# GA & PSO演習報告

**ID: 20202251201 Name: 陳実**

# 序論

　遺伝的アルゴリズム（GA）は、自然進化過程をシミュレーションすることによって最適解を探索するアルゴリズムである。複雑な組合せ最適化問題を解く場合、伝統的な最適化アルゴリズムと比べて、より良い最適化結果を迅速に得ることができる。しかし、遺伝的アルゴリズムは通常、伝統的な最適化アルゴリズムより効率が低い。

　粒子群最適化法（PSO）は、集団協調に基づくランダム探索アルゴリズムである。かなり速い近似最適解の速度を持ち、効率的にシステムのパラメータを最適化することができる。しかし、いくつかの問題では、性能が良くない。

　今回の演習では、遺伝的アルゴリズムと粒子群最適化法を実現し、目標関数：

の最大化及び最小化を求める。**最大化の結果は407.08333、最小化の結果は-602.00000。**私はPythonとVSCodeで遺伝的アルゴリズムと粒子群最適化法を実現し、matplotlibを利用して結果の画像を表現した。**完全なソースコードはpyファイルにある。**第2部分では、遺伝的アルゴリズムの部分ソースコードと結果の画像を示す。第3部分では、粒子群最適化法のを示す。最後に第4部分では、簡単な結論である。

# 遺伝的アルゴリズム

### 部分ソースコード

　詳細はGenetic\_Algorithm.pyを参照。

# GA\_function

def GA\_function(max\_or\_min, is\_square):

# generation 0

individuals = np.random.randint(2, size=(NUM, GENE\_LENGTH \* 3))

# calculate fitness

fitness = calculate\_fitness(individuals, max\_or\_min)

# 1~T generations

for \_ in range(T):

# roulette

individuals = roulette\_selection(individuals, fitness, is\_square)

# crossover & mutation

individuals = np.array(mutation(crossover(individuals)))

# calculate fitness

fitness = calculate\_fitness(individuals, max\_or\_min)

# print max or min fitness

if max\_or\_min == 1:

print("max fitness: %.5f" % (np.max(MAX\_FITNESS)))

elif max\_or\_min == -1:

print("min fitness: %.5f" % (np.min(MIN\_FITNESS)))

if \_\_name\_\_ == "\_\_main\_\_":

# get max fitness

GA\_function(1, IS\_SQUARE)

# get min fitness

GA\_function(-1, IS\_SQUARE)

# show the result by graph

plt.plot(MAX\_FITNESS)

# set titles

plt.title("Max fitness, P\_C = " + str(P\_C) + ", P\_M = " + str(P\_M) +

(", square" if IS\_SQUARE == 1 else ", no square"))

plt.show()

plt.plot(MIN\_FITNESS)

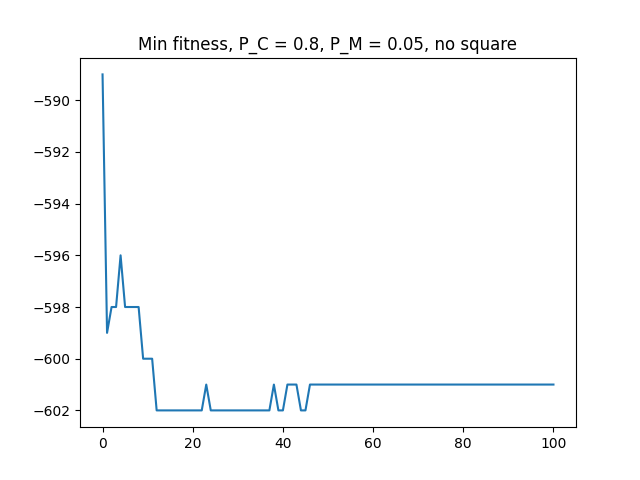
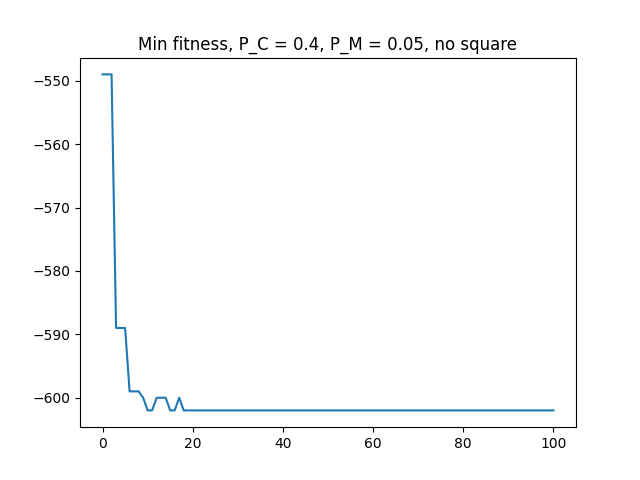
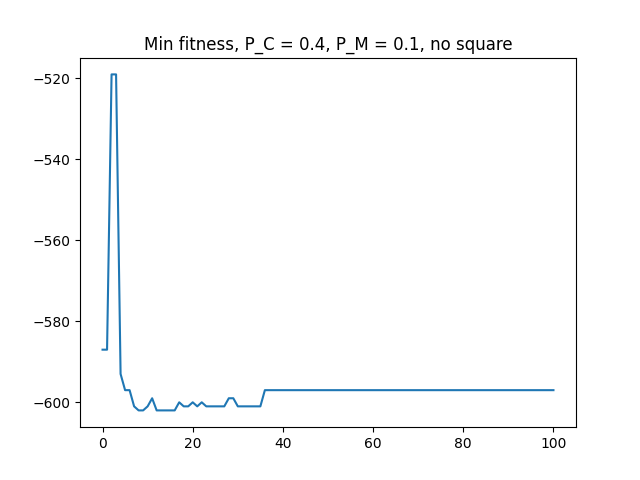
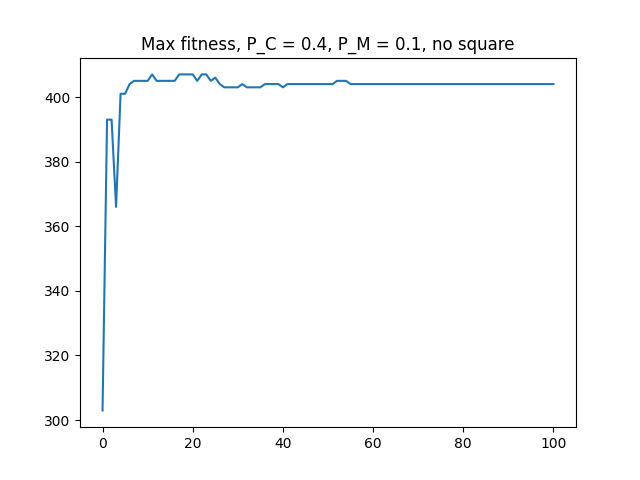
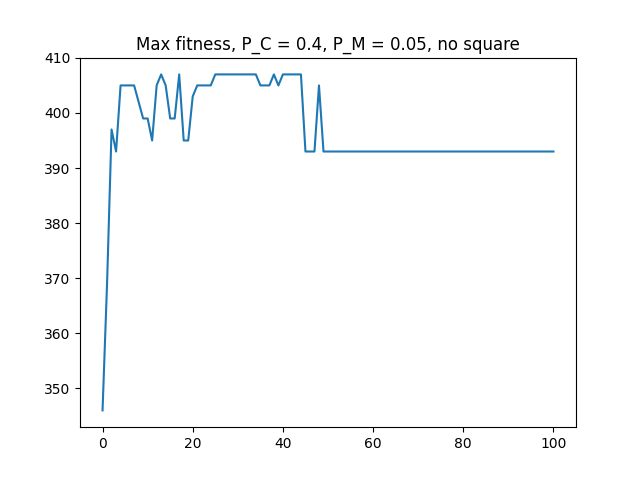
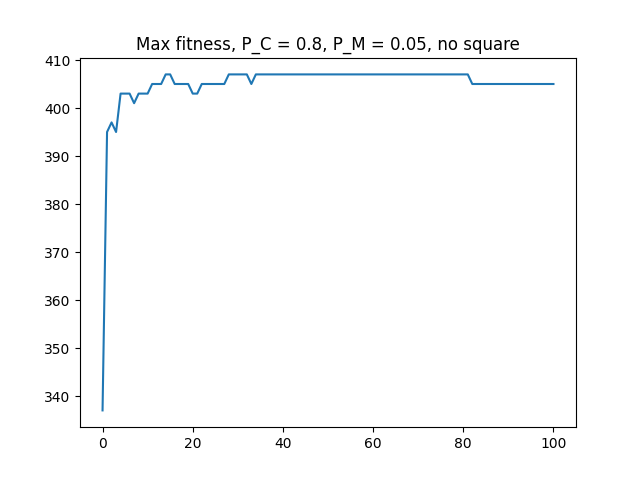
plt.title("Min fitness, P\_C = " + str(P\_C) + ", P\_M = " + str(P\_M) +

(", square" if IS\_SQUARE == 1 else ", no square"))

plt.show()

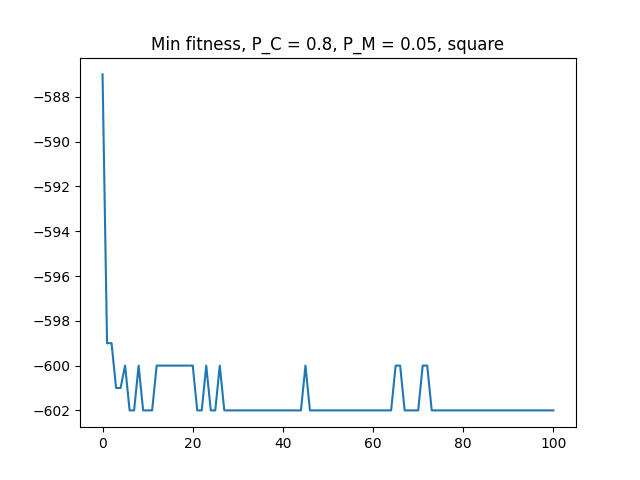
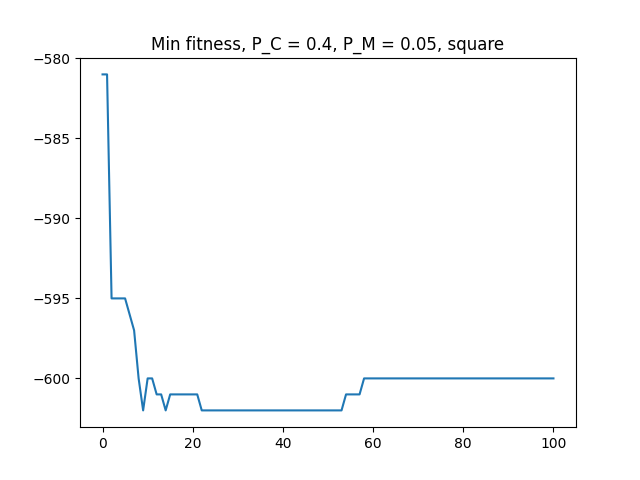
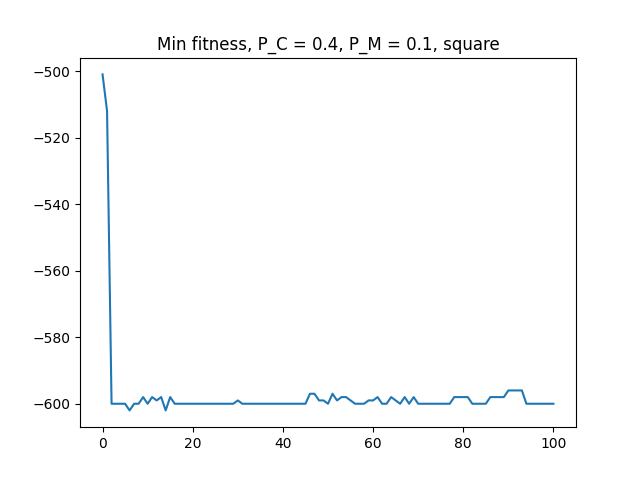
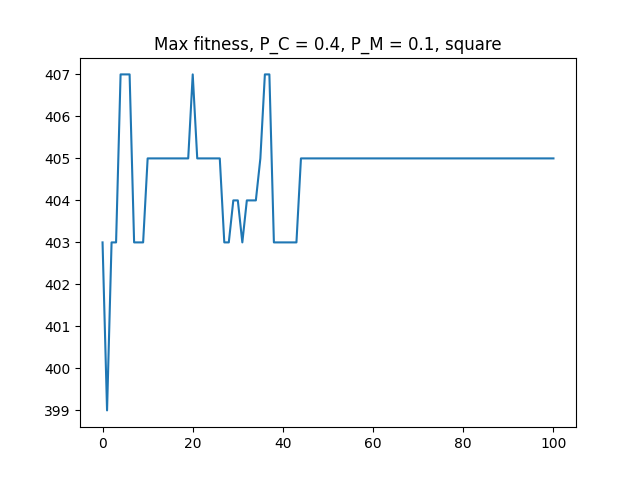
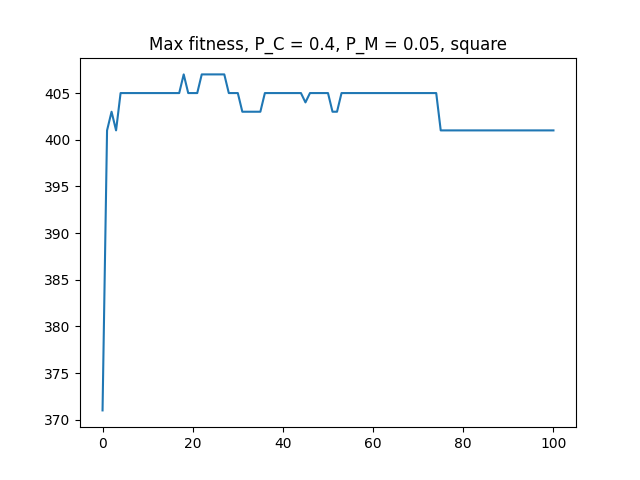
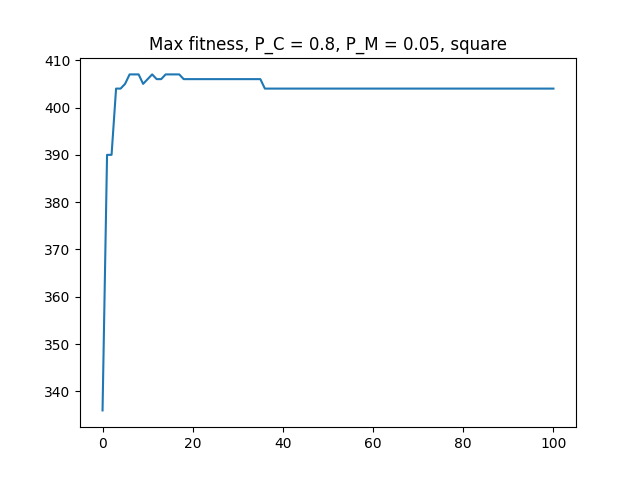
### 結果

　まずは結果の画像：



　上の6つの図は、適応度に比例させる場合に、パラメータを調整する結果の画像である。*P\_C*は交叉確率であり、*P\_M*は突然変異確率である。交叉確率は低いと、個体群の更新が遅くなるが、高いと、既存の最適解を破壊しやすく、最適解のランダム性が増大する。突然変異確率は低いと、収束不安定を招きやすいが、高いと、最適解のランダム性が増大する。

　上の6つの図は、適応度の2乗に比例させる場合に、パラメータを調整する結果の画像である。適応度の２乗に比例させる場合、適応度に比例させる場合より、最適解が選択される確率が高くなり、得られやすくなる。



# 粒子群最適化法

### 部分ソースコード

　詳細はParticle\_Swarm\_Optimization.pyを参照。

# PSO function

def PSO\_function(self, max\_or\_min):

for \_ in range(self.max\_steps):

# according to the formula

# r1 and r2 are random numbers, increasing the search randomness

r1 = np.random.rand(self.individual\_size, self.dim)

r2 = np.random.rand(self.individual\_size, self.dim)

# update velocity and C1, C2

self.v = W\*self.v+C1\*r1\*(self.p-self.x)+C2\*r2\*(self.pg-self.x)

minn = [[self.x\_bound[0]]\*self.dim]\*self.individual\_size

maxn = [[self.x\_bound[1]]\*self.dim]\*self.individual\_size

# limit the bound

self.x = clamp(self.v + self.x, minn, maxn)

# calculate fitness

fitness = self.calculate\_fitness(self.x, max\_or\_min)

# update one individual

update\_id = np.greater(self.individual\_best\_fitness, fitness)

self.p[update\_id] = self.x[update\_id]

self.individual\_best\_fitness[update\_id] = fitness[update\_id]

# update global best fitness

if np. max(fitness) > self.global\_best\_fitness:

self.pg = self.x[np.argmax(fitness)]

self.global\_best\_fitness = np.max(fitness)

# append into the sets

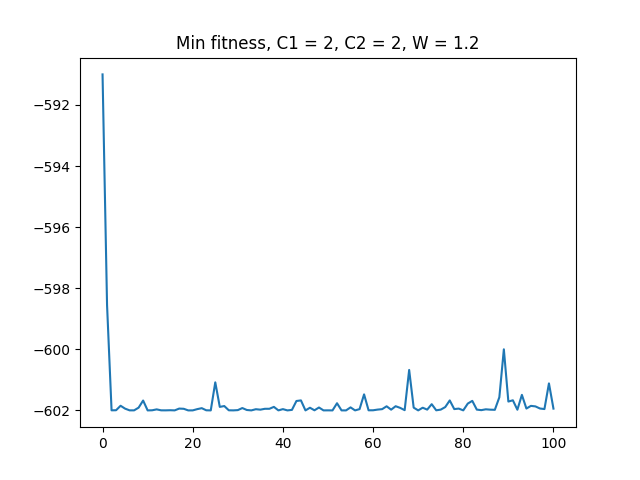
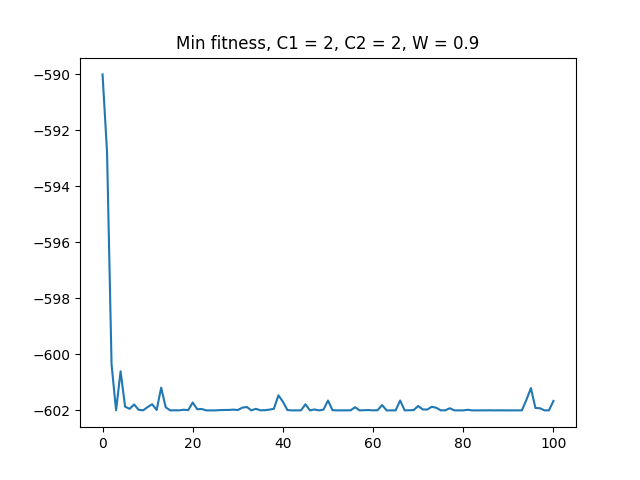
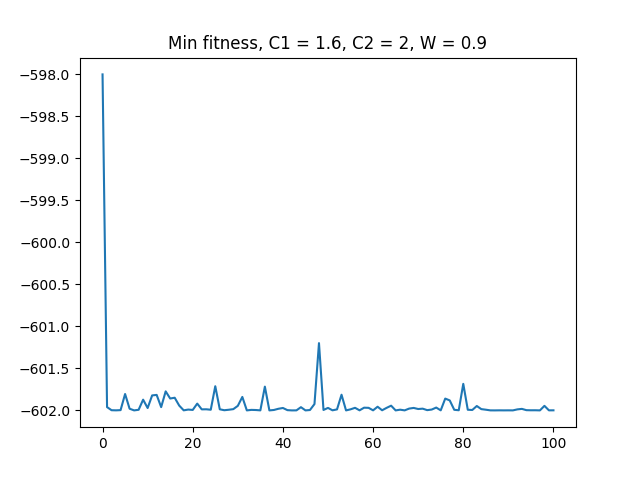
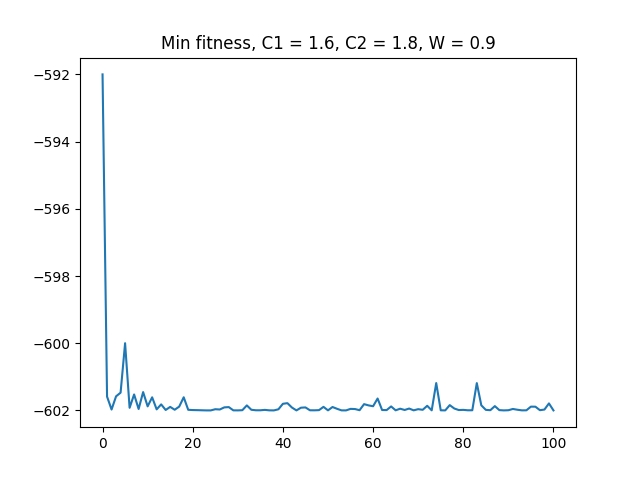
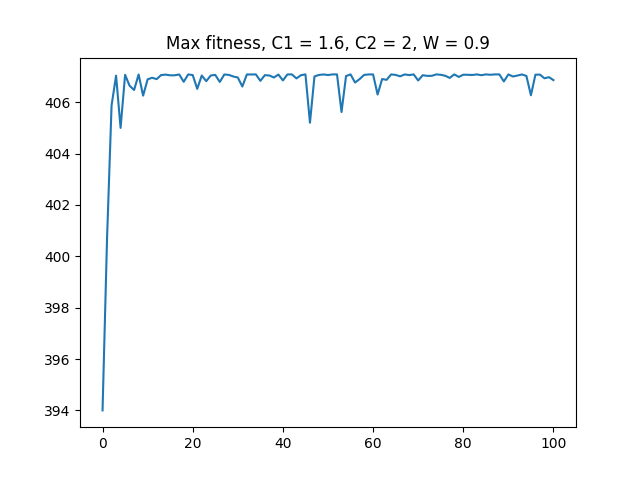
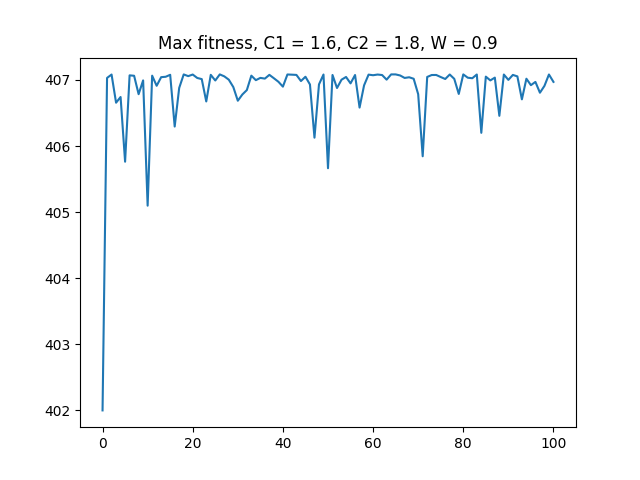
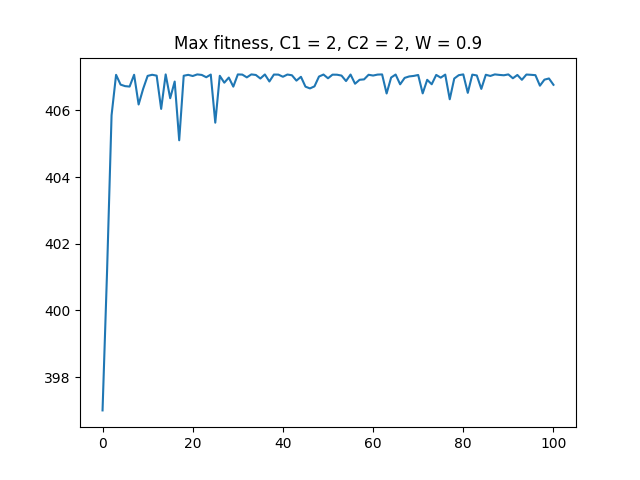
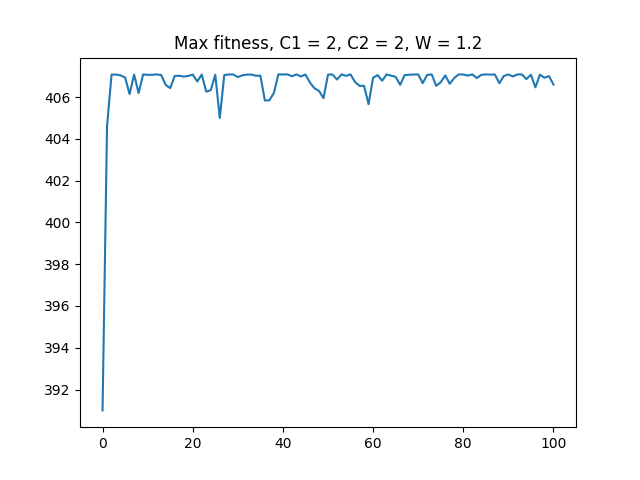
if max\_or\_min == 1:

MAX\_FITNESS.append(np.max(fitness))

elif max\_or\_min == -1:

MIN\_FITNESS.append(-np.max(fitness))

### 結果



　まずは結果の画像：

　上の8つの図は、諸パラメータを調整する結果の画像である。*C1*、*C2*は学習係数、*W*は慣性係数である。*W*は大きいと、グローバル検索に有利であり、局所最適に陥りにくいが、小さいと、局所探索に有利であり、最適解に迅速に収束させる。*C1*は小さいと、集団の多様性を失い、局所最適に陥りやすく飛び出せない。*C2*は小さいと、アルゴリズムの収束速度は遅くなる。

# 結論

　2つのアルゴリズムを比較すると、GAでは記憶がなく、以前の知識は個体群の変化に伴って破壊される。PSOでは記憶があり、優解の知識は全ての粒子が保存されている。両者の情報共有メカニズムも異なる：GAでは、染色体は互いに情報を共有し、個体群全体の移動は最適領域へ均一に移動するが、PSOでは、*gBest*だけが他の粒子に情報を与え、これは一方向の情報フローである。GAと比較して、ほとんどの場合、PSOの粒子はより早く最適解に収束する可能性がある。