汎用的なルービックキューブの解探索プログラムの作成

東京都立南多摩中等教育学校　4年　寺内駿

研究背景

ルービックキューブを6面揃えるには最低7つほどの手順の修得が必要だが, ルービックキューブを揃える過程で手順を回し間違えると1から揃え直しになる。これは, 初心者が練習する際の障壁となるだろう。

また, ルービックキューブを人が揃えるための手順はコンピュータによって複数生成され, 人が指で回しやすいと感じるものが選択されルービックキューブのコミュニティに広まってきた[1]。しかし今もルービックキューブを揃える上で, 回しにくいと感じる手順が多い。これらの問題をコンピュータを用いて解決する解探索プログラムを作成した。

研究目的

・ルービックキューブの揃え方(以降本稿では**解**と呼ぶ)を求める機能を開発する
・初心者の学習支援のため, ルービックキューブの展開図に色を塗り完成までの解を表示する機能を開発する
・より回しやすい手順を見つけるために, ルービックキューブの手順を探索する機能を開発する
・同様の機能を提供する既存のプログラム[2]よりも, 使いやすく, 多機能なものを開発する

解を求める機能の開発

解を求めるために, Two-Phase Algorithm[3]を用いた。これはIDA*探索を用い, ルービックキューブを2フェーズに分けて解くものであり, 最短解が見つかるまで探索し続けるという特徴を持つ。

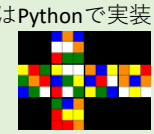
例

IDA*探索は深さが増えると探索範囲が指数関数的に増加するため, 2つのフェーズに分けることで探索効率が上がる。

また高速化のため探索部はC++, GUIはPythonで実装した。

探索部の速度の推移

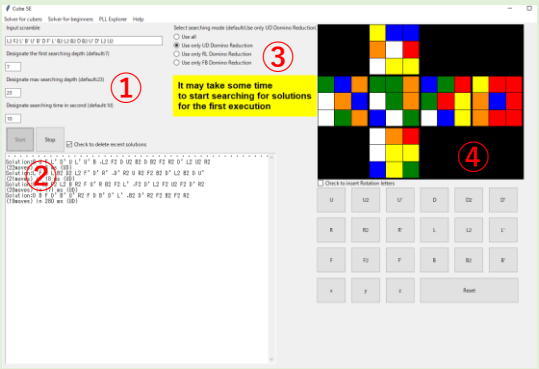
探索を始める深さ：7　　混ぜ方：



19手の解が見つかるまでの時間を測定

言語	Python	Cython	C++
20回平均(秒)	14.93652	15.43959	0.5266667

作成したGUI　　**動作の様子**



GUIの作成にはTkinterを用いた。GUI上のボタンを押すとC++で実装したexeファイルが実行され, 受け取った出力をテキストボックスに表示させる。入力に対する例外処理や, ボタンの連打対策も実装した。

主な機能

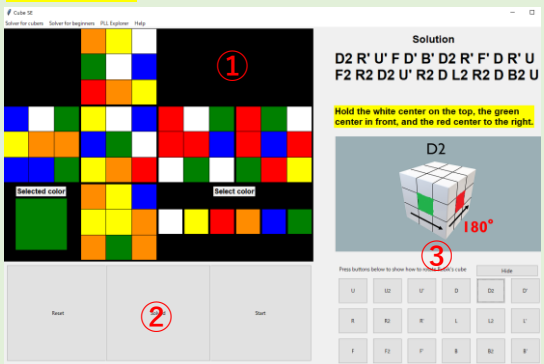
①
・混ぜ方
・探索を始める深さ
・探索する秒数
を指定できる。
②
探索の開始, 停止を行う。下のボックスに解が表示される。
③
3つの軸から探索に用いる軸を選択できる。3つ全て並列処理させると探索効率が最大になる。
④
混ぜた後のルービックキューブの状態を展開図で表示する。

※赤字は独自の機能
黒字は既存の機能[2]

初心者向けの機能の開発

ここでもTwo-Phase Algorithmを用いた。解は1つで良いため, 最初の解が見つかった時点で探索を終了させる。探索部はC++, GUIはPythonで実装した。

作成したGUI　　**動作の様子**



主な機能

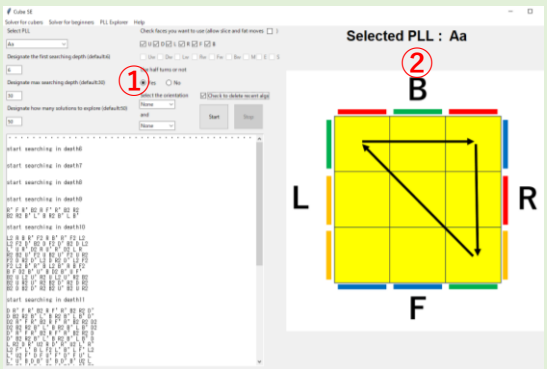
①
色を選択し, 展開図をクリックすることで色を塗れる。
色を塗り間違えるとエラーを表示する。
②
展開図の色の初期化, 探索の開始ができる。
③
ルービックキューブの回転を表す記号の意味を画像で確認できる。

※赤字は独自の機能
黒字は既存の機能[2]

手順を探索する機能の開発

Two-Phase Algorithmは短い解を求めることに特化しており, 手順の探索には不向きであるため探索は純粋なIDA*を使用する。前2つと同様の言語を用い, まずPLLという手順群の探索を開発した。

作成したGUI　　**動作の様子**



主な機能

①
・PLLの種類
・探索を始める深さ
・求める解の個数
・Generator*の指定
・180度回転の可否
・探索を始める向き
などを指定できる。
②
選択したPLLが表示される。

*Generatorとは手順を構成する回転の種類のことである。

※赤字は独自の機能
黒字は既存の機能[2]

手順探索の効率化

手順の探索は純粋なIDA*で行うため探索効率が良くない。そのため, 枝刈りを改良している。枝刈りには限定されたパーツ群2つを組み合わせて事前探索させ, 完成まで要した手数を表にして用いている。組み合わせにより性能に差が出るため, 刈れる手数の**期待値**を算出し比較した。

表の名前	要素数	μ(期待値)
co_cp	88179840	9.764133
cp_mep	479001600	9.064617
cp_eep	479001600	9.064617
cp_sep	479001600	9.064617

期待値が**9以上**となった表を抜粋した。期待値が高いと要素数も多くなるため表の読み込みに時間がかかることが分かった。そこで**表の保存形式**について実験している。

保存形式と読み込み速度の比較実験　　(単位：ミリ秒)

表の名前	要素数	JSON	CSV	binary	圧縮binary*
co_cp	88179840	4206.38	3994.7	21	10
cp_mep	479001600	56701.7	20665.3	102	56
cp_eep	479001600	56775.1	20647.9	100	55
cp_sep	479001600	56157.9	20681.7	105	56

*格納される値は全て4bit以内なので, 1byteに2つ値を格納したものが圧縮binary

今後の展望

解を求める機能と、初心者向けの機能は完成しており, 過去に2度試験配布を行いフィードバックを反映させた。同様に, 手順を探索する機能も探索効率を改善でき次第試験配布を行い, フィードバックを反映する。

手順の探索は現状, PLLという手順群についてしか探索できないが他の手順群についても探索できるようにする。他の手順群は, 探索に個別の関数が必要となる。

また, これまで機能面と使いやすさを重視し開発していたため探索された手順の評価ができていない。評価法を開発し改良を進めていきたい。

参考文献

[1] J perm. The Best V Perm Algorithms. YouTube. 2021-1-24. https://www.youtube.com/watch?v=1wUqCM94UIA, (参照 2022-01-22)
[2] Herbert Kociemba . Cube Explorer . Version 5.14 . 2020-08-19 . https://github.com/hkociemba/CubeExplorer . (参照 2022-01-22)
[3] 7y2n . ルービックキューブを解くプログラムを書いてみよう(後編:状態のindex化, Two-Phase-Algorithm) . Qiita . 2021-10-17 . https://qiita.com/7y2n/items/a840e44dba77b1859352 . (参照 2022-01-22)