人工智能原理设计报告题库

本文档为“人工智能原理-设计报告题库”，包括启发式搜索、对抗搜索、演化计算/群体智能、神经网络和强化学习等方向。每位同学可以从题库中任选 1 题。

评分规则：包括是否完成题目要求、工作量、提交时间、格式的规范等。鼓励“小题大做”：鼓励增加不同算法的对比、实验分析（包括大规模数据上的性能分析）、算法正确性分析、时间性能与空间性能分析、算法改进等。

报告模板：参考教学资料“实验报告模板：202021XXXX-王二-《人工智能原理》.doc”

报告命名方式：学号-姓名-《人工智能原理》-设计报告。

作业的提交方式：Educoder。

一、 启发式搜索

一、实验目的：

熟悉和掌握启发式搜索的定义、估价函数和算法过程，理解求解流程和搜索顺序。

二、实验方法：

1.先熟悉启发式搜索算法；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

1.估价函数

在对问题的状态空间进行搜索时，为提高搜索效率需要和被解问题的解有关的大量控制性知识作为搜索的辅助性策略。这些控制信息反映在估价函数中。

估价函数的任务就是估计待搜索节点的重要程度，给这些节点排定次序。估价函数可以是任意一种函数，如有的定义它是节点 x 处于最佳路径的概率上，或是 x 节点和目标节点之间的距离等等。在此，我们把估价函数 $f(n)$定义为从初始节点经过 n 节点到达目标节点的最小代价路径的代价估计值，它的一般形式是：

$$f(n) = g(n) + h(n)$$

其中 $g(n)$是从初始节点到节点 n 的实际代价，$g(n)$可以根据生成的搜索树实际计算出来；$h(n)$是从 n 到目标节点的最佳路径的代价估计，$h(n)$主要体现了搜索的启发信息。

2. 启发式搜索过程的特性

（1）可采纳性

当一个搜索算法在最短路径存在的时候能保证能找到它，我们就称该算法是可采纳的。所有 A*算法都是可采纳的。

（2）单调性

一个启发函数 h 是单调的，如果

a） 对所有的状态 n_i 和 n_j，其中 n_j 是 n_i 的子孙，$h(n_i)- h(n_j)\leq cost(n_i,n_j)$，其中 $cost(n_i,n_j)$是从 n_i 到 n_j 实际代价。

b） 目标状态的启发函数值为 0，即 $h(Goal)=0$.

具有单调性的启发式搜索算法在对状态进行扩展时能保证所有被扩展的状态的f值是单调递增（不减）。

（3）信息性

比较两个启发策略h1和h2，如果对搜索空间中的任何一个状态n都有$h1(n)\leq h2(n)$，就说h2比h1具有更多的信息性。

一般而言，若搜索策略h2比h1有更多的信息性，则h2比h1考察的状态要少。但必须注意的是更多信息性需要更多的计算时间，从而有可能抵消减少搜索空间所带来的益处。

3.常用的启发式搜索算法

（1）局部择优搜索算法（瞎子爬山法）

瞎子爬山法是最简单的启发式算法之一。该算法在搜索过程中扩展当前节点并估价它的子节点。最优的子节点别选择并进一步扩展；该子节点的兄弟节点

和父节点都不再被保留。当搜索到达一种状态，该状态比它的所有子状态都要好，则搜索停止。因此，该算法的估价函数可表示为f(n) = h(n)。

在一个限定的环境下，瞎子爬山法可能会极大的提高搜索的效率，但是对整个搜索空间而言，可能得不到全局最优解。

（2）最好优先搜索法（有序搜索法）

该算法的估价函数采用f(n) = g(n) + h(n)，在搜索过程中算法使用OPEN表和CLOSE表来记录节点信息：OPEN表中保留所有已生成而未考察的节点；CLOSE表中保留所有已访问过的节点。算法在每一次搜索过程中都会对OPEN表中的节点按照每个节点的f值进行排序，选择f值最小节点进行扩展。算法的描述如下：

① 把起始节点S放到OPEN表中，计算f(S),并把其值与节点S联系起来。

② 若OPEN是个空表，则算法失败退出，无解。

③ 从OPEN表中选择一个f值最小的节点i。结果有几个节点合格，当其中有一个为目标节点时，则选择此目标节点，否则就选择其中任一个节点作为节点 i 。

④ 把节点i从OPEN表中移出，并把它放入到CLOSED的扩展节点表中。

⑤ 若节点i是个目标节点，则成功退出，求得一个解。

⑥ 扩展i，生成其全部后继节点。对i的每个后继节点j:

（a） 计算f(j)。

（b） 如果j既不在OPEN表中，也不在CLOSED表中，则用估价函数f将其添加到OPEN表。从j加一指向其父辈节点i的指针，以便一旦找到目标节点时记住一个解答路径。

（c） 如果j已则OPEN表中或CLOSED表中，则比较刚刚对j计算过的f值和前面计算过的该节点在表中的f的值。若新的f值较小，则

（i） 以此值取代旧值。

（ii） 从j指向i，而不是指向它的父辈节点。

（iii） 若节点j在CLOSED表中，则把它移回OPEN表。

⑦ 转向②。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）状态表示的数据结构

（2）状态扩展规则的表示

（3）搜索产生的状态空间图（样例）

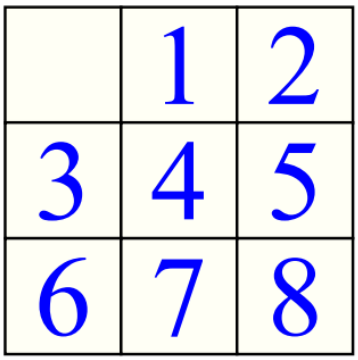
（4）OPEN表和CLOSE表变化过程（样例）

（5）衡量指标

（6）程序清单

（7）实验结果讨论

题目 1：问题描述：用启发式搜索方法求解九宫问题，目标状态如下图



延伸（选做）：

a. 随机生成至少10个初始状态作为测试样例，通过实验比较分析A*算法与盲目搜索算法（深度优先和宽度优先）。

b. 随机生成至少10个初始状态作为测试样例，通过实验比较分析不同的启发式搜索策略。

c. 实现A*算法的不同改进，并通过实验比较分析。

d. 比较不同的启发式函数对于启发式搜索算法性能的影响。

e. 若问题的初始状态是随机产生的，你的实验程序应该如何改进？（给于给定的任意初始状态S1，如何设计算法，使得能判定状态S1是否可以到达目标状态S2？）

题目 2：一般的传教士和野人问题

目的：熟练掌握启发式搜索算法及其应用。

问题背景：一般的传教士和野人问题（Missionaries and Cannibals）是指：有 N 个传教士和 N 个野人来到河边准备渡河。河岸有一条船，每次至多可供 K 人乘渡。问传教士为了安全起见，应如何规划摆渡方案，使得任何时刻，在河的两岸以及船上的野人数目总是不超过传教士的数目，但允许在河的某一岸只有野人而没有传教士。

基本任务：假设 K<= 3，请编程实现传教士和野人问题的求解，并以不同的 N 为例给出摆渡方案验证算法的正确性。

延伸：

A. 有哪些不同的盲目搜索算法和启发式搜索算法可以求解该问题，请实现不同的对比算法，并通过实验结果分析各算法的区别。

B. 实现 A*算法的不同改进，并通过实验比较分析。

C. 比较不同的启发式函数对于启发式搜索算法性能的影响。

题目 3：N 皇后问题

目的：熟练掌握搜索算法及其应用。

基本任务：在 N 行 N 列的国际象棋上摆放 N 个皇后，使其不能互相攻

击（即任意两个皇后都不能处于同一行、同一列或同一斜线上）。请问有多少种摆法，以及如何摆放这些皇后。请以八皇后问题为例给出摆放方案。

延伸：

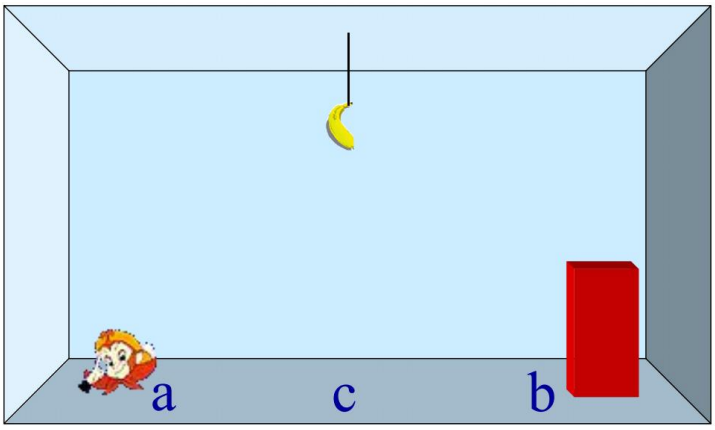
（1）有哪些算法可以求解该问题，请实现不同的算法，并从时间效率、空间效率、算法正确性等角度进行对比。

（2）从不同算法的对比结果中可以得出哪些有意义的结论？

（3）如何尽可能的提高搜索效率？

题目 4：猴子香蕉问题

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意 活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可 被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为 A，箱子位置为 B，香蕉位置在 C），如何行动可摘取到香蕉。



目的：熟练掌握搜索算法及其应用。

基本任务：对于上图中的地图，请编程求解并给出猴子可摘到香蕉的行动方案。

延伸：

（1）若换一个房间地图，问题求解有哪些变化？例如：房间中没有箱子、有 m 种不同的砖块（猴子只能通过把若干个砖块叠起来才能够到香蕉），每种砖块的数量是 n_i。每种砖块都是长方体，第 i 种砖块的大小是（a_i，b_i，c_i），位置是(x_i，y_i)。砖块能够翻转，可以将任意两边当作底面，剩下的那边作为高。问猴子应如何规划行动方案？

（2）有哪些算法可以求解该问题，请实现不同的算法，并从时间效率、空间效率、算法正确性等角度进行对比。

二、 对抗搜索

一、实验目的：

熟悉和掌握对抗搜索的定义、估价函数和算法过程，理解求解流程和基本算法。

二、实验方法：

1.先熟悉对抗搜索算法；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

在之前的实验中，我们仅研究单个智能体相关联的搜索策略，该智能体旨在找到通常以一系列动作的形式表达的解决方案。但是，可能存在多个智能体在同一搜索空间中搜索解决方案的情况，这种情况通常发生在博弈中。具有多个智能体的环境被称为多智能体环境，其中每个智能体可能是其他智能体的对手并且彼此竞争。每个智能体都需要考虑其他智能体的操作以及该操作对其性能的影响。因此，搜索两个或更多具有相互冲突目标的玩家试图探索解决方案的相同搜索空间的搜索称为对抗搜索，通常称为博弈。搜索算法和启发式评估函数是有助于在 AI 中建模和解决博弈问题的两个主要因素。

本实验中考虑的博弈问题是：有完整信息的、确定性的、轮流行动的、两个游戏者的零和游戏（如象棋）。

确定性的：表示在任何时间点上，玩家之间都有有限的互动；

零和游戏：意味着游戏双方有着相反的目标，换句话说，在游戏的任何终结状态下，所有玩家获得的总和等于零，有时这样的游戏也被称为严格竞争博弈。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）状态表示的数据结构

（2）状态扩展规则的表示

（3）搜索产生的状态空间图（样例）

（4）OPEN表和CLOSE表变化过程（样例）

（5）衡量指标

（6）程序清单

（7）实验结果讨论

题目 5：实现五子棋 AI

目的：熟练掌握极大值极小值算法、Alpha Beta 剪枝算法等对抗搜索算法及其应用。

基本任务：请编程实现一个五子棋 AI，给出算法设计思路与实现方案。完整的一次对弈过程可通过录屏的方式展示。

延伸：

A. 启发式函数应如何设计？分析比较不同的启发式函数带来的影响。

B. 通过实验分析搜索深度这一参数对于实验结果的影响。

C. 有哪些提高搜索效率的方法？

题目 6：实现井字棋 AI

目的：熟练掌握极大值极小值算法、Alpha Beta 剪枝算法等对抗搜索算法及其应用。

基本任务：请编程实现一个井字棋 AI，给出算法设计思路与实现方案。完整的一次对弈过程可通过录屏的方式展示。

延伸：

A. 启发式函数应如何设计？分析比较不同的启发式函数带来的影响。

B. 通过实验分析搜索深度这一参数对于实验结果的影响。

C. 有哪些提高搜索效率的方法？

三、 演化计算/群体智能

一、实验目的：

熟悉和掌握演化计算/群体智能的基本思想和基本方法，通过实验培养利用演化计算/群体智能进行问题求解的基本技能，并且了解其他分支的基本思想和基本方法。

二、实验方法：

1.先熟悉进化计算中演化计算/群体智能的基本思想和流程；

2.用 C、C++、JAVA 或 python 语言编程实现实验内容。

三、实验背景知识：

生物群体的生存过程普遍遵循达尔文的物竞天择、适者生存的进化准则。群体中的个体根据对环境的适应能力而被大自然所选择或淘汰。进化计算(evolutionary computation, 简称 EC；也称为演化计算)就是通过模拟自然物群进化的一类非常鲁棒的优化算法，它们模拟群体(其中组成群体的每个个体表示搜索问题解空间中的一个解)的集体学习过程，从任意初始群体出发，通过随机选择、变异和交叉过程，使群体进化学习到搜索空间中越来越好的区域。进化计算中选择过程使群体中适应性好的个体比适应性差的个体有更多的机会遗传到下一代中，交叉算子将父代信息结合在一起并他们遗传给下一代个体，而变异过程在群体中引入新的变种。进化计算包括遗传算法(evolutionary algorithm,EA)、进化策略(evolution strategy, ES)、进化编程(evolutionary programming, EP)、遗传编程(genetic programming, GP)等多个分支。

参考资料：

IEEE 动态优化社区主页：http://www.tech.dmu.ac.uk/~syang/IEEE_ECiDUE.html
IEEE CIS Task Force on Differential Evolution： http://labraj.feri.um.si/tf-cis-de/
IEEE CIS Task Force on Multi-Objective Evolutionary Algorithms:
http://www.is.ovgu.de/is_media/Research/IEEE_CIS_EMO_TF-p-1126.html
Suganthan (IEEE Fellow)：https://github.com/P-N-Suganthan/CODES
Yong Wang : http://ist.csu.edu.cn/YongWang.htm
Deb（IEEE Fellow）： http://www.egr.msu.edu/~kdeb/index.shtml
Xin Yao (IEEE Fellow)： https://www.cs.bham.ac.uk/~xin/
Shengxiang Yang http://www.tech.dmu.ac.uk/~syang/
Yaochu Jin (IEEE Fellow): http://www.soft-computing.de/
Michalewicz https://cs.adelaide.edu.au/~zbyszek/papers.html
Xiaodong Li：https://titan.csit.rmit.edu.au/~e46507/
PSO（部分开源代码）: http://www.adaptivebox.net/CILib/code/psocodes_link.html
全局优化算法（部分开源代码）：
http://www.mat.univie.ac.at/~neum/glopt/software_g.html
PSO（部分开源代码）: http://clerc.maurice.free.fr/pso/
多目标优化（部分开源代码）：
http://delta.cs.cinvestav.mx/~ccoello/EMOO/EMOOsoftware.html
http://www.macs.hw.ac.uk/~ml355/journals.htm

meta-heuristic 多种算法及 matlab 代码 https://yarpiz.com/
http://comopt.ifi.uni-heidelberg.de/software/

下面以简单的遗传算法为例介绍演化计算的基本求解过程。其他演化计算/群体智能算法可参考教材《计算智能导论》。

遗传算法是模仿生物遗传学和自然选择机理，通过人工方式构造的一类优化搜索算法，是对生物进化过程的一种数学仿真，是进化计算的一种最重要形式。遗传算法为那些那些难以找到传统数学模型的难题找出了一个解决方法。自从 Holland 于 1975 年在其著作《Adaptation in Natural and Artificial Systems》中首次提出遗传算法以来，经过近 30 年的研究，现在已发展到一个比较成熟的阶段，并且在实际中已经得到了很好的应用。

1.简单遗传算法的构成要素

（1）染色体编码和解码方法

在实现对一个问题用遗传算法之前，我们必须先对问题的解空间进行编码，以便使得它能够由遗传算法进行操作。解码就是把遗传算法操作的个体转换成原来问题结构的过程。常见的编码方法有：二进制编码、浮点数编码、格雷码等。以二进制为例：假设某一参数的取值范围是[A，B]，A＜B，我们有长度为 l 的二进制编码串来表示该参数，将[A，B]等分成 2^l -1 个子部分，每个等分的长度为 δ，则它能够产生 2^l 种不同的编码。

$$
\begin{aligned}
&00000000000\ldots0000000000 = 0 \quad \rightarrow \quad A\\
&00000000000\ldots0000000001 = 1 \quad \rightarrow \quad A+\delta\\
&\qquad\qquad\vdots\\
&11111111111\ldots\ldots1111111111 = 2^l-1 \quad \rightarrow \quad B
\end{aligned}
$$

二进制编码的最大缺点是使得遗传算法操作的个体位串太长，这容易降低遗传算法的运行效率，很多问题采用其他编码方法可能更有利。如对于函数优化问题，浮点数编码方法就更有效，而二进制编码方法不但容易降低遗传算法的运行效率，而且会产生精度损失。浮点数编码方法是指个体的每个染色体用某一范围内的一个浮点数表示，个体的编码长度就等于问题变量的个数。

在实际运用遗传算法解决问题时，一般都需要根据具体的问题采用合适的编码方法，这样更有利于遗传算法搜索到问题的最优解。

（2）适应度函数

遗传算法在进化搜索中基本上不用外部信息，仅用目标函数也就是适应度函数为依据。适应度函数是用于度量问题中每一个染色体优劣程度的函数，体现了染色体的适应能力。遗传算法的目标函数不受连续可微的约束且定义域可以是任意集合。但对适应度函数有一个要求就是针对输入可以计算出的能加以比较的结果必须是非负的。

在具体应用中，适应度函数的设计要结合求解问题本身的要求而设计。因为适应度函数对个体的评估是遗传算法进行个体选择的唯一依据，因此适应度函数的设计直接影响到遗传算法的性能。对于很多问题，可以直接把求解问题的目标函数作为适应度函数使用，但也存在很多问题需要进行一定的转换才能使得目标函数可以用作遗传算法的适应度函数。

（3）遗传算子

遗传算法主要有三种操作算子：选择(selection)、交叉(crossover)、变异(mutation)。

①选择算子

选择算子根据个体的适应度函数值所度量的优劣程度选择下一代个体。一般地，选择算子将使适应度函数值较大的个体有较大的机会被遗传到下一代，而适应度函数值较小的个体被遗传到下一代的机会较小。一般常采用的选择算子是赌轮选择机制。赌轮选择算子的基本原理如下。

令$\sum f_i$表示种群的适应度值之总和，f_i表示群体中第i个染色体的适应度值，则第i个个体产生后代的能力正好为其适应度值与群体适应度值的比值$f_i/\sum f_i$。

赌轮选择算子在个体数不太多时，有可能出现不正确反映个体适应度的选择过程，也就是说适应度高的个体有可能反而被淘汰了。为了改进赌轮选择算子的这种缺点，有很多改进的交叉选择算子。如：最佳个体保存法、期望值方法、排序选择方法、联赛选择方法、排挤方法等。

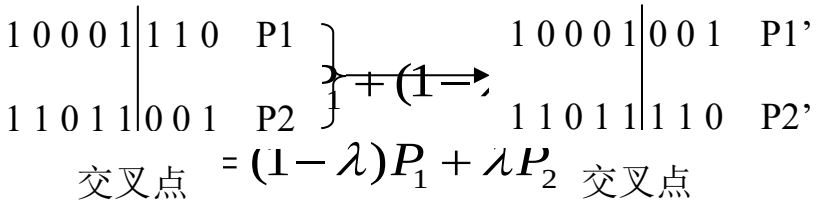
②交叉算子

在自然界生物进化过程中，起核心作用的是生物遗传基因的重组(加上变异)。同样，遗传算法中，起核心作用的是遗传操作的交叉算子。所谓交叉算子就是把两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃提高。

交叉算子设计一般与所求解的具体问题有关，但都应该考虑以下两点：Ⓐ设计的交叉算子必须能保证前一代中优秀个体的性状能在后一代的新个体中尽可能得到遗传和继承。Ⓑ交叉算子与问题的编码是相互联系的，因此，交叉算子设计和编码设计需协调操作，也就是所谓的编码—交叉设计。

对于字符串编码的遗传算法，交叉算子主要有一点交叉、两点交叉、多点交叉和一致交叉等。其中一点交叉的主要操作过程如下。假设两个配对个体为 P1、P2 分别如下所示。经过一点交叉后，得到两个新的个体 P1’、P2’。

对于实数编码的遗传算法，交叉算子主要是采用算术交叉算子。假设两个配对个体分别为 P1＝(x1,y1)和 P2=(x2,y2)。在 P1 和 P2 进行算术交叉后得到的两个新个体P_1'和P_2'分别可由下式计算得到。



$$= (1-\lambda)P_1 + \lambda P_2$$

③变异算子

变异算子是改变数码串中某个位置上的数码。遗传算法中引入变异算子有两个目的：其一是使遗传算法具有局部的随机搜索能力。当遗传算法通过交叉算

子已接近最优解领域时，利用变异算子的这种局部随机搜索能力可以加速遗传算法向最优解收敛。一般在这种情况下，遗传算法的变异概率应取较小的值，否则接近最优解的积木块会因为变异而遭到破坏。其二是使遗传算法可以维持较好的群体多样性，以防止遗传算法出现未成熟收敛现象。此时，遗传算法的变异概率应取较大的值。

遗传算法中，交叉算子具有很好的全局搜索能力。变异算子具有较好的局部搜索能力，是算法的辅助操作算子。遗传算法通过交叉和变异这一对相互配合又相互竞争的操作而使其具备兼顾全局和局部的均衡搜索能力。变异算子除了基本的变异算子坏，还有很多有效的变异算子。如逆转变异算子、自适应变异算子等。对于字符串编码的遗传算法，基本变异算子就是随机的从个体中选择某些基因位，然后以一定的概率对这些基因位进行翻转变异，也就是把这些基因位中为 0 的基因位以概率 *Pm* 变为 1，为 1 的基因为以概率 *Pm* 变为 0。对于实数编码的遗传算法，一般采用随机变异的方式，使变异个体的每一个变量分量加上一个随机数，一般使用均为分布的随机数或者是高斯分别的随机数。

（4）基本遗传算法运行参数

N：群体大小，即群体中所含个体的数量，一般取 20～100

T：遗传算法的终止进化代数，一般取 100～500

pc：杂交概率，一般取 0.4～0.99

pm：变异概率，一般取 0.0001～0.1

pr：复制概率

2.算法过程：

简单遗传算法的基本流程如下：

(1)初始化群体：随机产生一个由确定长度的特征串组成的初始群体

(2)计算群体上每个个体的适应度值

(3)按由个体适应度值所决定的某个规则选择将进入下一代的个体，也就是选择算子操作。

(4)按概率 Pc 对配对池中的个体进行交叉算子操作。

(5)按概率 Pm 对交叉算子产生的所有新个体进行变异操作。

(6)若没有满足某种停止条件，则转(2)，否则进入下一步。

(7)输出群体中适应度值最优的染色体作为问题的满意解或最优解。

四、实验内容：

本章实验中，设计报告应能覆盖以下内容：

（1）问题的编码和解码表示

（2）适应度函数

（3）算法参数设置

算法流程图

（4）选择算子、交叉算子、变异算子的设计

（7）算法的结束条件

（8）程序清单

（9）根据实验内容，给出相应结果并分析

题目 7：选择一种演化算法/群体智能算法，求解函数

f(x)=x*sin(10л+x)+1.0 在区间[-1，2]的最大值。

延伸：

a. 什么是过早收敛和过慢结束？造成上述状况的原因有哪些？

b. 你的程序有没有要改进之处？如何改进？

题目 8：用遗传算法求解下列函数的最大值，设定求解精度到 15 位小数。

$$f(x,y)=\frac{6.452(x+0.125y)(\cos(x)-\cos(2y))^2}{\sqrt{0.8+(x-4.2)^2+2(y-7)^2}}+3.226y$$

$$x\in[0,10], y\in[0,10]$$

1）给出适应度函数（Fitness Function）的 M 文件（Matlab 中要求适应度函数最小化）。

2）设计及选择上述问题的编码、选择操作、交叉操作、变异操作以及控制参数等，填入表 1，给出最佳适应度(Best fitness)和最佳个体（Best individual）图。

表 1 遗传算法参数的选择

编码	编码方式（population type）	
种群参数	种群规模（population size）	
	初始种群的个体取值范围（Initial range）	
选择操作	个体选择概率分配策略（对应 Fitness scaling）	
	个体选择方法（Selection function）	
最佳个体保存	优良个体保存数量（Elite count）	
交叉操作	交叉概率（Crossover fraction）	
	交叉方式（Crossover function）	
变异操作	变异方式(Mutation function)	
停止参数	最大迭代步数（Generations）	
	最大运行时间限制（Time limit）	
	最小适应度限制（Fitness limit）	
	停滞代数（Stall generations）	
	停滞时间限制（Stall time limit）	

3）使用相同的初始种群（Use random state from previous run），设置不同的种群规模（population size），例如 5、20 和 100，初始种群的个体取值范围（Initial range）为[0;1]，其他参数同表 1，然后求得相应的最佳适应度(Best fitness)、平均适应度（Mean fitness）和最佳个体（Best individual），填入下表 2，分析种群规模对算法性能的影响。

表 2 不同的种群规模的 GA 运行结果

种群规模	最佳适应度	平均适应度	最佳个体	
			x	*y*
5				
20				
100				

*4) 设置种群规模(population size)为 20，初始种群的个体取值范围(Initial range)为[0;10]，选择不同的选择操作、交叉操作和变异操作，其他参数同表 1，然后独立运行算法 10 次，完成下表 3，并分析比较采用不同的选择策略、交叉策略和变异策略的算法运行结果。

表 3 不同的选择策略、交叉策略和变异策略的算法运行结果

遗传算法参数设置（gaoptimset）			1	2	3	4
选择操作	个体选择概率分配 FitnessScalingFcn	Rank（排序）@fitscalingrank	√	√		√
		Proportional（比率）@fitscalingprop			√	
	个体选择 SelectionFcn	Roulette（轮盘赌选择）@selectionroulette	√	√		√
		Tournament（竞标赛选择）@selectiontournament			√	
交叉操作 CrossoverFcn	单点交叉 @crossoversinglepoint		√		√	√
	两点交叉 @crossovertwopoint			√		
变异操作 MutationFcn	Uniform（均匀变异）@mutationuniform		√	√	√	
	Gaussian（高斯变异）@mutationgaussian					√
最好适应度						
最差适应度						
平均适应度						

题目 9：用遗传算法求解下面一个 Rastrigin 函数的最小值，设定求解精度到 15 位小数。

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$$

1）给出适应度函数。

2）设计上述问题的编码、选择操作、交叉操作、变异操作以及控制参数等，填入表 4，并画出最佳适应度(Best fitness)和最佳个体（Best individual）图。

表 4 遗传算法参数的选择

编码	编码方式（population type）	
种群参数	种群规模（population size）	
	初始种群的个体取值范围（Initial range）	
选择操作	个体选择概率分配策略（对应 Fitness scaling）	
	个体选择方法（Selection function）	
最佳个体保存	优良个体保存数量（Elite count）	
交叉操作	交叉概率（Crossover fraction）	
	交叉方式（Crossover function）	
变异操作	变异方式(Mutation function)	
停止参数	最大迭代步数（Generations）	
	最大运行时间限制（Time limit）	
	最小适应度限制（Fitness limit）	
	停滞代数（Stall generations）	
	停滞时间限制（Stall time limit）	

3）设置种群的不同初始范围，例如[1;1.1]、[1;100]和[1;2]，画出相应的最佳适应度值(Best fitness)和平均距离（Distance）图，比较分析初始范围及种群多样性对遗传算法性能的影响。

4）设置不同的交叉概率（Crossover fraction=0、0.8、1），画出无变异的交叉（Crossover fraction=1）、无交叉的变异(Crossover fraction=0)以及交叉概率为 0.8 时最佳适应度值(Best fitness)和和平均距离（Distance）图，分析交叉和变异操作对算法性能的影响。

题目 10：针对 http://www.sfu.ca/~ssurjano/optimization.html 中列出的一个或多个测试问题，请实现至少一种演化算法/群体智能算法。

延伸：该问题还可以使用哪些算法进行求解？请实现不同的对比算法，通过实验结果分析各算法的区别，并分析参数对算法性能的影响。

题目 11：针对 http://www5.zzu.edu.cn/cilab/Benchmark/wysyhwtcsj.htm 中列出的一个或多个测试集，请实现至少一种演化算法/群体智能算法。

延伸：该问题还可以使用哪些算法进行求解？请实现不同的对比算法，通过实验结果分析各算法的区别，并分析参数对算法性能的影响。

四、神经网络及深度学习

题目 12：基于神经网络的优化计算

一、实验目的：

掌握连续 Hopfield 神经网络的结构和运行机制，理解连续 Hopfield 神经网络用于优化计算的基本原理,掌握连续 Hopfield 神经网络用于优化计算的一般步骤。

二、实验原理

连续 Hopfield 神经网络的能量函数的极小化过程表示了该神经网络从初始状态到稳定状态的一个演化过程。如果将约束优化问题的目标函数与连续 Hopfield 神经网络的能量函数对应起来，并把约束优化问题的解映射到连续 Hopfield 神经网络的一个稳定状态，那么当连续 Hopfield 神经网络的能量函数经演化达到最小值时，此时的连续 Hopfield 神经网络的稳定状态就对应于约束优化问题的最优解。

三、实验条件：

用 C、C++、JAVA 或 python 语言编程实现实验内容。

四、实验内容：

1、参考求解 TSP 问题的连续 Hopfield 神经网络源代码，给出 15 个城市和 20 个城市的求解结果（包括最短路径和最佳路线），分析连续 Hopfield 神经网络求解不同规模 TSP 问题的算法性能。

2、对于同一个 TSP 问题（例如 15 个城市的 TSP 问题），设置不同的网络参数，分析不同参数对算法结果的影响。

五、实验报告要求：

1、画出连续 Hopfield 神经网络求解 TSP 问题的流程图。

2、根据实验内容，给出相应结果及分析。

3、总结连续 Hopfield 神经网络和遗传算法用于 TSP 问题求解时的优缺点。

题目 13：人脸识别

1. 实验内容

特征脸（eigenface）是第一种有效的人脸识别方法，通过在一大组描述不同人脸的图像上进行主成分分析（PCA）获得。

本次实验要求构建一个自己的人脸库，可以通过现有的人脸库添加自己的人脸图像形成，人脸库中的每一张图像都只包含一张人脸且眼睛的中心位置已知。

在实验前首先根据眼睛的位置将图片对齐处理，在训练过程中，首先要求出平均脸，然后将前 K 个特征脸保存下来，利用这 K 个特征脸对测试人脸进行识别，此外对于任意给定的一张人脸图像，可以使用这 K 个特征脸对原图进行重建。

2. 实验要求

（1）不使用 OpenCV 的 Eigenface 接口，通过求解人脸图像的特征值与特征向量构建特征脸模型。

（2）利用特征脸模型进行人脸识别和重建，比较使用不同数量特征脸的识别与重建效果。

3. 实验环境

可以使用基于 Python 的 OpenCV 库进行图像相关处理，使用 Numpy 库进行相关数值运算。

题目 14：深度学习

1. 实验内容

自己选择一个机器学习相关的问题，并利用深度学习尝试解决这个问题。问题的选择可以是计算机视觉（人脸检测、人脸识别、图像分类等）、自然语言处理（语音识别、机器翻译、中文分词等）、推荐系统（兴趣地点推荐、用户评分预测等）、数据预测分析（红酒品质分类、金融数据预测等）。

目前多数深度学习模型都是依赖数据驱动的，所以在选择研究问题时，首先要考虑的是数据的获取，目前网络上有很多公开数据集可以供大家使用，例如 UCI 数据集，提供了大量机器学习的数据集。当然也可以自己收集数据，形成自

己的数据集。

模型的训练过程往往需要花费大量计算资源，推荐大家使用华为云 ModelArts 或者 Google Cloud Platform 等云端训练平台进行训练。

2. 实验要求

(1) 基本掌握一种深度学习开发库的使用。

(2) 能够使用深度学习库搭建自己的深度学习模型。

(3) 通过训练使模型在特定问题上得到较好的效果。

3. 实验环境

使用 Python 的 TensorFlow、PyTorch 等开源库。

题目 15：强化学习

1. 实验内容

使用强化学习 Q-Learning 算法设计一个可以在 4×4 大小的迷宫中找到宝藏的 agent。迷宫可以使用一个系列字符表示，字符“O”表示 agent 的位置，字符“T”表示宝藏的位置，该位置的收益为 1，且为游戏的终结状态，字符“P”表示陷阱的位置，该位置的收益为-1，且为游戏的终结状态，字符“_”表示收益为 0 的中间状态。

初始情况下，迷宫如下表示：

O _ _ _

_ P _ P

_ _ P _

_ _ _ T

每次行动 agent 可以选择向不超过迷宫边界的相邻位置移动一格，通过 Q-Learning 算法希望 agent 可以最终找到一条通向宝藏的路径。

通过改变决策 ε 的大小，考察 agent 选择路径方式的变化。

2. 实验要求

（1） 实现一个用字符表示的迷宫环境，能够对 agent 的行动以及收益返回相应的结果，并可视化地表示出来。

（2） 实现 Q-Learning 算法，包括 Q-Table 的更新以及根据 Q-Table 进行决策的函数。

（3） 利用 Q-Learning 算法训练一个能够在迷宫中找到宝藏的 agent。

3. 实验环境

不限制编程语言。