2.0
```

```
lennard 12 6
```

```
Mg O 1.50 0.00 0.00 6.0
```

```
Al O 1.50 0.00 0.00 6.0
```

```
O O 1.50 0.00 0.00 6.0
```

```
Mg Mg 1.50 0.00 0.00 6.0
```

```
Mg Al 1.50 0.00 0.00 6.0
```

```
Al Al 1.50 0.00 0.00 6.0
```

```
buck
```

```
Mg 0 1428.5 0.2945 0.0 0.0 7.0
Al 0 1114.9 0.3118 0.0 0.0 7.0
O 0 2023.8 0.2674 0.0 0.0 7.0
maxcyc 850
switch rfo 0.010
time 60
```

### 2.2.3 LAMMPS

in.relax

lammps 优化参数配置，可替换为分子动力学等等

```
clear
atom_style atomic
units metal
boundary p p p
read_data data          # 此行不可更改
### interactions
pair_style lj/cut 2.5
pair_coeff * * 1 1
mass 1 35.450000
mass 2 22.989769
### run
fix fix_nve all nvt temp 300.0 300.0 100
dump dump_all all custom 1 out.dump id type x y z vx vy vz
fx fy fz
# 最终输出必须为 out.dump
thermo_style custom step temp press pxx pyy pzz pxy pxz
pyz ke pe etotal
# 需要 pxx pyy pzz pxy pxz pyz
thermo 1
run 10
```

in.scf

lammps 自洽计算参数配置

```
clear
atom_style atomic
units metal
boundary p p p
read_data data
### interactions
pair_style lj/cut 2.5
pair_coeff * * 1 1
mass 1 35.450000
mass 2 22.989769
### run
fix fix_nve all nve
dump dump_all all custom 1 out.dump id type x y z vx vy vz
fx fy fz
thermo_style custom step temp press pxx pyy pzz pxy pxz
pyz ke pe etotal
thermo 1
run 10
```

#### 2.2.4 ASE 系列 (EMT, LJ, XTB...)

EMT, LJ, XTB...

建个文件夹就完事了，除非如 XTB 需要相关配置文件

xtb.yaml

下次一定

## 2.2.5 MTP

mlip.ini

active learning 控制参数, 详见 <https://git.nju.edu.cn/bigd4/mtp-api/-/blob/master/doc/manual/manual.pdf>

```

mtp-filename                pot.mtp
# 改不得
select                      TRUE
    select:site-en-weight    1.0
    select:energy-weight     0.0
    select:force-weight      0.0
    select:stress-weight     0.0
    select:threshold         1.5
    select:threshold-break   7.0
    select:save-selected     B-preselected.cfg
# 改不得
    select:load-state        A-state.als
# 改不得

```

pot.mtp

mtp 使用势场, 可从 untrained\_mtps 中拷贝, 仅可改变标注了作用的参数。

```

MTP
version = 1.1.0
potential_name = MTP1m
species_count = 1                # 原子种类数量
potential_tag =
radial_basis_type = RBChebyshev
    min_dist = 2                  # 原子最小间距
    max_dist = 5                  # 原子环境最大考虑半径
    radial_basis_size = 8         # 基函数个数
    radial_funcs_count = 3

```



```
alpha_moments_count = 84
alpha_index_basic_count = 46
```

```
train.cfg
```

训练集，若不存在，将自动生成空训练集

```
BEGIN_CFG
Size
4
Supercell
2.457244 0.0 0.0
0.0 -2.457244 0.0
0.0 0.0 -2.457244
AtomData: id type cartes_x cartes_y cartes_z fx fy fz
1 0 0.0 0.0 0.0 0.0 -0.0 0.0
2 0 0.0 -1.228622 -1.228622 0.0 -0.0 -0.0
3 0 1.228622 -1.228622 0.0 -0.0 0.0 0.0
4 0 1.228622 0.0 -1.228622 0.0 -0.0 -0.0
Energy
-15.79300182
EnergyWeight
0.02195049074783156
PlusStress: xx yy zz yz xz xy
26.253439887628005 26.253439887628005 26.253439887628005
-0.0 0.0 0.0
END_CFG
```

## 2.3 Seeds

POSCARS\_i 为第 i 代加入的种子文件，如有 POSCAR\_1~POSCAR\_9 希望在第一代加入，POSCAR\_10~POSCAR\_19 希望第二代加入，执行 `cat POSCAR_{1..9} > POSCAR_1` ； `cat POSCAR_{10..19} > POSCAR_2` 后将 POSCAR\_1 与

POSCARS\_2 放入 Seeds 中即可。

## 3 程序指令

`magus` 文件为程序运行的入口，通过运行 `magus` 可以得到所有的指令与介绍，通过 `magus [command] -h` 可获得其帮助。

指令	用途
<code>search 3.1</code>	结构搜索
<code>summary 3.2</code>	事后总结
<code>clean</code>	事后清理
<code>prepare</code>	事前准备
<code>calc 3.3</code>	批量计算
<code>gen</code>	批量产生

### 3.1 search

结构搜索模块，使用时直接提交命令 `magus search` 即可。可选择参数如下：

- `-h -help`  
展示帮助文档
- `-i INPUTFILE, -input-file INPUTFILE`  
指定输入参数文件，默认为 `input.yaml`
- `-l LEVEL, -log-level LEVEL`  
指定 `log.txt` 文件 logging 等级，可选项：DEBUG,INFO,WARNING,ERROR。默认为 INFO
- `-m, -use-ml`  
是否使用机器学习模块
- `-r, -restart`  
是否为继续上次任务，使用此选项目录内应保留上次作业的 `results` 与 `log.txt`

### 3.2 summary

用于总结一条或多条 ase traj 格式的轨迹，参数如下：

- -h -help  
展示帮助文档
- -p PREC, -prec PREC  
判断空间群的精度，默认值为 0.1
- -r, -reverse  
是否倒着输出，默认正着输出
- -s, -save  
是否将此轨迹中所有结构输出为 POSCAR，默认不输出，以防文件夹很乱
- -o OUTDIR, -outdir OUTDIR  
POSCAR 输出的目录
- -n SHOW\_NUMBER, -show-number SHOW\_NUMBER  
展示条目数量，默认为 20
- -sb SORTED\_BY, -sorted-by SORTED\_BY  
用哪个关键字排序，默认为 enthalpy
- -rm REMOVE\_FEATURES [REMOVE\_FEATURES ...], -remove-features REMOVE\_FEATURES [REMOVE\_FEATURES ...]  
需要移除展示的信息
- -a ADD\_FEATURES [ADD\_FEATURES ...], -add-features ADD\_FEATURES [ADD\_FEATURES ...]  
需要附加展示的其他信息

一个例子如下：

```
$ magus summary good.traj ref.traj -a volume -rm priSym
parentE -n 10
```

	symmetry	enthalpy	origin	fullSym	volume	source
1	I-43d (220)	0.412	random	Li16	123.164	ref
2	Cmc2_1 (36)	0.416	seed	Li88	647.965	good

3	Aea2 (41)	0.419	None	Li40	301.519	ref
4	Ama2 (40)	0.419	random	Li88	645.352	good
5	P3c1 (158)	0.422	Rattle	Li88	644.677	good
6	Cc (9)	0.423	Rattle	Li88	646.661	good
7	Cm (8)	0.427	Lattice	Li88	649.852	good
8	P1 (1)	0.427	Slip	Li88	647.470	good
9	Aea2 (41)	0.429	random	Li88	647.528	good
10	Aea2 (41)	0.429	Rattle	Li88	647.652	good

### 3.3 calc

根据 `input.yaml` 中定义的 calculator, 计算给出的 traj, 结果会输出于 `out.traj`, 可用 `magus summary out.traj` 命令查看。

- `-h -help`  
展示帮助文档
- `-m MODE, -mode MODE`  
计算类型, 可用值: scf, relax. 默认为 relax
- `-i INPUTFILE, -input-file INPUTFILE`  
指定参数所在文件, 默认为 `input.yaml`
- `-p PRESSURE, -pressure PRESSURE`  
指定压强, 若不给出则使用 `INPUTFILE` 中给出的压强

如若需对 `in.traj` 中的结构在 10GPa 下优化, 运行命令为:

```
$ magus calc in.traj -p 10
```

此命令与 `MtpCalculator` 或 `MTPLammpsCalculator` 结合, 可以 on the fly 的得到机器学习的训练集。

## 4 输出文件

结构搜索过程中, 将会产生 `log.txt`, `calcFold` 与 `results`, 如果搜索时加入 `-m` 选项, 会额外产生 `MLFold`。

### 4.1 calcFold

`calcFold` 中为 `input.yaml` 定义的计算器计算过程中所产生的文件。一个典型的结构搜索任务将产生如下结构的 `calcFold`:

```
calcFold
├── MTP
│   ├── epoch00
│   ├── epoch01
│   ├── epoch02
│   └── job_controller
└── Vasp
    ├── 00
    ├── 01
    ├── 02
    ├── 03
    └── job_controller
```

如果并行模式计算过程中发生错误, 报错信息将在对应的文件夹中出现。此外, 并行模式中计算器文件夹下会产生 `job_controller` 文件, 可通过修改其中内容改变作业提交的参数。如:

old\_job\_controller

```
kill_time: 100000
num_core: 48
pre_processing: ''
queue_name: 9242opa!
verbose: false
```

new\_job\_controller

```
kill_time: 100000
num_core: 64
pre_processing: ''
queue_name: 7702ib
verbose: false
```

这些改变将被记录在 `log.txt` 中:

`log.txt`

Be careful, the following settings are changed

`num_core: 48 -> 64`

`queue_name: 9242opa! -> 7702ib`

## 4.2 mlFold

`mlFold` 中为 `input.yaml` 中定义的机器学习模块执行挑选、训练的部分。如果提交任务时此文件夹不存在, 将会使用 `inputFold` 中对应的文件。训练或挑选中发生错误报错信息将在对应文件夹中出现, 此外, 可在 `train-out` 中查看训练集上的误差。

若已经进行过一次 MLMagus 搜索并得到了一个不错的势, 可以不用删除 `mlFold` 以在以后的搜索中使用; 或者复制 `pot.mtp` 与 `train.cfg` 到其他 `inputFold` 中反复使用。此时可将 `init_times` 设为 0, 代表不再在初始化时训练力场。

## 4.3 results

`results` 中记录了各代生成的各种结构, 可根据需要使用 `summary` 命令查看。

- `good.traj`  
目前最优的 `popSize` 个结构
- `good{i}.traj`  
第 `i` 代最优的 `popSize` 个结构
- `best.traj`  
历代最优的结构, 使用 `summary` 查看时注意需添加 `-sb None` 或 `-sorted_by None` 选项, 否则会默认按焓值排序显示而不是代数的顺序
- `keep{i}.traj`  
第 `i` 代经过聚类后保留的 `goodSize` 个结构

- `init{i}.traj`  
第 i 代产生的初始结构
- `raw{i}.traj`  
第 i 代的初始结构经过第一性结构优化后的结构，可用于 debug
- `gen{i}.traj`  
第 i 代种群，一般为 `raw{i}.traj` 或 `mlraw{i}.traj` 经过 check 后的结果若机器学习搜索使用第一性验证，则为 `mlraw{i}.traj` 中低能结构第一性优化后额结果。第一性验证则为
- `mlraw{i}.traj`  
第 i 代的初始结构经过机器学习结构优化后的结构，可用于 debug
- `mlgen{i}.traj`  
`mlraw{i}.traj` 经过 check 的结果



## 5 例子

### 5.1 $NH_4NO_3$ 分子晶体产生

目标：产生 10 个 *Pccn* 的  $NH_4NO_3$  分子晶体。

准备：input.yaml, NH4.xyz, N03.xyz

input.yaml

结构产生控制文件

```
minAt: 72                # 最小原子数
maxAt: 72                # 最大原子数
symbols: ['H', 'N', 'O'] # 待产生结构所含元素
molMode: True            # 分子晶体模式
molFile: ['NH4.xyz', 'N03.xyz'] # 所含分子结构
molFormula: [1, 1]       # 分子组分
molType: 'fix'
spacegroup: [56]         # 指定 56 号空间群 Pccn
dRatio: 0.8              # 原子间最小距离比
threshold_mol: 1.5       # 分子间最小距离比
volRatio: 8              # 体积比
```

NH4.xyz

```
5
NH4
H    4.511281    4.375470    3.210227
H    3.584655    4.486488    1.796710
H    4.670180    3.191076    2.019142
H    3.246077    3.272899    2.937012
N    4.000271    3.837356    2.488938
```

N03.xyz

```

4
N03
N    2.012707    2.014563    4.870574
O    1.714319    0.953807    5.478185
O    2.311095    3.075319    5.478185
O    2.012707    2.014563    3.582428

```

运行: `magus gen -n 10`

结果: 目标结构 `gen.traj`

## 5.2 MTP 大批量优化随机结构

目标: 使用 MTP 力场优化 1000 个  $B_{12}$  随机结构

准备: 待优化结构 `gen.traj`, `inputFold`, `input.yaml`, 其中, `inputFold` 结构为:

```

inputFold/
├── MTP
│   ├── mlip.ini
│   ├── pot.mtp
│   └── train.cfg
└── Vasp
    └── INCAR

```

input.yaml

定义主计算器与机器学习计算器

*#main calculator settings*

MainCalculator:

jobPrefix: Vasp

*# 标准能量由 Vasp 给出*

*#vasp settings*

```
xc: PBE
ppLabel: ['']
#parallel settings
numParallel: 20
numCore: 24
queueName: 9242opa!
waitTime: 30

MLCalculator:
  jobPrefix: MTP
  calculator: mtp
  min_dist: 1.2          # MTPrelax 最小距离
  queueName: 9242opa!
  numCore: 48
  waitTime: 10
  force_tolerance: 0.001
  stress_tolerance: 0.01
```

#### INCAR

不会有人不会写 INCAR 把

```
PREC = Accurate
EDIFF = 1e-4
IBRION = 2
ISIF = 3
NSW = 40
ISMEAR = 0
SIGMA = 0.050
POTIM = 0.250
ISTART = 0
LCHARG = FALSE
LWAVE = FALSE
```

```
#Crude optimisation
EDIFFG = 1e-3
KSPACING = 0.314
NCORE = 4
```

#### pot.mtp

MTP 势函数文件，这里只给出表头

```
MTP
version = 1.1.0
potential_name = MTP1m
scaling = 1.497018669914417e-02
species_count = 1                # 只有 B 原子一种
potential_tag =
radial_basis_type = RBChebyshev
    min_dist = 1.2771138600000000e+00
    # 最小距离 1.27, 由于 train 中设置了--update, 此项不必特别设置, 会自动更新
    max_dist = 6.0000000000000000e+00
    radial_basis_size = 12
    radial_funcs_count = 5
```

#### mlip.ini

active 控制文件

mtp-filename	pot.mtp
select	TRUE
select:site-en-weight	1.0
select:energy-weight	0.0
select:force-weight	0.0
select:stress-weight	0.0
select:threshold	1.5

```

select:threshold-break      7.0
select:save-selected       B-preselected.cfg
select:load-state          A-state.als

```

运行：提交 `magus calc gen.traj` 命令到队列

结果：MTP 优化后的结构 `out.traj`，势文件 `mlFold/MTP/pot.mtp`

### 5.3 $TiO_2$ 定组分结构搜索

目标：搜索 12 个原子  $TiO_2$  的结构

准备：input.yaml, inputFold

由于初始结构往往杂乱无章，因此往往使用多个 INCAR 分步优化，据说合适的做法是固定体积优化原子位置与晶格形状 (ISIF=4)，然后放开体积限制优化原子位置与晶格 (ISIF=3)，最后进行高精度的单点能自洽运算 (NSW=0)

input.yaml

给出搜索所需参数

```

calcType: fix                # 定组分搜索
pressure: 0                  # 0GPa
initSize: 20                 # 初始生成 20 个结构
popSize: 20                  # 每代维持 20 个结构
numGen: 10                   # 搜索 10 代
saveGood: 3                  # 每代保存 3 个结构

#structure parameters
minAt: 12
maxAt: 12
symbols: ['Ti', 'O']
formula: [1, 2]

```

```

dRatio: 0.6                # 最小半径比
volRatio: 3                # 最小体积比
addSym: False              # 不在父代中加入对称性

#main calculator settings
MainCalculator:
  jobPrefix: ['q', 'w', 'e', 'r']
  # 这里只是为了说明只要给出 calculator, jobPrefix 可以随意命名, 实际建议用更清晰的方式命名。
  calculator: vasp
  mode: parallel
  #vasp settings
  xc: PBE
  ppLabel: ['', '_s']
  #parallel settings
  numParallel: 5
  queueName: 7702ib
  waitTime: 30

```

## q/INCAR

```

PREC = Normal
EDIFF = 1e-3
IBRION = 2
ISIF = 4
NSW = 85
ISMear = 0
SIGMA = 0.06
POTIM = 0.20
LCHARG = FALSE
LWAVE = FALSE
#Crude optimisation

```

## w/INCAR

```

PREC = Normal
EDIFF = 1e-3
IBRION = 2
ISIF = 3
NSW = 100
ISMear = 0
SIGMA = 0.060
POTIM = 0.020
LCHARG = FALSE
LWAVE = FALSE
#Crude optimisation

```

```
EDIFFG = 1e-2
KSPACING = 1.256
LREAL = A
ALGO = Fast
```

```
EDIFFG = 1e-2
KSPACING = 0.942
LREAL = A
ALGO = Fast
```

#### e/INCAR

```
PREC = Normal
EDIFF = 1e-4
IBRION = 2
ISIF = 3
NSW = 70
ISMear = 0
SIGMA = 0.060
POTIM = 0.250
ISTART = 0
LCHARG = FALSE
LWAVE = FALSE
#Crude optimisation
EDIFFG = 1e-3
KSPACING = 0.618
#ALGO = Fast
```

#### r/INCAR

```
PREC = Normal
EDIFF = 1e-4
IBRION = 2
ISIF = 2
NSW = 0
ISMear = 0
SIGMA = 0.060
POTIM = 0.250
ISTART = 0
LCHARG = FALSE
LWAVE = FALSE
#Crude optimisation
EDIFFG = 1e-3
KSPACING = 0.618
#ALGO = Fast
```

运行：提交 `magus search` 到队列

结果：搜索结果 `results/good.traj`

## 5.4 $Zn_x(OH)_y$ 变组分结构搜索

目标：搜索 8-16 个原子  $Zn_x(OH)_y$  的结构

准备：input.yaml, inputFold

使用 Gulp 经验势优化，因此种群数目与代数可以大大增加。

input.yaml

给出搜索所需参数

```
calcType: var                # 变组分搜索
pressure: 0
initSize: 150
popSize: 150
numGen: 60
saveGood: 8
#structure parameters
minAt: 8
maxAt: 16
symbols: ['Zn', 'O', 'H']
formula: [[1,0,0],[0,1,1]]  # Zn : (OH) = 1 : 1
fullEles: True
eleSize: 5
dRatio: 0.5
volRatio: 10
addSym: False
#main calculator settings
MainCalculator:
  jobPrefix: ['Gulp1', 'Gulp2']
  # 若 jobPrefix 给出计算器名称, 可不指定 calculator
  mode: parallel
  #gulp settings
  exeCmd: gulp < input > output
  #parallel settings
  numParallel: 5
  numCore: 4
  queueName: e52692v2ib!
  waitTime: 30
```



**Gulp1/gpot**

定义 gulp 所使用的势，其中 ReaxFF.lib 是相应的反应力场文件

```
time 240
space
1
maxcyc 300
library ./ReaxFF.lib
lennard epsilon
Zn  Zn  0.0150 1.00 0.0 8.0
Zn  O   0.0150 1.00 0.0 8.0
Zn  H   0.0150 0.80 0.0 8.0
H   H   0.0150 0.60 0.0 8.0
O   O   0.0150 0.80 0.0 8.0
H   O   0.0150 0.80 0.0 8.0
```

**Gulp1/goption.relax**

relax 所使用命令，conv 代表第一代粗优固定晶格优化

```
opti spatial conj nosymmetry conv
```

类似的：

**Gulp2/gpot**

```
time 240
space
1
maxcyc 300
library ./ReaxFF.lib
```

**Gulp1/goption.relax**

```
opti spatial conj nosymmetry conv
```

运行: 提交 `magus search` 到队列

结果: 搜索结果 `results/good.traj`

## 5.5 $MgSiO_3$ 机器学习结构搜索

目标: 使用 MTP 加速搜索 10-20 个原子  $MgSiO_3$  的结构

准备: `input.yaml`, `inputFold`

与5.2类似, 准备好相应的 `pot.mtp`, `mlip.ini`, 不同的是需要把 `pot.mtp` 中的原子种类改为 3 种

`input.yaml`

搜索所需参数, 计算器部分与5.2类似, 不再赘述

```
calcType: fix
poolSize: 2000      # 预训练生成的代挑选随机结构, 可以设的很大
initSize: 400
popSize: 400
numGen: 60
saveGood: 3

#structure parameters
DFTRelax: False    # 不使用 DFT 验证能量
minAt: 10
maxAt: 20
symbols: ['Mg', 'Si', 'O']
formula: [1, 1, 3]
molDetector: 2     # 2 号分子探测算法
dRatio: 0.8
volRatio: 1.3
randFrac: 0.4
pressure: 150
addSym: True
```

```
softNum: 0
```

MTP/pot.mtp

表头部分，主要区别为分子种类被替换成了 3

```
MTP
version = 1.1.0
potential_name = MTP1m
scaling = 7.002314814814817e-01
species_count = 3
potential_tag =
radial_basis_type = RBChebyshev
    min_dist = 5.000000000000000e-01
    max_dist = 5.000000000000000e+00
    radial_basis_size = 12
    radial_funcs_count = 4
```

运行：提交 `magus search -m` 到队列

结果：搜索结果 `results/good.traj`

## 5.6 $CH_4$ 分子晶体搜索

目标：甲烷晶体结构搜索准备：input.yaml, inputFold, CH4.xyz

input.yaml 可参照5.1与5.3中的设置：

MTP/pot.mtp

```
calcType: fix
initSize: 20
popSize: 20
numGen: 40
saveGood: 3
```

```
pressure: 50
minAt: 20
maxAt: 20
symbols: ['C', 'H']
## mol crystal
molMode: True
molFile: ['CH4.xyz']
molFormula: [4]
molType: 'fix'
chkMol: True
addSym: True

dRatio: 0.8
volRatio: 5
randFrac: 0.4
molDetector: 2

#main calculator settings
MainCalculator:
  jobPrefix: ['Vasp1', 'Vasp2']
  mode: parallel
  #vasp settings
  xc: PBE
  ppLabel: ['', '']
  #parallel settings
  numParallel: 4
  numCore: 24
  queueName: 9242opa!
```

MTP/pot.mtp

```
5
C  H
C    2.260984    1.227715    2.255654
H    2.597307    0.217093    2.238728
H    1.194544    1.227505    2.236584
H    2.611534    1.725297    1.379429
H    2.590593    1.698611    3.156207
```

运行: 提交 `magus search` 到队列

结果: 搜索结果 `results/good.traj`

## 6 常见问题

使用时遇到疑问或 bug 可在<https://git.nju.edu.cn/gaaoooh/magus>中提出 issue.

1. 为啥 pip 安装时报错”ModuleNotFoundError: No module named 'yaml'”?

那你装一个啊

```
$ pip install pyyaml
```