课程论文题目

邹胜明

摘要

蛋白质-配体对接问题在药物发现过程中起着至关重要的作用。一个成功的蛋白质配体对接方法取决于两个关键因素: 一个有效的搜索策略和一个有效的评分函数。经典方法在算法有效性以及精度等方面具有一定的局限性。待复现论文尝试使用自适应差分进化 (DE) 算法作为搜索策略。通过引入参数自适应方案和改进的变异策略，提高了算法的搜索能力。

关键词：蛋白质对接；DE 算法；自适应机制

1 引言

随着医药学不断发展，众多新型药物结构被提出，旨在改善相关疾病的治疗过程。如何在庞大的药物中快速筛选出有效的药物结构成为了医药领域的重要问题。计算机辅助药物设计既可用于先导物的衍生，也可以用于先导物的优化，是实现基于结构和基于性质的药物设计的技术手段。

2 相关工作

差分进化 (DE) 是一种用于数值优化的强大的元启发式算法。自上世纪 90 年代 Storn 和 Price 首次提出[1]以来，DE 已经引起了计算智能研究人员的极大关注。DE 遵循通用进化算法的框架，由初始化、变异、交叉和选择四个基本部分组成。与其他常见进化算法（EAs）不同的是，DE 利用了当前种群中个体差异的信息。这一特性使得 DE 及其变体的性能优于其他 EAs。由于 DE 的可用性和有效性，它已扩展到许多复杂的优化任务。众所周知，DE 的优化性能对其控制参数和变异策略[2]非常敏感。因此，提高 DE 性能的研究主要集中在两个方面: 微调控制参数[3]和设计有效的突变策略[4]。具体来说，具有参数自适应机制的 DE 变体 (称为自适应 DE) 。论文尝试使用自适应差分进化 (DE) 算法作为搜索策略。通过引入参数自适应方案和改进的变异策略，提高了算法的搜索能力。

2.1 DE 方法

蛋白质-配体对接，将蛋白质视为刚性物质，配体视为柔性物质，那么空间位置、旋转角度以及配体可旋转建的扭角可以组合成一个向量，这个向量就是我们要优化的目标函数。DE 算法通过采用浮点矢量进行编码生成种群个体。在 DE 算法寻优的过程中，首先，从父代个体间选择两个个体进行向量做差生成差分矢量；其次，选择另外一个个体与差分矢量求和生成实验个体；然后，对父代个体与相应的实验个体进行交叉操作，生成新的子代个体；最后在父代个体和子代个体之间进行选择操作，将符合要求的个体保存到下一代群体中去。

2.2 DE 算法流程

(1) 确定差分进化算法控制参数，确定适应度函数。差分进化算法控制参数包括：种群大小 NP、缩放因子 F 与杂交概率 C_r。(2) 随机产生初始种群。(3) 对初始种群进行评价，即计算初始种群中每个个体的适应度值。(4) 判断是否达到终止条件或进化代数达到最大。若是，则终止进化，将得到最佳个

体作为最优解输出；若否，继续。(5) 进行变异和交叉操作，得到中间种群。(6) 在原种群和中间种群中选择个体，得到新一代种群。(7) 进化代数 g=g+1，转步骤 (4).

3 本文方法（ADE）

3.1 本文方法概述

在本研究中，我们试图通过结合先进的搜索策略来提高蛋白质配体对接程序的性能。由于自适应 DE 方法的高性能，我们采用自适应 DE 算法 (ADE) 作为解决蛋白质配体对接问题的搜索策略。通过引入参数自适应方案和改进的变异策略，提高了算法的搜索能力。

3.2 参数自适应

最近的研究表明，在许多实际问题中，DE 具有强大的全局优化能力。然而，DE 的性能对其控制参数的选择非常敏感[5]，因为选择合适的控制参数通常是问题相关的，在某些极端情况下，控制参数的理想值会随着演化过程而变化。在文献中，参数自适应是一种被广泛应用的有效方案，可以缓解选择合适的控制参数的问题[6]。由于正则 DE 方法中的敏感参数被不那么敏感的参数所取代，自适应 DE 算法在大多数问题中都能比正则 DE 算法表现得更好。在典型 DE 方法中有三个主要的控制参数: 总体规模 NP、缩放因子 F 和杂交概率 C_r。如文献所述，NP 不需要微调。我们将 NP 固定为一个典型值。对于另外两个参数 F 和 Cr，我们引入了一种主要受[7][8] 启发的自适应机制，在进化过程中调整控制参数。根据他们的成功经验，这两个参数的变化如下。在整个进化过程中保持两个池:S_F 用于存储第 t 代所有成功的比例因子 F_i, S_{C_r} 用于存储第 t 代所有成功的交叉率 C_r。在第 t+1 代，F 值从具有位置参数 u 和规模参数 0.1 的柯西分布中随机采样，用 Cauchy(μ_F,0.1) 表示。C_r 值随机抽样于均值为 μ_{C_r}，标准差为 0.1 的正态分布，用 Normal(μ_{C_r},0.1) 表示。然后，每个 F 和 Cri 对应于单独的 $X^{(i)(t)}$ 分别用于生成供体向量和目标向量。其中，F 和 Cr 的值在 [0,1] 范围之外时，会重新生成。参数自适应是通过逐渐调整 μ_F 和 μ_{C_r} 的值来实现的。在每一代中，μ_F 根据池 S_F 更新，μ_{C_r} 根据池 S_{C_r} 更新，如下所示:

$$\mu_F = \begin{cases} (1-c)\cdot\mu_F + c\cdot mean_L(S_F) & if \quad S_F \neq none \\ \mu_F & otherwise \end{cases} \tag{1}$$

$$\mu_{C_r} = \begin{cases} (1-c)\cdot\mu_{C_r} + c\cdot mean_A(S_{C_r}) & if \quad S_{C_r} \neq none \\ \mu_{C_r} & otherwise \end{cases} \tag{2}$$

其中 c 为适应相关常数。$mean_L$ 是用来计算 S_F 的 Lehmer 平均值的函数。$mean_A$ 是用来计算 S_{C_r} 的算术平均值的函数。

3.3 突变策略

突变策略“DE/rand/1”被认为是文献中应用最广泛的突变策略。与“DE/rand/1”相比，“DE/best/1”是一种贪婪策略，具有收敛速度快的优点。然而，这种策略会导致过早收敛的问题，容易陷入局部最优。两种算法如下：

$$\text{"}DE/rand/1\text{"} : \mathbf{V}^{(i)(t)} = \mathbf{X}^{(r_1)(t)} + F(\mathbf{X}^{(r_2)(t)} - \mathbf{X}^{(r_3)(t)}) \tag{3}$$

$$\text{"}DE/best/1\text{"} : \mathbf{V}^{(i)(t)} = \mathbf{X}^{(r_best)(t)} + F(\mathbf{X}^{(r_2)(t)} - \mathbf{X}^{(r_3)(t)}) \tag{4}$$

其中 F 是规模因子，它控制规模差异。r_1,r_2,r_3, 是从种群规模中随机选择的三个不同的整数。r_best 是第 t 代适应度函数最好的个体的索引。为了结合两种突变策略的优点，本方法中使用的“DE/pbest/1”策略定义如下:

$$\mathbf{V}^{(i)(t)} = \mathbf{X}^{(pbest)(t)} + F(\mathbf{X}^{(r_2)(t)} - \mathbf{X}^{(r_3)(t)}) \tag{5}$$

$\mathbf{X}^{(pbest)(t)}$ 是在当前群体中从前 p 个个体中随机选择的个体。当 p 等于 NP 时，“DE/pbest/1”策略等价于“DE/rand/1”策略，当 p 等于 1 时，“DE/best/1”策略等价于“DE/best/1”策略。

4 复现细节

4.1 与已有开源代码对比

在开源框架的结构上完成 ADE 算法编写，即主要编写 lshade.app 和 lshade.h 文件的编写工作。

算法伪代码如下所述。

Procedure 1 Main procedure of the ADE for the docking problem.

Initialize the docking enviroment.
Set the parameters c and p.
Set generation count t←0.
μ←0.5, μ_{C_r}←0.5
Initialize NP-size $\{\mathbf{X}^{(i)(t)}|i \in \{1, 2, ..., NP\}\}$.
Evaluate all $\mathbf{X}^{(i)(t)}$.
while *Stopping criterion is not meet* **do**
  Empty S_F and S_Cr.
  for *i in 1,2,...,NP* **do**
    Generate new F_i and C_{r_i}.
    Randomly select one of the top p individuals as $X^{(i)(t)}$.
    Create the donor vector $\mathbf{V}_{(i)(t)}$.
    Boundary control.
    Create the trial vector $\mathbf{U}^{(i)(t)}$.
    Determine whether $\mathbf{U}^{(i)(t)}$ is illegal ,and evaluate it.
    Select a better one as $X^{(i)(t+1)}$ from $X^{(i)(t)}$ and $U^{(i)(t)}$.
    if $\boldsymbol{U}^{(i)(t)}$ *win* $\boldsymbol{X}^{(i)(t)}$ **then**
      Add F_i into S_F.
      Add C_{r_i} into S_{C_r}.
    else
      pass
    end
  end
  Update μ_F and μ_{C_r}.
  t<-t+1.
end

4.2 实验环境搭建

本实验使用编译器位 vs2022，可从官网下载。配置 boost 库（可从官网下载）: 本实验选择版本 boost_1_79_0, 下载完毕后编译，编译成功。若编译有问题，查看 vs 版本与 boost 版本，若 vs 版本过新，建议选择较新 boost 版本，若 vs 版本过旧，建议选择较旧 boost 版本。编译成功后会有成功编译提示。配置项目属性，在 VC++ 目录里配置库目录。在 PDB 数据库里下载蛋白质-分子复合物，下载 AutoDock 软件对数据进行预处理。

4.3 界面分析与使用说明

根据预处理参数编写蛋白质-分子配置文件。完成数据预处理后，即可对数据集进行训练。预处理操作界面如图 1 所示

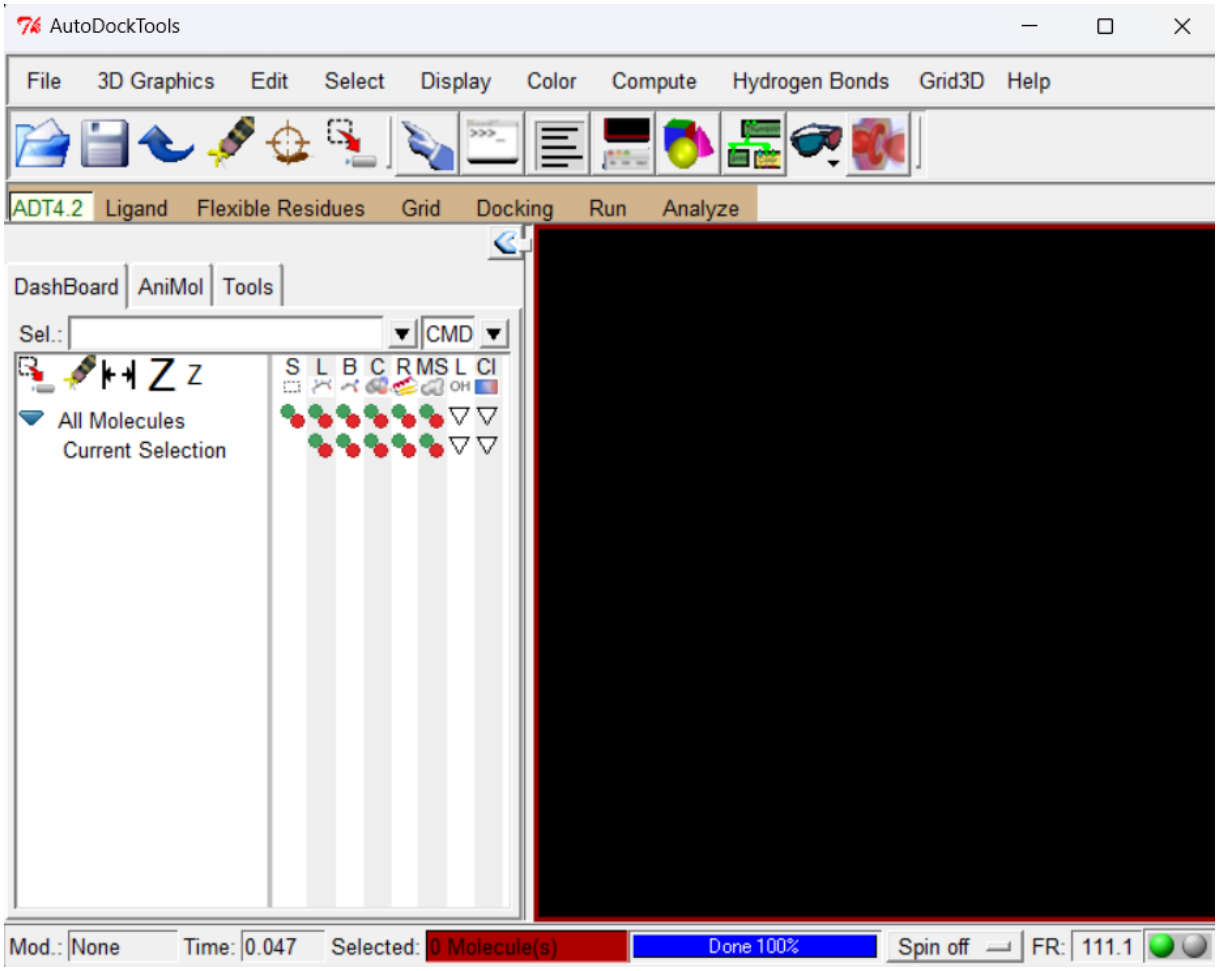

图 1: 预处理界面

导入蛋白质如图 2 所示。

图 2: protein

导入小分子如图 3 所示。

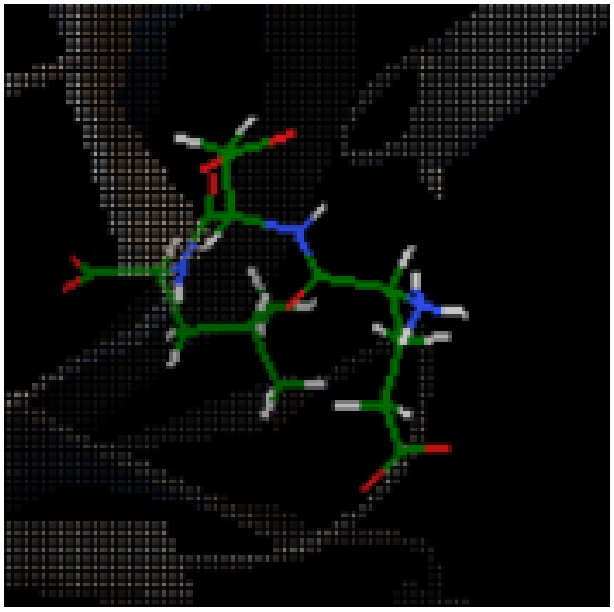

图 3: ligand

配置参数如图 4 所示。

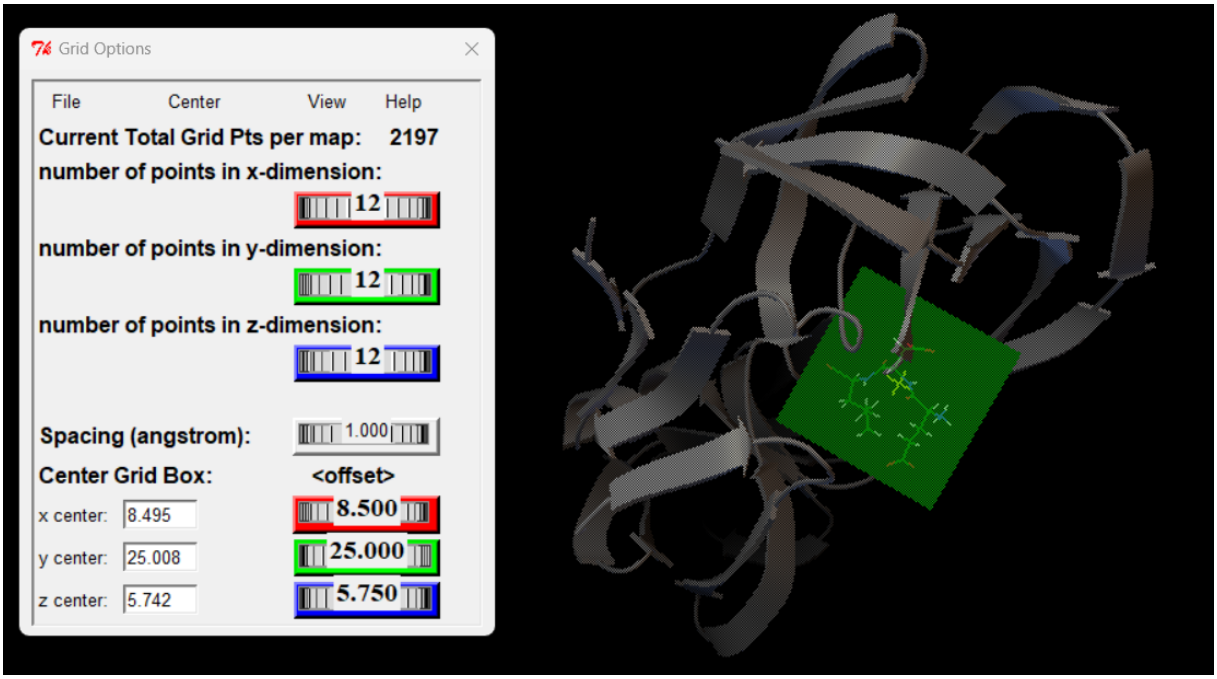

图 4: grid_box

4.4 创新点

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。对部分蛋白质进行训练测试，结果如图 5 所示。每对蛋白质分子配对输出其预测结构数、亲和力、结构对称性以及与实际配对原子距离。其中最重要的是第四列预测结构与基准结构之间的原子距离，这一指标小于 2 可认为预测成功。可以看出大部分蛋白质-分子配对预测正确，小部分略大于 2。整体效果较好。

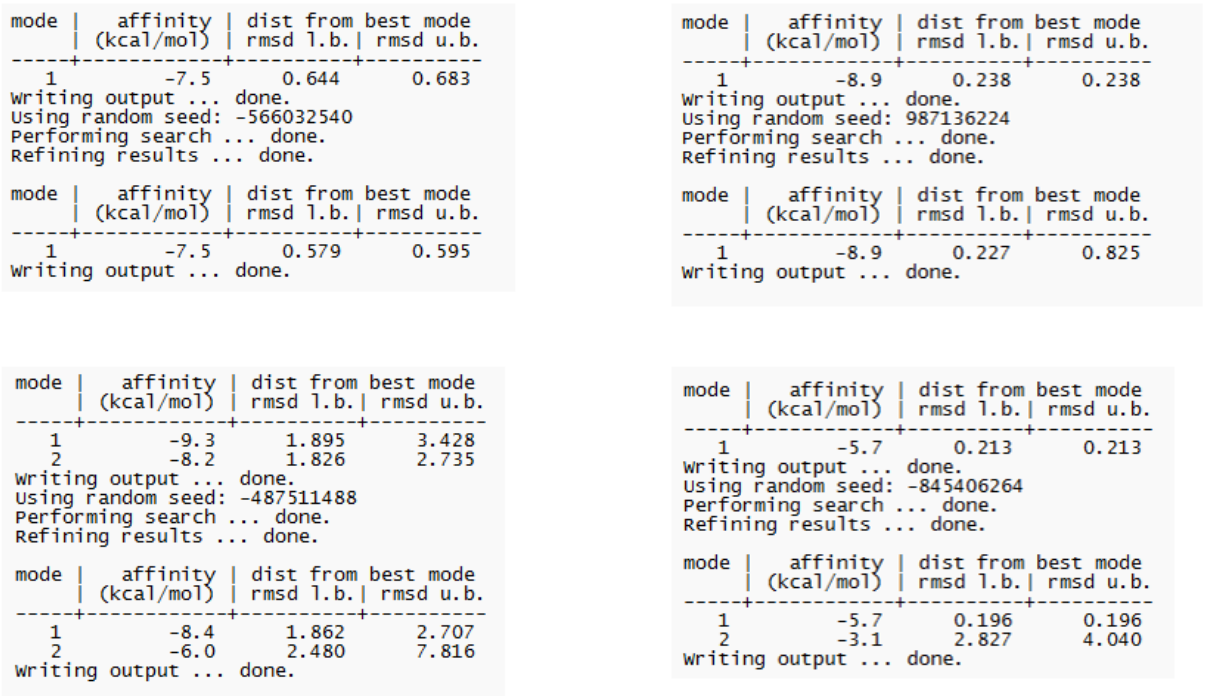

图 5: 实验结果

6 总结与展望

蛋白质-配体对接问题在药物发现过程中起着至关重要的作用。本文参考论文对 DE 算法进行改进，提出了 ADE 算法，主要贡献在增加参数自适应机制和突变策略。本文诠释蛋白质-分子对接大体流程，学习 DE 及改进的 ADE 算法，在其基础上对 ADE 算法进行实现。实验结果表明算法性能较好。但是本文参考论文实现算法还是基于进化计算这一主流解决方案，其计算代价开销非常大，准确率提升有限。近几年已经有工作将深度学习方法应用与此问题，在速度方面有较大提升，但在准确度方面提升不明显。最近有工作在将蛋白质-配体对接视为生成建模问题——给定配体和目标蛋白结构，我们学习配体姿势上的分布。由此建立一个生成扩散模型进行训练，其效果较之回归或搜索方案提升较大。或许可以将不同的算法方案应用到蛋白质-配体对接问题上，实验出新的好方法。

参考文献

[1] STORN R, PRICE K V. Differential Evolution－A Simple and Efficient Heuristic for global Optimization over Continuous Spaces[J]. Journal of Global Optimization, 1997, 11: 341-359.

[2] DRĂGOI E N, DAFINESCU V. Parameter control and hybridization techniques in differential evolution: a survey[J]. Artificial Intelligence Review, 2015, 45: 447-470.

[3] MENG Z, PAN J S, TSENG K K. PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization[J]. Knowl. Based Syst., 2019, 168: 80-99.

[4] WU G, MALLIPEDDI R, SUGANTHAN P N, et al. Differential evolution with multi-population based ensemble of mutation strategies[J]. Inf. Sci., 2016, 329: 329-345.

[5] DAS S, MULLICK S S, SUGANTHAN P N. Recent advances in differential evolution - An updated survey[J]. Swarm Evol. Comput., 2016, 27: 1-30.

[6] HUANG Q, ZHANG K, SONG J C, et al. Adaptive differential evolution with a Lagrange interpolation argument algorithm[J]. Inf. Sci., 2019, 472: 180-202.

[7] ZHANG J, SANDERSON A C. JADE: Adaptive Differential Evolution With Optional External Archive[J]. IEEE Transactions on Evolutionary Computation, 2009, 13: 945-958.

[8] BROWN C, JIN Y, LEACH M, et al. μJADE: adaptive differential evolution with a small population[J]. Soft Computing, 2016, 20: 4111-4120.