巡回セールスマンの問題と遺伝子アルゴリズム

2019 年 3 月 17 日

目次

概要

巡回セールスマンの問題に遺伝子アルゴリズムを適用するスクリプトを J 言語で作成し、そのの挙動を追ってみる。漸近法もなかなか使える。

AI としてもてはやすのもいいが、AI の挙動をよく理解することも必要である

1　巡回セールスマンの問題

巡回セールスマン問題はよく知られた計算困難クラスに属する。

$$\frac{(n-1)!}{2}$$

ここで用いる 12 都市モデルでも経路は次の数になる。

```
   x: -: ! 11
19958400
```

J の関数、自書式順序 (A. Anagram) を用いたイディオム（tap) はすべての組み合わせを表示する

```
tap=: i.@! A. i.
```

```
   tap 3
0 1 2
0 2 1
1 0 2
1 2 0
2 0 1
2 1 0
```

tap 12 では 8 GB のメモリーでしばらく計算して out of memory になる。総当たり法はスパコンでも 3 桁の都市で顎を出すようなので量子コンピューターに少しは期待しよう。

日経ソフトウエア 2019 年 3 月号に本田啓一郎「人工知能の強力手法ー遺伝子アルゴリズム」という記事が出ていた。小さなモデルを python のパッケージを使って構築している。記事に沿って J 言語で一から作ってみた。

WIKI によると巡回セールスマン問題（TSP) をヨーロッパは真正面から、US は遺伝子アルゴリズム（GA) を用いる傾向にあるとのことである。

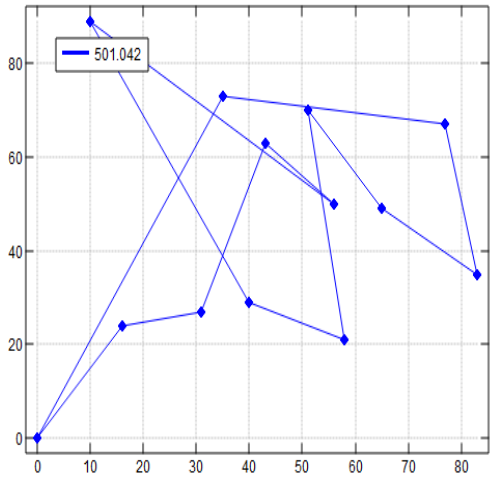
漸近法は逐次改善で良い方向に少しずつ進むアルゴリズムであり、ベストの解が得られることは少ないが真正面から突っ込んで爆死するよりもよりベターな解で満足することも大切である。

文はスクリプトの作成過程のメモが多く入っており、経過を追ったり、修正するときに有用である

遺伝子アルゴリズムの効果

1. 初期値 (14 本より選択)

```
roundn  {.sort_select pass0
```



2. GA　100 回反復の効果

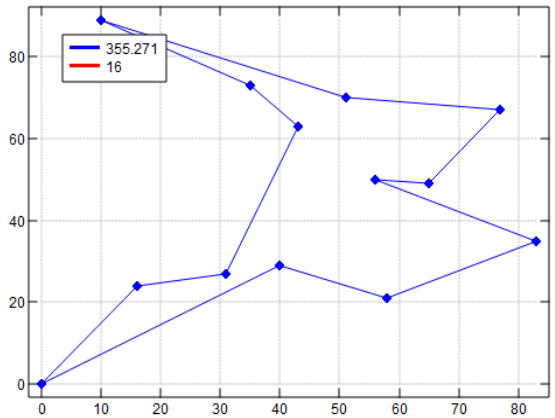
```
a=. (100;town)  ga_TS_modelx pass0
 plot |: a
```

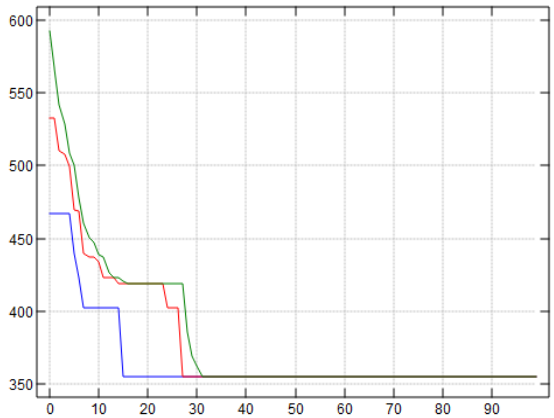
16 回で局所解に達し以降最後まで更新されなかった。

図から見れば最適解にかなり近い

GA による最良　四分位　中央値の漸近的推移

X 軸：反復回数　Y 軸:巡回経路の計





2　都市の配置:初期値の設定

到着順/ドローン方式

最初の乱数モデルは荷物を到着順に配送する単純モデルである

経路はドローンで都市上空を通過する、或いは列車で駅を通過しても下車しない、このように経路のクロスを許容し、厳密には一筆書きと言えぬ経路を許容する方式である。

都市のセットアップ

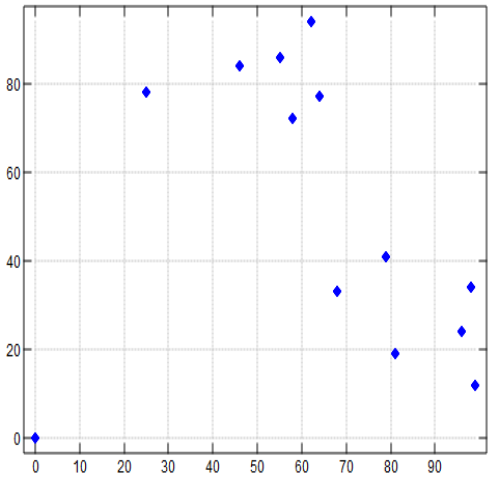
1. XY 座標なら複素数で一つの数で表した方が取り扱いが容易である
   2 つの数を複素数化 例:41j32

```
   compose_complex 12 100
   4j21 72j14 46j1 6j99 13j46 75j65 57j66 92j15 29j17 65j78 9j64 15j91

   pd 'save png c:/temp/compose_complex.png'



   compose_complex=: 3 : 0
NB. Usage: u 12 100
'n0 n1'=. y
( n0 ? n1) j. (n0 ? n1)
)
```

2. 初期値/都市のセットアップ
   - 乱数に?. を用いると毎回同じ乱数を生成でき、GA の純粋な検証に役立つ。
   - 始めと終わりに 0 を付加しスタートとゴールとする。

     ```
     setup_town=: 3 : ' 0,(/:~compose_complex y),0'
     NB. Usage: setup_town2 12 100
     ```
   - 次のように図を描いて良さそうな配置を選ぶ。(100 の範囲で乱数を XY 各々 12 個ずつ打ち出す)

     ```
     'marker' plot   town=: setup_town 12 100
     ```



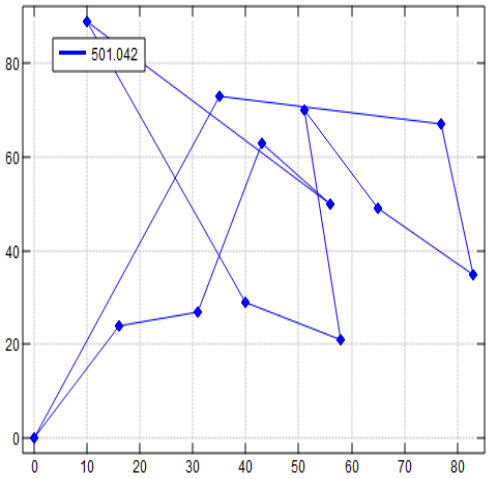
定数として扱うので保存しておく。

巡回路の打ち出し

モデルの初期値の設定で、乱数モデル、遺伝子モデル共用である

1. 配置された都市を定数として巡回順を乱数で決定し、一個の経路を打ち出す。
   荷の到着順に配送する単純モデルである。

```
pass=:    round_order setup_town 12 100
0 98j34 96j24 25j78 58j72 79j41 99j12
 68j33 64j77 55j86 46j84 62j94 81j19 0
```



```
round_order=: 3 : 0
NB. Usage: round_order a=. setup_town 12
nr=. # tmp=.}. }:  y
0,((nr ? nr){tmp),0
)
```

2. ピタゴレアン。 複素数の絶対値はピタゴレアンである　| 3j4 --> 5
   2 つの都市の差の絶対値もピタゴレアンとなる。これで距離が得られる

```
   |@-/ 0j1 2j4
3.60555
   %: 13
3.60555
```

3. 距離テーブルは不要。乱数で出した都市の差分を順次取ればよい。
   この方法だと乱数で示されたポイントの差分だけで全行程と距離が直ちに計算できる

```
   2<\ i.5
+---+---+---+---+
|0 1|1 2|2 3|3 4|
+---+---+---+---+
```

   ポイントの差分を採って複素数の絶対値でピタゴレアンを求める

```
dist=: 3 : ' > |@ -/(L:0)  2<\y'
```

```
   dist town
62 7.28011 50.9902 19.0263 40.025 38.4708 59.6657
68.469 43.0116 39.8497 27.0185 24.7386 119.017
```

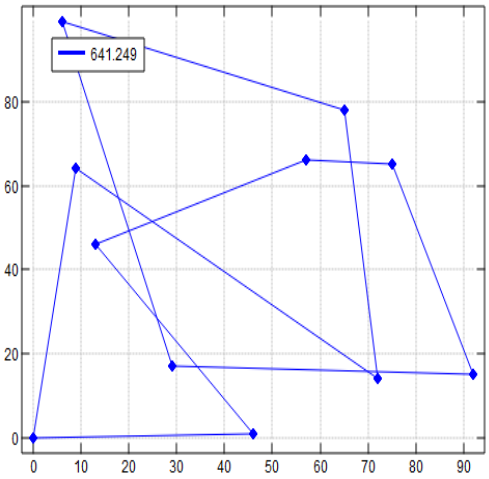
4. plot 巡回経路は plot で簡単に示すことができる

3　乱数モデル

一回の乱数モデル

経路と走行距離を plot に表示して画像出力まで。

```
rand_model=: 3 : 0
tmp=. round_order y
NB.------------------
ans=. +/ dist tmp
pd 'reset'
pd 'type marker'
pd tmp
pd 'type line'
pd tmp
pd 'key ',": ans
pd 'show'
pd 'save png c:/temp/rrs.png'
)
```



指定回数の乱数モデルの反復

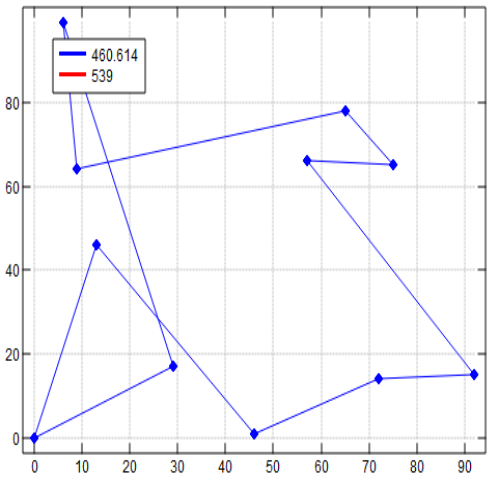
指定回数実行して最良値を選ぶ

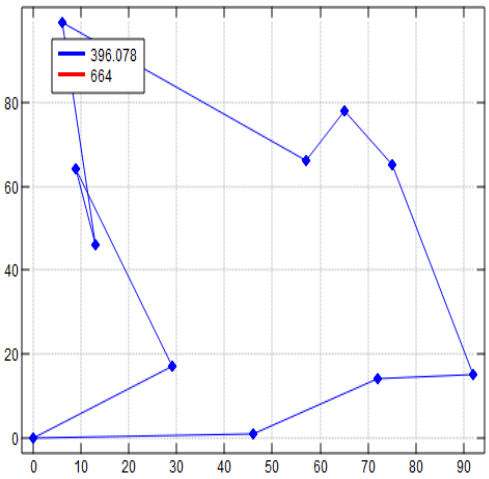
先の一回切りのモデルを指定回数反復し、最適値を選ぶと乱数モデルとなる。1000 回反復程度ならすぐにグラフィックスでみられる。

key で最少モデルの距離数と何回目かが表示されている

画像は最後に最良のを保存し画像の山に埋もれないようにしている。

```
1000 rand_modelx town
```





4　遺伝子アルゴリズムを用いた学習モデル

まったく学習しない乱数モデルは行きつ戻りつで運が良ければ最適解に行き着くだろうが、最適解に近いと思われるモデルを取り出して手で補正したくなる。

遺伝子アルゴリズムを用いた学習モデルを導入する

遺伝子の配置　乱数モデルをそのまま用いるが、遺伝子数と同じ程度の巡回路を染色体とする多数モデルを構築する。

交叉　交叉後の個体は遺伝子が重複することがあるので、重複を整理し、抜けたところには交叉で抜けた未使用の因子を補完する

突然変異　新世代の遺伝子を乱数で選んだ 2 ポイントを入れ替える。捻じれがほどける効果があるようだ

選択　優位性の指標は明確

街/家の配置

多数の巡回路の初期設定

1. 多数の巡回路の確率による設定/(n + 2) の多数モデルで遺伝子の数 + 出発/帰着である。
   `pass0=: round_ga0 town`
   これが初期設定となるので [pass0] も保存しておく
   起終点として +2 個の経路を含んでおり、個体数は遺伝子数となる。。

```
   ] pass0=.round_ga0 town
0  6j99  46j1 92j15 13j46 57j66 75j65 29j17 72j14 65j78  9j64 0
0 65j78 72j14  6j99 75j65 29j17 92j15  9j64  46j1 57j66 13j46 0
0 92j15 29j17  9j64  6j99 65j78 75j65 13j46  46j1 57j66 72j14 0
.....................
0  6j99 13j46  46j1  9j64 75j65 29j17 72j14 57j66 65j78 92j15 0
0 75j65 72j14 13j46  46j1  9j64 57j66 92j15  6j99 65j78 29j17 0
0 13j46 65j78  46j1 92j15  9j64 29j17 75j65 72j14 57j66  6j99 0
```

2. 各個体の経路合計を計算し評価しておこう

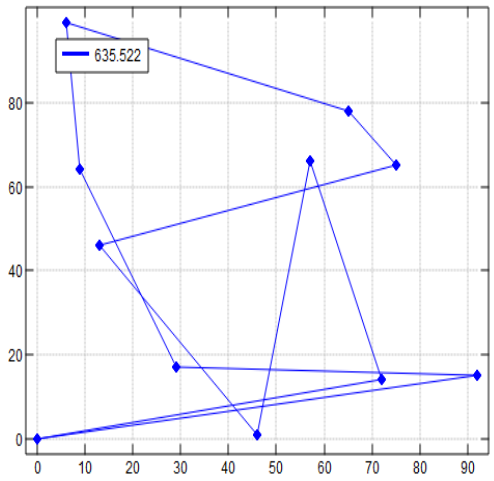
```
   calc_dist_all pass0
700.661 811.471 635.522 643.298 760.666 716.105
687.494 660.951 688.769 687.401 743.504 715.333
```

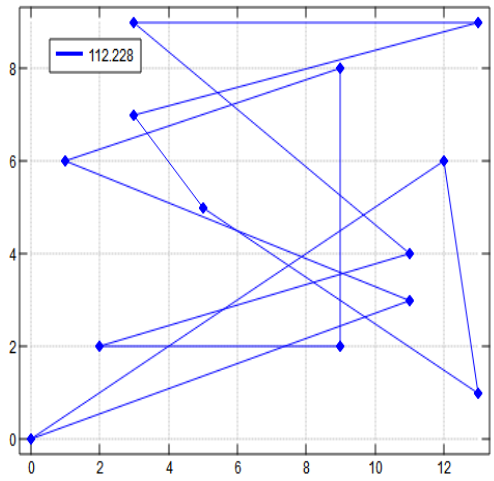
```
round_ga0=: 3 : 0
NB. Usage: round_ga0 town
NB. generate  # of y (town)
ans=: <(# y) # 1
for. i.# y do.
tmp=: round_order y
ans=. ans,<tmp
end.
>}. ans
)
```

```
calc_dist_all=: 3 : ' >+/@:dist L:0 { y '
```

3. 初期値の最短と最長の pass を図で見ておこう。

```
roundn  2{ pass0
```





交叉

1. 親を乱数でカップリングさせて次世代を一人づつ生む。遺伝子＝都市の重複は未チェック。TS_0 はクロスポイントが各カップル同一のモデル、TS_1 は乱数で各カップルのクロスポイントを決定するモデルである。
   - カップリング

カップリングとクロスは OneMax モデルと同一

```
   ,. cutx@couplingx tmp0
+------------------------------------------------------------------------+
|66j20 55j95 85j63 49j83 15j53 18j99 53j92 45j81 80j61 44j89  3j29 87j94|
|15j53 85j63 49j83 55j95  3j29 80j61 53j92 66j20 87j94 18j99 44j89 45j81|
+------------------------------------------------------------------------+
| 3j29 53j92 80j61 44j89 66j20 49j83 55j95 18j99 45j81 15j53 87j94 85j63|
|87j94 66j20 85j63 15j53  3j29 18j99 80j61 53j92 55j95 44j89 49j83 45j81|
+------------------------------------------------------------------------+
...
+------------------------------------------------------------------------+
|45j81 66j20 18j99 55j95 44j89  3j29 80j61 87j94 85j63 15j53 53j92 49j83|
|66j20 18j99  3j29 87j94 80j61 55j95 44j89 53j92 49j83 15j53 85j63 45j81|
+------------------------------------------------------------------------+
```

- クロス

```
   ] a=. crossTS_0 pass0
 3j29 66j20 85j63 44j89  3j29 49j83 80j61 18j99 45j81 44j89 87j94 45j81
45j81  3j29 53j92 55j95 44j89  3j29 55j95 87j94 85j63 66j20 53j92 49j83
45j81 18j99  3j29 49j83 80j61 44j89 44j89  3j29 87j94 15j53 66j20 45j81
80j61 87j94 55j95 53j92 15j53 66j20  3j29 55j95 49j83 80j61 45j81 49j83
18j99 55j95 49j83 53j92 53j92 45j81 85j63 80j61 87j94  3j29 44j89 80j61
15j53 55j95 85j63 55j95 15j53 80j61 53j92 66j20 87j94 44j89 44j89 87j94
45j81 80j61 18j99 49j83 44j89 44j89 55j95 80j61 15j53 53j92 87j94 66j20
```

- Script

```
crossTS_0=: 3 : 0
NB. Usage: crossTS_0 pass=. round_ga0 town
nr=. {:$ tmp0=: }."1 }:"1 y
CutTS=: {@|: L:0 cutx@couplingx tmp0  NB. {@|: make vertical 2 pair
Ind=: {@>genx nr
 >>(<Ind) { (L:0) CutTS            NB. Using double Box
NB. remove start and goal of 0
)
```

2. 重複の確認

   ~: で重複の確認ができる

```
   ~:"1   ] a=. crossTS_0 pass0
1 1 0 1 1 0 0 1 0 1 1 1
1 1 1 1 1 0 1 1 1 1 0 0
.....
1 1 1 1 0 1 1 0 1 1 1 0
```

3. 補充/complate
   重複の指標と書き換え用のアドレスを得て、補完する
   - e. (members of)
     含まれているものの指標, アドレス、抜き出し。指標を反転すると抜けているものを選び出すことができる

```
    ( (a=. 1 2 3 4 5) e.3 5 7)
0 0 1 0 1
   I. 1 2 3 4 5 e.3 5 7         NB. adress
2 4
   ( (a=. 1 2 3 4 5) e.3 5 7)#a
3 5                             NB. pickup data
```

   - 抜き出しの指標

```
   tmp1 e. {.tmp0
1 1 1 0 1 1 1 1 1 1 1 0
```

   - 指標を反転し 1 が立つ個所を抜き出す（未使用の遺伝子）

```
   (-.tmp1 e. {.tmp0)# tmp1
4j5 13j5
```

   - 書き換えアドレス

```
   ~: {. tmp0
1 1 1 1 1 1 1 1 1 1 0 0
 I. -.  ~: {. tmp0
10 11
```

4. 非重複の挿入データと書き換えアドレスの抽出

```
  >tmp1  writeback_sub L:0 {tmp0
+---------------+--------+
|4j5 6j1 13j6   |3 5 6   |
+---------------+--------+
|4j5 7j8 7j9 9j4|3 7 8 11|
+---------------+--------+
.............
+---------------+--------+
```

```
|6j1 6j7 7j6    |6 9 11  |
+---------------+--------+
```

5. 抜けた遺伝子の補完. ~:"1 ですべて 1 になれば完遂

```
   town complete pass0
13j6  7j9  7j8 13j5 11j2  7j6  4j5 10j4  6j1  4j9  6j7  9j4
 7j9 10j4  7j6  7j8  6j1 13j5 11j2  4j5  4j9 13j6  6j7  9j4
13j6 11j2  4j5  7j6  7j8  4j9 10j4 13j5  7j9  6j7  6j1  9j4
 4j9 10j4  7j9 11j2 13j5 13j6  6j1  7j6  7j8  4j5  6j7  9j4
13j6 13j5 10j4  7j8  4j9  7j9  9j4  7j6  4j5  6j1  6j7 11j2
 4j5  7j9  4j9  7j8  9j4 13j6  6j7  6j1  7j6 10j4 13j5 11j2
13j5  4j5  9j4  4j9  7j6 11j2  7j8  7j9 13j6 10j4  6j7  6j1
```

```
   ~:"1 a=. crossTS_0 pass0
1 0 1 1 1 1 1 1 0 0 1 0
1 1 1 1 0 0 1 0 0 1 1 0
1 1 0 1 1 1 1 0 1 0 1 0
1 1 0 1 1 1 1 0 1 1 0 0
1 1 1 0 1 0 1 0 1 1 1 0
1 1 1 0 1 0 1 1 1 1 1 1
1 1 1 1 0 1 1 0 1 1 0 1
```

```
   ~:"1 town complete a
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

6. Script

```
   complete=: 4 : 0
NB. Usage: town  complete  pass0

tmp0=: crossTS_0 y NB.  pass0=.  round ga0
tmp1=:}.}: x
data=:  >tmp1  writeback_sub L:0 {tmp0
NB. --------------------------
ans=. <''
for_ctr. i. # data do.
NB. pass1=: ({."1 data)({:"1 data)} L:0 {tmp0
'dat0 add'=. ctr{data
tmp2=. dat0 (add)} ctr{tmp0
ans=. ans,<tmp2
end.
>}.ans
```

```
writeback_sub=: 4 : 0
NB. y is tmp0
NB. x is tmp1
NB. usage: tmp1 writeback_sub L:0 { tmp0
adress=:   I. -.  ~:  y  NB. tmp0
seltown=:   (-. x e. y)# x
seltown adress
)
```

突然変異

1. 新世代の各個体の 2 個の遺伝子を選び前後を入れ替える

```
   ({.b0),: mutation_TS_sub {.b0
                *                                 *
4j9 4j5 9j4 6j1  7j6 13j6 7j9 11j2 7j8 6j7 10j4 13j5
4j9 4j5 9j4 6j1 13j5 13j6 7j9 11j2 7j8 6j7 10j4  7j6

   adress
11 4
```

2. Script

```
mutation_TS=: 3 : 0
tmp=. y
ans=. <''
for_ctr. i. # y do.
tmp=. mutation_TS_sub ctr{y
ans=. ans,<tmp
end.
>}.ans
)
```

```
mutation_TS_sub=: 3 : 0
NB. Usage: mutation_TS y
NB. y is  town  complete  pass0
 ind=: (i.nr) e. 2 ? nr=. # y
exdat=:  ind # y
adress=: |. I. ind
exdat (adress)}y
)
```

選択/淘汰

現世代と子の世代を合わせてソートし、各個体の経路合計を最小値から順に指定個数採るだけ。

優秀な個体は最上位につけ、なかなか世代交代の影響を受けない。丁度祖祖父が奥座敷でパイプをくゆらせながらロッキングチェアを揺らしているようなものである。

整理

一度整理しておこう

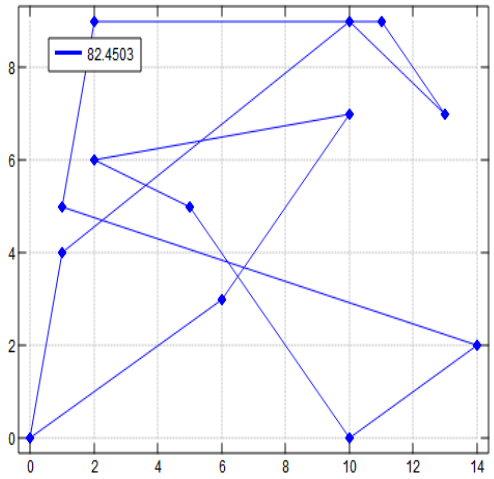
1. step0 `town=: setup_town 12 100`
2. step1 `'marker' plot town`
3. step2 `] pass0=. round_ga0 town`
4. step3 `calc_dist_all pass0`

```
  2 7 $ /:~  calc_dist_all pass0
82.4503 86.9225 88.2815  90.164 92.0198 92.1458  96.712
97.9372 107.182 107.886 109.876 110.209 113.295 124.389
```

5. GA 適用

```
a2=. town ga_TS_step pass0
```

```
   2 7 $  calc_dist_all a2
82.4503 85.2455 86.9225 88.2815  90.164 91.2837 92.0198
92.1458  96.712 97.9372 105.242 105.407 107.182 107.886
```



全体では良くなっている

初期値で良いのが入ると最小値だけ見ていると「だるま落し」のように何回でクリアできるかというつまらないものになる。

世代ごとに経路合計の最少値、四分位偏差、中央値、最大値を記録して世代ごとの推移を追うこととする

1. 処理するデータ（ソート済み）

```
   calc_dist_all town  ga_TS_step pass0
79.7404 82.4503 86.9225 88.2815 90.164 92.0198 92.1458
96.712 97.2284 97.879 97.9372 100.169 107.182 107.886
```

2. 整数と実数の区別

   x：で分数表示に戻し、2 x：で分子分母を分離して分母が 1 の数を区別する

```
   x: 2 2.5 0 _1 _1.2
2 5r2 0 _1 _6r5
  check_integer 2 2.5 0 _1 _1.2
1 0 1 1 0


 check_integer =: 3 : '({:("1)  2 x: y) e. 1 '
```

3. 四分位と中央値を取る（0 オリジンの場合）

```
    1r4 1r2 * 14
```

```
7r2 7
  check_integer  1r4 1r2 * 14
0 1
```

整数の場合は切り上げた数 4
実数の場合は当該数と次の数をとり、その平均

```
   take_42 pass0
82.4503 92.0198 102.56 124.389
```

4. Script

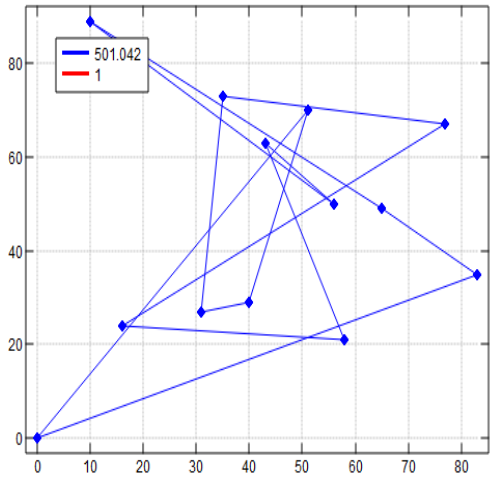
```
take_42=: 3 : 0
NB. usage:  take_42 pass0
NB. minimum, 1r4 ,1r2, maximum
nr=. # tmp=: /:~ calc_dist_all y NB. pass0
'Q M' =: check_integer nr * 1r4 1r2
'nrQ nrM'=: 1r4 1r2 * nr
NB. Q
if. 0 = Q do.
  QDATA=. (>. nrQ){ tmp
else. NB. integer
  QDATA=.  -: +/ (nrq , >: nrq){tmp
end.
NB. M
if. 0 = M do.
   MDATA=. (>. nrM){tmp
else. NB. integer
   MDATA=. -: +/ (nrM ,>: nrM){tmp
end.
({. tmp),QDATA,MDATA,{:tmp
)
```

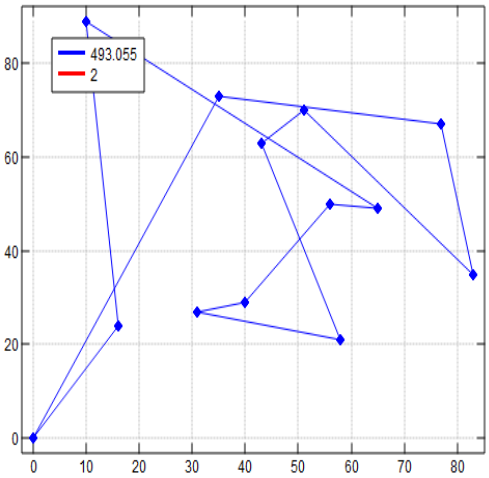
5　GA による巡回セールスマンのメインプログラムと経過

GA を一回適用

GA を一回適用。グラフは自動出力。

```
   town ga_TS_model pass0
501.042 567.001 627.774
493.055 533.099 566.279
   最少      四分位      中央値
```





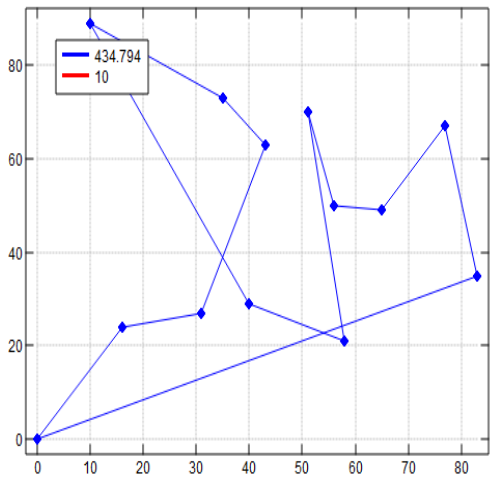

```
 ga_TS_model=: 4 : 0
NB.  only  1 step ga TS model
NB. usage: town ga_TS_model y is pass0
NB. x is town
NB. py is pass0=. round_ga0  town
NB. 0) initial step
tmpr0=: y   NB. (/: calc_dist_all y) { y
FST=. take_42 tmpr0
 (0, {. FST) plot_ga_TS {. y  NB. plot initial
NB. ga 1st step
tmpg0=. x  ga_TS_step y  NB. pass0
GA0=. take_42 tmpg0
(1,{.GA0) plot_ga_TS {. tmpg0
NB. --------------------
FST,:GA0
)
```

GA の反復モデル

1. GA の反復モデル。

```
   (10;town) ga_TS_modelx pass0
 499.47 533.099  580.12
 499.47 533.099 563.086
460.401  501.82 548.692
460.401  501.82 522.205
460.401  501.82 519.859
460.401  501.82 511.845
460.401  499.47 502.053
453.412 482.474 499.716
442.155 463.807 479.276
442.155 460.401 475.671
```

10 回の反復で 9 回目にベストが出たが何時もこのようになるわけでもない。がさがさと漸近し、四分位や中央値も改善している。
グラフの洪水を避けるため BEST が改善された場合にのみ出力するようにしている



2. Script

```
ga_TS_modelx=: 4 : 0
NB. x is 10;town
NB. y is pass0
NB. Usage: (100;town) ga_TS_modelx pass0
NB. --initial step--
'times town'=: x
tmp1=. y
FST=. take_42 tmp1                    NB. top Quarter middle
 (0, {. FST) plot_ga_TS {. tmp1    NB. plot top pass
BEST=. {. > ANS=. < FST
NB. --ga_step--
for_ctr. i. times do.
 tmp1=. town  ga_TS_step tmp1
 GA0=. take_42 tmp1
 ANS=. ANS,<GA0
  if. BEST > {.GA0 do.  NB. for plot
    BEST=. {. GA0
    (ctr,{.GA0) plot_ga_TS {. tmp1 NB. plot
  end.
end.
 >}.ANS
)
```

3. 100 回反復した最小値、四分位、中央値の変化を示す。がさがさと漸近する遺伝子アルゴリズムの様子を見て取ることができる
(結果は最初に提示した）

References

本田啓一郎「人工知能の強力手法ー遺伝子アルゴリズム」日経ソフトウエア 2019 年 3 月号
Script は http://japla.sakura.ne.jp の Workshop 2019/03 から DL できる