基于 GA－ANN－GA 系统的启发式遗传算法用于模拟星际航行控制的研究

作　　者： 华南理工大学 张颖鹏
E-mail：tczyp@yahoo.com.cn

作　　者： 华南理工大学 张智卓
E-mail：zzz2010@gmail.com

摘要：本文从星际航行的控制问题出发，通过对遗传算法（GA）与传统算法的分析、比较，提出运用遗传算法解决这一复杂的动作控制问题，并实践证明其可行性。在实现遗传算法的过程中，创造性地提出了新的编码方案，新的变异算子，摒弃了累赘的交叉算子。并在生物科学中“获得性遗传”学说的启发下提出并且实现一种新的启发式遗传算法模式，进一步提炼成具有普遍意义的“GA－ANN－GA”混合智能系统模型，并通过比较实验验证了该模型的先进性和可靠性。同时在软件开发过程中融入了 MVC（Model－View－Controller）设计模式和测试驱动开发（TDD）等先进的软件开发理念，从而保证了代码的健壮性和可移植性，力求使得具有普遍意义的研究成果便于移植与扩展。在文章的最后，总结了我们研究过程中获得的经验和存在的不足并提出相应的改进方案。

关键词：遗传算法 人工神经网络 启发式 GA－ANN－GA 混合智能系统模型 MVC

The Research of Heuristic Genetic Algorithms based on GA－ANN－GA system in the control of interstellar aeronautics

Author： ZHANG Ying-Peng of South China University Of Technology
E-mail：tczyp@yahoo.com.cn

Abstract: The article was in the context of the control of interstellar aeronautics. By analysis and comparison of the Genetic Algorithm (GA) and the traditional algorithms in the area, a new algorithm based on GA is put forward to settle the complex control problem. And the experimental results show that the scheme is feasible in practice. In our algorithm, we improve the performance of GA by the following scheme: we bring forward a new coding scheme based on which new mutation operators are designed and the cross operator is discarded. A new heuristic Genetic pattern derived from "Required Genetic" is put forward and realized. Further, a kind of universal model of mixed intelligent system: "GA－ANN－GA" is brought forward and is proved to be advanced and reliable.What's more, we adopted a lot of advanced ideal of software development, such as Model－View－Controller (MVC), Test Driven Design(TDD) and so on .So the code will be robust and transplantable. We expect this universal laboratory achievement is easy to transplant

and expand. Finally, we sum up the experiences of the work, discuss its deficiencies and propose some possible corresponding solutions

Key Words: Genetic Algorithm Artificial Neural Network Heuristic GA－ANN－GA Model of Mixed Intelligent System MVC

目 录

1.引言

人类渴望进入宇宙空间，怀着两个目的：第一个目的是开发利用空间资源。有人认为 21 世纪国家对航天能力的依赖程度可与 19，20 世纪国家对电力和石油的依赖程度相比拟。可见开发利用空间资源的重要性。第二个目的是满足人类的求知欲望。宇宙空间是陆海空之外的第四环境，广袤无垠，蕴藏无穷的奥秘。人类对宇宙的探索已经持续几千年，揭开并解读了一些奥秘，还有更多的奥秘值得人类去探索，去认识。比如是否存在外星生物，是否存在全新的未知粒子与及其他未知物质，等等。

幻想是无止境的，求知创新的梦想更是无边无际！当人类的认识已经不再满足于我们自身所生活的地球时，“登天入地”的追求便是很自然的事情。求知、创新是人类的一种天性，在不断认识自然改造自然的过程中，逐渐扩展自己的生存空间，从陆地到浩瀚的海洋，到稠密的大气层，再到广袤的宇宙空间。人类探索宇宙的脚步永远不会停止。在技术的推动下，一定会从现阶段的无人和载人航天，发展到无人和载人航宇，实现自由进出宇宙空间的远大理想。

对神州五号飞船成功发射报予的鲜花与掌声还没有停息，神州六号完成任务的捷报紧接而至。在台前幕后无数航天人的努力下，中国航天技术已经取得了举世瞩目的成就，并跻身航天大国之列。中国的成就表明人类探索宇宙空间的队伍又有了一支新的生力军。中国的航天力量，作为国际航天力量的一部分，将为人类探索太空的发展和人类的文明发挥我们自己的作用。

作为身处这样一个航天时代的中国人，也希望能利用自己所学到的知识为推动航天技术的发展尽一分绵力。

由此我们提出星际航行中的控制问题，并试图借助计算机配合高效的算法给出有效的解决方案。

事实上星际航行是一个非常大的主题，影响星际航行的因素也绝不仅仅是控制问题。一般来说，航行于太空，除了受各种引力影响外，还受到高真空、温度变化剧烈、高辐射、空间碎片等各种因素的影响。而在本文仅描述并解决在一个模拟真实的物理环境中的控制问题。

2.创新点

由于本次研究涉及较前缘的技术与设想，没有现成的研究成果以供参考，所以从算法到编码均在一些基本原则的指导下原创出来的。比较显著的创新点有下面几点：

1) 提出一种新型的混合编码方式：**命令－时间**编码方式，并验证其可行性。
2) 提出一种新型的变异算子：**插入式变异**，并通过实验验证其先进性。
3) 从生物和仿真的角度论证交叉算子对于解决这类问题是**累赘**的。
4) 受到生物学中“**获得性遗传**”学说的启发，设计并实现一种基因编码能在“一生中”发生基因突变的新型启发式遗传算法。并由此提炼出具有普遍意义的**混合智能系统模型—**

—GA－ANN－GA 模型。

3.星际航行中的控制问题

3.1 问题描述

古往今来，人类都向往着头顶上这一片星空。经过一代又一代的努力，在上个世纪初，人类终于能够如愿地飞上了天空。而经过短短的几十年，人类又坐着火箭飞上了太空。现在看来，对于古人来说的星际旅行的“梦”现在离我们不那么遥远了。

由于星际航行是高速的并受多个引力影响的运动，不可能像驾驶汽车那样由人来控制。不过幸好，即使航行于星际，也严格遵循物理定律的，所以只要我们尽可能精确地把这个物理环境描述出来，并配合计算机和高效的算法，那么我们就可以实现人工智能控制星际航行。下面我们将对复杂的物理环境进行简化，并建立对应的数学模型，从而把实际问题转化成可由计算机解决的数学问题。

3.2 建立数学模型

为了使用计算机来解决具体的实际问题，必须先把实际问题抽象成一个可真实反映实际问题的数学模型。因为航天器航行于太空中是实时进行的。那么我们为了把无限的搜索空间转换成一个精度可以接受的有限的搜索空间，即把无限维的连续问题转换为有限维的参数优化问题，必须把时间和距离离散化。那么对航天器的控制也变成一个实时的指令序列，即在预定的时间段内对航天器发出预定的指令（比如向哪个方向加速），这是符合实际操作过程的。

因为我们研究的目的着重于遗传算法的设计与优化，为了减轻工作量同时又能从一定程度上反映遗传算法解决实际问题的能力。物理引擎经过了适当的简化处理。**物理引擎**是一个仅考虑航天器自身不同方向喷射所引起的反作用力和一个向下的地心引力的影响，忽略周围的各个行星的地心引力影响，忽略各行星的运行速度，忽略物体高速运动而导致的复杂的状态变化，的**二维经典力学模型。**

对物理引擎的简化并不会影响对算法的有效性和可行性的评估。

1) 因为本质上遗传算法是在**高维空间中作一维搜索**。所以随着问题复杂度增加，(典型如：环境变量的增加，使到搜索维数增加)遗传算法的搜索复杂度只会呈线性增加而不像传统的解释法那样呈指数级增加。(组合优化中启发式算法的研究分析。)
2) 又由于遗传算法是一种只对结果敏感而无视过程的算法，所以当换上新的物理环境的数学模型的时候，只需要重新设计适应度函数，算法本身基本不需要修改，就能处理新的物理环境。

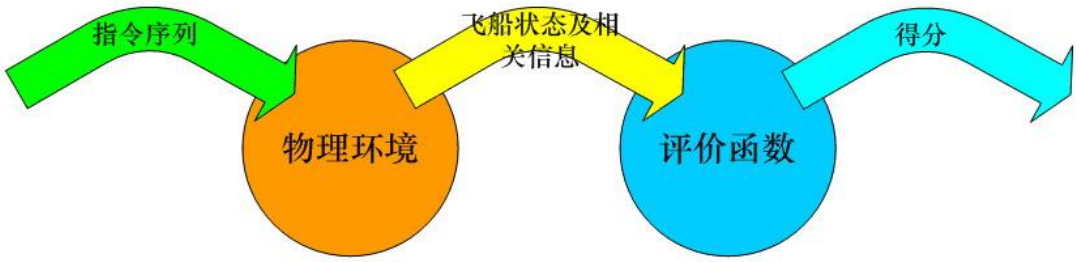
所以作为对遗传算法本身的研究，为了减低研究成本，而又能反映算法解决问题的能力，在研究初期使用简化的物理模型是恰到好处的。

定义：指令序列：$c=(c_1,c_2,c_3,c_4,c_5,\cdots\cdots)$ 其中 $c_1,c_2,c_3,c_4,c_5,\cdots\cdots$ 分别是离散时刻 $t_1,t_2,t_3,t_4,t_5,\cdots\cdots$ 对应的指令。

我们把物理模型定义成一个函数：$f(c)$。在这个函数里面实现，指令序列映射到物理引擎里面进行运算。具体过程是这样的：在每一个离散时间单位里面物理引擎从指令序列里

按顺序抽取出一个有效指令c_k（c_k取 5 种值：0，1，2，3，4。分别代表不向任何方向加速，向左加速，向右加速，向上加速，向下加速）然后物理引擎会综合飞行器当前速度矢量，所受合力矢量，与及当前位置，按照经典力学法则计算出飞行器经过一个单位时间（离散时间的单位）段飞行器的新状态，包括飞行器新速度矢量，和新位置，并进行碰撞检测。不断重复上述取指令和计算状态的过程，直到飞行器碰撞，或指令序列结束，或飞船完成既定的任务的时候物理引擎将返回经过 c 序列指令后飞船的状态与及一些相关的信息 s（比如飞船的速度，位置等信息。）

然后我们把 s 作为参数传给一个评估结果优劣的函数（在遗传算法中就是适应性函数）：$g(s)$返回这一次模拟航行的客观评分。这个过程如下图所示：

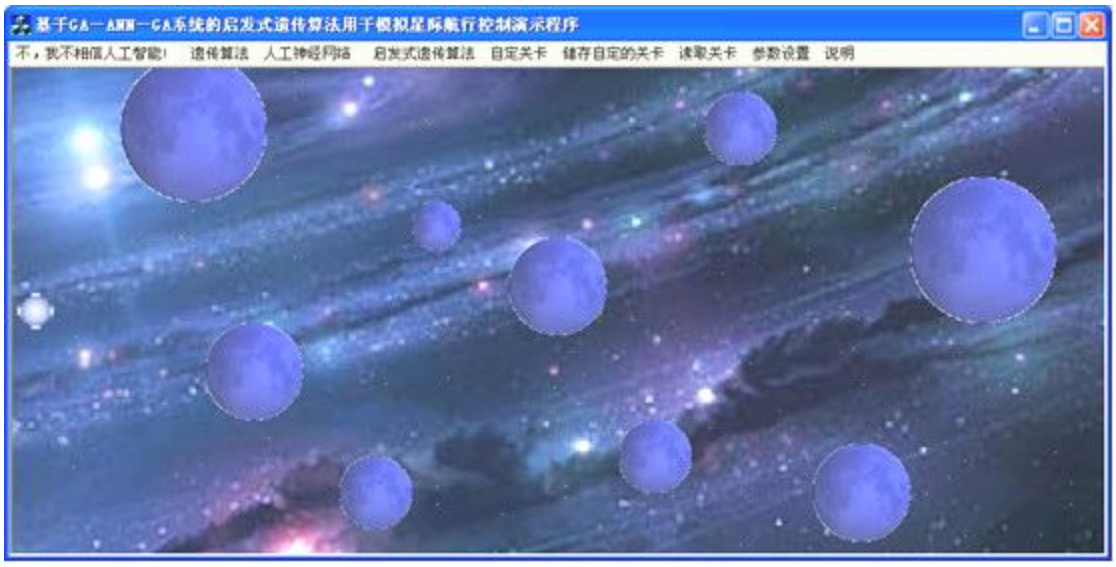


我们可以把这个过程用一个复合函数来表示：评价分值$=F(c)=g(f(c))$。这就是说每一个实时命令队列 c 对应着一个分值$F(c)$，那么这个分值就是指令序列 c 对应的解。而我们希望分值越高越好，就是解越大越好，那么实际问题就转变成在一个在高维的搜索空间寻找函数$F(c)$的全局最优解或高效的次优解的问题。至此物理引擎的数学模型基本建立完成。

物理引擎的相关参数可以通过参数设置对话框的对应属性页进行设置，如下图所示：（不经特别说明，后续实验中使用的物理环境变量值如下图所设。）

为了方便后续的相关实验，我们分别预设了具有不同特征的地图作为比较实验的平台。它们分别如下：（不经特别指明，论文里面提到的默认地图分别指这三幅地图。）

地图一：



星球数目：9

地图数据：

	横坐标	纵坐标	半径
星球 1	200	250	40
星球 2	150	50	60
星球 3	350	130	20
星球 4	300	350	30
星球 5	450	180	40
星球 6	530	320	30
星球 7	600	50	30
星球 8	700	350	40
星球 9	800	150	60

特点：星球数目适中，星球分布较疏散，有较多可行路径。

地图二：

星球数目：7

地图数据：

	横坐标	纵坐标	半径
星球 1	225	121	53
星球 2	228	282	48
星球 3	469	16	56
星球 4	471	188	42
星球 5	473	369	63
星球 6	676	94	58
星球 7	679	286	45

特点：星球数目较少，星球分布较集中，可行路径较少。

地图三：

星球数目： 14

地图数据：

	横坐标	纵坐标	半径
星球 1	154	97	23
星球 2	154	243	25
星球 3	326	21	23
星球 4	461	94	25
星球 5	458	259	26
星球 6	597	173	27
星球 7	595	342	24
星球 8	716	100	15
星球 9	705	270	20
星球 10	821	22	23
星球 11	809	184	23
星球 12	814	344	24
星球 13	323	341	24
星球 14	608	13	22

特点： 星球数目较多，星球分布匀称而密集，可行路径较多。

4.利用一般性遗传算法解决太空旅行问题

4.1 遗传算法与传统的最优化算法的比较

$F(c)$ 是一个复杂的非线性函数，不能直接用解析的方法求解函数的最大值，即问题的全局最优解。（主要是因为算法的复杂度太大）而经过大量的实践已经证明使用遗传算法来

寻找复杂解空间中的最优解或高效的次优解相对于传统的最优化方法（如最速下降法，牛顿法和共轭梯度法）有以下的优势：简便，灵活，全局搜索能力强。

简便：遗传算法只使用报酬信息（适应性函数）而不使用导数或其它辅助信息。而传统的最优化理论必须涉及到导数等信息，在这个问题里面这类信息是不可求的，至少是难求的。使用遗传算法将大大简化了问题的求解过程。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去“找”最优解。而只要简单的“否定”一些表现不好的解就行了。

灵活：这主要是针对软件工程思想来说的，遗传算法引擎具有良好的**可移植性**和**可扩充性**。这是因为遗传算法可以忽略问题的求解过程，而只关心对结果的评价。所以遗传算法引擎可以把函数 $F(c)$（代解决问题）本身当成一个“**黑盒**”。因为不同的问题就是不同的“黑盒”当我们把遗传算法引擎用于解决新问题的时候我们只需要重新设计遗传算法引擎跟“黑盒”的相关接口（必须重写的只是编码方式和适应性函数）就可以把引擎用于解决全新的问题。当然为了使更高效地解决具体的问题，我们可以在测试了软件的安全性后再给遗传算法引擎重写适合于解决该具体问题的交叉函数，变异函数，选择函数等。这是符合软件开发的**迭代过程**的。使用遗传算法的可重用性和可扩充性和安全性相对于传统的依赖于问题本身的最优化算法有绝对的优势。（具体细节请参阅 MVC 的模式）

全局搜索能力强：因为遗传算法在搜索过程中维护的是一个解的种群而不是一个解。传统的最优化算法，单纯是一个迭代的过程，每迭代一次都忽略过去曾经出现在迭代过程中的点（解）。这说明传统的最优化方法“目光短浅”。只着眼于某一个点。而遗传算法却始终维护着一个解的群或者说是着眼于一个面，这使得遗传算法有较强的全局搜索能力。

4.2 对遗传算法过程的设计与及对存在问题的解决方案

经典的遗传算法的实现过程实际上就像自然界的进化过程那样。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。）然后用随机数初始化一个种群，种群里面的个体就是这些数字化的编码。接下来，通过适当的解码过程之后，在物理引擎之中“考验”这条基因，然后用适应性函数对每一个基因个体的“表现”作一次适应度评估。用选择函数按照某种原则择优选择。让个体基因交叉变异，然后产生子代，至此遗传算法的一个世代结束。

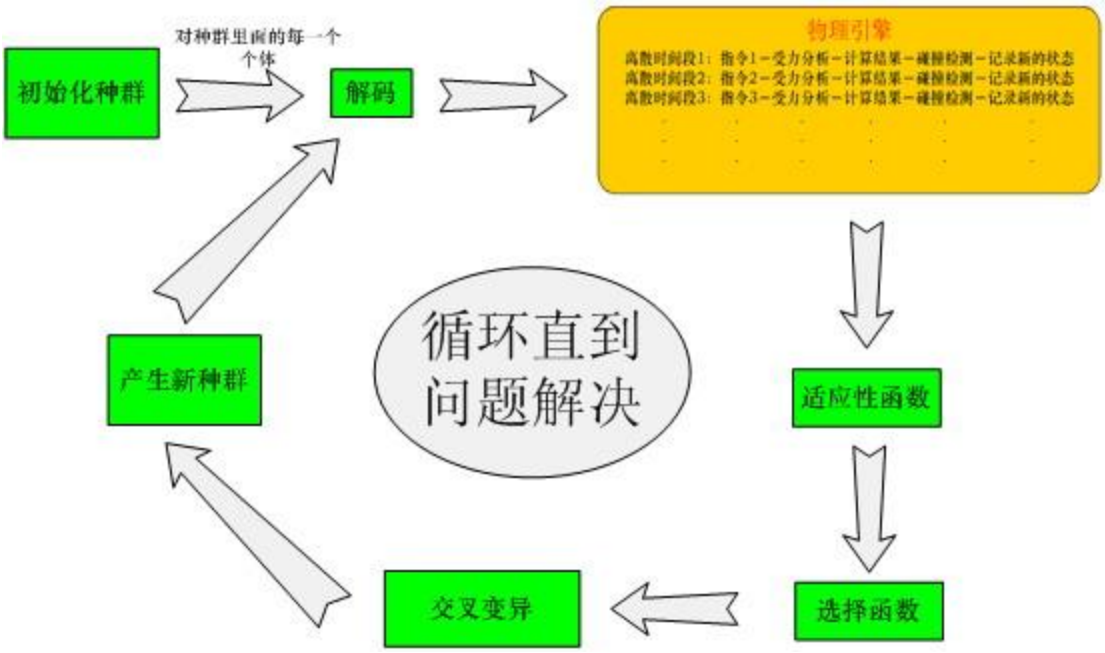
这个过程总结如下：

用随机数初始化一个种群，并开始循环。

1) 对基因进行解码，并在物理引擎里面测试每一个基因对应的个体。
2) 评估每个基因所对应个体的适应度。
3) 遵照适应度越高，选择概率越大的原则，每次从种群中选择两个个体作为父方和母方。
4) 抽取父母双方的基因，进行交叉，产生子代。
5) 对子代的基因进行变异。
6) 重复 3，4，5 步骤，直到新种群的产生。

如果没有找到满意的解继续循环，否则结束循环。

经典的遗传算法过程如下图所示：



遗传算法流程图，其中绿色区域代表遗传算法的流程，橙色代表物理引擎的流程

4.2.1 基因的编码方式——两种编码方式的比较

遗传算法的编码过程是整个遗传算法的基础步骤。怎么样的编码就相对应有怎么样的解码，交叉和变异过程。受到指令序列的启发，我们最容易想到的编码方式就是：直接用一个指令序列作为编码。形式如下：$c=(c_1,c_2,c_3,c_4,c_5,\cdots\cdots)$ 其中 $c_1,c_2,c_3,c_4,c_5,\cdots\cdots$ 分别是离散时刻 $t_1,t_2,t_3,t_4,t_5,\cdots\cdots$ 对应的指令。但是经过实验，发现这样的编码方式有以下的缺点：1）基因长度过长，而且随航行用时的增加而线性递增。2）由于基因长度过长，搜索空间庞大。

经过仔细的思索，原创出一种混合编码方式：**命令－时间**编码方式。其中每个基因块包含两个部分，一个是代表对飞行器发出的指令，用整型变量储存。另外一部分是该指令持续的时间（单位为离散时间单位），同样可以用整型变量存储。用这种编码方式的基因形式如下：$c=\{(c_1,T_1),(c_2,T_2),(c_3,T_3),(c_4,T_4)\cdots\cdots\}$，其中 $c_1,c_2,c_3,c_4,c_5,\cdots$ 是指令，$T_1,T_2,T_3,T_4,T_5\cdots\cdots$ 代表对应的指令持续的时间（单位为离散时间单位）。

这种编码方式跟物理引擎所需要的指令模式有点不同，所以需要一个解码过程。该解码过程如下：

$$c=\{(c_1,T_1),(c_2,T_2),(c_3,T_3),(c_4,T_4)\cdots\cdots\}$$

$$=(\underbrace{c_1,c_1,c_1\cdots}_{T_1\text{个}c_1},\underbrace{c_2,c_2,c_2\cdots}_{T_2\text{个}c_2},\underbrace{c_3,c_3,c_3\cdots}_{T_3\text{个}c_3},\underbrace{c_4,c_4,c_4\cdots}_{T_4\text{个}c_4},\cdots\cdots)$$

对比原则：

为遗传算法设计编码方式常常需要考虑到以下几个原则：

1) **完全性**，原则上问题的所有可能解都能找到与之对应的编码组合。
2) **合法性**，每个基因编码都对应一个可接受的个体。
3) **多重性**，多个基因型解码成一个表现型，即从基因型到相应的表现型空间是多对一的关系，这是基因的多重性。若相同的基因型被解码成不同的表现型，这是表现型多重性。当然，基因型到表现型的映射关系最好是一对一的关系。
4) **紧致性**,若两种基因编码能解码成相同的个体，那么占用空间越小的编码方式就越紧致。
5) **复杂性**，指基因型结构的复杂性，解码的复杂性，计算时空的复杂性。

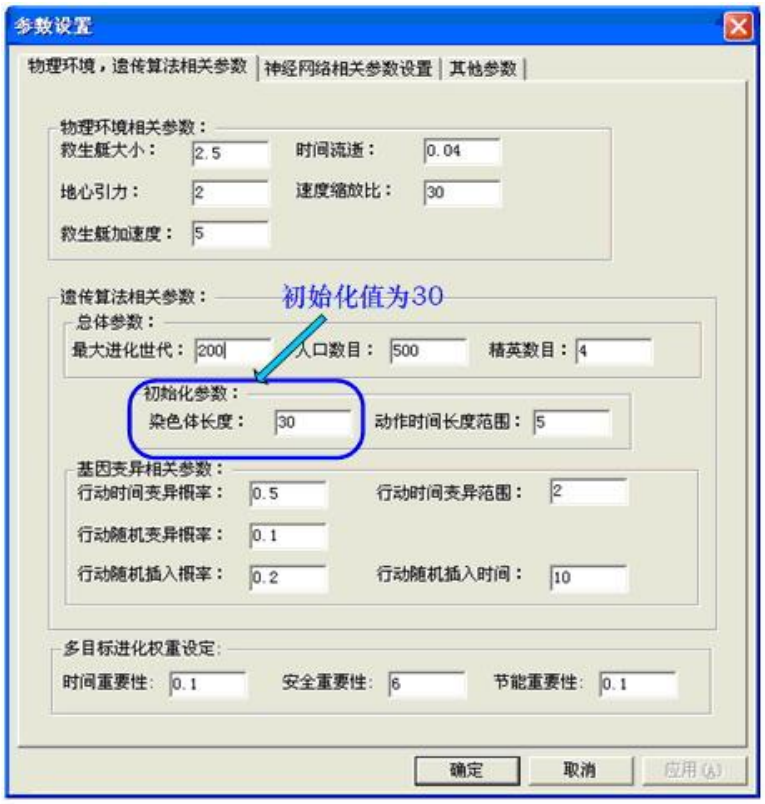
对比结果：

由于两种编码可以相互等价变换，所以它们都同时符合完全性，合法性，多重性和紧致性。下面我们从各个方面比较它们的复杂性。

	命令序列编码方式	命令－时间编码方式
基因结果复杂度	简单	复杂
解码复杂度	简单	复杂
计算时间复杂度	较复杂（这主要是因为遗传算子：交叉和变异的时候比较费时）	较简单
计算空间复杂度	较复杂（因为每一个遗传算子都需要处理一条很长的基因，所以空间消耗较大）	较简单

因为计算时空复杂度对算法的优劣性影响较大，而且经过比较实验证明命令－时间这种混和编码方式是优于命令序列式的编码方式的。所以在解决这个问题的时候我们采用命令－时间混合编码方式。

基因的长度可以设置，如下图所示：



4.2.2 基因的变异——几种变异方式的设计与比较

当基因的编码方式确定以后，就可以设计与之相对应的基因变异方式。在还没提到启发式遗传算法之前，基因变异和接下来提到的基因重组是使到子代不同于父代的唯一原因。

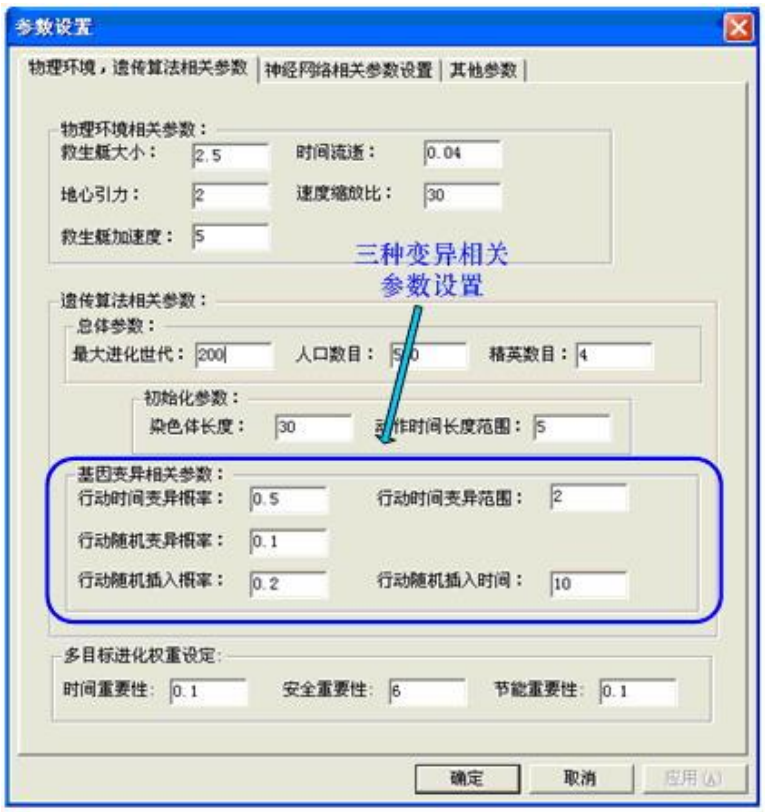
从基因的编码方式出发比较容易得到的变异方式是：指令随机变异和持续时间随机变异。它们都很好理解，就是基因编码 $c=\{(c_1,T_1),(c_2,T_2),(c_3,T_3),(c_4,T_4)\cdots\cdots\}$ 中的 c 会随机变成 0 到 4 之中的一个值。T 会随机增大或减小一个小随机数。从而得到一个新的基因组。

但是在实际操作过程中，我们发现这两种变异未能高效的完成任务，有时候进化进程停留在某个局面无法继续。典型表现是航天器遇到某个障碍，经过很多世代也不能进化出一个能绕过去的命令序列。我们经过大量的观察与实验，慢慢思索总结出：命令－时间编码方式有时候会出现这样一种编码形式，时间的跨度比较长，刚好在这个较长的时间段里面航天器遇到了障碍，那么航天器的控制**细度**受到编码方式的限制，导致很难进化出一种能绕过障碍物的命令序列。为了增加控制细度，我们创造出一种全新的变异方式：**插入式变异**。（在这个过程中我们也慢慢体会到算法宏观表现的倾向，是根源于这个算法的微观实现的。有时候一点小小的变动会从宏观上很直观的体现出来。）

这种变异方式是这样的：随机给基因某个地方插入一个基因块，包括随机的指令和随机的小持续时间段。这种变异不会对原来基因的**表现型**产生明显的改变，只是起到微调的作用。但是当这个新基因插入到适当的地方，会提高航天器在该处的控制细度，从而有可能在后续的变异过程中出现一种能绕过障碍物的命令序列。

经过实验验证（详情请参看实验一）

在软件中，使用者可以调整这三种变异的相关数据，如下图所示：



4.2.3 对基因交叉的研究——摒弃

基因交叉也常常作为遗传算法过程中一个重要环节。但是在这里我们分别从生物角度，和仿真实验角度对交叉算子进行研究，最后决定摒弃这一步骤。

生物角度：

由于基因交叉和两性有莫大的关联，所以我们可以从这个角度去深入了解基因交叉出现的必要性。性别的出现是在生物已经进化到相对复杂的时候。那个时候生物的基因基本形成了一种功能分块的架构。而自然界的基因交叉过程又一般不是单个基因，或者随便几个基因的交叉，而是一块基因，往往是表现某种个体性状的那块基因，所以从宏观上看，基因交叉的表现是性状的分离（孟德尔在实验中总结出来的规律）。而性状又往往表现相对独立的个体特征。比如豌豆的高茎和矮茎，圆滑和皱缩。这些都是相对独立的特征，它们之间可以自由组合互相搭配。这时候，交叉过程将起到从宏观上调整基因块之间搭配的作用。经过物竞天择的过程，最后就能得到相对较好的特征组合方式，从而产生更优的个体。这才是基因交叉的意义所在。所以对于很多问题，使用基因交叉操作的效果不太明显，往往只能充当跳出局部最优解，相当于大变异的功能。真正意义上的基因交叉应该使用在大规模参数的优化过程当中，它将承担起对基因块进行组合优化的职责，从更宏观的角度去优化个体。

仿真角度：

为了从比较实验过程中证实以上的想法，我们进行了使用基因交叉技术和不使用基因交叉技术的两个遗传算法进行比较实验。（具体实验请参阅实验二）在这里我们采用的基因交叉操作是这样的：随机交换 $c=\{(c_1,T_1),(c_2,T_2),(c_3,T_3),\cdots\}$ 和 $c'=\{(c_1,T_1),(c_2,T_2),(c_3,T_3),(c_4,T_4)\cdots\cdots\}$ 基因编码中对应位置的基因块，从而产生子代。

仿真实验结果表明，不使用基因交叉这个步骤的遗传算法大体上比使用基因交叉这个步骤的遗传算法要更快找到问题的解。这些数据反映和印证了从生物角度的思考和研究结果。在这个解决这个问题的时候使用基因交叉这一遗传算子是累赘的，至少是低效的。

4.2.4 选择算子比较与选取

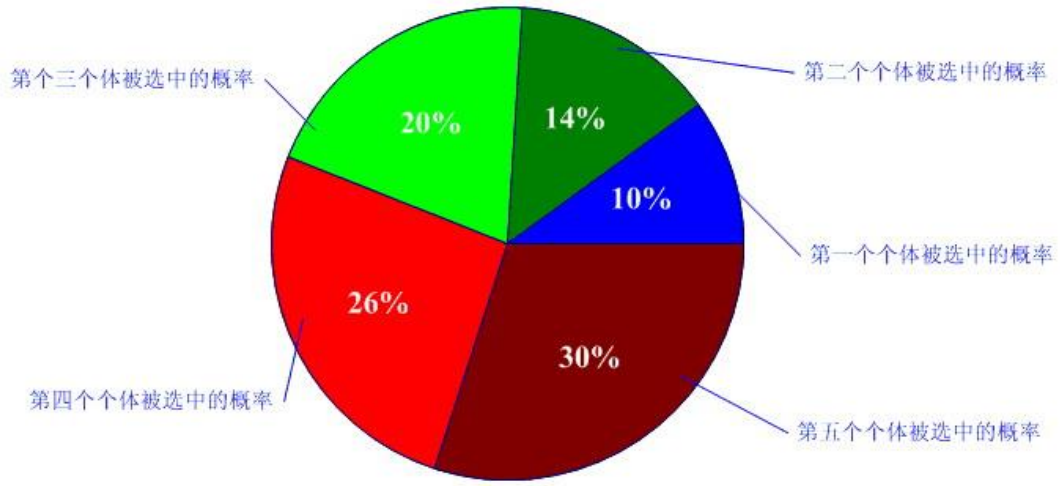
选择算子是决定父代种群中那些个体，并能以多大可能性被挑选来复制或遗传到下一代的进化操作，它模拟的是“优胜劣汰，适者生存”的自然进化原理。选择算子以对个体的适应度评价为基础，其主要的原则是为算法的搜索提供导向：挑选最优秀的个体，使算法避免无效的搜索且能以较快的速度搜索到问题的最优解。

下面我们比较几种选择算法的优劣性：轮盘赌选择法，锦标赛选择法，排序选择法。

1） **轮盘赌选择法**（Roulette Wheel Selection）。这个算法受启发于赌徒所使用的轮盘。假设种群数目 n，某个个体 i 其适应度为 f_i，则其被选中的概率为：

$$P_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

这个过程就像在一个轮盘上，适应度分值越高的个体占据的面积越大，也越容易被选中。比如，加入 5 个基因的适应度分值分别是 5，7，10，13，15。那么它们构成的轮盘如下图所示：



2） **锦标赛选择法**（tournament selection）。这个算法是以体育联赛方式，以个体适应度比较为基础的选择个体策略。为了选择个体 i，先指定一个竞赛规模量 s，然后在种群中随机抽取 s 个个体，然后从中选择最优的个体。

3） **排序选择法**。这种选择算法必须先对种群中的所有个体按照适应度评分作一次排序。然后按照某一个比例从较优个体中选择出繁殖新种群的父方和母方。

而在选择算法过程中，我们常常需要用上一种叫**杰出者选择**（elitest selection）的技巧。即把上一代种群中最优的若干个个体不经过交叉和变异，原封不动地复制到新的一代。这种算法有效的对遗传算法进行了保优，保证迄今为止的最优个体不会被交叉、变异等遗传运算所破坏，它是遗传算法收敛性的一个重要保证条件，同时也体现了“优胜劣汰”的自然生存法则。经过实验，在解决这个问题上面，这个算法是非常有效的。

下面我们考虑用那种选择算子配合杰出者选择来作为我们的遗传算法的选择算子。下面

列表对轮盘赌选择法、锦标赛选择法、排序选择法进行了比较。

各项属性对照表：

	轮盘赌选择法	锦标赛选择法	排序选择法
时间复杂度	复杂	较复杂	简单
空间复杂度	较简单	简单	简单
是否需要排序	不需要	不需要	需要
算法的效果	中等	较差	较好
算法的灵活性	较灵活	不灵活	灵活

结论：

从比较表中可以看出来，选择排序法各方面的指标都具有一定的优势，但是这种算法必须先对个体的适应度评分进行排序，这是非常耗时的，所以如果单单考虑这三种算法本身的时候我们一般都折衷地选择轮盘赌选择法。但是由于杰出者选择技巧对于解决这个问题特别有效，而且杰出者选择需要对种群预先做好排序。所以无论用哪一种算法我们都必须有排序这个耗销。在这种情况下，排序选择法就更显优势。

经过比较实验（具体请参阅实验三），我们可以看出在大多数时候排序选择法是优于其它两种选择算法的，所以我们解决这个问题的时候使用排序选择法作为遗传算法的选择函数。

4.2.5 适应度评价算子的设计

适应度（fitness）是生物学家在研究自然界中生物的遗传和进化现象时用以度量某个生物对其生存环境适应程度的术语。遗传算法借用这个术语来度量个体作为全局最优解的可接受程度。它是选择算子的度量依据。

适应度函数应该遵循的几个原则：

1) 适应性评分应该是非负的.
2) 合理、一致性：要求适应度值反映对应解的优劣程度。
3) 计算量小：适应度函数设计应尽可能简单，这样可以减少计算时间和空间上的复杂性，降低计算成本。

算法的目的是产生一个命令序列使到航天器能够躲避所有障碍，到达地图的另一端。综合考虑到算法的目的和上面几条原则。最后决定使用每次物理引擎传回的状态数据中的航天器碰撞时的横坐标为该个体的适应度。以这个数值作为适应度既保证是非负的，而且一定程度上反映个体（命令序列）的优劣性。同时计算成本很低。进一步的改进设想，请参阅**不足、改进与扩展。**

5.利用启发式遗传算法解决太空旅行问题

5.1 获得性遗传的启示

我们这里提到的启发式遗传算法的思想是来自于对生物学里面的进化论的发展过程深入研究的结果。在研究进化论的发展历史的过程中我们注意到有这样的一种曾经盛行的进化

理论：**拉马克主义的进化论**。

法国学者拉马克（Jean-Baptiste de Lamarck,1744～1891）的进化论观点表述在他的《动物学哲学》(1809)一书中。该书提出生物自身存在一种是结构更加复杂化的“内驱力”，这种内驱力是与生俱来的，在动物中表现为“动物体新器官的产生来自它不断感觉到的新需要。”不过具体的生物能否变化，向什么方向变化，则要受环境的影响。拉马克称其环境机制为“**获得性遗传**”，这一机制分为两个阶段：一是动物器官的用与不用（即“用进废退”：在环境的作用下，某一器官越用越发达，不使用就会退化，甚至消失。）；二是在环境作用下，动物用与不用导致的后天变异通过繁殖传给后代（即“获得性遗传”）。

德国动物学家魏斯曼（August Weismann,1834～1914）对获得性遗传提出坚决的质疑。他用老鼠做了一个著名的“去尾实验”，他切去老鼠的尾巴，并使之适应了短尾的生活。用这样的老鼠进行繁殖，下一代老鼠再切去尾巴，一连切了 22 代老鼠的尾巴，第 23 代老鼠仍然长出正常的尾巴。由此魏斯曼认为后天后天获得性不能遗传。（择自《怀疑----科学探索的起点》）

遗传算法虽然是一种仿生的算法,但我们不需要局限于仿生本身。大自然是非常智慧的，但不代表某些细节上人不能比她更智慧。另外，具体地说，大自然要解决的问题，毕竟不等同于我们要解决的问题，所以解决方法上的偏差是非常正常和在所难免的。譬如**杰出者选择（elitest selection）**也并非存在于自然界的，但是却非常有效的解决很多实际问题。上面这个“获得性遗传”我们先不管它在自然界存不存在，但是对于遗传算法的本身，有非常大的利用价值。即变异不一定发生在产生子代的过程中，而且变异方向不一定是随机性的。变异可以发生在适应性评估的过程当中，而且可以是有方向性的。

我们从上面这样的启发出发，寻找一种改进遗传算法的方法，使得遗传算法更好的解决航天飞行控制的问题。

为简化搜索并减少搜索过程中出现大量可选的搜索方向,从与待解问题有关信息中所得到的启发知识或“经验法则”可用来确定搜索方向。精心选择的启发信息可在极大程度上加大找出解答的机会，且在同时减少求解所需的时间。遵照“获得性遗传”的假设，我们想办法实现：当一条基因在物理环境中演绎的时候，通过一些经验的手段使得基因得到“后天”强加的变异，那么遗传算法就得到一种参考进化方向，往往这种参考方向是准确的，这相对于传统遗传算法的随机搜索，大大节约了搜索随机性所耗销的时间。

5.2 人工神经网络作为启发手段的可行性分析

比较几种可作为启发手段的算法：模拟退火法、列表寻优法、人工神经网络（Artifical neural network)、专家系统、有限状态机和经典的最优化等算法。首先，前面已经提到过，模拟退火和经典的最优化算法由于必须涉及到导数等信息,所以不能作为解决这个问题的算法。而列表寻优、专家系统和有限状态机器都可以作为一种“积累经验”的算法，但是它们有一个共同的缺点就是它们无法处理**非线性**而且**状态无限**的的问题。而航天飞行控制问题就属于这么一类问题。而人工神经网络恰恰具有解决该问题所必须的几个性质。而且更吸引人的是神经网络的**自学习能力**。这暗示着，当我们使用这个引擎去处理新的问题的时候，我们不需要改动该部分的代码，只需要重新训练一次神经网络就可以了。这也是其它算法所不具备的性质，比如用有限状态机的话，需要**人工方式**积累经验，然后以代码的方式写到有限状态机的不同的状态里面。当问题改变了，那么那些代码必须跟着改变。

下面总结出人工神经网络几个性质：

1. 非线性

非线性关系是自然界诸规律的普遍特征，线性关系往往只是为简化问题而作的局部近似，世间许多奇妙的事物都源于非线性，大脑的智慧就是一种非线性现象。基本单元模拟了脑细胞的一种生物学属性，即它可处在抑制或激发两种不同的状态.这种二态行为在数学上表现为一种非线性关系。这种非线性由神经元的输出与输入信号强度比来表现：小输入信号受较大倍数的放大后输出；较大的输入信号受较小倍数的放大后输出；强输入信号不经放大或衰减后输出；当输入超过某个阈值时以饱和值输出.。可以说非线性是神经网络具有高容错性的基础。

2. 适应性（自学习能力）

适应性是神经网络嵌入了一个调整自身突触权值以适应外界变化的能力。特别是，一个在特定运行环境下接受训练的神经网络，对环境条件不大的变化可以容易进行重新训练。而且，当它在一个时变环境（即它的统计特性随时间变化）中运行时，网络突触权值就可以设计成随时间变化。

另外自适应性也导致了神经网络具有强大的学习能力。人工神经网络可以接受用户提交的样本集合，依照系统给定的算法，不断地修正用来确定系统行为的神经元之间连接的强度，有些算法还能够动态改变神经网络的拓扑结构。这种改变是根据其接受的样本集合自然的进行的，中间不需要人工的干预。在传统人工智能的系统研究中，虽然人们对“机器学习”，给予了高度的重视，和付出了极大的努力。但是系统的自学能力差常常成为广泛应用的最大障碍。而人工神经网络却具有“天生的学习能力”这一特征。所以人工神经网络将成为实现机器学习的最主要手段。

3. 容错性

这一特征又是神经网络引人注目的一个重要原因。神经网络的容错性可以说成是神经网络的鲁棒性，也表现为神经网络的普化能力，外推能力（这就像人的举一反三的能力）。这个能力在生物界是普遍存在的，一只蚯蚓被揣成两段，但是依然能各自生存下去；一棵树也不会因为被砍掉几根树枝而失去了生长的能力；人可以根据残缺的图案来推测出原来的形状等等。一般来说，经过亿万年进化而来的生物的“容错性”都是非常强的，这种天赋使得他们拥有坚拔的韧性，和顽强的生命力。而神经网络的容错性表现在：当人工神经网络被训练后，神经网络对输入的微小变化是不会敏感的反映出来的。就是说，相差不远的输入对应的输出也是相差不远的。而这种不敏感的特性就赋予了神经网络“去噪音，容残缺”的能力。

4. 并行性

神经网络的大规模并行性使它具有快速处理任务的潜在能力。人工神经网络在结构上与目前的计算机有本质上的不同。它是由很多小的处理单元互相连接而成的，每个处理单元的功能简单，但大量简单的处理单元集体的，并行的活动就能得到预期的计算结果。

特别是因为遗传算法也是一种并行的算法，当它们两者配合起来使用的时候，这种并行性的潜在优越性得到更好的体现。

正因为人工神经网络的非线性，自学习性，和高容错性使到它非常适合在解决这个问题的时候作为遗传算法的启发手段。我们可以借助人工神经网络从过去的经验中找寻到某种“规律”，并让这个网络为当前情况提出解决方案（搜索方向）。虽然这个方案不一定是绝对准确的，但神经网络只充当启发（提出建议）的职能，它只需保证它所提供的方案在大部分情况下对进化有帮助的就能达到提高搜索效率的目的。

5.3 设计神经网络

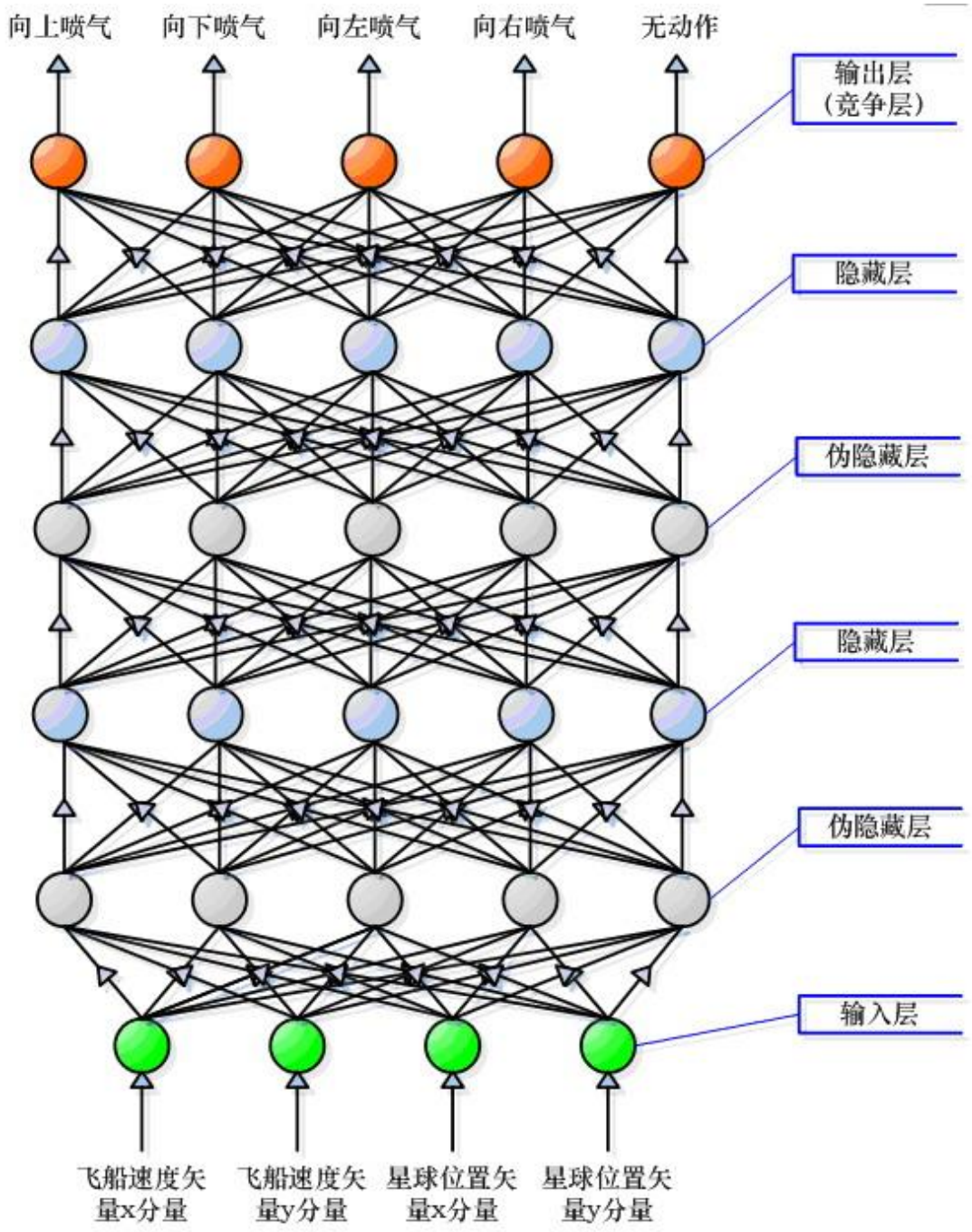
既然决定使用人工神经网络作为遗传算法的启发手段，那么我们必须设计一个有效解决问题的人工神经网络，这包括两个方面：1. 设计人工神经网络的拓扑结构。2. 设计人工神经网络的训练算法。

5.3.1 设计人工神经网络的拓扑结构

我们希望人工神经网络能总结出一种“经验”，比如当航天器正前方很近的地方有一个星球的时候，神经网络应该能凭经验判断出现在应该适当的减速（就是向前喷气）以避免撞到星球上。

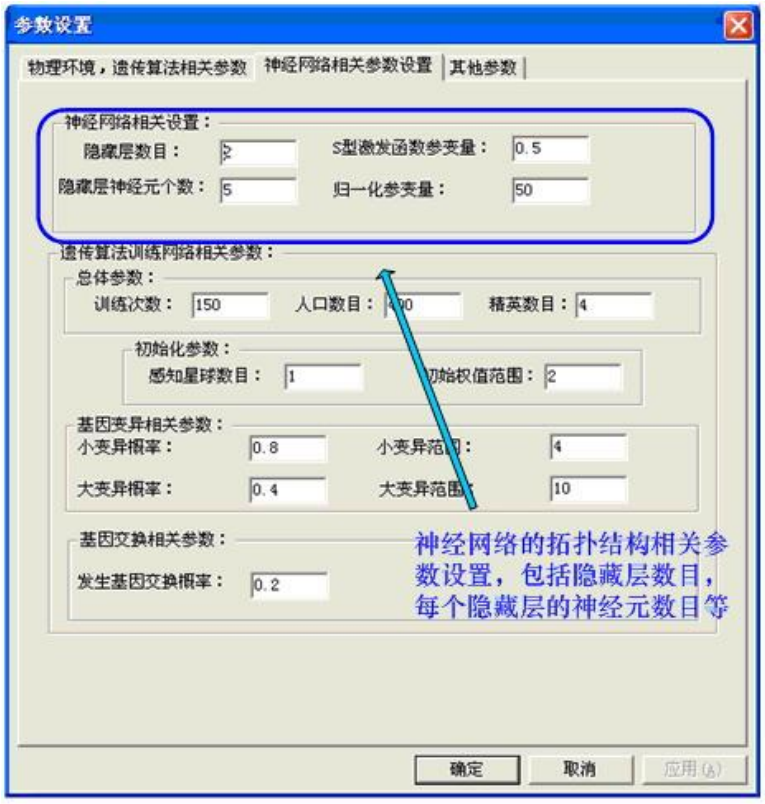
基于上面的思考，我们首先为人工神经网络设计好输入与输出值。因为神经网络必须**感知**到目前飞行器自身和外部环境所处的**状态**才能相应的“**想**”出应该的指令。同时为了减轻神经网络的学习难度，我们的输入信息不是绝对坐标，而是相对坐标——矢量。所以神经网络的输入值设计为：航天飞船本身的速度矢量和离开航天飞船最近的几个星球的位置矢量。（大部分例子我们仅仅让神经网络感知到一个星球，从而减低神经网络的进化难度。而用户可以在参数设置对话框里面设置感知的星球数目。）而神经网络的输出设置为一个竞争层（compete 层，也叫 Kohonen 层）5 个神经元，分别代表向前、后、左、右加速和无动作。而那个神经元的输出大就使用那个神经元所代表的指令方案作为神经网络的输出，竞争层的引入使到神经网络成为一个有机的整体。

神经网络的隐藏层个数和每个隐藏层神经元的个数都是凭借大量实验的经验值，并按照尽量少的原则——《神经网络原理》得到的。隐藏层上面的激活函数采用经典的 S 型函数（Sigmod），在这里我们还用上了“伪隐藏层”技术——《遗传算法进化神经网络的新技术》大大增强了遗传算法跳出局部最优解的能力，并缩短了神经网络的进化时间。最后我们使用的神经网络的结构是这样的：4 - 5 - 5 - 5 - 5 - 5 的全连接分层前馈网络（BP 网络）。而用户每添加一个可感知星球数目，那么神经网络的输入层就会多两个神经元。这个网络结构图如下图所示：



神经网络的拓扑结构示意图

用户也可以通过参数设置对话框动态修改神经网络的拓扑结构：



5.3.2 设计人工神经网络的训练算法

神经网络的拓扑结构设定好了，那么接下来应该确定用什么样的机制和什么样的算法来训练神经网络。

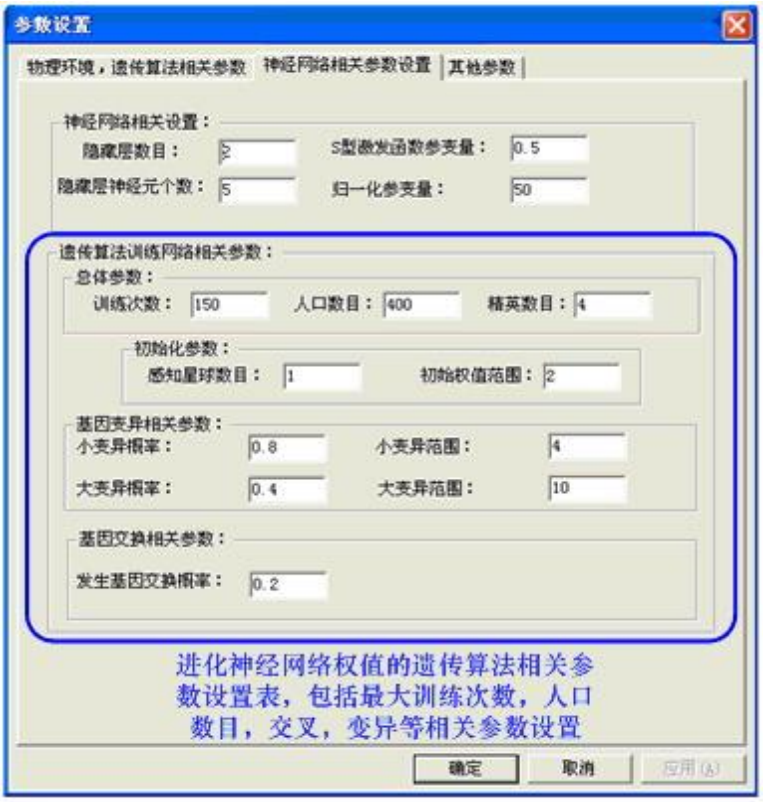
因为我们使用的网络模型是多层全连接前馈网络（BP 网络），而训练这种网络的经典算法是误差反向传播算法（Back propagation）与及它的一些改进（包括借助数值优化技术，如最速下降法，牛顿法，共轭梯度法的改进方案）。近年来，遗传算法也因为它强大的全局搜索能力而越来越受到到重视。下面表格显示了两种算法之间的比较。

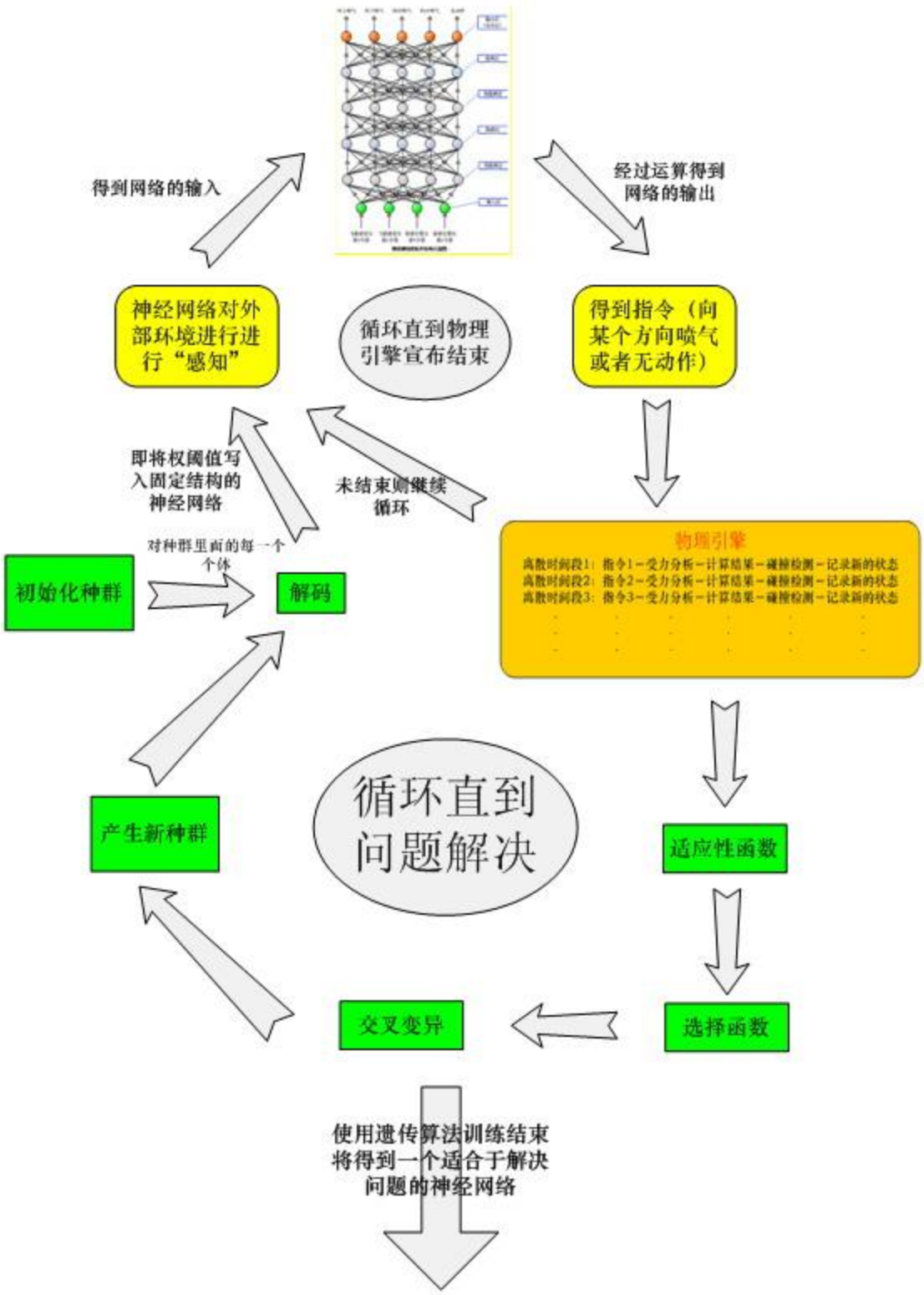
	BP 算法及其改进	遗传算法
是否需要导师	需要	不需要
算法收敛速度	较快	较慢
局部搜索性能	较强	较弱
全局搜索性能	弱	较强
算法模块的可扩充性	弱	较强
算法模块的可移植性	难以移植	方便移植

BP 算法与遗传算法性能对照表

因为 BP 算法是有导师型算法，所以我们必须为算法提供目标样本。而在待解问题中，是没有直接提供目标样本的。需要额外借助其它技术，比如专家系统，或者人工凭借经验为算法拟制目标样本。而遗传算法则不需要这个步骤，不但节省了麻烦和时间，而且大大提高了算法的可移植性，因为解决不同的问题需要不同的目标样本，所以对于有导师型算法而言，每当待解问题发生变化，那么必须又重新为其提供适合于新的待解问题的目标样本，而无导

师型算法则可以直接移植过去。另外，遗传算法的全局搜索能力强，虽然具备搜索性能不好，但是由于遗传算法是一种新兴算法，很多改进方案还在酝酿中，具有很强的可扩充性。比如为了解决遗传算法局部搜索能力较弱的问题，可以采用变异步长自适应的技术，或者混和遗传算法的技术——用模拟退火来调整变异步长，又或者直接用 BP 算法作为局部搜索辅助算法。综合上面的研究结果，我们决定采用遗传算法作为神经网络的训练算法。请注意，这里所用的遗传算法与前面提到的，用于取得航天器命令序列的遗传算法是完全不同的。当然这个遗传算法里面也包括编码，交叉，变异，选择，适应度评估等遗传算子。由于本文的重点是介绍启发式遗传算法，在这里就不细说这个用于进化神经网络权值的遗传算法的步骤了，下面给出遗传算法进化神经网络的流程图。（**注意：** 在训练神经网络的过程中基因编码是一组神经网络权阈值组成的串，另外，我们把几幅默认的地图作为神经网络的训练地图。）其中训练神经网络的遗传算法的相关参数可以通过参数设置对话框的对应属性页进行动态设置。




得到网络的输入
经过运算得到网络的输出
神经网络对外部环境进行进行“感知”
循环直到物理引擎宣布结束
得到指令（向某个方向喷气或者无动作）
即将权阈值写入固定结构的神经网络
未结束则继续循环
对种群里面的每一个个体
初始化种群
解码
物理引擎
离散时间段1：指令1－受力分析－计算结果－碰撞检测－记录新的状态
离散时间段2：指令2－受力分析－计算结果－碰撞检测－记录新的状态
离散时间段3：指令3－受力分析－计算结果－碰撞检测－记录新的状态
循环直到问题解决
产生新种群
适应性函数
交叉变异
选择函数
使用遗传算法训练结束将得到一个适合于解决问题的神经网络

5.3.3 测试 GA－ANN 模块

由于使用遗传算法（GA）来训练人工神经网络（ANN）是一个相对独立的过程，遵照软件工程独立模块独立测试的原则。程序里面也设计了相应的独立测试模块。在这个模块里面单单利用训练好的人工神经网络作为航天器的决策系统。在测试过程中，用户可以看到航天器某一时刻感知到的星球（如图蓝色圈圈球），从航天器的喷气方向可以看出神经网络的决策是怎样的。经过实验表明，经过充分的训练后，神经网络决策系统能够很好的处理用于训练网络的几个地图（缺省的地图）同时因为神经网络的**推广，联想**能力，使用这个神经网络处理全新的地图（软件提供用户绘制地图的功能）也能得到非常满意的结果。（如图）这为下面的用神经网络作为启发式手段建立了良好的基础，因为作为启发手段本身必须在大多数情况下能作出正确的判断。

如图，蓝色圈圈着的星球是神经网络当前“感知”到的星球

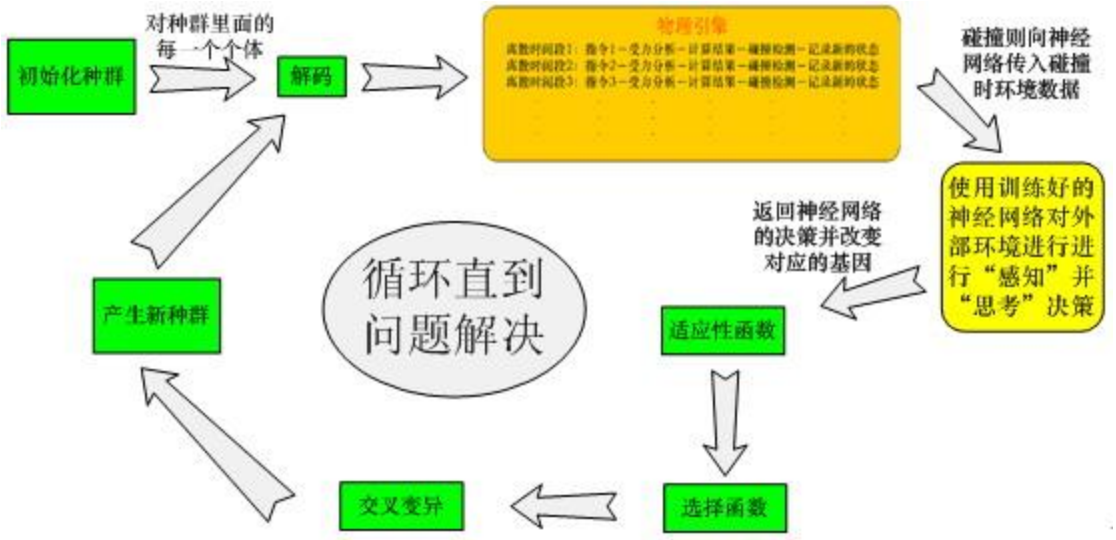
面对全新的地图能够根据以往的经验灵活处理

这个实验（具体的实验过程请查阅实验四）证明训练好的神经网络得到了预期的效果，训练完成的时候这个神经网络几乎拥有了低级的小飞虫的脑袋了。

5.4 使用训练完成的神经网络作为遗传算法的启发手段。

到目前为止，两个主要的引擎：独立解决问题的 GA 引擎和 GA－ANN 引擎都已经完成，并且分别进行了测试。那么接下来需要把它们组装起来并实现启发式遗传算法（HGA）引擎。

正如前面所说的，我们受到拉马克的进化论思想影响，生物的遗传信息会受到“环境”的影响。就是说基因的变化不但来自于变异和交叉这两个遗传过程。还有可能在生物的一生中因为周遭的环境而改变。那么我们参考这一思想，考虑到对航天器的命令序列并不一定仅仅在遗传算法的变异过程才能随机的发生变化。应该允许这个命令序列在物理环境里面执行的时候能够有目的性的改变。由于航天器在没有碰撞之前的指令序列可以被认为是可以接受的，而当航天器发生碰撞的时候的那一个指令，或者连续几个指令则可以认为是导致这次碰撞的原因。那么我们就在这个时候使用训练好的神经网络“感知”周遭环境的情况，并作出决策。这个决策将被提交给遗传算法引擎从而在一定机率上改写原来指令序列中与神经网络的决策所相异的指令。这一过程是在原来的遗传算法过程中进行一些改动而得到的。其流程图如下所示：



通过比较实验（实验五），显示启发式遗传算法比一般性的遗传算法，无论在效率上还是可靠性上都略胜一筹，这种优势在更加完善的物理引擎上面将会变得更加明显。（不足、改进与扩展部分会有所提及）

6. GA－ANN－GA 模型

6.1 总结并提出 GA－ANN－GA 模型

下面我们总结一下启发式遗传算法的流程，并提出一个具普遍意义的**混合智能系统模型**——GA－ANN－GA 模型。

回顾一下启发式遗传算法的整个流程：首先使用一般的遗传算法训练一个设定好的神经

网络模型，直到这个神经网络具有根据环境参数独立作出接近正确判断的能力。然后使用这个训练好的神经网络作为启发手段辅助原来的遗传算法来解决待解问题。(流程如下图所示)

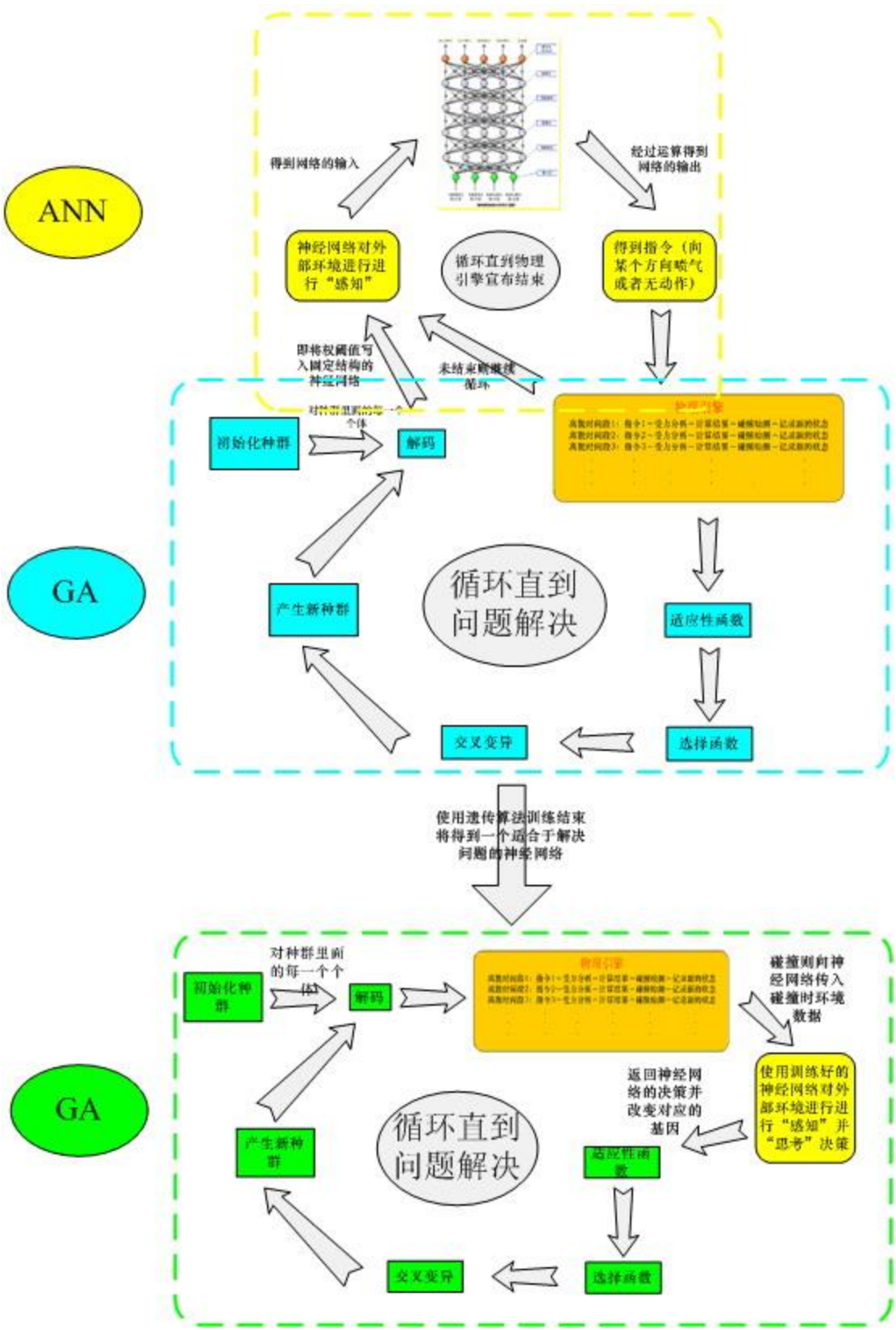
这是一种构造智能系统的模型，不单单适用于解决当前问题。这样的智能系统模型可以从人类的高级思维模式中找到相对应的模式。即积累经验，再从积累的经验中抽象出（或者归纳出）一些规律，然后在指导实践。当然，也许不经过对规律的抽象，仅仅凭借经验，人类也能实践。就像单纯使用遗传算法而不用启发式遗传算法那样，也可以解决问题，但是从效率的角度来说，前者要高很多，而且对于一些特别复杂的问题，后者也许会是不可解的。比如我们要证明一道几何题目，假设做题者只要知道直线交一对平行线时候的角的关系和三角形内角和是 180 度等一些公理级的定理就能作出证明。但是相对来说如果他能从以往的经验中总结出平行四边形的对角是相等的那么他的证明过程就会更容易些。当然如果他还能从经验中总结出一些更抽象的规律，比如见到某种模式的图形就应该使用何种解题思路。那么他的证明会更容易些。在 GA－ANN－GA 模型中，实现这一抽象工作的就是 GA—ANN 引擎。当然人类的智能系统是把 GA－ANN 这个进化过程和原来的 GA 进化过程并行执行的，而在我们的软件实现里面把它们分开了。

6.2 GA－ANN－GA 模型的本质

我们还是举上面那个做几何证明题的例子，因为从本质上来说人类寻找证明过程就像是一个**搜索过程**——不断用已知的定理去堆砌出一个个推论，然后试图让这个推论逼近题目让你去证明的结论。而如果智能体对经验具有更高层次的归纳总结那么智能体就拥有启发式搜索的能力。这种启发式搜索能对原来的庞大的搜索树进行大量的**剪枝**，使得智能体能够更快的解决较复杂的问题。对经验的抽象化程度越高，抽象层次越深，那么启发能力也越高，解决问题的效率也相应越大。

这就像是人类在这几千年来的工作，特别是对数学的研究过程的缩影——不断在原有规律的基础上抽象出更高层次的规律。最初学的代数是描述一些离散的数据之间的简单规律。然后初等代数就是基于这些简单规律上面来研究更复杂的规律。而高等代数研究又是基于这些规律，而同时又把这些规律推广到处理很多数据组成的集合——矩阵。所以很多时候使用一些更低层次的规律来解决问题比直接使用高度抽象的规律来解决问题要麻烦很多。

这种复杂的具有归纳能力的智能系统不仅仅适用于解决这个星际航行的动作控制问题。也不仅仅能适用于解决动作控制问题。事实上这个系统的思想反映了人类的思维模式，非常适合于解决一些**状态不能穷举**，**搜索空间庞大**，**规律难以量化描述**，的**复杂模糊性**问题。



解决本问题的 GA－ANN－GA 模型流程图（橙色代表物理引擎模块，黄色块代表 ANN 模块，蓝色代表进化神经网络的 GA 引擎，绿色代表启发式遗传算法模块。）

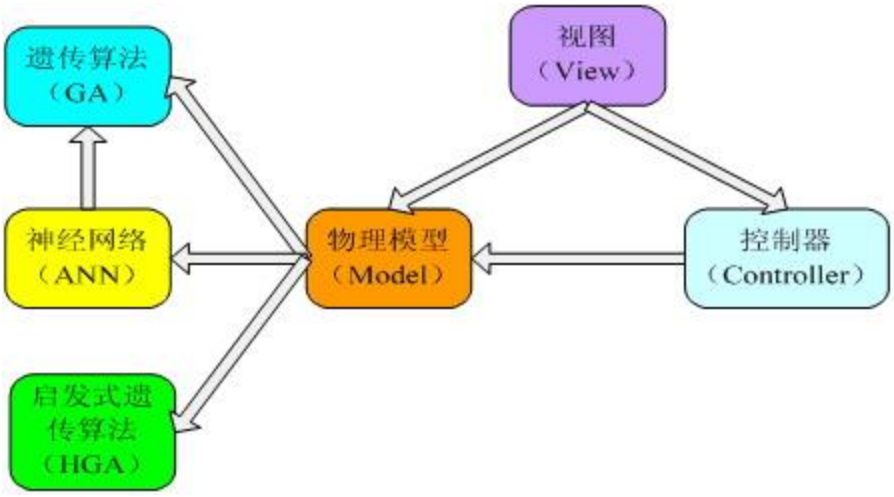
7. MVC 模式——软件工程理念

整个软件的设计过程中我们始终关注并遵循着一些先进的软件设计理念。比如软件开发流程采用测试驱动开发的敏捷开发模式，软件的架构则采用 MVC 模式及其理念。这里重点介绍一下 MVC 模式的理念以及其对于这个软件的意义。

MVC 的全称是“**模型－视图－控制器（Model－View－Controller）**”。MVC 模式主要的设计意图在于：

1） 把负责界面显示或用户输入的对象和负责业务操作的对象分开，使用户界面相关代码不必关心如何完成业务操作等问题，而业务代码也不必关心如何和用户交互。
2） 上述两类对象分离后，它们可以相互独立地发生变化，而不给对方造成影响。例如，界面是软件中较容易发生变化的部分，使用 MVC 模式后，开发者可以相对独立地改变用户界面的代码。
3） 两类对象可以由不同的项目组并行开发，同时可以保证系统代码的可读性、易维护性和可扩展性。
4） 在一个系统中添加新的视图（新的数据显示方式）非常容易，对系统中原有的代码没有任何影响。
5） 可以用多个视图，从不同角度展现同一份数据的内容。
6） 在运行期间，软件可以根据工作流程、用户习惯或系统状态动态选择不同的用户界面。
7） 两类对象分离后，更多的对象可以成为可复用对象。例如，负责业务处理的对象可以很容易地被复用于不同的项目，甚至可以被复用到不同平台的软件系统中。
8） 负责业务处理的对象可以很容易地脱离用户界面系统。我们能单独地对它们进行测试，也可以在另一个项目中以后台方式调用这些对象的功能，或者利用它们进行批量处理。
9） 通过使用分布式的协议，我们可以很容易地把负责业务处理的对象部署到其他服务器上运行，同时把负责用户界面的对象部署在客户端与前者进行交互。

事实上我们在实际设计的过程中不但使用了 MVC 模式本身，还遵照 MVC 模式的思想理念——提炼类（Extract Class），进一步把模型类提出四种模型：物理模型、进化神经网络的 GA 模型、ANN 模型和启发式遗传算法 HGA 模型。所以整个程序的类结构图如下（并未包括一些用于测试的类结构）：



从图中可以看到视图、控制器、物理模型、GA、ANN、HGA 等模块都是相互独立的。这样做为测试带来了方便，并且为以后我们的进一步的扩展研究提供了基础，比如如果我们要把这个二维的航天旅行问题推广到三维（即真实世界）我们所要做的主要是对视图类和物

理引擎的修改，其它的部分的修改量相当小。而且因为原来的代码经过了多次的运行与测试，保证了代码的安全性，所以继续沿用成熟的代码模块会大大减少软件开发过程中的风险，和缩短软件开发的周期。而应用 MVC 模式的最大的优势还在于软件模块的可移植性高。因为 GA－ANN－GA 模型是一个具有普遍意义的混合智能系统模型，可以应用到很多相关领域。那么软件的可移植性就显得特别重要了。正如前面所阐述的 GA、ANN 和 HGA 引擎的特点——它们对具体问题的依赖性不强。比如解决不同的问题的时候 GA 只需要改变编码方式和适应度函数就基本可以用于解决新的问题了。而 ANN 基本不用改动，只要重新训练就可以了。所以这些引擎本身的特性配合 MVC 模式，我们就可以非常方便地把 GA－ANN－GA 模型移植到其它软件作为其处理问题的核心系统。而极大程度上实现了代码重用。

8. 主要的实验及其结论

8.1 引入插入式变异算子前后的比较实验

实验目的：验证引入插入式变异算子的可行性

实验步骤：在其它条件等同的情况下，分别采用和不采用插入式变异，然后比较遗传算法在三个各具特征的默认地图上面进行问题的求解，记录每次问题解决的时候遗传算法进化的世代数。

实验数据：

实验次数	默认地图 1		默认地图 2		默认地图 3	
	使用插入式变异	不使用插入式变异	使用插入式变异	不使用插入式变异	使用插入式变异	不使用插入式变异
1	23	29	56	107	38	38
2	25	35	41	105	31	45
3	19	27	81	49	18	34
4	15	23	35	66	25	42
5	30	38	51	84	41	38
6	22	33	42	56	31	24
7	26	30	60	99	33	42
8	17	27	55	125	28	33
9	24	22	30	68	25	56
10	14	28	35	75	31	50
11	22	19	43	52	26	37
12	12	30	50	58	24	39
13	25	29	38	115	31	38
14	20	33	44	84	35	44
15	23	27	48	64	33	28
平均值	21.13	28.67	47.27	80.47	30	39.2

结论：

经过实验可以看到，引入新的变异方式后，在三幅不同特征的地图上面，算法的收敛速度都得到不同程度上的提高。这证明了这种变异方式的引入明显地提高了算法的效率和搜索能力。而且从得分曲线能看出算法很少陷入局部最优解而无法跳出。

8.2 验证交叉算子效率的比较实验

实验目的：检验使用交叉算子的有效性。

实验步骤：在不改变其它参数的情况下，分别进行和不进行基因交叉算子，然后比较遗传算法在三个各具特征的默认地图上面进行问题的求解，记录每次问题解决的时候遗传算法进化的世代数。

实验数据：

实验次数	默认地图 1		默认地图 2		默认地图 3	
	进行交叉算子	摒弃交叉算子	进行交叉算子	摒弃交叉算子	进行交叉算子	摒弃交叉算子
1	31	23	44	56	33	38
2	28	25	33	41	41	31
3	19	19	37	81	23	18
4	23	15	85	35	28	25
5	33	30	63	51	35	41
6	17	22	38	42	43	31
7	33	26	53	60	27	33
8	22	17	35	55	36	28
9	32	24	52	30	27	25
10	18	14	48	35	30	31
11	27	22	40	43	27	26
12	18	12	42	50	36	24
13	28	25	39	38	31	31
14	25	20	50	44	32	35
15	20	23	51	48	41	33
平均值	24.93	21.13	47.33	47.27	32.67	30

结论：

仿真实验结果表明，不使用基因交叉这个步骤的遗传算法大体上比使用基因交叉这个步骤的遗传算法要更快找到问题的解。这些数据反映和印证了从生物角度的思考和研究结果。在这个解决这个问题的时候使用基因交叉这一遗传算子是累赘的，至少是低效的。

8.3 几种选择算子效率的比较实验

实验目的：比较三种选择算法的优劣。

实验步骤：在不改变其它参数的情况下，先后使用轮盘赌选择法、锦标赛选择法、排序选择法作为遗传算法的选择函数，然后比较遗传算法在三个各具特征的默认地图上面进行问题的求解，记录每次问题解决的时候遗传算法进化的世代数。

实验数据：

实验次数	默认地图 1			默认地图 2			默认地图 3		
	轮盘赌	锦标赛	排序选	轮盘赌	锦标赛	排序选	轮盘赌	锦标赛	排序选

	选择法	选择法	择法	选择法	选择法	择法	选择法	选择法	择法
1	41	29	23	68	121	56	37	43	38
2	34	55	17	95	102	41	55	66	31
3	44	47	18	88	99	81	47	47	18
4	28	52	28	83	152	35	52	58	25
5	29	61	17	122	121	51	61	47	41
6	36	38	31	91	133	42	38	49	31
7	33	44	26	75	107	60	44	52	33
8	35	55	22	110	113	55	55	45	28
9	49	40	35	105	91	30	40	61	25
10	39	53	21	113	124	35	53	72	31
平均值	36.8	47.4	23.8	95	116.3	48.6	48.2	54	30.1

结论:

实验结果表明排序选择法在三种具有不同特征的默认地图上解决问题的效率都高于其它两种算法。

8.4 GA－ANN 模块测试实验

实验目的：验证 GA－ANN 模块的有效性，以保证神经网络能作为遗传算法的启发手段。

实验原理：为了验证神经网络能起到启发的作用，那么我们先通过实验检验直接用神经网络控制飞船的解决问题能力。由于物理环境的变化会影响神经网络的训练，比如如果飞船的体积比较大，那么就更容易碰撞，神经网络的训练难度会增加，如果地心引力比较大，而飞船的加速比较小，同样会使到飞船的控制难度增加，从而神经网络的训练难度增加。所以为了检验神经网络解决问题的能力，我们需要在不同的物理环境中测试神经网络的性能。

实验步骤：每次实验中，分别调整物理引擎的相关环境变量（飞船大小，喷射加速度，地心引力）针对具体情况训练神经网络，记录每次实验中神经网络操控飞船在每幅默认地图和一幅新地图（非训练地图）的完成度（飞船碰撞的时候处于地图横向宽度的百分比），并记录进化世代数，训练用时和总得分。并同时截下训练进步曲线。

实验数据：

实验	实验一				实验二				实验三			
物理引擎环境变量	飞船大小	地心引力	飞船加速度		飞船大小	地心引力	飞船加速度		飞船大小	地心引力	飞船加速度	
	2.5	2	5		3.5	2	6		2.5	3	4.5	
地图完成度（%）	默认地图一	默认地图二	默认地图三	非训练地图	默认地图一	默认地图二	默认地图三	非训练地图	默认地图一	默认地图二	默认地图三	非训练地图
	100	100	100	100	100	100	100	75	100	100	100	85

进化世代数	122	157	182
训练用时（秒）	48	71	88
总得分	15490.0	14826.6	14664.32

结论：

以上是分别在不同的物理环境中训练神经网络，并再使用训练完成的神经网络控制飞船的实验。从实验结果可以看出，经过有限次训练，神经网络都能达到预期的训练目的，并较好地完成任务，即使面对全新的地图也能有 70%以上的完成度。这为后续的启发式遗传算法打下良好的基础。

以下分别是三个实验相对应的神经网络训练进步曲线（上面是最优个体进步曲线，下面是种群平均分数进步曲线）。

实验一的神经网络进步曲线

实验二的神经网络进步曲线

实验三的神经网络进步曲线

8.5 比较一般性遗传算法和启发式遗传算法的效率和可靠性

实验目的：在不同的环境因素和不同的星球分布场景中，比较一般遗传算法和启发式遗传算法的效率和可靠性。

实验原理：因为启发式遗传算法是基于之前训练完成的神经网络的，所以影响启发式遗传算法的效率和可靠性的因素非常多。为了更全面的比较一般遗传算法和启发式遗传算法的效率和可靠性。我们把实验分成三组，每一组实验对应于不同的物理环境因素。而在每一组实验中，我们分别训练一次神经网络，然后让遗传算法分别在默认的三幅地图上面和一幅全新的地图上面寻求问题的解，重复实验 10 次，从而得到求解的进化世代数的平均值和均方差，该平均值反映算法的效率，均方差反映算法的可靠性。

实验步骤：进行每组实验前调整一次物理环境参数，然后进行神经网络的训练，接着分别对默认的三幅地图，和自定义的一幅新地图使用一般遗传算法和启发式遗传算法，并记录下求解所需要的世代数，重复实验 10 次。

实验数据：

实验	第一组实验			第二组实验			第三组实验		
物理引擎环境变量	飞船大小	地心引力	飞船加速度	飞船大小	地心引力	飞船加速度	飞船大小	地心引力	飞船加速度
	2.5	2	5	3.5	2	6	2.5	3	4.5
神经网络	16190.50			13962.4			14562.32		

得分																								
地图	默认地图一		默认地图二		默认地图三		非训练地图		默认地图一		默认地图二		默认地图三		非训练地图		默认地图一		默认地图二		默认地图三		非训练地图	
进化世代数	原	启	原	启	原	启	原	启	原	启	原	启	原	启	原	启	原	启	原	启	原	启	原	启
	12	11	46	22	33	52	63	45	14	16	39	22	33	37	82	45	17	9	39	32	45	29	41	45
	12	15	33	26	35	28	47	33	13	15	51	33	39	51	51	39	12	15	41	26	35	37	47	35
	15	6	35	21	41	22	59	56	15	8	35	21	48	28	55	56	17	11	35	21	35	28	59	56
	26	10	28	35	44	26	48	37	25	12	28	35	47	32	57	61	21	16	37	41	51	31	48	37
	22	22	33	40	61	41	56	37	27	22	39	48	66	41	56	37	18	22	33	40	61	41	56	51
	11	8	29	23	25	44	41	41	11	8	33	26	29	44	41	41	20	8	25	23	33	44	47	36
	16	10	55	44	27	29	62	35	15	14	55	44	27	35	65	35	16	10	61	44	27	29	62	35
	14	16	24	30	44	33	53	44	14	16	28	30	51	33	53	47	16	16	29	30	49	39	53	44
	13	13	34	22	33	48	49	50	15	12	38	27	33	48	58	46	13	13	51	30	38	48	49	50
	14	9	35	25	38	35	51	49	17	16	51	25	44	38	51	49	14	9	35	25	41	35	51	49
平均值	15.5	12	35.2	28.8	38.1	35.8	52.9	42.7	16.6	13.9	39.7	31.1	41.7	38.7	56.9	45.6	16.4	12.9	38.6	31.2	41.5	36.1	51.3	43.8
均方差	20.9	19.6	73.6	60.6	95.9	90.8	45.1	49.8	24.4	15.7	83.4	73.7	128.6	48	103.9	61	17.4	17.3	99.8	57.8	91.9	43.1	35.8	53

（注意：“原”表示一般的遗传算法，“启”表示启发式遗传算法）

结论：

实验结果表明，在不同的物理环境下，不同特征的地图上面，用神经网络作为遗传算法的启发手段都会对原来的遗传算法在效率和可靠性有不同程度的提高。

9. 不足、改进与扩展

在我们研究的过程中就始终注意着这个软件的，具体地说就是这些算法的不足之处与及其改进方案和扩展的空间。我们并不仅仅止步于对我们所面临的问题本身的研究，而是始终关注如何解决问题本身的同时，还关注如何把研究这个航天飞行控制问题过程中所得到的宝贵经验，所创新的有效方法类比地用到其它领域。就是说我们希望我们的研究成果是具有普遍意义的，而不仅仅局限于解决我们所面对的问题本身。比如 GA－ANN－GA 模块同样适

用于充当其它智能系统的核心。当然，我们所面对的问题被定位为一个实验的场所，一个测试的平台，其本身就有很多的外延，所以能够非常有效地解决我们所面对的问题本身，就说明算法能解决与之类似的一大类问题。

总的来说我们的研究还存在以下的不足和改进扩展的方向：

9.1 物理引擎的不足与改进

物理引擎仿真度不足，虽然我们研究的重点是在于对求解算法本身的研究，不宜过多地把精力分散到物理引擎的完善。但是为了更真实直接地测试算法解决实际问题的效率，完善物理引擎又是非常必须的，所以我们下一步打算就是把 2D 化的物理引擎，做成 3D 的真实世界物理引擎，并同时把星球的地心引力，星球的移动考虑进去。当然，根据我们对以往研究的经验，和有效的推理，我们可以预知当加入这些因素之后，遗传算法也将能应付，因为正如前面所提到过的，遗传算法是在高维的搜索空间进行一维的搜索，搜索空间维数增大，遗传算法的搜索复杂度只会呈线性增加。同时我们可以预知到，在 3D 化的场景中，我们提出的启发式遗传算法模型——GA－ANN－GA 模型将更加发挥作用。因为在 2D 的场景中，每一个搜索状态的子节点可以分成 5 条不同的分支，而在 3D 化的场景中搜索状态中的每个节点将分成 7 个不同的分支，那么如果我们使用神经网络作为启发手段将会大大增加剪枝的比例，使到算法的效率得到更大的提高。

9.2 适应度函数的不足与改进

在这个程序中，对于适应度函数我们采取比较简洁的设定方法，其中原因这里就不再赘说了。这样的适应性评分，只能保证进化的趋势是让个体走得越来越远，但是并不保证航天器走的路径是不是比较安全（如果航天器的航行路线跟某些星球擦身而过，那么我们认为那是不太安全的，因为物理引擎毕竟不能考虑现实世界的所有情况，所以为了让解决方案具有更好的**鲁棒性**，我们必须提高安全性。）也不能保证航天器走到终点所用的时间是不是比较短，同时也不能保证航天器走到终点所消耗的能耗是不是比较少（这一点能反映在航天器的喷射次数上面）。而恰恰，安全性，时耗，和能耗是人类在太空航行中所比较关心的问题。事实上，这些要求都可以通过修改适应性函数来得到满足。如果人们对这些指标都有要求的话，比如人们希望在安全性过关的情况下，旅行得越快越好的话，那么我们在设计适应度函数的时候就应该在安全性评价值达到一个阈值的时候加大消耗时间评价值的权量。那样就可以达到多目标同时照顾的效果。这就是多目标遗传算法。也是我们接下来要完成的研究工作。

9.3 神经网络模块的不足与改进

在我们研究神经网络模块的时候，发现当我们训练好神经网络后，尽管神经网络的得分已经相对高，但是神经网络控制飞行器在星球稀少的空间都显得很“保守”地向前飞行，不像人类，如果人类觉得某一个地方星球非常稀少，会毫不犹豫地加快前进速度的。但是为什么我们无论怎么进化也进化不出这种类人的思维模式呢，主要有两点原因，其一是我们没能给神经网络输入足够的信息，从某种程度上来说神经网络是非常“盲目”的。另外一个原因是神经网络本身比较简单——单个神经网络，很难进化出那么高级的“思维”水平。所以我们经过对神经网络最新技术的研究，发现**神经网络集成技术——**利用有限个神经网络共同解决某一问题，能以较小的运算代价显著提高系统的泛化能力。这正是我们所需要的，如果我们能使到神经网络本身解决问题的效果更优，那么使用神经网络作为遗传算法启发式手段，

必定也能得到更准确的启发方向，从而进一步提高启发式遗传算法的效率。

9.4 GA－ANN－GA 模块的不足与改进

正如前面所提到过的那样，GA－ANN－GA 模块非常类似于人类的思维模式，而人类的思维模式中把 GA－ANN 这个进化过程和原来的 GA 进化过程并行执行的，而在我们的软件实现里面把它们分开了。事实上两者并行执行是有好处的，这样会使到同一次经验分享给不同 GA 引擎实现进化，而不是各自执行，各自进化，这样会进一步缩短算法求解的周期。使到我们不需要事先花一段时间来训练神经网络，经过改进后，我们可以一边解决问题一边训练神经网络，在这个改进的结果中将会看到，算法从一开始效率偏低，到后来效率越来越高。因为背后的神经网络在不断的“学习”。

9.5 进一步提高软件可扩充性，可移植性。

虽然在这个软件的编写过程中我们始终以一些先进的软件工程的理念作为指导和原则，比如 MVC 模式，测试驱动开发的敏捷开发理念与及始终注意 GA－ANN－GA 模型的可移植性。但是一方面是时间仓促，早期研究精力集中于对算法效率的提高上面，另一方面在一开始的软件架构设计过程中没有充分考虑到问题的复杂性，从而错过了一些非常有效的软件**设计模式（Design Patterns）**，比如**观察者模式（Observer）**、**单件模式(Singleton)**、**模板方法模式（Template Method）**等。后来编码过程发现，最初如果使用这些模式将会大大降低软件的编写难度，增加代码的健壮性，也同时会提高引擎的可移植性。所以打算在后续的研究过程中重写部分代码，以引入上面提到的设计模式。

9.6 其它改进思路

事实上对于我们的研究还有非常多不足与及可改进方案，以上提到的只是比较明显和比较迫切的几个方面，其实引入**自适应遗传算法**进化神经网络权值，引入遗传算法进化**神经网络拓扑结构**，引入其它类型的神经网络作为遗传算法的启发手段，比如**自适应谐振网络（ART）**，**Hopfield 网络**等。都是非常值得研究的改进方向。

10. 总结与感想

我们在算法的研究和软件的开发过程中遇到并且解决了很多问题，在解决这些问题的过程中我们总结到一些对于研究问题非常有益的方法和思维习惯，现在总结如下：

1.解决问题的思路

由于遗传算法带有极大的随机性和不确定性，即使是当前被大部分人所接受的**模式定理**理论中大部分结论都未得严格的数学证明（计算智能中的仿生学），所以我们在研究遗传算法的过程中常常采用理性分析和大量实践结果作为理论的基础和支持，还不时从生物界寻找一些有益的启示。

我们的研究过程是按照这样一个思路而进行的：

2.始终关注生物科学并从中得到启发

因为遗传算法包括我们在解决这个问题过程中使用的人工神经网络技术是仿生学高度发展的结果（从原来的模仿生物的形态，生物的器官，到现在模仿生物界的方法，智慧）。所以我们在研究过程中始终关注着生物学的一些理论研究成果，并从中得到了许多启发，我们的很多创新点都源自于这样的启发。比如对交叉算子的摒弃与及建立 GA－ANN－GA 模型的设想，都源自于生物学的启发。

3. 算法宏观表现的倾向，根源于算法的微观实现。

在我们研究怎么改进原来的遗传算法的过程中，不断尝试通过参数的修改，算法的替换来让原来的算法更加有效，更加稳定。在这个过程中我们深刻地体会到算法宏观表现的倾向，是根源于这个算法的微观实现的。有时候一点小小的变动会从宏观上很直观的体现出来。我们通过深入的研究更加把握这种研究的技巧，渐渐能预知微观上修改能够对宏观上有什么影响，或者宏观上所反映的问题怎么从微观上去解决。从而使到研究的进度大大加快。

4.融入软件工程的理念

由于我们所编写的软件涉及到混合的复杂智能系统，具有随机性和复杂性等特点。很多 bug 都**掩饰于系统的随机性当中**。所以我们在编写整个软件的过程中不但考虑到算法效能上的优劣性，还始终从软件工程的角度关注代码实现过程中是否符合一些软件工程的原则和理念。我们遵循化整为零，模块独立开发，独立测试，测试驱动开发的敏捷软件开发理念。尽可能降低类与类之间的耦合度，方便代码的扩延和重用，从而降低风险，提高代码的健壮性。

总之，在研究的过程中，我们获得了很多宝贵的经验，并总结出很多行之有效的研究方法。这些经验和研究方法必将有助于我们进一步的研究。

参考文献

[1] 张颖鹏. 遗传算法进化神经网络权值的新技术. 中国人工智进展论文集. 2005

[2] 万琼, 姚望舒, 王金根, 陈世福, 谢俊元. 进化算法在人工神经网络中的应用研究. 中国人工智能学会第 10 届全国学术年会论文集. 2003

[3] 徐宗本，张讲社 郑亚林. 计算智能中的仿生学：理论与算法. 科学出版社 2004

[4] (日)玄光男 程润伟 著,汪定伟等译. 遗传算法与工程设计 北京:科学出版社 2000

[3] [美]Stuart Russell,Peter Norvig 著, 姜哲 金奕江 张敏 等译. 人工智能——一种现代方法(第二版) (Artifical Intelligence: A modern approach) 人民邮电出版社 2004

[6] 蒋宗礼. 人工神经网络导论. 高等教育出版社. 2001

[7] (美)Martin T.Hagan Howard B.Demuth Mark H.Beale 著，戴葵 等译. 神经网络设计（Neural Network Design）. 北京:机械工业出版社,2002.

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

[9] T.Mitchell 著,曾华军,张银奎译.机器学习(MachineLearning).北京:机械工业出版社,2003.

[10] 阎平凡，张长水编著. 网络与模拟进化计算 北京：清华大学出版社 2000

[11] （美）米凯利维茨（Michalewice, Z.）著， 周家驹 何险峰译. 演化程序——遗传算法和数据编码的结合 北京:科学出版社 2000

[12] 张昀著.生物进化.北京大学出版社.1998

[13] （美）Alan Shalloway, James R.Trott 著， 雄节 译. 设计模式精解(Design Patterns Explained:A New Perspective on Object-Oriented Design) 北京：清华大学出版社 2004

[14] （美）Erich Gamma Richard Helm 等著, 李英军 译. 设计模式---可复用面向对象软件的基础 北京:机械工业出版社,2004.

[15] Xiangping Meng, Huaguang Zhang, Wanyu.Tan. A hybrid method of GA and BP for short-term economic dispatch of hydrothermal power systems. Issue 3-4, pp.341~348, January 2000

[15] W.E.Hart,R.K.Belew.Optimization with genetic algorithm shybrids that use local search.In:R.K.Belew,M.Mitchell(eds.).Adaptive Individual sin Evolving Populations.AddisonWesley,1996

[16] E.Lamma,F.Riguzzi,L.M.Pereira.Belief Revision by Lamarckian Evolution.Appli
Cations of Evolutionary Computing:EvoWorkshops2001.

[17] YewSoonOng,AndyJ.Keane.Meta-LamarkianLearninginMemeticAlgorithms.IEEETransactiononEvolutionaryComputation.Vol.8.No.2.2004:99-110.

[18] D.Dasgupta.Artificial neural networks and artificial immune systems: similarities and differences.1997 IEEE International Conferenceon Computational Cyberneticsand Simulation.Institute of Electrical and Electronics Engineers,Incorporated.1997:873~878.

[19] Shigeru Obayashi, Takanori Tsukahara, and Takashi Nakamura.Multiobjective Genetic Algorithm Applied to Aerodynamic Design of Cascade Airfoils.IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 47, NO. 1, FEBRUARY 2000 211

[20] Ricardo M. Ramos, Rodney R. Saldanha, Ricardo H. C. Takahashi,and Fernando

J. S. Moreira.The Real-Biased Multiobjective Genetic Algorithm and Its Application to the Design of Wire Antennas.IEEE TRANSACTIONS ON MAGNETICS, VOL. 39, NO. 3, MAY 2003 1329

[21] Byoung-Ki Jeon, Jeong-Hun Jang, and Ki-Sang Hong.Road Detection in Spaceborne SAR Images Using a Genetic Algorithm.IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 40, NO. 1, JANUARY 2002

[22] Hussain Aziz SALEHa & Rachid CHELOUAHb. The Design of the Global Navigation Satellite System Surveying Networks using Genetic Algorithms. A Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificielle, IRIDIA, CP 194/6

[23] Zeng, Guoqiang’ Xi, Xiaoning Ren, Xuan’. Use of Genetic Algorithm in Optimal Orbit Maneuver. Proceedings of the 3* World Congress on Intelligent Control and Automation June 28-July 2,2000, Hefei,P.R China