第4章 机器进化

4.1 进化计算

上一章的学习系统，通过改变其行为来使其符合一组训练实例。这种学习模仿了生物系统学习的某些方面。生物系统采用另一种方法进行学习，即进化——子孙比祖先更先进。能运用与进化类似的过程来产生有用的程序么？本章将介绍一种达到此目的的技术。

生物进化过程中，先是双亲生产后代，然后这些后代中的一部分“有选择地生存”下来，并生产更多的后代。繁殖和选择性生存这两个方面足以生产繁殖能力越来越强的一代代个体。下面这个几何类比简单地描述了这一过程：想像这样一幅数学“风景画”：那里有山峰、山谷和平地，还居住着一群个体，它们随机地分布在这幅画的不同地方。个体在风景画中所处的高度是衡量此个体相对其他个体完成任务的能力。那些处于海拔低的个体停止“生存”，其概率随着其海拔越来越低而增大。那些处于海拔高的个体进行“繁殖”，其概率随着其海拔越来越高而增大。繁殖就是生产新的个体，使这些个体在风景画中的位置与其单亲或双亲的位置相关但不同。最有趣且有效的一种繁殖就是父亲和母亲联合生产新个体。这些子女在风景画中所处的位置是其双亲的位置的函数。在一些进化过程中，一个子女的位置（从某中意义上讲）处于其双亲的位置“之间”。

可把此过程视为搜索风景画的高峰过程。新生成的个体占据新地形，而那些处于海拔低的个体逐渐消失。不久，一些个体将位于峰顶。这个过程的有效度依个体不同于其单 /双亲的方式和风景画的性质而定。子女所处的海拔高度也许会比它们的单 /双亲低，但偶而也会更高。

把进化搜寻过程运用于计算机科学的主要目的有两个。函数优化是最直接的应用，在这种应用中，我们试图求出一个函数的最大值，如$f(x_1,\cdots,x_n)$，其中自变量$(x_1,\cdots,x_n)$说明个体的位置，f值为高度。John Holland[Holland 1975]提出了一种解决这类问题的算法——“遗传算法（GA）”。许多教材和论文均对此算法做了详细的介绍（请参阅 [Goldberg 1989，Mitchell，M.1996，Michalewicz 1992]）。

另一种应用就是用进化程序解决具体问题——如控制响应agent的程序。本书将对此进行介绍。“分类（*classifier*）系统”是GA的分支之一[Holland 1986]，且[Holland 1975, pp.171以后，第2版。已经成功地用于进化程序这一目的。而“遗传编程 (*genetic programming*)（GP）”[Koza 1992, Koza 1994]是另外一种技术，它采用一种比GA更直接的方式来进化程序。在下一节我将举例说明GP过程。

4.2 遗传编程

4.2.1 遗传编程的程序表示

在遗传编程中，我们改进功能程序 (*functional program*)，如LISP函数。这样的程序可以表

示为具有节点标记、有限的树。内部节点为带有一个或多个参数的函数、谓词或动作。叶节点为程序常数、或不带自变量的动作函数。在图 4-1中，说明了如何用树结构来表示计算 3+(5×4)/7的程序。本例中，叶节点为3、4、5和7，内部节点为函数 +、×和 /。

这里，我将介绍如何运用遗传编程来进化一个沿墙运动的机器人，这个机器人的任务与第2章所述相同。设此机器人所处的世界为二维网格世界，如图 4-2所示。我们进化一个程序，此程序把机器人当前的传感器数据作为输入，并计算出一个动作。我们希望重复运行此程序来控制机器人，并把机器人从任一初始位置移到与墙毗邻的一个单元中，使其永远沿墙运动。

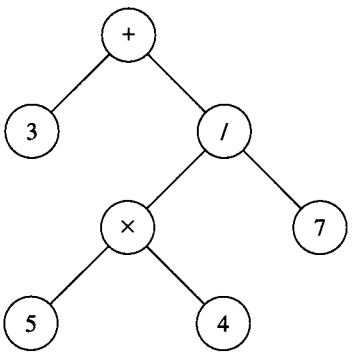


图4-1 用树结构表示的一个程序

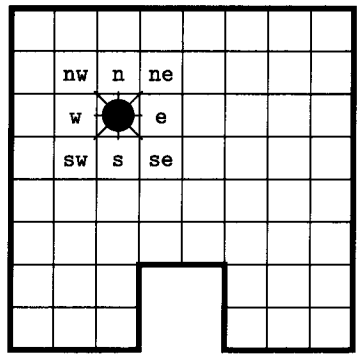


图4-2 网格世界里的一个机器人

此程序的基本函数包括：四个布尔函数——AND、OR、NOT和IF；四个动作——north、east、south和west。布尔函数具有它们通常的定义：

1) 当x=0时，AND(x,y)=0；否则为y。

2) 当x=1时，OR(x,y)=1；否则为y。

3) 当x=1时，NOT(x)=0；否则为1。

4) 当x=1时，IF(x,y,z)=y；否则为z。

四个动作定义如前：

north：将机器人在网格中向上移动一个单元。

east：将机器人在网格中向右移动一个单元。

south：将机器人在网格中向下移动一个单元。

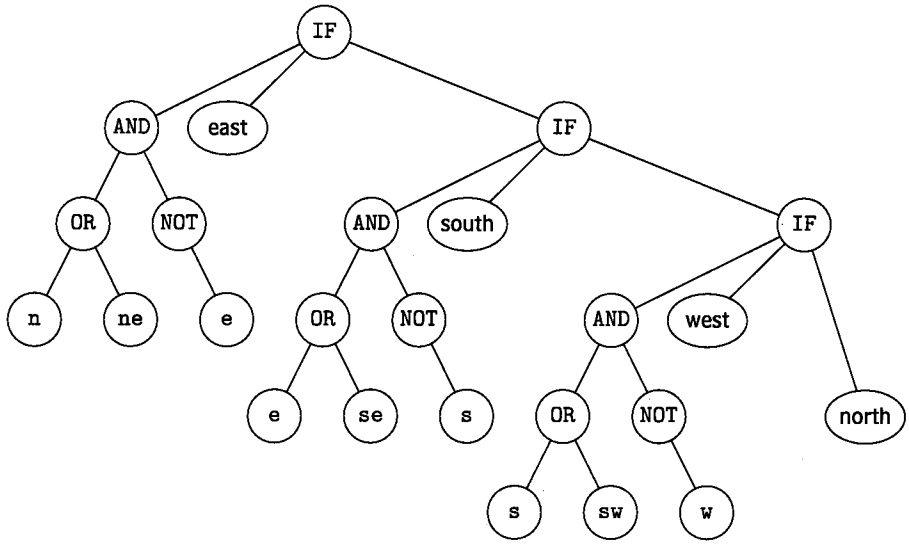
west：将机器人在网格中向左移动一个单元。

这些动作函数本身都有效果但无数值；因为对它们任意一个赋值便会终止程序，所以无须沿树结构向上传值。所有这些动作函数均有其指示的效果，当机器人企图移入墙里时，该动作无效，只能终止程序。当然，重复运行无效的程序最终也无效。

运用与以前一样的传感器输入，但在这里，为了便于记忆，我不用 $s_1,\cdots,s_8$ 这种表示方式，而运用n、ne、e、se、s、sw、w和nw来表示。当相应的单元空缺使得机器人可移入时，输入值为0；否则为1。图4-2展示了将被感知的、与机器人相关的单元的位置。

在遗传编程中，必须保证程序中所有的表达式和子表达式（除了用来终止程序的表达式）对所有可能的参数来说均有值。例如，用函数 *Divides*(*x*,*y*)来表示*x*/*y*的值。当*y*=0时，也应给*x*某一个值（也许是0）。这样，可保证树结构中的每个函数均有适当数量的参数，从而使此树结构描述一个可执行的程序。不久我们便会明白这一点的重要性。

在用遗传编程来进化一个沿墙运动的程序之前，图 4-3给出了一个沿墙运动程序，同时也列出了此程序的树状表示和表结构表示。重复运行此程序将使机器人向北走到墙边，然后顺时针沿墙运动。这个程序可与第2章中为沿边界运动开发的产生式系统相比较。




```
(IF (AND (OR (n) (ne)) (NOT (e)))
    (east)
    (IF (AND (OR (e) (se)) (NOT (s)))
        (south)
        (IF (AND (OR (s) (sw)) (NOT (w)))
            (west)
            (north))))
```

图4-3 一个沿墙运动的程序

4.2.2 遗传编程过程

在遗传编程中，我们从随机程序开始，并运用那些能使程序对感兴趣的问题域有效的函数、常数和传感器输入。这些初始程序构成0代(*generation* 0)。0代程序的大小是遗传编程中的一个参数。在对遗传编程的阐述中，我们将从 5000个随机程序开始来进化一个沿墙运动的机器人。从基本函数AND、OF、NOT和IF，动作north、east、south和west，传感器函数n、ne、e、se、s、sw、w和nw以及常数0和1中产生这些初始随机程序。每一代程序均接受评估，而且直到有一个程序的表现十分令人满意时才产生新的一代程序。

依据一个程序如何完成所设定的任务来对它进行评估。这里，我们把一个程序运行 60次，并计算在这60次运行中被访问的与墙毗邻的单元个数（共有 32个单元与墙毗邻，因此，从未走近墙边的程序的计数为0，而理想的程序的计数为32)。然后，我们让机器人分别在10个随机选择的不同初始位置进行前面这个步骤。在这 10个运行中被访问的与墙毗邻的单元的总数即为此程序的“ 合理度（*fitness*）”。可能的合理度的最高值为 320 —— 机器人只有在这10个步骤的每一步的60次运行中均访问了所有与墙毗邻的单元，才可能达到这一最高值。

第*i*代按照下列方式建构第*i*+1代：

1) 把第*i*代的500个程序（10%）直接复制到第*i*+1代。用下列“锦标赛选拔（*Tournament Selection*）”过程来选择这500个程序：从5000个程序中随机选出7个，再从中选择最合适的那一个（“锦标赛选拔”是选择合适个体的一种有效的方法。数字7和直接复制的程序的百分比是遗传编程过程中的附加参数）。

2) 把4500个新的“孩子(*child*)”程序（90%）归入第*i*+1代。每一个孩子程序通过以下“交叉”操作由一个母亲(*mother*)程序和一个父亲(*father*)程序共同生产出来：这些双亲程序均从第*i*代的锦标赛选拔中选出，把母亲程序中一个随机选择的子树替换为父亲程序中的一个随机选择的子树，便产生出孩子程序（以下要求保证孩子程序的可执行度：即此程序的所有函数对所有参数的可能值来说均可执行）。在图4-4中，通过图解说明了这一交叉操作。双亲程序中的阴影节点即为随机选出的交叉点。这个孩子程序的合适度可能比其双亲高，也可能比其双亲低。因为一个主程序以及其合适的双亲的子表达式结合到了孩子程序中，所以我们认为此交叉操作可能有效。

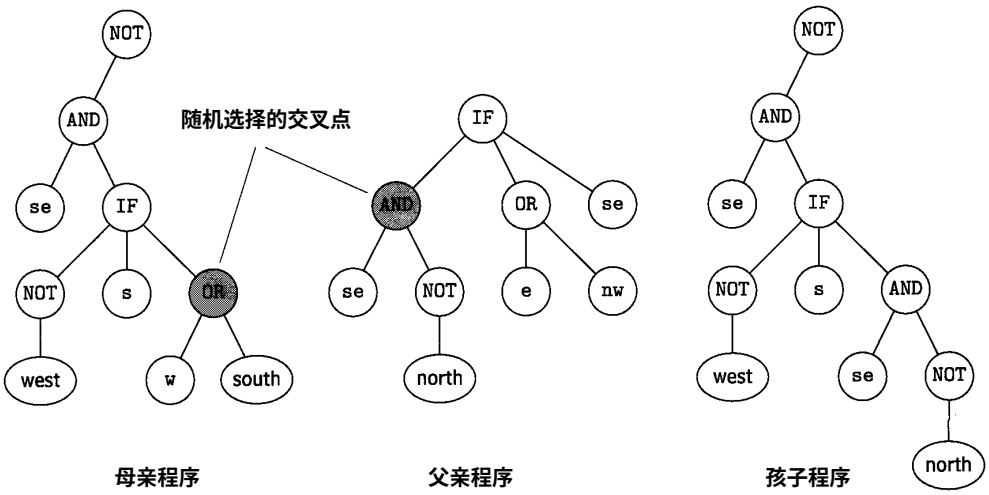


图4-4 两个双亲程序及其孩子程序

3) 有时，还用一个“变异算子(*mutation operator*)”来构造下一代个体，但它构造的下一代数目很少（也许只有1%）。这个变异算子用锦标赛选拔从*i*代中选出一个单亲，删除此单亲程序中随机选择的子树并替换为一个新成长的随机子树（此子树以0代个体生成的方式生成）。本例未用到变异。

我们注意到，在构造下一代的过程中，要设置几个任意参数，包括直接复制的数目、交叉生产数目、参与锦标赛选拔的程序数目以及变异百分比。示例中所用参数是遗传编程专家推荐的参数。

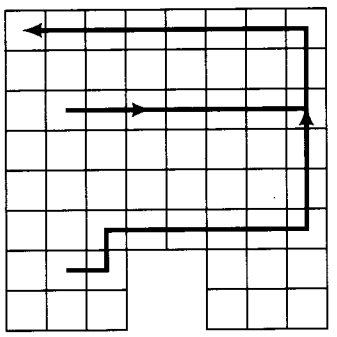
4.2.3 进化一个沿墙运动的机器人

从5000个随机程序开始，并运用刚才描述的技术，来开始进化一个沿墙运动的机器人的遗传编程[⊖]。许多0代随机程序根本什么也不会：如，(AND(sw)(ne))(合适度为0)给它的第一个参数赋值，若结果为0，则终止运行；否则，继续给第二个参数赋值，再终止运行。一些程

⊖ 感谢David Andrew为此实例编程。

序不管输入如何，只往一个方向运动。如，当程序（ OR(e)(west)）赋值“ west ”，它朝西运动，然后终止运行。这个程序的合适度为5（一些随机程序会恰巧运动到与墙毗邻的单元中）。0代的这5000个程序可视为利用解决这类特殊进化问题资源来进行盲目空间搜索的计算机程序。

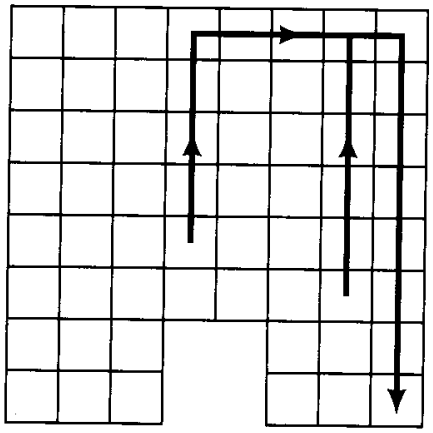
在图4-5中，给出了0代的最合适的程序（合适度为92）的表结构形式以及它的两个合适度的运行。和通常的遗传编程程序一样，这个程序很难理解而且存在许多冗余的操作（其中一些可被后处理器删除）。这个程序从任意单元开始东移直至到达与墙毗邻的单元；然后北移直至能再次东移，或者西移直至被困于左上角的单元中。



```
(AND (NOT (NOT (IF (IF (NOT (nw))
                       (IF (e)(north) (east))
                       (IF (west)(0) (south))
                   (OR (IF (nw)(ne)(w))
                       (NOT (sw)))
                   (NOT (NOT (north))))))
     (IF (OR (NOT (AND (IF (sw)(north)(ne))
                       (AND (south)(1))))
             (OR (OR (NOT (s))
                     (OR (e)(e)))
                 (AND (IF (west)(ne)(se))
                      (IF (1) (e)(e)))))
         (OR (NOT (AND (NOT (ne))(IF (east)(s)(n))))
             (OR (NOT (IF (nw)(east)(s)))
                 (AND (IF (w)(sw)(1))
                      (OR (sw)(nw)))))
         (OR (NOT (IF (OR (n)(w))
                      (OR (0)(se))
                      (OR (1)(east))))
             (OR (AND (OR (1)(ne))
                      (AND (nw)(east)))
                 (IF (NOT (west))
                     (AND (west)(east))
                     (IF (1)(north)(w)))))))
```

图4-5 0代的最合适个体

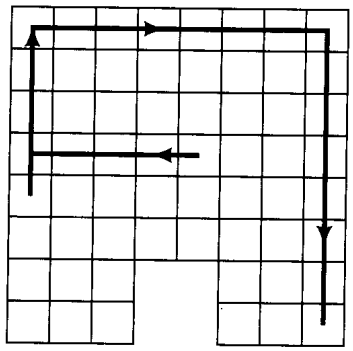
第2代的最佳程序的合适度为117。图4-6展示了这个程序及其两次典型的合适度运行的表现。这个程序比0代最佳程序短，但仍将被困于右下角。



```
(NOT (AND (IF (ne)
              (IF (se)(south)(east))
              (north))
          (NOT (NOT (e)))))
```

图4-6 第2代的最合适个体

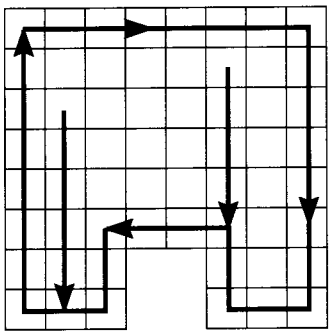
到第6代，程序的最高合适度达到163。其中最佳程序可理想地沿边界运动，但仍会被困于右下角，如图4-7所示。



```
(IF (AND (NOT (e))
         (IF (e)(s)(nw)))
    (OR (IF (1)(e)(south))
        (IF (north)(east)(nw)))
    (IF (OR (AND (0)(north))
            (AND (e)(IF (e)
                        (IF (se)(south)(east))
                        (north))))
        (AND (e)
             (NOT (IF (s)(sw)(e))))
        (OR (OR (AND (nw)(east))
                (west))
            (nw))))
```

图4-7 第6代的最合适个体

最后到第10代时，遗传编程过程已进化出一个能十分完美地沿墙运动的程序。图 4-8展示了此程序以及从不同起点开始的两条路径。此程序顺时针沿墙运动，并且当它一开始不与墙毗邻时就南移并朝墙运动。



```
(IF (IF (IF (se)(0)(ne))
        (OR (se)(east))
        (IF (OR (AND (e)(0))
                (sw))
            (OR (sw)(0))
            (AND (NOT (NOT (AND (s)(se))))
                 (se))))
    (IF (w)
        (OR (north)
            (NOT (NOT (s))))
        (west))
    (NOT (NOT (NOT (AND (IF (NOT (south))
                            (se)
                            (w))
                        (NOT (n)))))))
```

图4-8 第10代的最合适个体

图4-9为每一代的最合适个体的合适度曲线。注意一代代逐渐的进步（通常进步不大）。

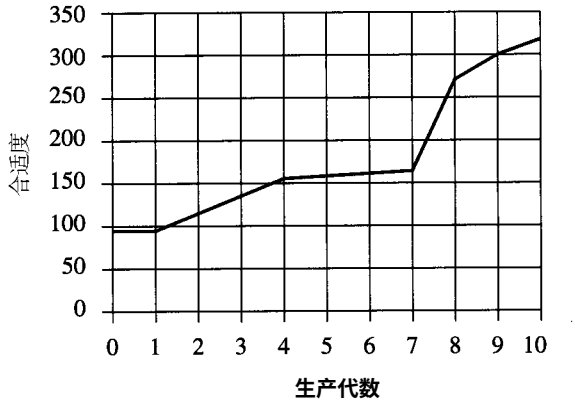


图4-9 合适度作为生产代数（*generation number*）的函数

4.3 补充读物和讨论

遗传算法和遗传编程已被广泛地应用于各种领域。遗传编程已经成功地用来进化响应agent，这些agent与其他人工智能研究者所研究的响应agent相似，其中包括推箱子、跟踪蚂蚁

路线、为载重拖车倒车及平衡反向钟摆等响应 agent。[Koza 1992]描述并演示了这些应用以及其他技术的广泛应用。当我们扩充此方法使其允许进化子路线，继而借此来把主要程序当成基本函数时进行进化，这种方法会变得更加强壮。[Koza 1994]阐述了这个特征及其应用。最复杂且成功的遗传编程应用应是合成电子滤波器、放大器以及其他电路装置 [Koza, et al. 1996]。已经有人做过用遗传编程来进化产生及运用储存器的程序、搜寻程序和递归程序等等这些基础性研究[Andrew 1995, Teller 1994, Brave 1996]。

研究遗传编程和遗传算法的论文出现于遗传编程会议的会议论文集、《IEEE Transactions on Evolutionary Computation》以及国际遗传算法会议的会议论文集中。

习题

4.1 指定用于进化下列agent的合适度函数
   1) 控制一个电梯的agent
   2) 控制一个城市主街道上的停车灯的agent

4.2 确定（生物学）进化论中“基因型（*genotype*）”与“表现型(*phenotype*)”这两个词的意义。如何运用它们来描述遗传编程？

4.3 若不严格要求“每个子树运行后必须返回一个值”，遗传编程交叉过程将会有何改变？

4.4 遗传编程中的交叉操作是从双亲的程序中各自随机选一个子树。谈谈你对随机选择出现以下偏见时所产生的影响的看法：
   1) 倾向于选择在合适度测试中表现相当积极的子树；
   2) 倾向于选择大的子树而不是小的子树，然后反之。

4.5 怎样用进化过程，如遗传编程过程，来进化
   1) 神经网络？
   2) 产生式系统？

   请详细描述这些进化过程，特别是其中的交叉操作。谈谈你是否允许在神经网络的进化过程中运用Lamarckian进化。

4.6 你为什么认为变异有助于或不利于采用交叉的进化过程？