按分子清单组织构象生成、失败记录和CPU并发

完整操作教程 · 参考固定 · 往返恢复验收

RDKit批量生成三维构象，先保证分子身份和化学状态，再讨论CPU并发。SMILES能解析、嵌入成功、力场优化收敛是三道不同关卡。本篇对已经完成化学标准化的单组分配体生成多个ETKDGv3构象，保留一个已收敛的

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

RDKit批量生成三维构象，先保证分子身份和化学状态，再讨论CPU并发。SMILES能解析、嵌入成功、力场优化收敛是三道不同关卡。本篇对已经完成化学标准化的单组分配体生成多个ETKDGv3构象，保留一个已收敛的低能候选，并为失败分子写明状态。

输入先明确质子化与立体化学

ligands.tsv包含id、smiles两列表头，TAB分隔，例如L001与CCO、L002与c1ccccc1。ID唯一且不可用空值。盐拆分、互变异构、质子化及立体状态由项目方案先确定；代码拒绝多组分和未指定四面体中心，只覆盖部分常见检查，不代表完成了全部化学标准化。

步骤	本例做法	验收
解析	MolFromSmiles并加氢	失败不混入成功SDF
构象	ETKDGv3、每分子20个示例	可能少于请求数
优化	MMFF参数完整后优化	仅接受返回状态0
选择	同分子已收敛构象最低能	不可跨分子比较绝对能量
输出	每分子SDF与status.csv	失败列表不能丢失

生成构象并记录每条结果

将代码保存为conformers.py，在装有兼容RDKit、NumPy的环境运行。随机种子、20个构象、去重阈值0.5 Å及优化迭代200均为教学协议，正式项目按分子柔性与目标验证。嵌入和优化内部各用一线程，先建立顺序基线，再测试外层进程并发。

```
import csv,math,re,sys,numpy as np
from pathlib import Path
from rdkit import Chem,rdBase
from rdkit.Chem import AllChem
with open("ligands.tsv",newline="") as f:rows=list(csv.DictReader(f,delimiter="\t"))
ids=[r["id"] for r in rows]
assert rows and len(set(ids))==len(ids) and all(re.fullmatch(r"[A-Za-z0-9_-]+",x) for x
in ids)
root=Path("conformers-001");root.mkdir();(root/"rdkit-version.txt").write_
text(rdBase.rdkitVersion)
failed=0
with (root/"status.csv").open("x",newline="") as f:
    report=csv.writer(f);report.writerow(["id","status","embedded","energy_kcal_
mol","detail"])
    for row in rows:
        try:
            mol=Chem.MolFromSmiles(row["smiles"])
            if mol is None:raise ValueError("SMILES parse failed")
            if len(Chem.GetMolFrags(mol))!=1:raise ValueError("multiple components")
            if any(tag=="?" for _,tag in
Chem.FindMolChiralCenters(mol,includeUnassigned=True)):
                raise ValueError("unassigned tetrahedral center")
            mol=Chem.AddHs(mol)
            params=AllChem.ETKDGv3();params.randomSeed=2026;params.numThreads=1;
params.pruneRmsThresh=0.5
            confs=list(AllChem.EmbedMultipleConfs(mol,numConfs=20,params=params))
            if not confs:raise ValueError("no conformer embedded")
```

```
            if not AllChem.MMFFHasAllMoleculeParams(mol):raise ValueError("missing MMFF
parameters")
            opt=AllChem.MMFFOptimizeMoleculeConfs(mol,numThreads=1,maxIters=200)
            assert len(opt)==len(confs)
            valid=[(energy,cid) for cid,(flag,energy) in zip(confs,opt) if flag==0 and
math.isfinite(energy)]
            if not valid:raise ValueError("no converged finite-energy conformer")
            energy,cid=min(valid);assert
np.isfinite(mol.GetConformer(cid).GetPositions()).all()
            mol.SetProp("_Name",row["id"]);mol.SetDoubleProp("MMFF_energy_kcal_
mol",energy)
            writer=Chem.SDWriter(str(root/(row["id"]+".sdf")))
            writer.write(mol,confId=int(cid));writer.close()
            report.writerow([row["id"],"ok",len(confs),energy,""])
        except Exception as e:
            failed+=1;report.writerow([row["id"],"failed","","",str(e)])
        f.flush()
sys.exit(1 if failed else 0)
```

重读SDF再安排后续用途

成功表中的每个ID应对应一份可重读SDF，原子/键、立体标识、分子名和有限三维坐标与预期一致。嵌入数量可能因RMS去重少于二十，不能因此认定失败；优化返回非零则不能无说明当成收敛。整批有失败时脚本返回非零并保留成功产物，任务中心应显示部分完成。

MMFF能量用于同一分子、同一模型下比较构象，不是结合自由能或活性。最低能的游离构象也不一定是受体中的结合姿势；需要构象集合时应按批准的能量/RMS规则导出多份，不能把本例单个最低能结构当成所有对接方案的唯一正确输入。

解析错误先查键价和芳香表示；缺参数先核元素与化学范围；大环或柔性分子嵌入失败时另立重试协议，记录改动而不是无限增加构象数。不要自动切换力场并仍用同一列能量排序，更不能将未确定的质子化或立体状态随机补齐后冒充原分子。

先用代表分子测CPU与内存

用简单、柔性、大环等代表类别记录每分子完整耗时、内存和成功率，不能只以平均分子大小估计全库。外层进程数乘内部线程数共同计算，逐步增加并发观察吞吐；有些分子会成为长尾任务，全部ID均有明确结果后才能验收批次。

保存RDKit版本、输入校验、标准化方案、随机种子与参数，升级后先回归同一小清单。生成三维构象不自动完成对接需要的电荷、原子类型、受体准备或指定文件格式转换，这些步骤另按目标引擎验收；资源选择以必要的完整预处理任务为依据。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

RDKit官方三维构象入门 · 来源资料

ETKDG与EmbedMultipleConfs真实接口 · 来源资料

MMFF完整性与优化返回值 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。