分子プログラミングの最適化システムのための Python ラッパー

大西春寧　(指導教員：オベル加藤ナタナエル)

1　はじめに

分子プログラミングは, 分子の相互作用を利用して計算や情報処理を行うことを目的としている. しかし, 特定の機能を実現するシステムを構成するには, 多くの試行錯誤が必要で, これを実験で行うことは困難である. そこで, コンピュータ上での最適化アルゴリズムを用いたシミュレーションが有効になる. 本研究では, シミュレーションライブラリである DACCAD を用いた最適化システムである DACCADEvo の高速化を図り, シミュレーションと機能拡張を実行する [1, 2].

2　背景

2.1　PEN DNA toolbox

PEN DNA toolbox は,Montagne らによって提案された DNA の相互作用と酵素反応による化学反応ネットワークの分子プログラミングフレームワークである [3]. これには, 活性と抑制というモジュールがある. 活性では, まず, 短い DNA 鎖を入力シグナルとして, テンプレートと呼ばれる DNA 鎖とハイブリダイゼーションする. そして, ポリメラーゼによってシグナルが伸長して, ニッカーゼが入力シグナルと伸長した部分である出力シグナルをニックし, その後テンプレートから両方のシグナルが解放される. 抑制では, シグナルの一種であるインヒビターとテンプレートがハイブリダイゼーションするが, 伸長や切断が十分に行われないことで, テンプレートの反応を抑制する. また, 時間経過とともに, シグナルはエキソヌクレアーゼによって分解されることでシステムは平衡状態を保つ. これらを組み合わせて、ネットワークを構成する.

2.2　DACCAD

DACCAD は,Aubert らによって開発された DNA 反応ネットワークのためのシミュレーションライブラリである [1].Java 言語で実装されている. その数理モデルは, 分子の濃度やスタビリティからシグナルやテンプレートの濃度を計算する.

2.3　Quality-Diversity 最適化と QDPy

Quality-Diversity 最適化（QD 最適化）は, 多様で高品質な解のコレクションを得ることを目的にしている [4]. そのアルゴリズムでは, 解の品質だけでなく, 新規性を評価に加えることで, 様々なタイプの解の探索が可能になり, 各タイプごとの最適な解のコレクションが得られる. QDPy は,QD アルゴリズムを Python ライブラリとして提供している [5].

2.4　DACCADEvo

DACCADEvo は,DACCAD と QDPy を組み合わせた最適化システムである [2]. その一連のプロセスでは, 1.QDPy によって、解の候補となる Individual として, シグナルのスタビリティとテンプレートの濃度の配列を生成される. 2. その配列を Java で開発された DACCAD に提出するために,JSon ファイルに変換する. 3.DACCAD でシミュレーションを実行する. 4. 解の品質と特徴を評価する. 5. 同じタイプの解の中で, 品質的に最適であれば,Grid に追加する.（Grid は 2 つの特徴を軸として解のタイプを形成する.）これを繰り返すことで, 最終的に, 多様性があり高品質な解のコレクションを得る. 実際には, この繰り返しは数万回単位で行われるため, ファイル操作の処理が, 時間の面でもメモリ空間の面でも膨大なコストとなり, 現状の DACCADEvo の課題であり, 機能を拡張する上で障害となっている.

3　実験

2.4 で述べた課題を解決するために, ファイル操作を省略して高速化を図る. 次に, 高速化が実現された場合のメリットを活かし, 機能拡張も行う. そして, 新たな DACCADEvo を用いて, 振動（oscillation）の最適解を探索するシミュレーションを行う.

3.1　実験 1: コンピューティングの高速化

3.1.1　方法

JPype[6] という Python モジュールを利用し,Python で書かれた DACCADEvo から Java のライブラリに直接アクセスすることで, ファイル操作を省略する. そして, オリジナルバージョンと今回新たに開発したバージョンをそれぞれ 10 回ずつ実行して、実行時間の平均をとり, 結果を比較する. 使用するマシンのスペックは, 以下の通りである. OS:buntu 18.04, CPU:AMD Ryzen 9 3900X 12-Core Processor, 24CPUs, 3.8GHz　メモリ:64GB

3.1.2　結果

実験結果を表 1 にまとめた. 実行時間の平均は, オリジナル版では約 130 時間, 新しいバージョンでは約 76 時間と約 4 割の削減に成功した. この結果から, 十分に高速化が達成できたと言えるだろう.

3.2　実験 2: シミュレーション

3.2.1　方法

振動についての最適解を求め, 解のタイプが品質に与える影響を観察する. 振動の品質は, 連続した振動の振れ幅の大きさの平均値と振幅の最大値の比を 1 から引いた値に, 振動の回数を正規化した値と各振動のピークの際立ち度合いの平均を掛けることで定義され,0 以上 1 以下の値となる. 次に, 特徴として用いたパラメータについて説明する.「PeaksNumber」は, 振動の回数を表し, その値は 0 以上 1 以下に正規化される.「PeaksLastValue」は, 最も大きく振れた振動と最後の振動の振幅の比を表す.「AveragePeriod」は, 各振動周期の平均をとった値である. さらに, 以下の 2 つを新たに追加した.「NumActiveNodes」は, 相互反応をするシグナルの個数で, この値は 1 以上 7 以下と事前に定義されている.「NumConn」は, 相互反応をするテンプレートとインヒビターの個数であり, この値も 1 以上 13 以下と事前に定義されている. これらの特徴を組み合わせてシミュレーションを実行する.

バージョン	適応度	特徴 1	特徴 2	Grid の形	反復回数	実行回数	平均実行時間 (秒)
original	oscillation	PeaksNumber	PeaksLastValues	(25, 25)	50000	10	469976.46766526334
new	oscillation	PeaksNumber	PeaksLastValues	(25, 25)	50000	10	273829.88811366784

表 1: 実行条件と平均実行時間

3.2.2 結果

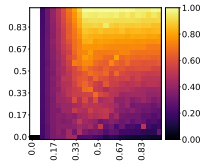

図 1: PeaksNumber と PeaksLastValue

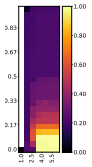

図 2: NumActiveNodes と AveragePeriod

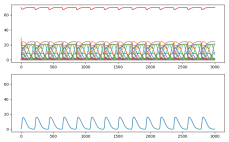

図 3: 図 1 の最適解（座標 (14,24)）のシミュレーション結果

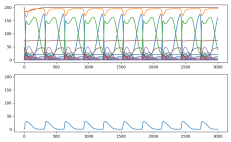

図 4: 図 2 の最適解（座標 (6,2)）のシミュレーション結果

図 3, 図 4 の上段は, 系全体のシミュレーション結果
下段は, 上段のグラフから評価に使われたシグナルを取り出している

最初の実験では,PeaksNumber と PeaksLastValue を特徴として設定した. 図 1 から,PeaksNumber が大きく, かつ PeaksLastValue が 1 に近い場合は, 安定した振動が得られていると考えられる. しかし, 最適とされた解では,PeaksLastValue は約 0.99 であったが,PeaksNumber は 0.4 とあまり大きくない. また, 行単位では,PeaksNumber が中程度の解が最適であり, 列単位では,PeaksLastValue が最大の解が最適であるとわかった. これらは, 振動の品質の定義による影響だと考えられる. 次に, 特徴を NumActiveNodes と AveragePeriod をとした. 図 2 からは,NumActiveNodes が 4 以上, かつ AveragePeriod が小さい時, 高品質な振動が得られている. また, 振動には少なくとも 3 つのシグナルが必要だとわかっているが [3], この結果からも, シグナル数が 4 つ以上であれば安定した振動を実現でき, 現実にネットワークを構成することを考慮すると, できるだけコンパクトなシステムが好ましく, 小規模のシステムでも十分な結果が得られるということが言えるだろう.

3.3 実験 3: 擬似テンプレートの拡張

3.3.1 擬似テンプレート

擬似テンプレートは, 特定のシグナルと反応する通常のテンプレートによく似た鎖を持ち, そのシグナルとハイブリダイゼーションする. しかし, 入力シグナルの 3' 末端を変更し, デハイブリディゼーションがゆっくり進行することで, そのシグナルと通常のテンプレートとの反応を抑制する. この時, 出力はなく, 擬似テンプレートは元の状態に復元する [7].

3.3.2 実験方法

まず, 擬似テンプレートの濃度を加えた配列を生成するために,QDPy の新しい Individual クラスを作成する. 配列の変異のために, 擬似テンプレートの追加や消去、変異を行う関数を定義する. そして, 新たに,「NumPt」を擬似テンプレートの個数を表す特徴として追加する.

3.3.3 結果

最初に,PeaksNumber と PeaksLastValue を特徴として設定した. 図 5 から, 図 1 と似た傾向はあるものの, 擬似テンプレートを追加した方が, 中程度の品質の結果が得られる範囲が広がっていると読み取れ, 安定した振動を得られるシステムの幅が広がったと考えられる. 次に, 特徴を NumActiveNodes と AveragePeriod と設定した. 図 6 から, 実験 2 にの時と比べて,NumActiveNodes が 3 の時も, 高品質の結果が得られるようになっている. 擬似テンプレートの追加は, シグナルを増やすよりは簡単であるので, これらの結果は擬似テンプレートの追加が有効である可能性を示している.

4 まとめと今後の展望

DACCADEvo の高速化が達成できたため, 様々な特徴を利用した最適化シミュレーションや機能拡張が容易に行えるようになった. そして, シミュレーション結果から擬似テンプレートの追加が有効であると考えられる. 今後は, 品質や特徴の再定義をした実験を行い, 適切な評価方法を検討したい. そして, 捕食者-非食者テンプレート [8] などの拡張を行い, より複雑なシステムを生成し, その機能を確認したい.

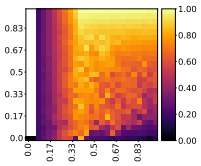

図 5: PeaksNumber と PeaksLastValue

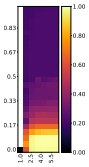

図 6: NumActiveNodes と AveragePeriod

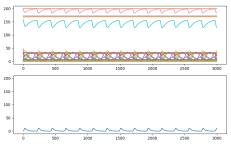

図 7: 図 5 の最適解（座標 (14,24)）のシミュレーション結果

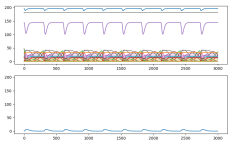

図 8: 図 6 の最適解（座標 (5,2)）のシミュレーション結果

参考文献

[1] Nathanaël Aubert, Clément Mosca, Teruo Fujii, Masami Hagiya, and Yannick Rondelez. Computer-assisted design for scaling up systems based on dna reaction networks. *Interface*, Vol. 11, No. 93, 2018.

[2] Mika Ito and Nathanael Aubert-Kato. Daccadevo, a python wrapper to combine daccad with qdpy. `https://bitbucket.org/AubertKato/daccadevo/src/master/`, 2021.

[3] Kevin Montagne, Raphael Plasson, Yasuyuki Sakai, Teruo Fujii, and Yannick Rondelez. Programming an in vitro dna oscillator using a molecular networking strategy. *Molecular Systems Biology*, Vol. 7, No. 1, p. 466, 2011.

[4] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, Vol. 22, No. 2, pp. 245–249, 2021.

[5] L. Cazenille. Qdpy: A python framework for quality-diversity. `https://gitlab.com/leo.cazenille/qdpy`, 2018.

[6] Steve Menard. Jpype. `https://github.com/jpype-project/jpype`.

[7] Kevin Montagne, Guillaume Gines, Teruo Fujii, and Yannick Rondelez. Boosting functionality of synthetic dna circuits with tailored deactivation. *Nature Communications*, Vol. 7, , 2016. 13474.

[8] Teruo Fujii and Yannick Rondelez. Predator–prey molecular ecosystems. *ACS Nano*, Vol. 7, No. 1, p. 27—34, 2013.

5 付録

5.1 実験 2 の結果

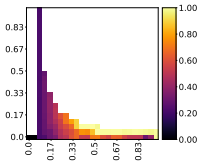

図 9: PeaksNumber と AveragePeriod

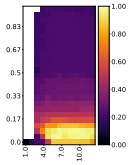

図 10: NumConn と AveragePeriod

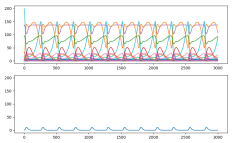

図 11: 図 9 の最適解のシミュレーション結果
Grid の座標は,(12,2).

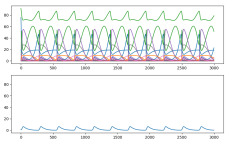

図 12: 図 10 の最適解のシミュレーション結果
Grid の座標は,(6,2).
7 つのテンプレートがある

図 11, 図 12 の上段は, 最適解の系全体のシミュレーション結果
下段は, 上段のグラフから評価に使われたシグナルを取り出したもの

5.2 実験 3 の結果

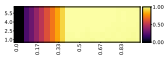

図 13: PeaksNumber と NumPt

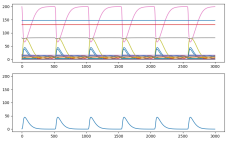

図 14: 図 13 の最適解のシミュレーション結果
Grid の座標は,(5,13).

図 14 の上段は, 系全体のシミュレーション結果
下段は, 上段のグラフから評価に使われたシグナルを取り出している