# GA & PSO演習報告

**ID: 20202251201 Name: 陳実**

# **序論**

遗传算法 (GA) 是一种通过模拟自然进化过程搜索最优解的算法。在求解较为复杂的组合优化问题时，相对一些常规的优化算法，通常能够较快地获得较好的优化结果。但是遗传算法通常的效率比其他传统的优化方法低。

粒子群最优化算法 (PSO) 是一种基于群体协作的随机搜索算法。具有相当快的逼近最优解的速度，可以有效的对系统的参数进行优化。但是在某些问题上性能并不是特别好。

这次的演习，需要我们利用遗传算法和粒子群最优化算法，对目标函数

进行最大值和最小值的求解。我利用了 Python 和 VSCode，对遗传算法和粒子群最优化算法进行了实现，同时利用 matplotlib 库将结果的图像展现出来。完整源码在py文件中。在第2部分，将展示遗传算法的部分代码和结果；在第3部分，将展示粒子群最优化算法的部分代码和结果；在第4部分，将进行简短的总结。

# **遗传算法**

### **部分代码**

# 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)

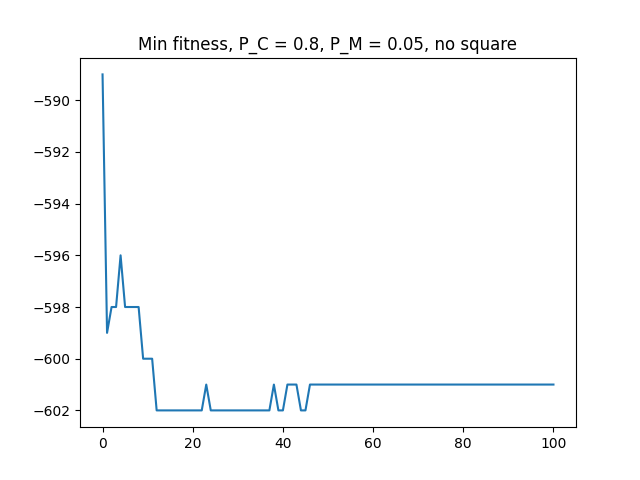
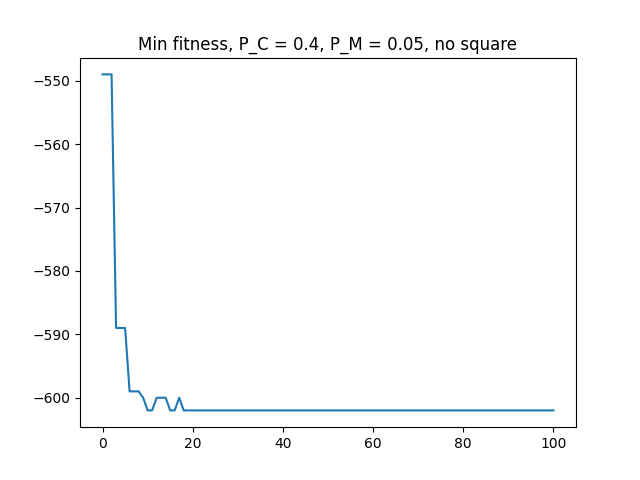
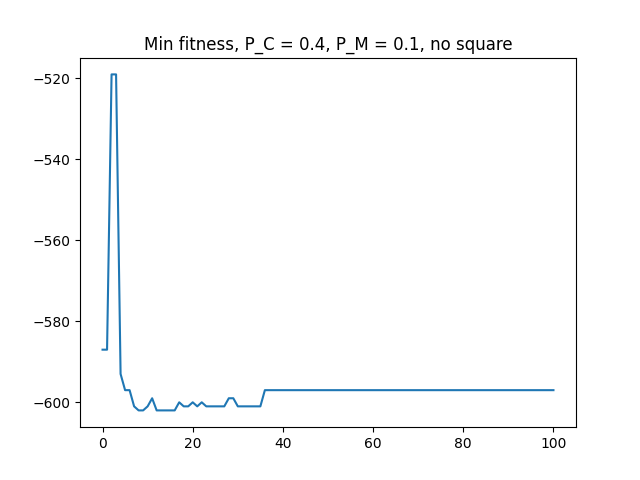
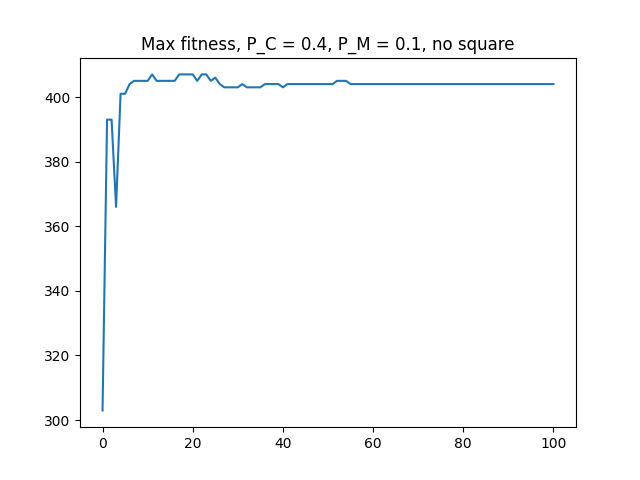
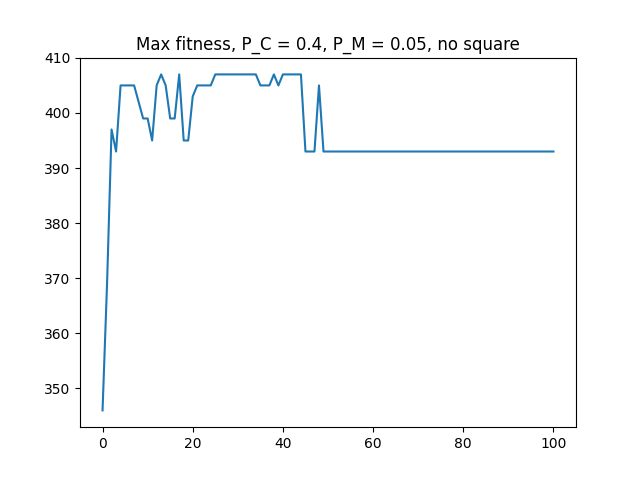
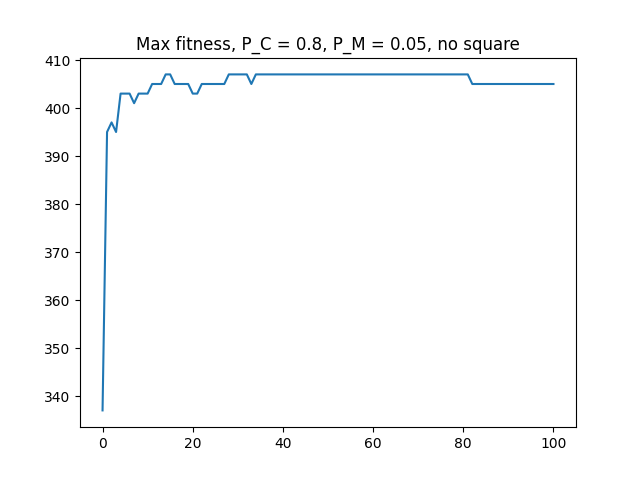
plt.title("Min fitness, P\_C = " + str(P\_C) + ", P\_M = " + str(P\_M) +

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

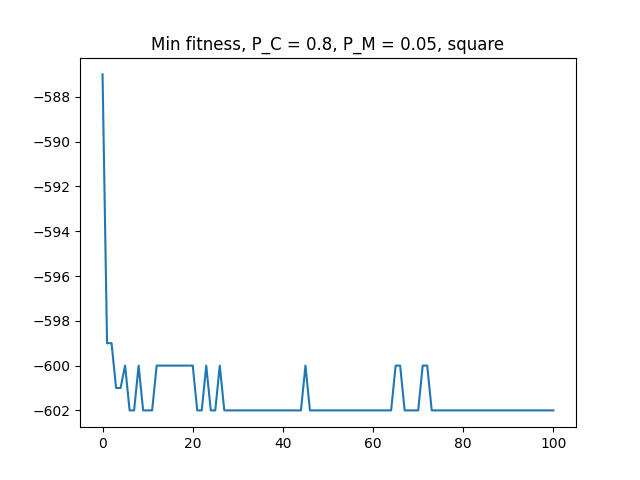
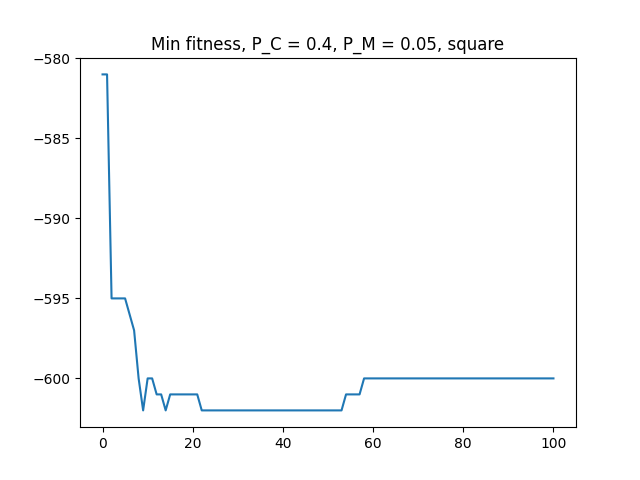
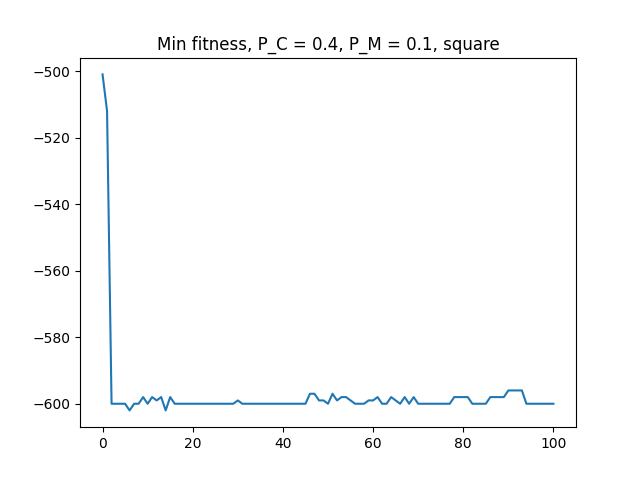
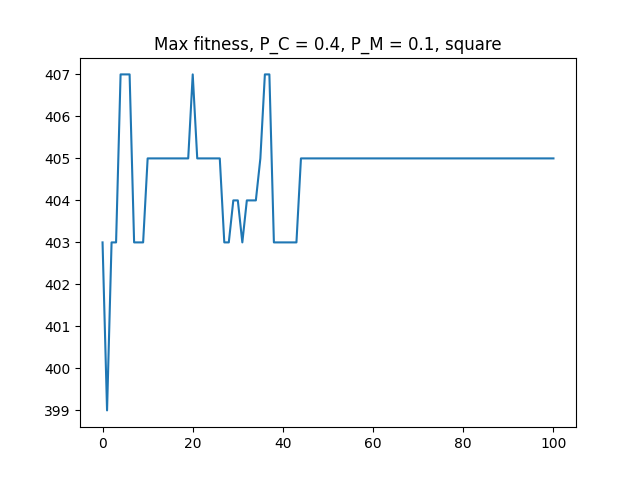
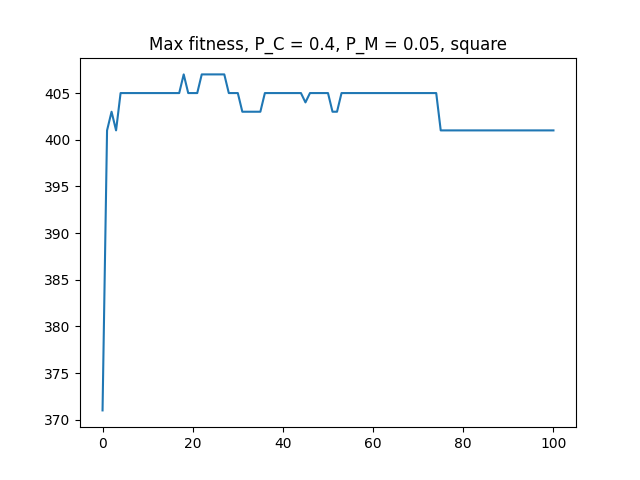
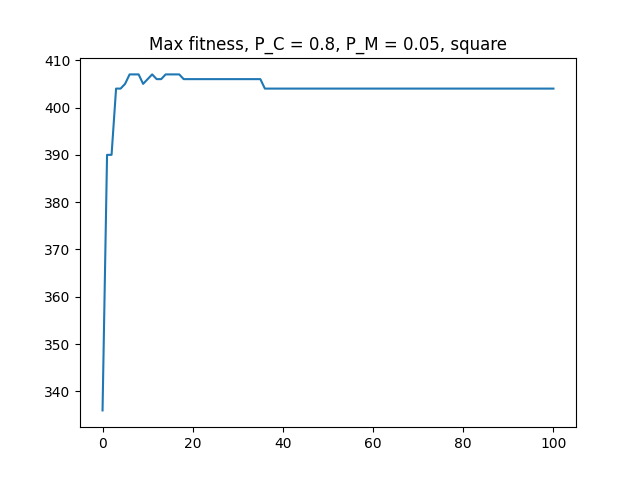
plt.show()

### **结果**

首先是图片结果：



上面六个图是在与适应度成比例的情况下、调整参数的结果图像。P\_C是交叉概率，P\_M是变异概率。交叉概率小，种群更新减慢；交叉概率大，容易破坏已有的最适解，随机性增大。变异概率小，容易导致收敛不稳定；变异概率大，最适解的随机性增大。



上面六个图是在与适应度的二次方成比例的情况下、调整参数的结果图像。当与适应度的二次方成比例的情况下，与适应度成比例相比，最适解被选择的概率增大，更容易得到最适解。

# 粒子群最优化算法

### 部分代码

# 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

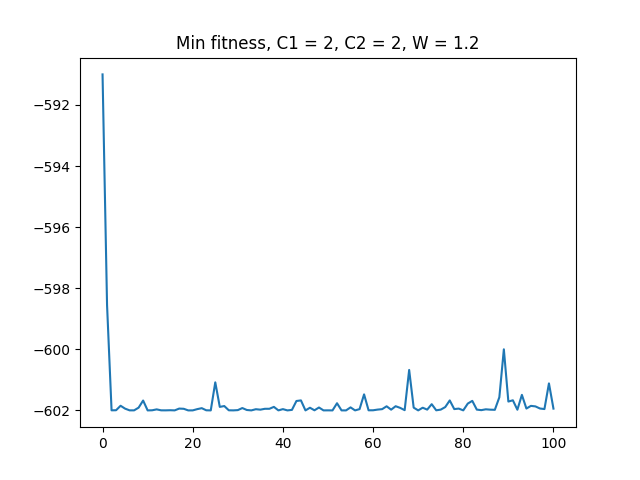
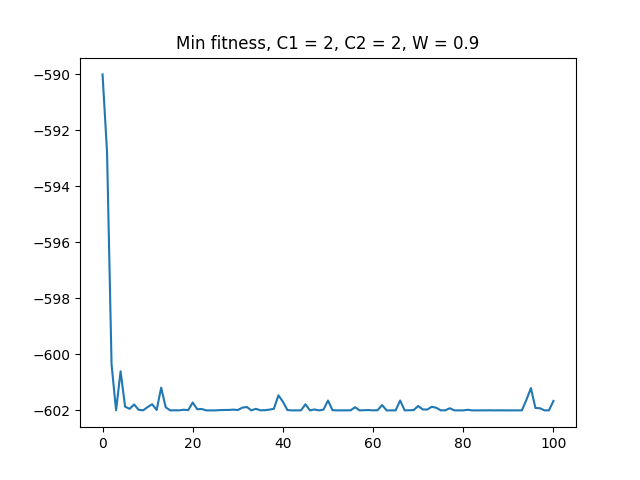
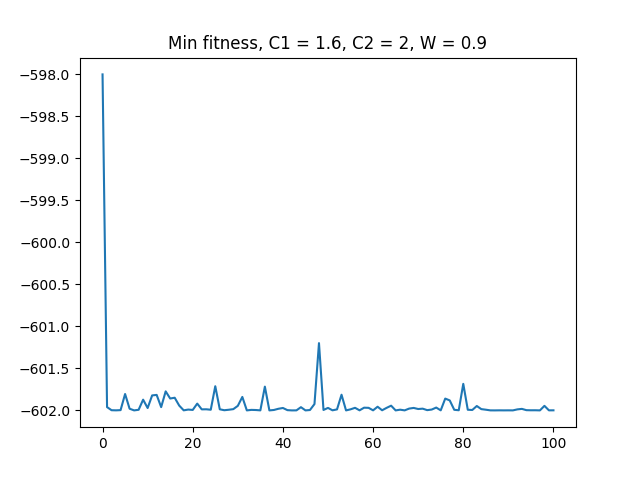
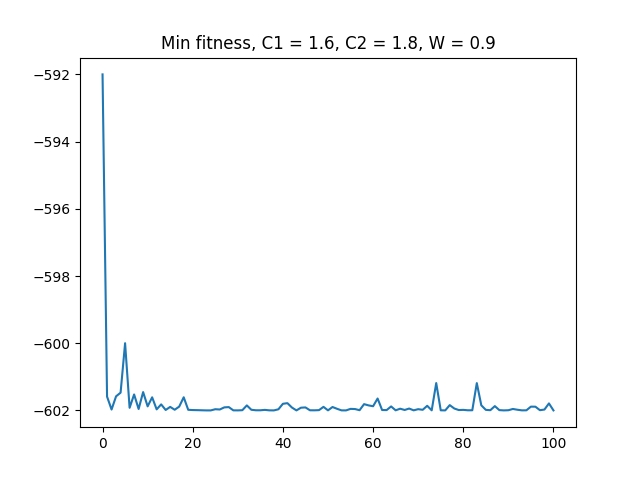
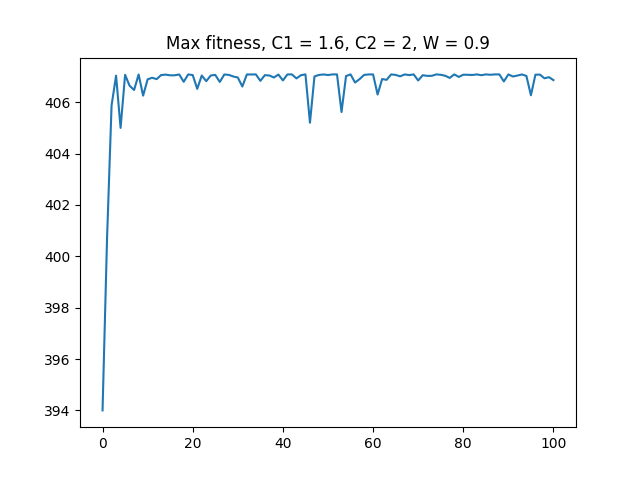
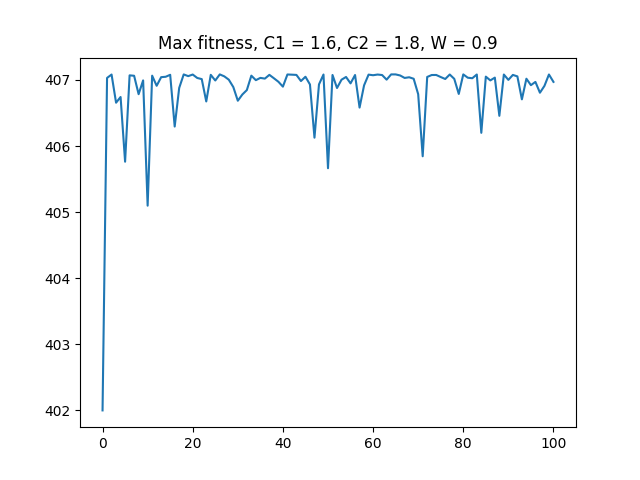
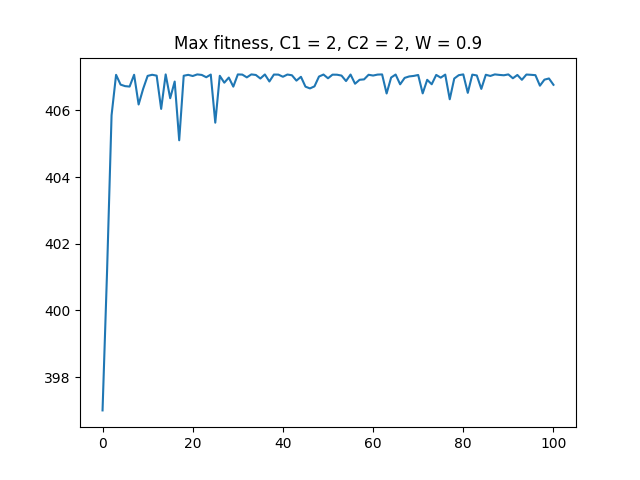
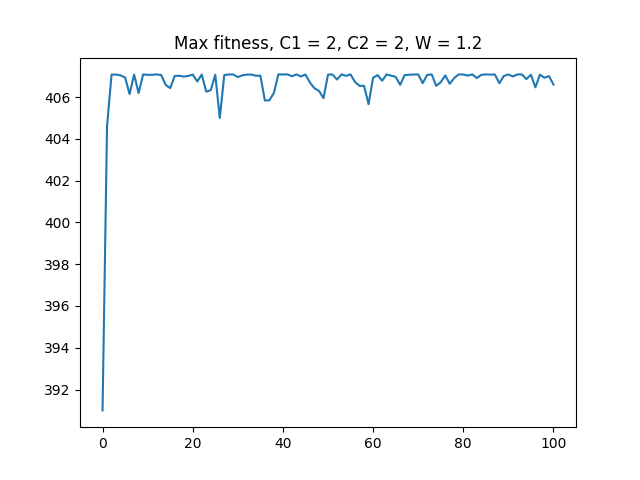
if max\_or\_min == 1:

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

elif max\_or\_min == -1:

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

### 结果



上面八个图是在调整不同参数的结果图像。C1、C2是学习系数，W是惯性系数。W较大，有利于全局搜索，不容易陷入局部最优；W较小，有利于局部搜索，让算法快速收敛到最优解。C1较小，丧失群体多样性，易陷入局部最优而无法跳出；C2较小，算法收敛速度会减慢。

# 总结

两个算法相比较，GA没有记忆，以前的知识随着种群的改变被破坏；PSO有记忆，优解的知识所有粒子都保存。两者的信息共享机制不同：在GA中，染色体互相共享信息，整个种群的移动是均匀向最优区域移动；在PSO中, 只有gBest给出信息给其他的粒子， 这是单向的信息流动。与GA相比, 在大多数的情况下，PSO的粒子可能更快收敛于最优解。