成　绩	
评卷人	

研究生	李雪
学　号	2015363039

武漢紡織大學

研 究 生 课 程 论 文

论文题目	遗传算法
完成时间	2020.11.15
课程名称	人工智能
专　　业	电子信息
年　　级	2020 级

武汉纺织大学研究生处制

一、 引言

遗传算法(Geneticl Agoirhtms，下称 GAs)是一类模拟生物进化过程与机制求解问题的自适应人工智能技术。它的核心思想是基于 Darwin 的进化论和 Mendel 的遗传学说：从简单到复杂、从低级到高级的生物进化过程本身是一个自然、并行发生的、稳健的优化过程，这一优化过程的目标是对环境的适应性，而生物种群通过“优胜劣汰”及遗传变异来达到进化的目的。如果把待解决的问题描述作为对某个目标函数的全局优化，则 GAs 求解问题的基本作法是：把待优化的目标函数解释作生物种群对环境的适应性，把优化变量对应作生物种群的个体，而由当前种群出发，利用合适的复制、杂交、变异与选择操作生成新的一代种群。重复这一过程，直至获得合乎要求的种群或规定的进化时限。

由于具有鲜明的生物背景和对任何函数类(特别可以无表达式)可用等突出特点，GAs 自 80 年代中期以来引起人工智能领域的普遍关注，并被广泛应用于机器学习、人工神经网络训练、程序自动生成、专家系统的知识库维护等一系列超大规模、高度非线性、不连续、极多峰函数的优化。90 年代以来，对该类算法的研究日趋成为计算机科学、信息科学与最优化领域研究的热点。

然而与通常优化技术中的搜索方法不同，GAs 作为一种自适应的随机搜索方法，其搜索方式是由当前种群所提供的信息，而不是由单一的方向或结构来决定。同时，它将多个个体作为可能的解并考虑搜索空间全局范围内的抽样，如此导致其能以更大的可能性收敛到全局最优解。由于这些特性，GAs 能够成功地用于求解众多不同的复杂而困难的优化问题(包括非数值优化问题)。

本文将从遗传算法的理论和技术两方面概述目前的研究现状，描述遗传算法的主要特点、基本原理以及各种改进算法，介绍遗传算法的应用领域，并对遗传算法的性能进行分析。

二、 相关工作

通过学习，了解到演化计算最初形成了三大分支，即遗传算法（Genetic Algorithm，GA）、演化规划（Evolutionary Programming，EP）、演化策略

（Evolution Strategy，ES）。在 20 世纪 90 年代初，在遗传算法的基础上又发展出了另一个分支——遗传程序设计（Genetic Programming，GP）。虽然这几个分支在算法的实现方面有一些差别，但是它们都是借助于生物演化的思想与原理来进行问题求解。以下将针对演化规划和演化策略的原理进行详解。

2.1 连续空间的演化规划

演化规划是 60 年代 Fogel 等人提出的一类演化算法，其基本思想可概括为：(a)将待优化问题的目标函数转换到某生物种群对环境的适应性；(b)将优化变量对应作生物种群的个体；(c)将所发展的求解优化问题的算法与生物种群的演化过程类比。

考虑全局优化问题

$$(P) \qquad \min\{f(x)；x\in S\subset R^n\}，f：S\subset R^n\rightarrow R^1 \qquad （1）$$

则(P)的多个可行解的一个集合称为一个种群，种群中的每一个元素(可行解)称为一个个体，种群中个体的数目称为种群的规模。设 S^*是 S 的全体子集构成的集合。

引入参数集合 A、B、C，定义四个函数：

(1) 竞争选取函数 f_c：$A\times S^*\rightarrow \psi(S^*)$，满足$\forall y\in f_c(\alpha，x)$：$y\subseteq x$，其中$\psi(S^*)$为 S^*的幂集。

(2) 变异函数 f_m：$B\times S^*\rightarrow \psi(S^*)$，满足$\forall \beta\in B, x\in S^*, f_m(\beta，x)\subseteq \bigcup_{s\in x} N(s)$，N(s)为个体 s 的邻域。

(3) 选择函数 f_s：$C\times S^*\rightarrow S^*$，满足$\forall \gamma\in C$，$x\in S^*$：$f_s(\gamma，x)\subseteq x$。

(4) 终止函数 f_e：$S^*\rightarrow${true，false}。

求解问题(P)的演化规划如下：

步骤 1 初始化，在 S^*中随机选取 x_0 为初始种群，置 k：=0；

步骤 2 产生新一代种群：1)依据某个竞争规则，选取获胜率最高的 N 个个体作为父本，使 $q_k=f_c(\beta，x_k)$；2)通过变异函数作用于父本，产生中间种群 $y_k=\bigcup_{\tau\in fc} f_m(\beta，\tau)$，$x'_k=x_k\cup y_k$；3)选择新一代种群 $x_{k+1}=f_s(\gamma，x'_k)$。

步骤 3 终止检验，如果 $f_e(x_{k+1})$=true，则停止演化，并输出最优解；否则，

置 k：=k+1，转步骤 2。

2.2 演化策略

2.2.1 Pareto 最优解

不失一般性，考虑下列 m 个自变量和 n 个目标函数的多目标函数优化问题：

Maximize　$y=f(x)=(f_1(x_1，\cdots\cdots，x_m)，\cdots\cdots，f_n(x_1，\cdots\cdots，x_m))$

其中 $x=(x_1，\cdots\cdots，x_m)\in X\subset R^m$，$y=(y_1，\cdots\cdots，y_n\in Y\subset R^n$，X 为决策(参数)向量，X 为决策空间，y 为目标向量，Y 为目标空间。

设 a∈X，b∈X，称 a Pareto 优于 b（Pareto dominate 记 a ≻ b）或 b 劣于 a 当且仅当∀ i{1，……，n}：$f_i(a)\geq f_i(b)$，称 a 覆盖 b（cover 记 a ≻=b）当且仅当 a ≻ b 或 f(a)=f(b)。

显然 Pareto 优于关系是偏序。下面给出非劣解或 Pareto 最优解的定义：

定义 设 a∈X，X′ 为 X 的子集，若 X′ 中不存在优于 a 的向量，则称 a 为关于子集 X'的非劣解或关于子集 X'的 Pareto 最优解，若 a 为关于 X 的非劣解，则 a 简称为非劣解或 Pareto 最优解。所有的 Pareto 最优解对应的目标函数值形成的区域称为均衡面(Pareto Optimal Front or Trade-off Surface)。

2.2.2 自适应(μ+1)演化策略

我们采用(μ+1)演化策略(ES Evolution Strategy)来实现算法，编码采用实数向量编码，对于实函数的优化问题，实数编码的性能优于传统的二进制编码。由父体产生后代时，不使用杂交算子，只使用变异算子，因为直观地来看，两个(或几个)相距较远的 Pareto 最优个体杂交后可能会产生一个较差的个体，杂交不仅没有优势，甚至还会有劣势。一般认为自适应变异法要优于非自适应变异法，我们使用高斯变异的自适应变异法。

自适应(μ+1)演化策略算法描述如下：

(1) 随机产生μ个个体组成初始群体，置 k=1 每个个体表示为(x_i，η_i)，i∈{1，2，……，μ}，x_i 为 m 维自变量，η_i 为高斯变异的标准方差，x_i 和 η_i 均为 m 维向量；

(2) 在μ个个体中任意取一个个体(x_i，η_i)，i∈{1，2，……，μ}由下式产生(x_i，η_i)的后代(x_i，η_i)，对 j=1，2，……，m，

$x_i'(j)=x_i(j)+\eta_i(j)N_j(0，1)$

$\eta'_i(j)=\eta_i(j)\exp(\tau'N(0，1)+\tau N_j(0，1))$

其中 $x_i(j)$、$x_i'(j)$、$\eta_i(j)$和$\eta'i(j)$分别是向量 x_i、x_i'、η_i 和η'_i 的第 j 个分量，N(0，1)表示一维的标准正态分布随机数，$N_j(0，1)$表示标准正态分布随机数随着每一个 j 而更新，步长 τ 和τ分别取为$(\sqrt{2\sqrt{m}})$和$(\sqrt{2m})^{-1}$；

(3) 按式(1)计算(μ+1)个个体的适应值，去掉适应值最大的个体；

(4)如停机规则满足则停机，否则 k=k+1，并转(2)。

三、 算法描述（伪代码）

遗传算法的实现过程实际上就像自然界的进化过程那样。下面以袋鼠的自然进化为例对遗传算法的运行过程进行描述。

(1) 首先寻找一种对问题潜在解进行“数字化”编码的方案（建立表现型和基因型的映射关系）；

(2) 然后用随机数初始化一个种群（那么第一批袋鼠就被随意地分散在山脉上），种群里面的个体就是这些数字化的编码；

(3) 接下来，通过适当的解码过程之后（得到袋鼠的位置坐标），用适应性函数对每一个基因个体作一次适应度评估（袋鼠爬得越高，越是受我们的喜爱，所以适应度相应越高）；

(4) 最后用选择函数按照某种规定择优选择（我们要每隔一段时间，在山上射杀一些所在海拔较低的袋鼠，以保证袋鼠总体数目持平）。让个体基因变异（让袋鼠随机地跳一跳）。然后产生子代（希望存活下来的袋鼠是多产的，并在那里生儿育女）。

遗传算法并不保证你能获得问题的最优解，但是使用遗传算法的最大优点在于你不必去了解和操心如何“找”最优解。（不必去指导袋鼠向那边跳，跳多远。）而只要简单的“否定”一些表现不好的个体。（把那些总是爱走下坡路的袋鼠射杀，这就是遗传算法的优胜劣汰的准则！）

如图 3.1 所示，为遗传算法的一般步骤流程图如下：

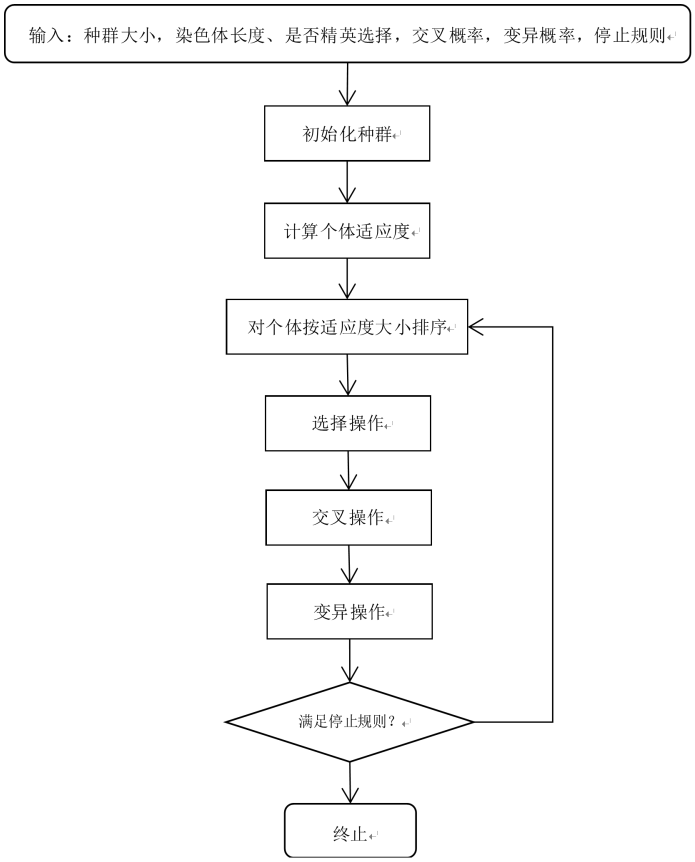

图 3.1 遗传算法运行流程图

TicTacToe 的遗传算法伪代码如下：

```
Input:
    None
Output:
    the best solution
def GA():
    generation_num = 3000
    population_num = 800
    prob_crossover = 0.15
    prob_replicate = 0.10
    prob_mutation = 0.001
    INDIVIDUAL_TEMPLATE = {}
    STATE = {}
    POPULATION = []
```

```
FITNESS = [0]*population_num
PROB = [0]*population_num

Init()
for t in range(generation_num):
    P_TMP = copy of POPULATION
    for i in range(population_num):
        seed()
        if random() <= prob_replicate:
            POPULATION[i] = Select(P_TMP)
        else:
            d1 = Select(P_TMP)
            d2 = Select(P_TMP)
            d = Crossover(d1, d2)
            Mutate(d)
            POPULATION[i] = d
    fitness_sum = CalculateFitness()
    #Update the statistics of population
    PROB[0] = FITNESS[0]/fitness_sum
    for i in range(1, len(FITNESS)):
        PROB[i] = PROB[i-1]+FITNESS[i]/fitness_sum
return the individual with MAX_FITNESS of POPULATION
def Init():
    ttt = TicTacToe()
    GenerateIndividualTemplate(ttt)
    items = INDIVIDUAL_TEMPLATE.items()
    for i in range(population_num):
        individual = GenRandomIndividual(items)
        POPULATION.append(individual)
    fitness_sum = CalculateFitness()
```

```
    PROB[0] = FITNESS[0]/fitness_sum
    for i in range(1, len(FITNESS)):
        PROB[i] = PROB[i-1]+FITNESS[i]/fitness_sum

    def GenerateIndividualTemplate(ttt):
        if ttt.isGameOver() != None:
            return
        moves = ttt.getAllMoves()
        ttt_str = ttt.ToString()
        if STATE.get(ttt_str) == None:
            for equ_str in GenEquivalent(ttt_str):
                STATE[equ_str] = ttt_str #base state
            INDIVIDUAL_TEMPLATE[ttt_str] = moves
        for move in moves:
            node = ttt.clone()
            node.play(move)
            GenerateIndividualTemplate(node)

def GenRandomIndividual(items):
    seed()
    individual = {}
    for ttt_str, moves in items:
        individual[ttt_str] = Random(moves)
    return individual

def Select(population):
    r = random()
    for i in range(len(PROB)):
        if r <= PROB[i]:
            return copy of population[i]
```

```
#d1, d2: two individuals
def Crossover(d1, d2):
    d = {}
    for key in d1.keys():
        r = random()
        if r <= prob_crossover:
            d[key] = d1[key]
        else:
            d[key] = d2[key]
    return d


#d: individual
#d[i][0]: encode of state i
#d[i][1]: move of state i
def Mutate(d):
    for key in d.keys():
        if random() <= prob_mutation:
            moves = INDIVIDUAL_TEMPLATE[key]
            d[key] = Random(moves)


def CalculateFitness():
    PLAY_NUM = [0]*population_num
    LOST_NUM = [0]*population_num
    for i in range(population_num):
        ttt = TicTacToe()
        lost_num, play_num = PlayGameAsFirst(ttt,
        →POPULATION[i])#from ttt to all states, as the first
        →player
```

```
            LOST_NUM[i] += lost_num
            PLAY_NUM[i] += play_num
            ttt = TicTacToe()
            lost_num, play_num = PlayGameAsSecond(ttt,
            →POPULATION[i])#from ttt to all states, as the second
            →player
            LOST_NUM[i] += lost_num
            PLAY_NUM[i] += play_num
        fitness_sum = 0
        for i in range(population_num):
            FITNESS[i] = 1 - LOST_NUM[i]/PLAY_NUM[i]
            fitness_sum += FITNESS[i]
        return fitness_sum
```

四、 算法验证（结果与分析）

4.1 函数 f(x) = x + 2 sin 2x + 3 sin 3x + 4 sin4x 在定义域 [0, 10]上的最大值

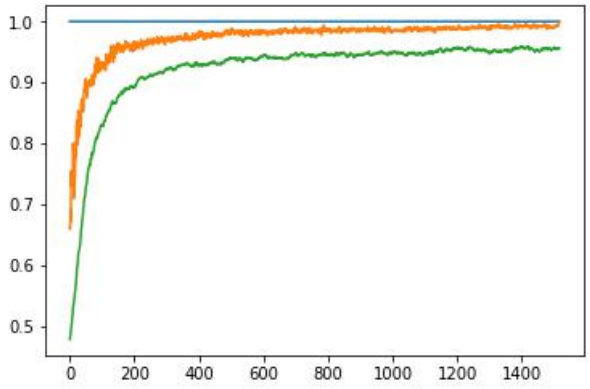

图 4.1 TicTacToe 遗传程序的迭代曲线

遗传迭代曲线如图 4.1 所示。程序在迭代到 1516 次时，最大适应度收敛到 1，表明策略可以保证算法不会输棋（因为胜与平的得分都是 1，而输的得分是 0）。

4.2 TicTacToe 的遗传算法对弈程序

程序在运行过程中，如图 4.1 所示为 TicTacToe 对弈的初始状态，图 4.2 中展示了双人对弈的过程，“X”为甲方，“O”为乙方。首先由甲方先出棋，乙方其后，轮流出棋，图 4.2 中棋盘的左上角均标注了下棋的顺序。

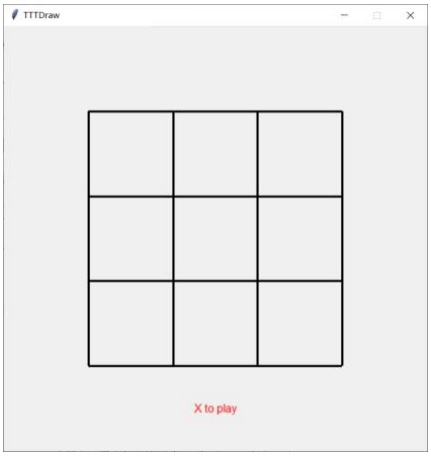

图 4.1 TicTacToe 对弈的初始状态

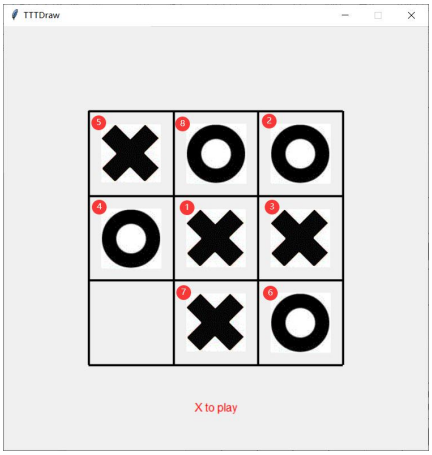

图 4.2 TicTacToe 对弈的目标状态

对于 3*3 的方格棋盘，一共有 8 种等价情况，首先通过其中一种进行旋转，镜像操作可以得到另其七种等价。针对此例题，对于一个个体的评价，要综合考虑其先手下棋与后手下棋的表现。通过统计其输棋数量与下棋总数，可以计算出输棋率，进而得到其胜率。将胜率作为其评价值。GenRandomIndividual 为随机从所有下发中选取一个，及随机选取一个孩子节点。Select 则是依据累积适应度加以限制，随机从种群中选取一个。

五、 算法应用

遗传算法已被成功地应用于工业、经济管理、交通运输、工业设计等不同领域，解决了许多问题。例如，可靠性优化、流水车间调度、作业车间调度、机器调度、设备布局设计、图像处理以及数据挖掘等。下面是遗传算法的一些主要应用领域。

5.1 组合优化

随着问题规模的增大,组合优化问题的搜索空间也急剧扩大，有时在目前的计算机上用枚举法很难或甚至不可能求出其精确最优解。对这类复杂问题，人们已意识到应把主要精力放在寻求其满意解上，而遗传算法是寻求这种满意解的最佳工具之一。实践证明，遗传算法已经在求解旅行商问题、背包问题、装箱问题、布局优化、图形划分问题等各种具有 NP 难度的问题得到成功的应用。

5.2 生产调度问题

生产调度问题在很多情况下建立起来的数学模型难以精确求解，即使经过一些简化之后可以进行求解，也会因简化得太多而使得求解结果与实际相

差甚远。目前在现实生产中主要是靠一些经验来进行调度。现在遗传算法已成为解决复杂调度问题的有效工具，在单件生产车间调度、流水线生产间调度、生产规划、任务分配等方面遗传算法都得到了有效的应用。

5.3 自动控制

在自动控制领域中有很多与优化相关的问题需要求解，遗传算法已在其中得到了初步的应用，并显示出良好的效果。例如用遗传算法进行航空控制系统的优化、使用遗传算法设计空间交会控制器、基于遗传算法的模糊控制器的优化设计、基于遗传算法的参数辨识、基于遗传算法的模糊控制规则的学习、利用遗传算法进行人工神经网络的结构优化设计和权值学习等，都显示出了遗传算法在这些领域中应用的可能性。

5.4 机器人学

机器人是一类复杂的难以精确建模的人工系统，而遗传算法的起源就来自于人工自适应系统的研究，所以，机器人学理所当然地成为遗传算法的一个重要应用领域。例如，遗传算法已经在移动机器人路径规划、关节机器人运动轨迹规划、机器人逆运动学求解、细胞机器人的结构优化和行为协调等方面得到研究和应用。

5.5 图像处理

图像处理是计算机视觉中的一个重要研究领域。在图像处理过程中，如扫描、特征提取、图像分割等不可避免地会存在一些误差，从而影响图像的效果。如何使这些误差最小是使计算机视觉达到实用化的重要要求。遗传算法在这些图像处理中的优化计算方面找到了用武之地，目前已在模式识别(包括汉字识别)、图像恢复、图像边缘特征提取等方面得到了应用。

六、 结论与展望

本文从遗传算法的理论和技术两方面概述目前的研究现状，描述遗传算法的主要特点、基本原理以及各种改进算法，介绍遗传算法的应用领域，并对遗传算法的性能进行了分析。

遗传算法的研究归纳起来分为理论与技术研究、应用研究两个方面。理论与技术研究主要从遗传操作、群体大小、参数控制、适应度评价以及并行实现技术等方面来提高遗传算法的性能。应用研究则是遗传算法的主要方

向，开发遗传算法的商业软件、开拓更广泛的遗传算法应用领域是今后应用研究的主要任务。

除遗传算法外，演化规划和演化策略也是演化计算的两大分支。演化算法特别适合于求解多目标优化问题，同时，演化算法求解多目标优化问题的设计比单目标问题的设计要困难得多。现有的算法大都采用两个群体、精英保留策略、聚类分析等方法，算法的整体设计显得较复杂。本文介绍了(μ+1)演化策略求解多目标优化问题，提出了使用拥挤密度来保持群体中个体的均匀分布，将个体的Pareto强度值和拥挤密度合并到个体的适应值定义中。

综上所述，遗传算法是一个十分活跃的研究领域，遗传算法的研究正在从理论的深度、技术的多样化以及应用的广度不断地探索，朝着计算机拥有甚至超过人类智能的方向努力。

七、 参考文献

[1] 吉根林. 遗传算法研究综述[J]. 计算机应用与软件, 2004, 21(2):69-73.

[2] 王仲民;戴怡;;A New Chaotic Genetic Hybrid Algorithm and Its Applications in Mechanical Optimization Design[J];Journal of China Ordnance;2010 年 03 期

[3] 阚玉峰;徐新华;李云;;遗传算法在 UTP 优化中的应用研究[J];重庆科技学院学报(自然科学版);2008 年 01 期

[4] 高世刚;基于云遗传算法的软件项目资源调度研究[D];武汉科技大学;2010 年

[5] 陈国良, 王熙法, 庄镇泉, et al. 遗传算法及其应用[M]. 人民邮电出版社, 1999.

[6] 王小平, 曹立明. 遗传算法——理论、应用与软件实现[M]. 西安交通大学出版社, 2002.

[7] 陈根社, 陈新海. 遗传算法的研究与进展[J]. 信息与控制, 1994.

[8] 甄文祥,王文田.遗传算法及其应用[J].计算机应用研究，1994.11(5):9-10.

[9] 董红斌, 黄厚宽, 何军,等. 一种求解约束优化问题的演化规划算法[J]. 计算机研究与发展, 2006(05):841-850.

[10] 郭崇慧, 唐焕文. 演化策略的全局收敛性[J]. 计算数学,

2001(01):105-110.

[11] 杜小勤。《人工智能》课程系列，Part I: Python 程序设计基础，2018/06/13。

[12] 杜小勤。《人工智能》课程系列，Part II: Python 算法基础，2018/07/31。

[13] 杜小勤。《人工智能》课程系列，Chapter 5：博弈树搜索技术，2018/10/23。

[14] Gregor Hochmuth. On the Genetic Evolution of a Perfect Tic-Tac-Toe Strategy. Stanford University.

[15] 徐宗本, 陈志平, 章祥荪. 遗传算法基础理论研究的新近发展[J]. 数学进展, 2000(02):97-114.