人工知能第二回レポート第 10 問 (カエル探索)

工学部電子情報工学科
学生証番号 03-213009 柚木隼人

1　アルゴリズムの説明

ナップザック問題とは、1 つのカバンの容量 (重さ) と、カバンに入れる複数の荷物の重さと価値が与えられ、カバンに入れられる荷物の価値が最大になる組み合わせを探す問題である。

ここでは、カエル探索 : Shuffled Frog-Leaping Algorithm (以下 SFLA) を用いて 01 ナップザック問題を解くためのアルゴリズムを説明する。

(1) カエルを P 匹用意する。それぞれのカエルはどの荷物を入れるかを表す遺伝子 x(リストになっていて入れる荷物は 1, 入れない荷物は 0 に対応する) と、そのときの価値の合計 $fitness$ を持っている。

(2) P 匹のカエルを $fitness$ について降順に並べ替える。

(3) P 匹のカエルを m 個の memeplex に分ける。1 番目のカエルは 1 番目の memeplex に、2 番目のカエルは 2 番目の memeplex に、m 番目のカエルは m 番目の memeplex に、m+1 番目の memeplex は 1 番目の memeplex に、というように分ける。

(4) 全ての memeplex に対して以下の Local Search を行う。

（a） P 匹の中で最大の $fitness$ を持つカエルの遺伝子を X_g、memeplex の中で最大の $fitness$ を持つカエルの遺伝子を X_b、memeplex の中で最小の $fitness$ を持つカエルの遺伝子を X_w とする。

（b） 式 (1) によって $X_w(new)$ を作る。ただし $Rand() \sim U(0,1)$ である。$X_w(new)$ を作る方法は他にもあるが、参考論文 ([2]) において最も成績が良かったこの方法を採用した。

$$
\begin{aligned}
D &= Rand() \times (X_b - X_w) \\
t &= 1/(1+\exp(-D)) \\
X_w(new) &= \begin{cases} 0 & (t \le \alpha) \\ X_w & (\alpha < t < (1+\alpha)/2) \\ 1 & (t \ge (1+\alpha)/2) \end{cases}
\end{aligned} \tag{1}
$$

（c） $X_w(new)$ の $fitness$ が X_w の $fitness$ より改善されていれば X_w を $X_w(new)$ に置き換える。

（d） 改善されていなければ、式 (1) の X_b を X_g に置き替えて $X_w(new)$ を作る。これで改善されれば X_w を $X_w(new)$ に置き換える。

（e） これでも改善されなければ X_w をランダムに作り直す。

（f） 以上を $iMax$ 回繰り返す。

(5) 全ての memeplex から全てのカエルを取り出す。

(6) 確率 p_m で全てのカエルの遺伝子 x に突然変異を起こさせる。

(7) P 匹の中で $fitness$ が最も良いカエルを取り出し、以前のそれと変わっていない、つまり解が収束しているとみなせれば、そのカエルの遺伝子 x が解となる。収束していない場合は (2)に戻る。

2 実現したシステム・ソースリストの説明

システムは Python を用いて実現した。クラスは、P 匹のカエルを管理する population クラス、個々の memeplex を管理する memeplex クラス、個別のカエルを管理する individual クラスからなる。

2.1 population クラス

属性として、最も $fitness$ が高いカエルを表す global_best_frog やカエルの集まりである frog_list、memeplex の集まりである memeplex_list などを持つ。主なメソッドは以下である。

- init_population：P 匹のカエルをランダムに作成する。
- calc_fitness：P 匹のカエルの $fitness$ を計算する。
- sort_population：P 匹のカエルを $fitness$ について降順にソートする。
- partition：P 匹のカエルを m 個の memeplex に分ける。
- search：全ての memeplex に対して Local Search を行う。
- mutation：P 匹のカエルに対して確率 p_m で突然変異を行う。
- judge_termination：解が収束したかどうかを確かめる。収束条件を様々に変えて実験を行った。

2.2 memeplex クラス

属性として、memplex の中で最も $fitness$ の大きいカエルを表す best_frog や memeplex の中で最も $fitness$ の小さいカエルを表す worst_frog、Local Search で作るカエルを表す worst_frog_new、memeplex 内のカエルの集まりを表す frog_in_memeplex などを持つ。主なメソッドは以下である。

- local_search：その memeplex に対して Local Search を行う。

- sort_memeplex：memeplex 内のカエルを $fitness$ について降順にソートする。
- mk_worst_new：$X_w(new)$ を作成する。ここで式 (1) を適用するが、ここで用いられている α は、参考論文 ([2]) において最も成績が良かった $\alpha = 0.4$ とした。

2.3 individual クラス

属性として、カエルの遺伝子を表す x やカエルの適合度を表す $fitness$ を持つ。主なメソッドは以下である。

- calc_fitness：そのカエルの $fitness$ を計算する。
- repair_x：遺伝子 x を用いた際の重さの合計がカバンの容量 $capacity$ を超えていた時に x を修正する。修正する際は x の要素が 1 になっている荷物について、重さあたりの価値が最も小さいものを削除 (x の要素を 0 に) し、これを重さの合計が $capacity$ 以下になるまで繰り返す。修正方法または $capacity$ という制約を考慮しながら探索を行う方法は他にも存在するが、参考論文 ([2]) において最も成績が良かったこの方法を採用した。
- calc_w_in_bag：遺伝子 x を用いた際の荷物の重さの合計を計算する。
- ind_mutation：そのカエルの遺伝子 x に対して確率 p_m で突然変異を行う。p_m を様々に変えて実験を行った。

3 結果

ナップザック問題の問題設定のベンチマークとして参考文献 [1] のデータセットを用いた。
突然変異率 p_m や収束条件、Local Search の回数 $iMax$ を変えたときの SFLA の性能を調べた。

3.1 参考論文の条件での性能評価

まず参考論文 ([2]) において最も成績が良いとされていた条件で様々な問題パターンで性能評価を行った。突然変異率 $p_m = 0.06$ である。収束条件はアルゴリズムの (7)で述べた、以前と変わらなかった回数を Δ としたとき、$\lceil iMax/20 \rceil \leq \Delta$ である。$iMax = 500$ とした。結果を表 1に示す。

表**1:** 参考論文の条件で性能

問題番号	荷物の個数	価値の最適値 y	最適値の探索結果 (3 回の平均値)x	x/y
P01	10	309	309	1.000
P02	5	51	51	1.000
P03	6	150	150	1.000
P04	7	107	107	1.000
P05	8	900	900	1.000
P06	7	1735	1735	1.000
P07	15	1458	1454	0.997
P08	24	13549094	13371121	0.987

3.2 突然変異率 p_m を変えたとき

収束条件 $\lceil iMax/20 \rceil \leq \Delta$ と $iMax = 500$ を固定し、p_m を様々に変えて、最適値を探索した。結果を図 1に示す。問題は P07 を用いた (P08 は p_m を大きくするとなかなか収束しなかったため用いなかった)。価値の最適値を y、最適値の探索結果 (3 回の平均値) を x としたときの x/y を評価値として用いている。

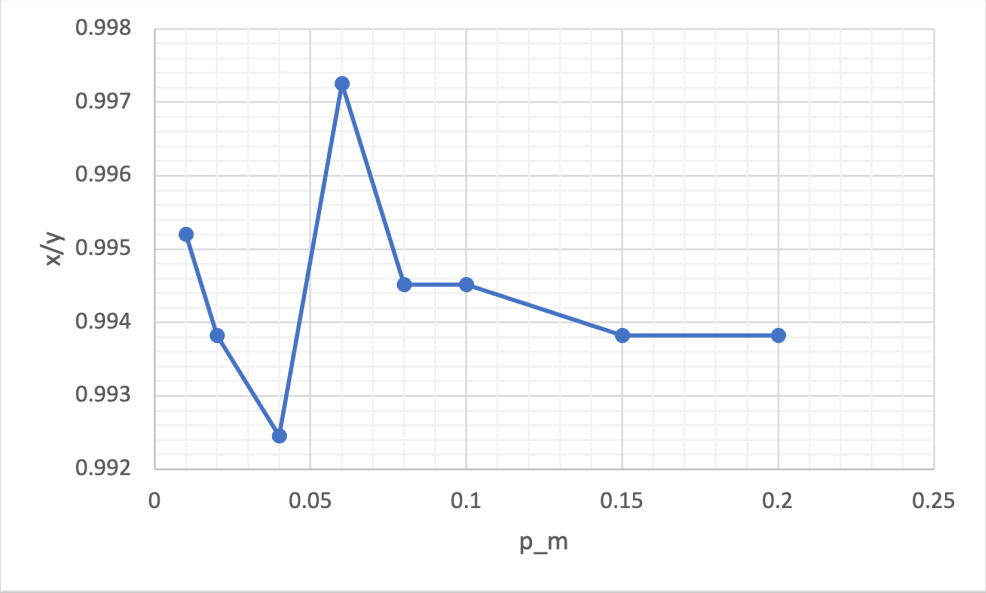

図**1:** p_m を変えたときの性能

3.3　収束条件を変えたとき

$p_m = 0.06$ と $iMax = 500$ を固定し、収束条件を $\lceil iMax/N \rceil \leq \Delta$ をしたときの N を変化させることで、収束条件を様々に変えて、最適値を探索した。結果を図 2に示す。問題は P07 と P08 を用いた。価値の最適値を y、最適値の探索結果 (3 回の平均値) を x としたときの x/y を評価値として用いている。

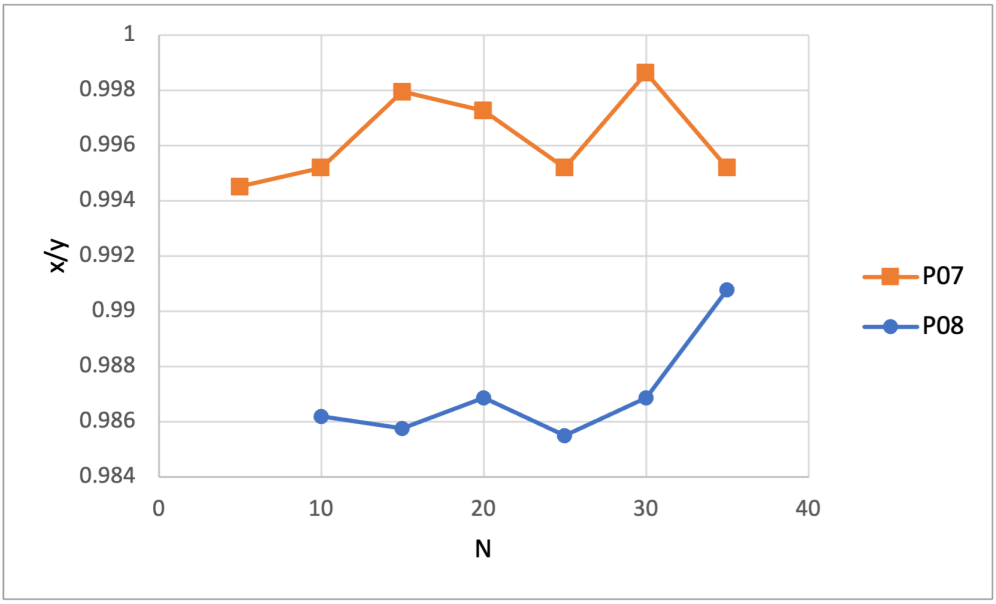

図**2:** 収束条件を変えたときの性能

3.4　Local Search の回数 $iMax$ を変えたとき

収束条件 $\lceil iMax/20 \rceil \leq \Delta$ と $p_m = 0.06$(P07 のとき), $p_m = 0.01$(P08 のとき) を固定し、$iMax$ を様々に変えて、最適値を探索した。結果を図 3に示す。問題は P07 と P08 を用いた。価値の最適値を y、最適値の探索結果 (3 回の平均値) を x としたときの x/y を評価値として用いている。

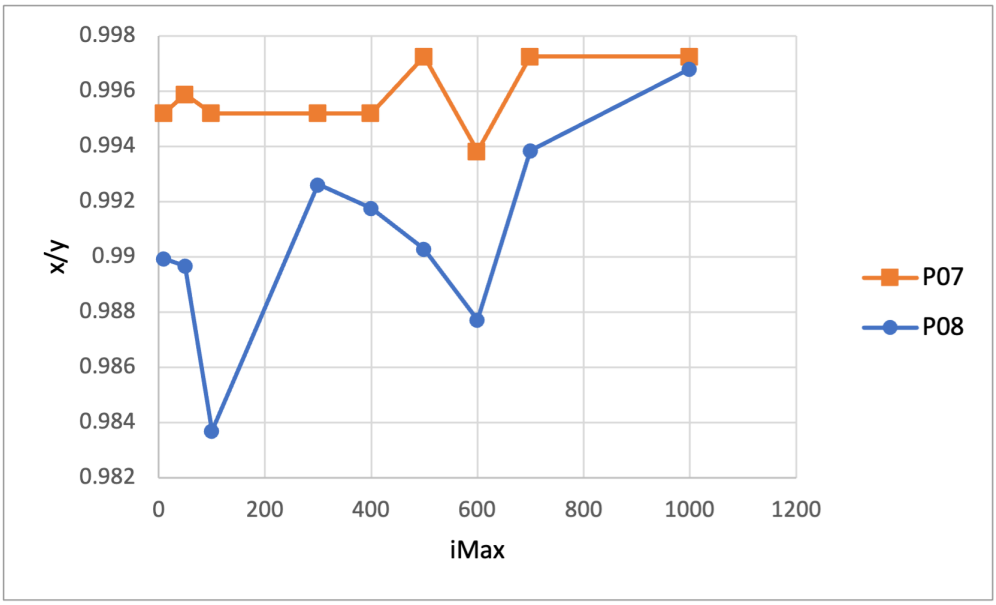

図**3:** iMax を変えたときの性能

4　考察と評価

4.1　参考論文の条件での性能評価

表 1より、ハイパーパラメータを参考論文で性能の良かった値にしたとき、P01∼P06 の比較的荷物の個数が少ない場合には探索は成功し、P07, P08 の比較的荷物の個数が多い場合には探索結果は最適値に近くなる (成功はしない) ということが分かる。荷物の個数が大きくなればなるほど表 1の x/y の値は悪化すると考えられる。

4.2　p_m、収束条件、$iMax$ を変化させたとき

p_m は小さすぎると解が局所解に陥る可能性が高くなり、大きすぎると解が最適解から離れていってしまう可能性が高くなる。図 1において p_m が比較的小さいときと大きいときであまり良い結果が得られなかったのはそのためであると考えられる。

図 2において N は大きいときほど収束条件が緩い。一般に収束条件がきついほど性能が良くなると考えられるが、図 2よりそのような傾向は見られない。つまり今回実験した収束条件の範囲程度では性能にそれほど影響しないと考えられる。収束条件はきつくするほど計算時間がかかるので、それほどきつくしすぎる必要はないことが分かる。

図 3より、P07 のときは iMax は性能にほとんど影響していないことが分かる。P07 は P08 と比べて問題が複雑ではない (荷物の個数が多くない) ため、Local Search の回数、つまり iMax を大きくしなくてもある程度性能の高いカエルを生み出すことが可能であるためだと考えられる。P08

のときは iMax が大きいときほど性能が良くなる傾向があることが分かる。これは P08 の問題が複雑であるため iMax を大きくしないと性能の高いカエルが生まれる確率が低くなってしまうためであると考えられる。ただ iMax を大きくすればするほど計算時間がかかるので問題の難易度によって iMax を調整する必要があると考えられる。

5　授業に関するコメント、要望など

成績評価を試験ではなく、ものを作って提出するという形で行うスタイルがすごく良かったです。試験勉強するよりもよっぽど自分の力になりました。

参考文献

[1] KNAPSACK_01 Data for the 01 Knapsack Problem, https://people.sc.fsu.edu/ jburkardt/-datasets/knapsack_01/knapsack_01.html, 参照年月日:2023/02/08

[2] Shuffled frog leaping algorithm and its application to 0/1 knapsack problem, Kaushik Kumar Bhattacharjee, S.P. Sarmah, 2014