使用 MolAICal 和 LightDock 进行蛋白质-DNA 和蛋白质-RNA 分子对接教程

Qifeng Bai
电子邮箱: molaical@yeah.net
主页: https://molaical.github.io 或 https://molaical.gitlab.io

1. 引言

MolAICal 可以使用开源的 LightDock [1]软件(https://lightdock.org)并通过 MM/GBSA 或 MM/PBSA 方法进一步提高生物大分子对接的准确性。LightDock 是一个开源(GPL-3.0 许可证)的对接框架，用于蛋白质-蛋白质、蛋白质-肽、蛋白质-RNA 和蛋白质-DNA 复合物。它特别擅长处理柔性且具有挑战性的案例(例如，瞬时相互作用、低亲和力复合物，或需要主链/侧链柔性的系统)，并作为一个可扩展平台用于测试新的评分函数、约束条件或优化策略。LightDock 采用了萤火虫群优化(GSO)算法 [2]，这是一种最初为多模态函数优化开发的受生物启发的群体智能方法。

本教程以小型嗜热染色体蛋白 Sac7d（来自嗜酸热硫化叶菌，古菌）、双链 DNA 双链体（蛋白质数据银行编号 PDB ID: 1AZP）以及 PDB ID: 1A1T 中的 RNA 链作为演示案例。用户也可参考 LightDock 官方教程：https://lightdock.org/tutorials 或 https://lightdock.org/tutorials/0.9.3/index.html。

本教程仅适用于在 Linux 操作系统上完成！ 只有部分软件可以在 Windows 上完成。用户可以通过 SSH shell 或其他工具在 Linux 和 Windows 之间传输文件，以便更好地观察运行结果。

2. 材料

2.1. 软件需求

1) MolAICal: https://molaical.github.io or https://molaical.gitlab.io
2) NAMD (CPU 版本): https://www.ks.uiuc.edu/Research/namd/
   (注意: 本教程可以使用 NAMD 2.x、3.x 或更高版本运行。例如，如果您使用 NAMD 3.x 版本，命令"namd3"将替代本教程中的命令"namd2"。对于更高版本的 NAMD，用户可以使用与前面示例类似的替换方式。)
3) PyMol: https://github.com/cgohlke/pymol-open-source-wheels
4) VMD: https://www.ks.uiuc.edu/Research/vmd

2.2. 示例文件

1) 所有必要的教程文件可从以下网址下载:
   https://gitee.com/molaical/tutorials/tree/master/026-protein_pn_docking

3. 操作流程

3.1. 分子准备

为了解决 Linux 中的库依赖问题，Linux 版本的 MolAICal (Windows 版本的 MolAICal 没有采用容器)采用了基于容器的方法。如果需要调用外部程序，建议在 MolAICal 容器内安装这些程序。如果不需要外部程序，可以忽略此步骤。本教程需要外部程序 NAMD 和 VMD，具体步骤如下：

1. 首先，将文件复制到 MolAICal 容器中

```
# 进入容器文件系统（将进入 "/root" 目录）
#> molaical.exe -eset shell in

# 将 VMD 和 NAMD 安装包从本地机器复制到容器中，'cp' 命令的第一部分（源路径）
# 位于本地主机，第二部分（目标路径）在容器内；VMD 和 NAMD 软件包可通过以下命
# 令移入容器。
#> cp /home/user/<本地文件> /root/soft

# 退出容器文件系统
#> exit
```

2. 其次，进入 MolAICal 容器的虚拟环境

```
#> molaical.exe -eset sys run molaical
```

注：molaical 是容器名称。

3. 在 MolAICal 容器虚拟环境中安装软件的方式与在本地主机上安装相同。以下以安装 VMD 和 NAMD 为例：

1) 安装 NAMD：

解压 NAMD 文件（假设解压后的文件夹名为 namdcpu），然后使用以下命令将其路径告知 MolAICal：

```
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注：请将上述 VMD 和 NAMD 的路径替换为您系统中的实际路径。-n 后面的 "VMD" 和 "NAMD"（大小写不敏感）是固定的标识符。为确保 MM/GBSA 结果的可重复性，建议使用 NAMD 的 CPU 版本，因为 CUDA 版本中的 seed 参数似乎对结果可重复性无效。

2) 安装 VMD：

按以下步骤操作：

◆ **解压 VMD 文件：**

```
#> tar -xzvf vmd-xxx.tar.gz
```

注：请将上述路径替换为您系统中的实际路径。

◆ **修改 VMD 解压目录中名为 configure 的文件中的安装路径：**

```
# 默认值：
$install_bin_dir="/usr/local/bin";
$install_library_dir="/usr/local/lib/$install_name";
```

```
# 修改为:
$install_bin_dir="/root/soft/vmd193/bin";
$install_library_dir="/root/soft/vmd193/lib/$install_name";
```

◆ **安装 VMD：**

```
#> cd vmd-xxx
#> ./configure LINUXAMD64
#> cd src
#> make install
```

注：请运行 `./configure` 并根据所用计算机选择正确的类型，此处为 `"LINUXAMD64"`。

◆ **然后使用以下命令将 VMD 路径告知 MolAICal：**

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
```

至此，NAMD 和 VMD 在 MolAICal 容器内的安装与配置已完成。为防止 MolAICal 出现问题时丢失已安装的程序（例如 VMD 和 NAMD），请参阅**附录 1** 中的第 **3 步**。

✧ **记得使用 exit 命令退出 MolAICal 虚拟环境，返回本地计算机进行计算（主要是为了省去文件拷贝步骤；在容器内运行也可行，但需手动将数据从本地计算机复制到 MolAICal 容器中）。**

```
#> exit
```

假设用户已使用以下说明在 MolAICal 中配置了 VMD 和 NAMD 的内部命令:

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"

#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注意: 请将上面 VMD 和 NAMD 的路径替换为用户系统上的实际路径。"-n"后面的字符串"VMD"和"NAMD"(不区分大小写)对于 VMD 和 NAMD 的路径是固定的。为了确保 MM/GBSA 结果的可重现性，建议使用 NAMD 的 CPU 版本，因为在 NAMD 的 CUDA 版本中，"seed"参数对可重现性似乎无效。请注意，Windows 版与 Linux 版的 MolAICal 配置存在差异：Linux 版本采用基于 udocker 容器的技术方案，需在容器内部完成设置，而 Windows 版本则无需此步骤。

打开教程材料文件夹"026-protein_pn_docking ":

```
#> cd 026-protein_pn_docking
```

◆ 将 PDB 文件"1AZP.pdb"加载到 PyMol 中，首先删除复合物的氢原子（可选）和水分子，然后使用图 1 中的步骤分别保存"protein_A.pdb（链 A）“和"DNA_BC.pdb（链 B，C）“。

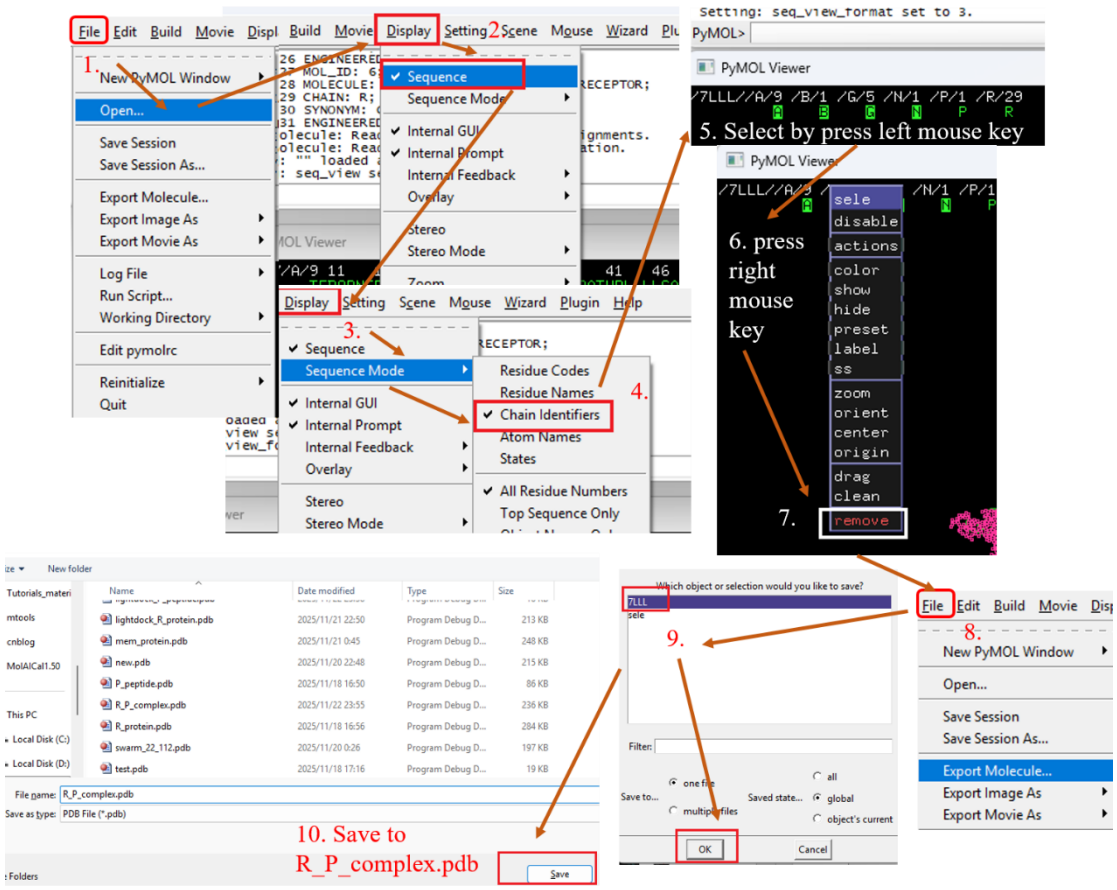

图 1

这里，“DNA_BC.pdb"包含两个不同的链，B 和 C，这不符合两个分子对接的要求，因此需要将"DNA_BC.pdb"的两个链合并为单个链 B。命令如下：

```
#> molaical.exe -call run -c vmdargs -i -pdb DNA_BC.pdb -args none set sel [atomselect top all], \\\$sel set chain B, \\\$sel writepdb DNA_B.pdb
```

注意:

- 它包括外部程序的所有命令行参数，但'**-i**'参数必须是最后指定的，而所有其他标识符(例如，'-call'、'-c'等)必须在'-i'之前。
- '**-pdb**'表示将 pdb 文件加载到 VMD 中。
- **-c:** 如果其值为"vmdargs"，它将在 Tk 控制台中运行 VMD 命令一次。
- '-args'后面的字符串表示可以在 VMD 的 Tk 控制台中运行的命令。
  - '-args'后的第一个参数可以是 VMD 的包名称或'none'。如果'-args'后第一个参数是'none'，它将省略加载包。例如，如果'-args'后第一个参数是'autopsf'，它将在 VMD 的 Tk 控制台中运行命令"package require autopsf"。如果'-args'后第一个参数是'none'，它将不会运行命令"package"。
  - '-args'后面的字符串包含一个可以在 VMD 的 Tk 控制台中执行的命令，每个逗号','之前；换句话说，逗号字符代替了在 VMD 的 Tk 控制台中输入每个命令并按 Enter 键的过程。这使得只需一个命令就可以执行多行命令。
  - 一些特殊字符，例如字符"$"，需要使用转义字符""。对于特殊字符，MolAICal 使用三个转义字符""(即"\")。

◆ 以同样的方式，将 PDB 文件"1A1T.pdb"加载到 PyMol 中，首先删除复合物的氢原子（可选）和水分子，然后使用图 1 中的步骤保存"RNA_B.pdb（链 B）“。

3.2. 制作残基约束文件

为残基创建约束文件类似于识别活性口袋中的关键残基位点，分子将在其周围进行对接。用户可以根据文献介绍或直接根据经验自定义这些关键残基。

约束文件中残基的表示格式为:

```
Chain.Residue_Name.Residue_Number[.Residue_Insertion]
```

注意: 这里，Residue_Insertion 是一个可选的单字符标识符，附加到标准残基编号后面，通常用于 PDB 文件中区分插入残基而不重新编号整个序列。在生物信息学和结构生物学中，插入代码对于准确跟踪和描述特定蛋白质残基至关重要，特别是在处理序列变异或比较时。

当省略 Residue_Insertion 时，一般格式为:

```
Chain.Residue_Name.Residue_Number
```

或者

```
Chain.Residue_Name.Residue_Number P
```

注意: 标签"P"表示此约束是被动的(与没有标签的主动约束相反)。除非完全理解其含义，否则不推荐使用被动残基约束。

在这里，MolAICal 用于生成受体(蛋白质)在配体(肽)4.0Å 范围内的残基列表文件(命名为'rec.dat')作为关键残基，如下所示:

```
#> molaical.exe -call run -c sfile -i get_res.tcl -pdb 7LLL.pdb -args R P 4.0 rec.dat R
```

注意: 以下 1st、2nd、3rd 和 4th 分别表示第一个参数、第二个参数、第三个参数和第四个参数，依此类推。

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本，这里调用脚本文件'get_res.tcl'。
- '-pdb'表示将 pdb 文件加载到 VMD 中。
- 1st: 所选分子的链名；
- 2nd: 用于选择的参考分子的链名；
- 3rd: 在指定距离内从参考分子选择的所选分子的残基；
- 4th: 保存的文件名；
- 5th: 指定分子类型，'R'代表受体，'L'代表配体。

类似地，MolAICal 用于生成配体(肽)在受体(蛋白质)4.0Å 范围内的残基列表文件(命名为'lig.dat')作为关键残基，如下所示:

```
#> molaical.exe -call run -c sfile -i get_res.tcl -pdb 7LLL.pdb -args P R 4.0 lig.dat L
```

将'rec.dat'和'lig.dat'合并到名为'restraints.list'的约束文件中，如下所示:

```
#> molaical.exe -call run -c ecommand -i cat rec.dat lig.dat > restraints.list
```

在许多情况下，配体的具体细节是未知的。**在本教程中，不使用上述命令，**上述命令仅提供更多信息选择。本教程仅使用受体（第一个分子）的关键残基，**这些残基是根据经验和文献报告手动添加到**名为"restraints.list"的约束文件中，其内容如下：

```
R A.TRP.24

R A.VAL.26

R A.ARG.42
```

3.3. 蛋白质和核酸的准备

蛋白质伙伴必须首先拥有在 DNA 评分函数中参数化的氢原子（基于 AMBER94 力场）。如果下面的命令不起作用，请将下面的命令替换为 UCSF Chimera（https://www.cgl.ucsf.edu/chimera）的操作（例如，删除和添加氢原子），该软件基于 AMBER 力场处理分子。

3.3.1. 蛋白质

运行以下命令：

删除氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -Trim protein_A.pdb > protein_A_noH.pdb
```

添加氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -BUILD protein_A_noH.pdb > protein_A_H.pdb
```

重新编号原子：

```
#> molaical.exe -call run -c ldock -i pdb_reatom protein_A_H.pdb > protein.pdb
```

3.3.2. DNA

运行以下命令：

删除氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -Trim DNA_B.pdb > DNA_B_noH.pdb
```

对于添加氢原子，核酸（DNA 或 RNA）存在"resname"问题：“reduce"命令似乎对 resname 的位置很敏感，要求在核酸允许的 resname 范围内，名称必须右对齐。**这个问题来自上面 VMD 的链设置和保存操作。**

修复 DNA 的 resname：

```
#> molaical.exe -call run -c sfile -i fix_pdb.tcl -args DNA_B_noH.pdb DNA_B_nh.pdb
```

添加氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -BUILD DNA_B_nh.pdb > DNA_B_H.pdb
```

更新氢原子 H 类型。“reduce_to_amber.py"中的代码是：“*atom.name = translation[atom.name]*"，其中 translation 包含更改的原子类型。

```
#> molaical.exe -call run -c sfile -i reduce_to_amber.py DNA_B_H.pdb DNA_dh.pdb
```

删除与 AMBER94 力场冲突的氢原子"HO'3"和"HO'5"。“purge_prime_H.py"中的代码是：“*if atom.name in ("HO'3", “HO'5"): continue*"，其中 translation 包含更改的原子类型。

```
#> molaical.exe -call run -c sfile -i purge_prime_H.py DNA_dh.pdb DNA.pdb
```

3.3.3. RNA

运行以下命令：

删除氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -Trim RNA_B.pdb > RNA_nh.pdb
```

添加氢原子：

```
#> molaical.exe -call run -c ldock -i reduce -BUILD RNA_nh.pdb > RNA_h.pdb
```

对于 RNA 分子的进一步准备，仍需满足以下三个要求：

- ✧ "reduce"为核酸添加的氢原子与 AMBER94 力场不完全名称兼容。（reduce_to_amber.py：用于重命名和/或删除不兼容原子类型的脚本）。
- ✧ 此外，Reduce 在 RNA 的末端（3'和/或 5'端）追加氢原子，这与 AMBER94 力场冲突。（purge_prime_H.py：用于移除末端（3'和/或 5'）氢原子的脚本）。
- ✧ 最后，AMBER94 力场要求 RNA 残基标记有"R"前缀。（retag_rna.py：用于为 RNA 残基添加或删除"R"前缀的脚本）

1） 在 DNA 的原始 resname 前添加"R"。“retag_rna.py"中的代码是"*atom.residue_name = "R" + atom.residue_name*"

```
#> molaical.exe -call run -c sfile -i retag_rna.py RNA_h.pdb rna_pre.pdb
```

2）更新 H 类型。“reduce_to_amber.py"中的代码是：“*atom.name = translation[atom.name]*"，其中 translation 包含更改的原子类型。

```
#> molaical.exe -call run -c sfile -i reduce_to_amber.py rna_pre.pdb rna_pre2.pdb
```

3）删除与 AMBER94 力场冲突的氢原子"HO'3"和"HO'5"。“purge_prime_H.py"中的代码是：“*if atom.name in ("HO'3", "HO'5"): continue*"，其中 translation 包含更改的原子类型。

```
#> molaical.exe -call run -c sfile -i purge_prime_H.py rna_pre2.pdb rna_tagged.pdb
```

4） “retag_rna.py"的"-r"参数表示为设置移除添加的标签"R"。

```
#> molaical.exe -call run -c sfile -i retag_rna.py -r rna_tagged.pdb RNA.pdb
```

3.3.4. 为 DNA 和 RNA 分别创建工作目录

为了避免 DNA 和 RNA 项目之间的文件冲突，应将它们移动到不同的目录，如下命令所示：

```
# 1. 通过下面的命令为 DNA 和 RNA 构建工作目录：

#> molaical.exe -call run -c ecommand -i mkdir DNA RNA
```

```
# 2. 将蛋白质和 DNA 文件移动到 DNA 文件夹，命令如下：
```

```
#> molaical.exe -call run -c ecommand -i cp -f protein.pdb DNA

#> molaical.exe -call run -c ecommand -i cp -f DNA.pdb DNA
```

```
# 3. 将蛋白质和 RNA 文件移动到 RNA 文件夹，命令如下：

#> molaical.exe -call run -c ecommand -i cp -f protein.pdb RNA

#> molaical.exe -call run -c ecommand -i cp -f RNA.pdb RNA
```

3.4. 分子对接设置和模拟

首先，运行群体(swarm)生成的设置，如下所示:

1）对于 DNA，打开 DNA 文件夹，运行设置和模拟：

```
#> cd DNA
```

```
#> molaical.exe -call run -c ldock -i lightdock3_setup.py protein.pdb DNA.pdb --noxt -sp -rst ../restraints.list
```

```
#> molaical.exe -call run -c ldock -i lightdock3.py setup.json 100 -s dna -c 8
```

2）对于 RNA，打开 RNA 文件夹，运行设置和模拟：

```
#> cd ../RNA
```

```
#> molaical.exe -call run -c ldock -i lightdock3_setup.py protein.pdb RNA.pdb --noxt -sp -rst ../restraints.list
```

```
## 模拟需要 AMBER94 力场，这要求在 RNA 的"resname"前添加 R 标签

#> molaical.exe -call run -c ecommand -i mv lightdock_RNA.pdb lightdock_rna_untagged.pdb

#> molaical.exe -call run -c sfile -i retag_rna.py lightdock_rna_untagged.pdb lightdock_RNA.pdb
```

```
#> molaical.exe -call run -c ldock -i lightdock3.py setup.json 100 -s dna -c 8
```

- '-call': 与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- -c: 如果其值为"ldock"，它将通过调用"LightDock"运行分子对接(蛋白质-蛋白质、蛋白质-肽、蛋白质-DNA 或蛋白质-RNA)。
- "lightdock3_setup.py"后的第一个和第二个参数分别是受体和配体文件。
- --noxt: 如果启用此选项，LightDock 将忽略 OXT 原子。这对于几种不理解这种特殊类型原子的评分函数很有用。

- ✧ --anm: 如果启用此选项，将激活 ANM 模式，并使用 ANM(通过 ProDy)建模主链柔性。该参数通过 ANM（使用 ProDy）启用骨架柔性建模，可能导致对接结果的结构畸变。**除非实施更优构象采样（如能量最小化），否则应避免使用。**替代方案：预生成多重构象以规避对接过程中的骨架柔性问题。
- ✧ -sp: 如果启用(默认为 False)，将在 init 文件夹中写入调试额外文件。
- ✧ -r, -rst restraints_file: 如果提供了 restraints_file，在设置和模拟过程中将考虑残基约束。如果没有提供 restraints_file，设置和模拟将集中在整个受体上。
- ✧ "lightdock3.py"后的第一个和第二个参数分别是设置步骤生成的配置文件和模拟的步数(这里是 100 步)。
- ✧ -s SCORING_FUNCTION: 使用此标志的默认评分函数(DFIRE，快速 C 实现 fastdfire)。接受评分函数的名称或包含多个评分函数名称和权重的文件。
- ✧ -c CORES: 默认情况下，使用硬件上可用的 CPU 核心总数来运行模拟，但可以通过此选项指定不同数量的 CPU 核心。

- ■ 它将生成名为'**init**'的文件夹，该文件夹包含以 PDB 格式表示的分子，这些分子代表了用于进一步对接的萤火虫群。
- ■ 新生成的文件是：'lightdock_protein.pdb'，'lightdock_DNA.pdb'和'lightdock_RNA.pdb'

通过 PyMol 以类似于图 2 的方式打开'lightdock_protein.pdb'和'**init**'文件夹中所有前缀为"starting_positions*"的 PDB 文件：

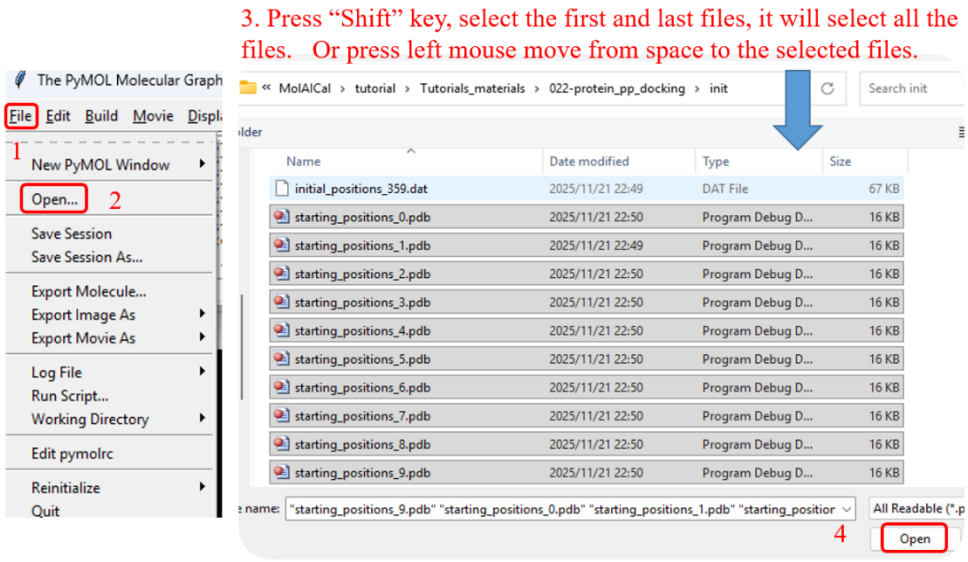

图 2

请参见图 3 中的结果。很明显，许多萤火虫位于蛋白质周围。文件"restraints.list"可以限制萤火虫的数量；没有此约束文件，将生成更多萤火虫。

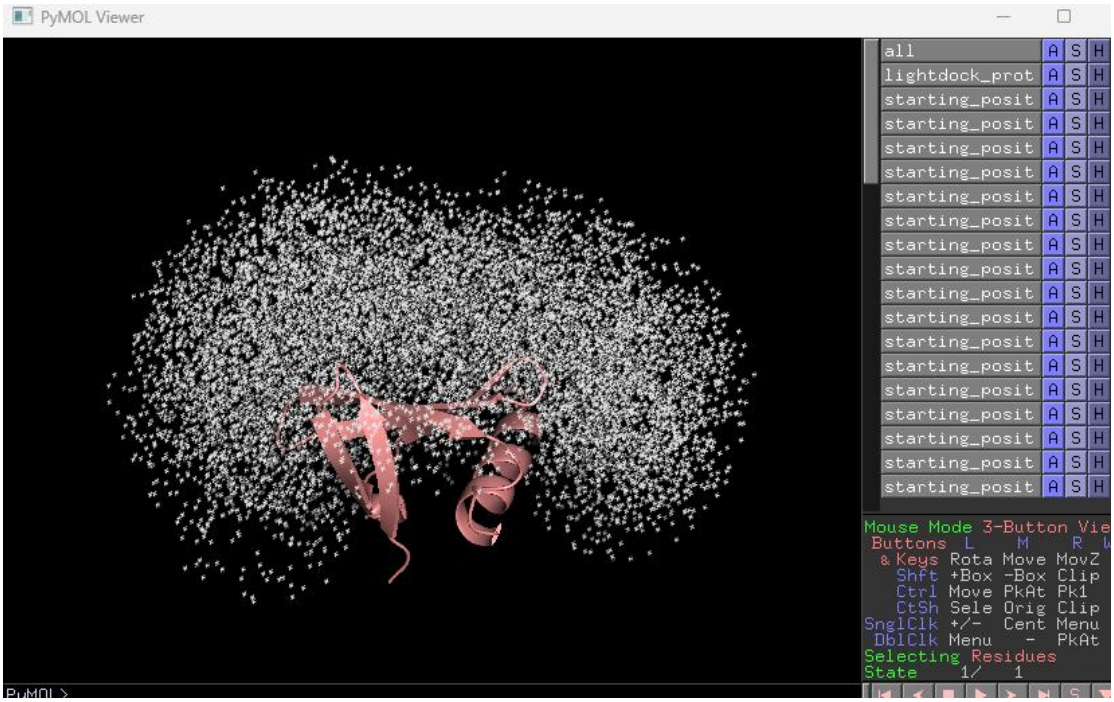

图 3

3.5. 生成模型、聚类、排名和过滤

模拟完成后，MolAICal 调用脚本来生成模型、聚类、排名和过滤结果，如下所示：

1）对于 DNA

```
#> molaical.exe -call run -c sfile -i lrank.sh 1::=protein.pdb 2::=DNA.pdb 3::=A 4::=B 5::=8
6::=../restraints.list 13::=0.7 17::=--lnuc
```

2）对于 RNA

此步骤需要移除 RNA 的"R"标签。

```
#> molaical.exe -call run -c ecommand -i mv lightdock_RNA.pdb lightdock_RNA_tagged.pdb

#> molaical.exe -call run -c ecommand -i mv lightdock_rna_untagged.pdb lightdock_RNA.pdb
```

注意：似乎只有 RNA 模拟步骤和"reduce_to_amber.py"需要"R"标签以适应 AMBER 力场，因此可以独立准备未标记和已标记的文件。

```
#> molaical.exe -call run -c sfile -i lrank.sh 1::=protein.pdb 2::=RNA.pdb 3::=A 4::=B 5::=8
6::=../restraints.list 13::=0.7 17::=--lnuc
```

- '-call'：与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。
- '::='：在此，它用于按指定顺序为参数赋值。
- '-c'：如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本。
- '1::='：它是第一个输入分子（受体）的路径。必需参数。
- '2::='：它是第二个输入分子（配体）的路径。必需参数。
- '3::='：受体伙伴（第 1 个分子）上的链。默认值为'R'。
- '4::='：配体伙伴（第 2 个分子）上的链。默认值为'P'。
- '5::='：使用的 CPU 核心数。默认值为 12。
- '6::='：包含约束的文件。默认值为'restraints.list'。
- '13::='：用于过滤步骤的，至少具有此比例本地接触的结构。默认值为 0.4。为了识别与受体活性位点中更多关键残基结合的配体，此处将约束比例设置为 0.7。
- '17::='：用于"lgd_filter_restraints.py"的更多选项。默认值为空字符串，空字符串表示为""。如果此输入包含多个参数，使用逗号字符','分隔它们。MolAICal 将自动将每个逗号','转换为空格" "。此处，参数"--lnuc"代表是核酸分子。
- 其他参数使用默认值。更多信息，请参阅 MolAICal 手册。

聚类和过滤过程完成后，将生成一个名为**"filtered"**的新目录。该目录保存所有符**合默认 70%过滤标准的预测结构**。其中，将找到一个名为**"rank_filtered.list"**的文件，该文件根据"dna"分数对这些结构进行排名，其中更高（更正）的分数表示更好的排名。

- dna 分数：实现 pyDockDNA 评分函数（无去溶剂化）和用于蛋白质-DNA 对接的自定义范德华权重。使用 Python C-API 实现。它也适用于 RNA 评分。

表 1

核酸类型	名称	分数	百分比
DNA	swarm_7_101.pdb	217.290	1.000
	swarm_13_71.pdb	175.002	1.000
	swarm_48_142.pdb	172.741	1.000
	……..	……..	……..
RNA	swarm_18_31.pdb	237.259	1.000
	swarm_46_142.pdb	218.034	1.000
	swarm_14_41.pdb	213.206	1.000
	……..	……..	……..

注意：与> **13::=0.7** 对应的百分比通过 len(contacts_receptor & restraints_receptor) + len(contacts_ligand & restraints_ligand) / (len(restraints_receptor) + len(restraints_ligand))计算。分子命名格式：swarm_{id_swarm}_{id_glowworm}.pdb，其中："id_swarm" 对应"setup"阶段生成的 swarm 文件夹名称；"id_glowworm" 指该 swarm 文件夹内 PDB 格式的分子名称。

表 1 仅显示部分对接结果的分数。将"1AZP.pdb（DNA 复合物）"或"1A1T.pdb（RNA 复合物）"（即天然结合构象）和表 1 中的分子加载到 PyMol 中，将"1AZP.pdb（DNA 复合物）"或"1A1T.pdb（RNA 复合物）"与所需的分子对齐（参见图 4）。对齐后，可以进一步查看对接的构象等结果。

此外，用户可以将最佳的一个或指定的前几个分子放入新文件夹"filtered/candidates"，命令如下：

```
#> molaical.exe -call run -c sfile -i mrank.py -ft 1
```

➢ **-t**：复制到候选者的前几个分子数量（默认：0）

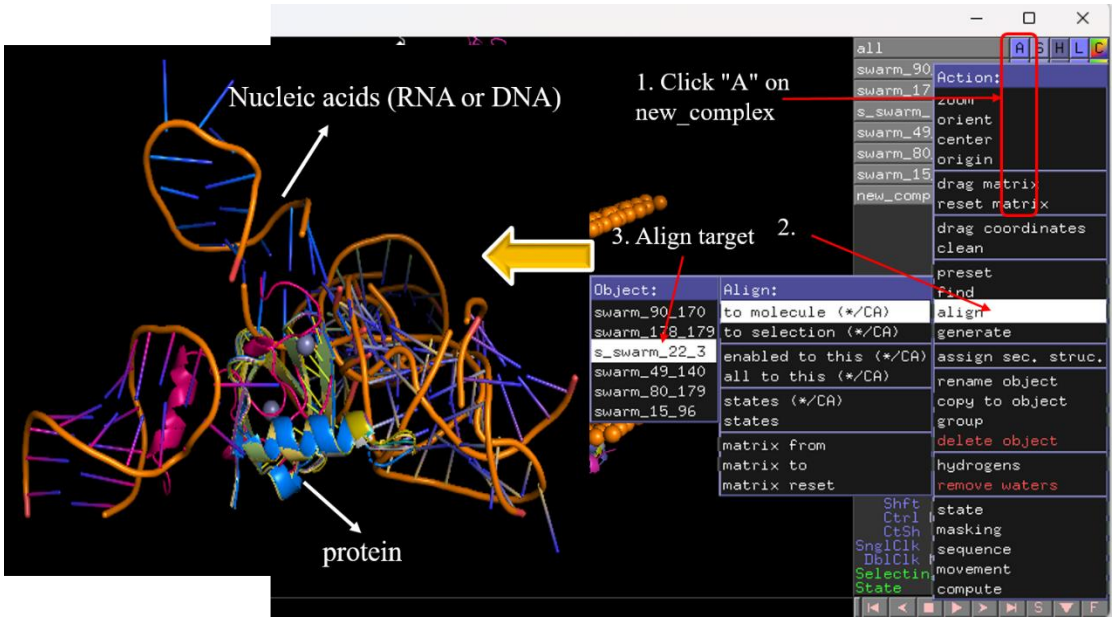

图 4

3.6. 进一步优化的 MM/GBSA 计算

MolAICal 计算 MM/GBSA 和 MM/PBSA 结合自由能。根据文献证据和经验验证，对于蛋白质-蛋白质和蛋白质-核酸对接系统，MM/GBSA 的表现优于 MM/PBSA。然而，这并不意味着 MM/GBSA 在所有系统中都优于 MM/PBSA。我们建议研究人员根据其特定的生物分子系统和计算目标选择最佳方法。此处，工作流程采用 MM/GBSA 计算对接姿态的结合亲和力，命令如下：

1）对于 DNA

```
#> molaical.exe -call run -c sfile -i mmgbpbsa_batch.sh 1::=A 2::=B
```

2）对于 RNA

```
#> molaical.exe -call run -c sfile -i mmgbpbsa_batch.sh 1::=A 2::=B
```

✧ '-call'：与外部程序或命令交互。其值可以是'set'或'run'。'set'表示设置外部程序的环境，包括名称和路径。'run'表示调用外部程序，可以从设置的环境文件中搜索程序的路径。

✧ '::='：在此，它用于按指定顺序为参数赋值。

✧ '-c'：如果其值为"sfile"，它将运行 MolAICal 的 VMD 脚本。

✧ '1::='：第一个分子的链。默认值为'R'。

✧ '2::='：第二个分子的链。默认值为'P'。

✧ '3::='：使用的 CPU 核心数。默认值为 12。

✧ 其他参数使用默认值。更多信息，请参阅 MolAICal 手册。

注意：

1）首次运行时将自动生成参数文件**"cal_mmgbpbsa.sh"**。若该文件已存在，后续运行不会覆盖，以目录中现有版本为准。用户可自定义 **"cal_mmgbpbsa.sh"** 中的参数，此脚本遵循 MolAICal 格式，支持逐行批量执行 MolAICal 命令（命令行通常以 molaical.exe 开头）

2）MolAICal 目前仅支持对标准蛋白质残基和核酸自动计算 MM/GBSA 和 MM/PBSA。对于小分子和非标准氨基酸，需要额外的 CHARMM 力场，并且可以通过添加"-top"和"-ff"等参数来修改目录中的"cal_mmgbpbsa.sh"脚本，以扩展额外的 CHARMM 力场。具体来说，更改 "cal_mmgbpbsa.sh"的第 5 行可以将计算切换到 MM/PBSA。详细信息可以在 MolAICal 手册和相应的 MM/GBSA 和 MM/PBSA 教程中找到。

对'./mmgbpbsa/candidates'中的分子进行排名或提取所需的分子，命令如下：

1. 仅打印 MM/GB/PBSA 的排名结果，运行以下命令：

```
#>molaical.exe -call run -c sfile -i mrank.py
```

2. 将结果的前 2 个分子复制到'./mmgbpbsa/candidates'，运行以下命令：

```
#> molaical.exe -call run -c sfile -i mrank.py -t 2
```

➢ -t：复制到候选者的前几个分子数量（默认：0）

它将显示如下结果：

1）对于 DNA

Name	G_binding (kcal/mol)	±	SE	SD
swarm_7_101.pdb	-33.6310	+/-	0.0389	0.2750
swarm_49_130.pdb	-18.4513	+/-	0.0032	0.0225
swarm_13_71.pdb	-17.9355	+/-	0.0163	0.1154
swarm_22_101.pdb	-10.2513	+/-	0.0042	0.0297
swarm_33_151.pdb	-5.6938	+/-	0.0115	0.0814
swarm_48_142.pdb	-1.5483	+/-	0.0135	0.0952

2）对于 RNA

Name	G_binding (kcal/mol)	±	SE	SD
swarm_14_41.pdb	-25.2272	+/-	0.0159	0.1123
swarm_18_31.pdb	-22.5430	+/-	0.0067	0.0471
swarm_14_142.pdb	-19.1931	+/-	0.0147	0.1041
swarm_46_142.pdb	-10.4946	+/-	0.0197	0.1395
swarm_18_83.pdb	-8.8363	+/-	0.0149	0.1052
swarm_33_158.pdb	-6.7762	+/-	0.0346	0.2444

一般情况下，G_binding 值越负越好。

注意：本教程提供详细的多步骤指导，**主要用于教学目的。**教程中的步骤已更新，但示例结果未同步刷新；若你的输出与教程不同，**以你的实际结果为准。**

若程序被异常终止（例如通过键盘快捷键 "Ctrl+C" 或其他非标准方式），**大量残留进程将持续失控运行**。为解决此问题，用户可执行以下命令，删除当前目录下所有关联进程：

```
#> molaical.exe -eset pdclean
```

重要提示：

- **重复执行：**需多次运行此命令以确保彻底清除相关进程。
- **严重警告：**该操作会同时终止同目录下其他正在运行的程序进程。
- 执行前必须严格验证：
  - 若当前目录存在多个运行中的程序，须明确确认哪些目录关联进程可安全删除。
  - 切勿在共享目录/生产环境目录中执行此命令，除非经过充分风险评估与明确验证。

参考文献

1. Jiménez-García, B.; Roel-Touris, J.; Romero-Durana, M.; Vidal, M.; Jiménez-González, D.; Fernández-Recio, J., LightDock: a new multi-scale approach to protein-protein docking. Bioinformatics 2018, 34 (1), 49-55.
2. Krishnanand, K. N.; Ghose, D., Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intelligence 2009, 3 (2), 87-124.

附录 1

在 MolAICal 容器内安装外部程序（推荐，适用于 Linux 版本的 MolAICal）

请注意，Windows 版与 Linux 版的 MolAICal 配置存在差异：Linux 版本采用基于 udocker 容器的技术方案，需在容器内部完成设置，而 Windows 版本则无需此步骤。

1. 首先，将文件复制到 MolAICal 容器中

```
# 进入容器文件系统（将进入 "/root" 目录）
#> molaical.exe -eset shell in

# 将 VMD 和 NAMD 安装包从本地机器复制到容器中，'cp' 命令的第一部分（源路径）位于
# 本地主机，第二部分（目标路径）在容器内；VMD 和 NAMD 软件包可通过以下命令移入容
# 器。
#> cp /home/user/<本地文件> /root/soft

# 退出容器文件系统
#> exit
```

2. 其次，进入 MolAICal 容器的虚拟环境

```
#> molaical.exe -eset sys run molaical
```

注：molaical 是容器名称。

3. 在 MolAICal 容器虚拟环境中安装软件的方式与在本地主机上安装相同。以下以安装 VMD 和 NAMD 为例：

1) 安装 NAMD：

解压 NAMD 文件（假设解压后的文件夹名为 namdcpu），然后使用以下命令将其路径告知 MolAICal：

```
#> molaical.exe -call set -n NAMD -p "/root/soft/namdcpu/namd3"
```

注：请将上述 VMD 和 NAMD 的路径替换为您系统中的实际路径。-n 后面的 "VMD" 和 "NAMD"（大小写不敏感）是固定的标识符。为确保 MM/GBSA 结果的可重复性，建议使用 NAMD 的 CPU 版本，因为 CUDA 版本中的 seed 参数似乎对结果可重复性无效。

2) 安装 VMD：

按以下步骤操作：

◆ **解压 VMD 文件：**

```
#> tar -xzvf vmd-xxx.tar.gz
```

注：请将上述路径替换为您系统中的实际路径。

◆ **修改 VMD 解压目录中名为 configure 的文件中的安装路径：**

```
# 默认值：
$install_bin_dir="/usr/local/bin";
$install_library_dir="/usr/local/lib/$install_name";
```

```
# 修改为:
$install_bin_dir="/root/soft/vmd193/bin";
$install_library_dir="/root/soft/vmd193/lib/$install_name";
```

◆ **安装 VMD：**

```
#> cd vmd-xxx
#> ./configure LINUXAMD64
#> cd src
#> make install
```

注：请运行 `./configure` 并根据所用计算机选择正确的类型，此处为 `"LINUXAMD64"`。

◆ **然后使用以下命令将 VMD 路径告知 MolAICal：**

```
#> molaical.exe -call set -n VMD -p "/root/soft/vmd193/bin/vmd"
```

至此，NAMD 和 VMD 在 MolAICal 容器内的安装与配置已完成。

✧ **记得使用 exit 命令退出 MolAICal 虚拟环境，返回本地计算机进行计算（主要是为了省去文件拷贝步骤；在容器内运行也可行，但需手动将数据从本地计算机复制到 MolAICal 容器中）。**

```
#> exit
```

3) 保存并加载已修改的容器（可选）

为保留容器内所做的更改，并避免日后重新安装软件（因为一旦容器被删除，所有数据都会丢失），可执行以下操作：

✧ **获取容器 ID 和名称：**

```
#> molaical.exe -eset sys ps
```

✧ **克隆该容器：**

```
#> molaical.exe -eset sys export --clone -o molaicalv2.tar molaical
```

注：`molaical` 是容器名称，也可使用容器 ID。如果报错，那可能是因为挂载的文件系统的权限问题，可以忽略。

✧ **假如需要，导入克隆的容器：**

```
#> molaical.exe -eset sys import --clone --name molaicalv2 molaicalv2.tar
```

✧ **将新导入的容器名称更改为默认容器名 `"molaical"`，原容器重命名为 `"bakmolaical"`：**

```
#> molaical.exe -eset sys rename molaical bakmolaical
#> molaical.exe -eset sys rename molaicalv2 molaical
```

注意：容器（动态）是从镜像（静态）生成的。软件安装等操作必须在动态容器中进行；如果克隆了该容器，恢复时仍基于原始底层镜像。

更多详情请参阅 MolAICal 用户手册，或运行以下命令获取帮助：

```
#> molaical.exe -eset sys --help
```