遗传算法

Outline

Topic

一种受生物进化启发的学习方法。

- 它不再是从一般到特殊或从简单到复杂地搜索假设，而是通过变异和重组当前已知的最好假设来生成后续的假设。
- 在每一步，被称为当前群体（population）的一组假设被更新，方法是通过使用目前适应度最高的假设的后代替代群体的某个部分。
- 这个过程形成了对假设的生成并测试（generate-and-test）柱状搜索（beam-search），其中若干个最佳当前假设的变体最有可能在下一步被考虑。

一种受生物进化启发的学习方法。

- 它不再是从一般到特殊或从简单到复杂地搜索假设，而是通过变异和重组当前已知的最好假设来生成后续的假设。
- 在每一步，被称为当前群体（population）的一组假设被更新，方法是通过使用目前适应度最高的假设的后代替代群体的某个部分。
- 这个过程形成了对假设的生成并测试（generate-and-test）柱状搜索（beam-search），其中若干个最佳当前假设的变体最有可能在下一步被考虑。

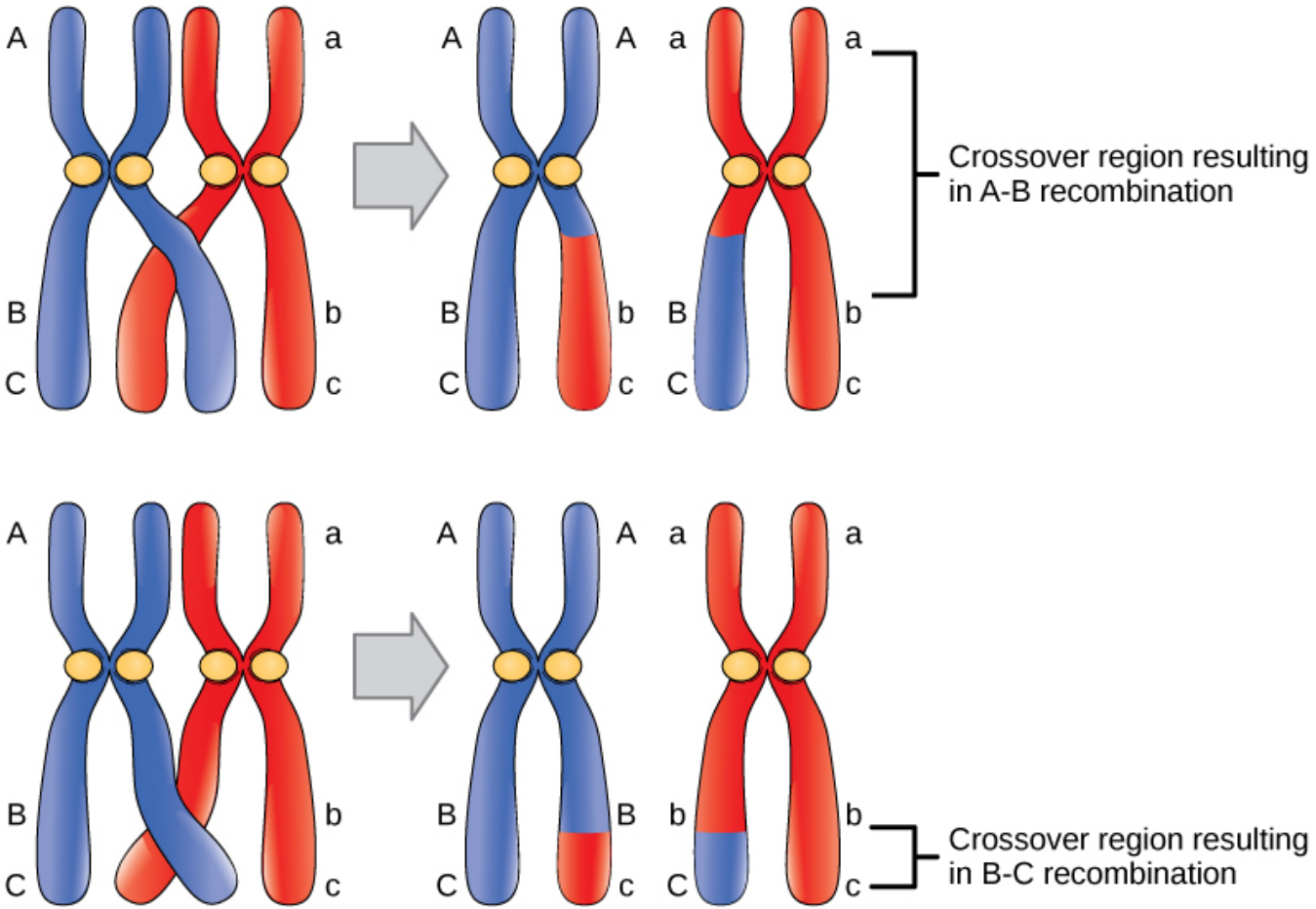
一种受生物进化启发的学习方法。

- 它不再是从一般到特殊或从简单到复杂地搜索假设，而是通过变异和重组当前已知的最好假设来生成后续的假设。
- 在每一步，被称为当前群体（population）的一组假设被更新，方法是通过使用目前适应度最高的假设的后代替代群体的某个部分。
- 这个过程形成了对假设的生成并测试（generate-and-test）柱状搜索（beam-search），其中若干个最佳当前假设的变体最有可能在下一步被考虑。

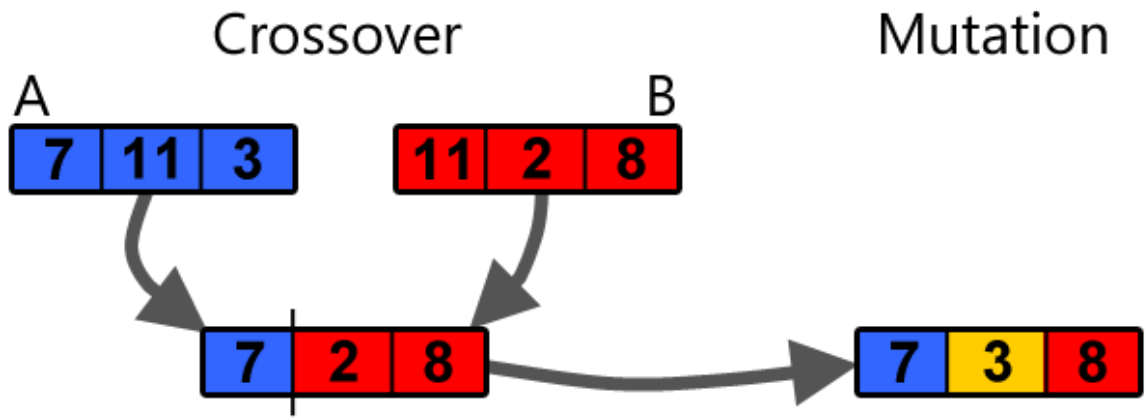
一种受生物进化启发的学习方法。

- 它不再是从一般到特殊或从简单到复杂地搜索假设，而是通过变异和重组当前已知的最好假设来生成后续的假设。
- 在每一步，被称为当前群体（population）的一组假设被更新，方法是通过使用目前适应度最高的假设的后代替代群体的某个部分。
- 这个过程形成了对假设的生成并测试（generate-and-test）柱状搜索（beam-search），其中若干个最佳当前假设的变体最有可能在下一步被考虑。

遗传算子


A
a
B
b
C
c
Crossover region resulting in A-B recombination
Crossover region resulting in B-C recombination

遗传算子



GA 的普及和发展得益于：

- 在生物系统中进化被认为是一种成功的自适应方法，并且具有很好的鲁棒性。
- GA 搜索的假设空间中，假设的各个部分相互作用，每一部分对总的假设适应度的影响难以建模。
- 遗传算法易于并行化，且可降低由于使用超强计算机硬件的带来的昂贵费用。

GA 的普及和发展得益于：

- 在生物系统中进化被认为是一种成功的自适应方法，并且具有很好的鲁棒性。
- GA 搜索的假设空间中，假设的各个部分相互作用，每一部分对总的假设适应度的影响难以建模。
- 遗传算法易于并行化，且可降低由于使用超强计算机硬件的带来的昂贵费用。

GA 的普及和发展得益于：

- 在生物系统中进化被认为是一种成功的自适应方法，并且具有很好的鲁棒性。
- GA 搜索的假设空间中，假设的各个部分相互作用，每一部分对总的假设适应度的影响难以建模。
- 遗传算法易于并行化，且可降低由于使用超强计算机硬件的带来的昂贵费用。

GA 的普及和发展得益于：

- 在生物系统中进化被认为是一种成功的自适应方法，并且具有很好的鲁棒性。
- GA 搜索的假设空间中，假设的各个部分相互作用，每一部分对总的假设适应度的影响难以建模。
- 遗传算法易于并行化，且可降低由于使用超强计算机硬件的带来的昂贵费用。

Topic

问题

求二元函数的最大值：

$$f(x_1, x_2) = x_1^2 + x_2^2$$

其中：

$$x_1 \in \{1, 2, 3, 4, 5, 6, 7\}$$
$$x_2 \in \{1, 2, 3, 4, 5, 6, 7\}$$

个体编码

- 遗传算法的运算对象是表示个体的符号串，所以必须把变量 x_1, x_2 编码为一种符号串。
- 因 x1, x2 为 $[0, 7]$ 之间的整数，所以分别用 3 位无符号二进制整数来表示，将它们连接在一起所组成的 6 位无符号二进制数就形成了个体的基因型，表示一个可行解。
  - 例如，
    - 基因型 $X = 101110$
    - 所对应的表现型是：$x = (5, 6)$
    - 个体的表现型 x 和基因型 X 之间可通过编码和解码来进行转换。

个体编码

- 遗传算法的运算对象是表示个体的符号串，所以必须把变量 x_1, x_2 编码为一种符号串。
- 因 x1, x2 为 $[0, 7]$ 之间的整数，所以分别用 3 位无符号二进制整数来表示，将它们连接在一起所组成的 6 位无符号二进制数就形成了个体的基因型，表示一个可行解。
  - 例如，
    - 基因型 $X = 101110$
    - 所对应的表现型是：$x = (5, 6)$
    - 个体的表现型 x 和基因型 X 之间可通过编码和解码来进行转换。

个体编码

- 遗传算法的运算对象是表示个体的符号串，所以必须把变量 x_1, x_2 编码为一种符号串。
- 因 x1, x2 为 $[0, 7]$ 之间的整数，所以分别用 3 位无符号二进制整数来表示，将它们连接在一起所组成的 6 位无符号二进制数就形成了个体的基因型，表示一个可行解。
  - 例如，
    - 基因型 $X = 101110$
    - 所对应的表现型是：$x = (5, 6)$
    - 个体的表现型 x 和基因型 X 之间可通过编码和解码程序相互转换。

个体编码

- 遗传算法的运算对象是表示个体的符号串，所以必须把变量 x_1, x_2 编码为一种符号串。
- 因 x1, x2 为 $[0, 7]$ 之间的整数，所以分别用 3 位无符号二进制整数来表示，将它们连接在一起所组成的 6 位无符号二进制数就形成了个体的基因型，表示一个可行解。
  - 例如，
    - 基因型 $X = 101110$
    - 所对应的表现型是：$x = (5, 6)$
    - 个体的表现型 x 和基因型 X 之间可通过编码和解码程序相互转换。

个体编码

- 遗传算法的运算对象是表示个体的符号串，所以必须把变量 x_1, x_2 编码为一种符号串。
- 因 x1, x2 为 $[0, 7]$ 之间的整数，所以分别用 3 位无符号二进制整数来表示，将它们连接在一起所组成的 6 位无符号二进制数就形成了个体的基因型，表示一个可行解。
  - 例如，
    - 基因型 $X = 101110$
    - 所对应的表现型是：$x = (5, 6)$
    - 个体的表现型 x 和基因型 X 之间可通过编码和解码程序相互转换。

产生初始种群

- 遗传算法是对群体进行的进化操作，需要给其准备一些表示起始搜索点的初始群体数据。
- 本例中，群体规模的大小取为 4，即群体由 4 个个体组成，每个个体可通过随机方法产生。
  - 如：011101，101011，011100，111001

产生初始种群

- 遗传算法是对群体进行的进化操作，需要给其准备一些表示起始搜索点的初始群体数据。
- 本例中，群体规模的大小取为 4，即群体由 4 个个体组成，每个个体可通过随机方法产生。
  - 如：011101，101011，011100，111001

产生初始种群

- 遗传算法是对群体进行的进化操作，需要给其准备一些表示起始搜索点的初始群体数据。
- 本例中，群体规模的大小取为 4，即群体由 4 个个体组成，每个个体可通过随机方法产生。
  - 如：011101，101011，011100，111001

适应度计算

- 遗传算法中以个体适应度的大小来评定各个个体的优劣程度，从而决定其遗传机会的大小。
- 本例中，目标函数总取非负值，并且是以求函数最大值为优化目标，故可直接利用目标函数值作为个体的适应度。

适应度计算

- 遗传算法中以个体适应度的大小来评定各个个体的优劣程度，从而决定其遗传机会的大小。
- 本例中，目标函数总取非负值，并且是以求函数最大值为优化目标，故可直接利用目标函数值作为个体的适应度。

适应度计算

- 遗传算法中以个体适应度的大小来评定各个个体的优劣程度，从而决定其遗传机会的大小。
- 本例中，目标函数总取非负值，并且是以求函数最大值为优化目标，故可直接利用目标函数值作为个体的适应度。

选择运算

- 选择运算 (或称为复制运算) 把当前群体中适应度较高的个体按某种规则或模型遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中。
- 本例中，我们采用与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。
  - 计算出每个个体的相对适应度的大小 $\frac{f_i}{\sum f_i}$，它即为每个个体被遗传到下一代群体中的概率，

选择过程

编号	种群 $p(0)$	x_1	x_2	适应值	数比例	选择次数
1	011101	3	5	34	0.24	1
2	101011	5	3	34	0.24	1
3	011100	3	4	25	0.17	0
4	111001	7	1	50	0.35	2

选择结果：

011101
111001
101011
111001

选择运算

- 选择运算 (或称为复制运算) 把当前群体中适应度较高的个体按某种规则或模型遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中。
- 本例中，我们采用与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。
  - 计算出每个个体的相对适应度的大小 $\frac{f_i}{\sum f_i}$，它即为每个个体被遗传到下一代群体中的概率，

选择过程

编号	种群 $p(0)$	x_1	x_2	适应值	数比例	选择次数
1	011101	3	5	34	0.24	1
2	101011	5	3	34	0.24	1
3	011100	3	4	25	0.17	0
4	111001	7	1	50	0.35	2

选择结果：

011101
111001
101011
111001

选择运算

- 选择运算 (或称为复制运算) 把当前群体中适应度较高的个体按某种规则或模型遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中。
- 本例中，我们采用与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。
  - 计算出每个个体的相对适应度的大小 $\frac{f_i}{\sum f_i}$，它即为每个个体被遗传到下一代群体中的概率，

选择过程

编号	种群 $p(0)$	x_1	x_2	适应值	数比例	选择次数
1	011101	3	5	34	0.24	1
2	101011	5	3	34	0.24	1
3	011100	3	4	25	0.17	0
4	111001	7	1	50	0.35	2

选择结果：

011101
111001
101011
111001

选择运算

- 选择运算 (或称为复制运算) 把当前群体中适应度较高的个体按某种规则或模型遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中。
- 本例中，我们采用与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。
  - 计算出每个个体的相对适应度的大小 $\frac{f_i}{\sum f_i}$，它即为每个个体被遗传到下一代群体中的概率，

选择过程

编号	种群 $p(0)$	x_1	x_2	适应值	数比例	选择次数
1	011101	3	5	34	0.24	1
2	101011	5	3	34	0.24	1
3	011100	3	4	25	0.17	0
4	111001	7	1	50	0.35	2

选择结果：

011101
111001
101011
111001

选择运算

- 选择运算 (或称为复制运算) 把当前群体中适应度较高的个体按某种规则或模型遗传到下一代群体中。一般要求适应度较高的个体将有更多的机会遗传到下一代群体中。
- 本例中，我们采用与适应度成正比的概率来确定各个个体复制到下一代群体中的数量。
  - 计算出每个个体的相对适应度的大小 $\frac{f_i}{\sum f_i}$，它即为每个个体被遗传到下一代群体中的概率，

选择过程

编号	种群 $p(0)$	x_1	x_2	适应值	数比例	选择次数
1	011101	3	5	34	0.24	1
2	101011	5	3	34	0.24	1
3	011100	3	4	25	0.17	0
4	111001	7	1	50	0.35	2

选择结果：

011101
111001
101011
111001

交叉运算

- 交叉运算是遗传算法中产生新个体的主要操作过程，它以某一概率相互交换某两个个体之间的部分染色体。
- 本例采用单点交叉的方法，其具体操作过程是：
  - 先对群体进行随机配对；
  - 其次随机设置交叉点位置；
  - 最后再相互交换配对染色体之间的部分基因。

交叉过程

编号	选择结果	配对	交叉点位置
1	01,1101	2	2
2	11,1001	1	2
3	1010,11	4	4
4	1110,01	3	4

交叉结果

011001
111101
101001
111011

交叉运算

- 交叉运算是遗传算法中产生新个体的主要操作过程，它以某一概率相互交换某两个个体之间的部分染色体。
- 本例采用单点交叉的方法，其具体操作过程是：
  - 先对群体进行随机配对；
  - 其次随机设置交叉点位置；
  - 最后再相互交换配对染色体之间的部分基因。

交叉过程

编号	选择结果	配对	交叉点位置
1	01,1101	2	2
2	11,1001	1	2
3	1010,11	4	4
4	1110,01	3	4

交叉结果

011001
111101
101001
111011

交叉运算

- 交叉运算是遗传算法中产生新个体的主要操作过程，它以某一概率相互交换某两个个体之间的部分染色体。
- 本例采用单点交叉的方法，其具体操作过程是：
  - 先对群体进行随机配对；
  - 其次随机设置交叉点位置；
  - 最后再相互交换配对染色体之间的部分基因。

交叉过程

编号	选择结果	配对	交叉点位置
1	01,1101	2	2
2	11,1001	1	2
3	1010,11	4	4
4	1110,01	3	4

交叉结果

011001
111101
101001
111011

交叉运算

- 交叉运算是遗传算法中产生新个体的主要操作过程，它以某一概率相互交换某两个个体之间的部分染色体。
- 本例采用单点交叉的方法，其具体操作过程是：
  - 先对群体进行随机配对；
  - 其次随机设置交叉点位置；
  - 最后再相互交换配对染色体之间的部分基因。

交叉过程

编号	选择结果	配对	交叉点位置
1	01,1101	2	2
2	11,1001	1	2
3	1010,11	4	4
4	1110,01	3	4

交叉结果

011001
111101
101001
111011

交叉运算

- 交叉运算是遗传算法中产生新个体的主要操作过程，它以某一概率相互交换某两个个体之间的部分染色体。
- 本例采用单点交叉的方法，其具体操作过程是：
  - 先对群体进行随机配对；
  - 其次随机设置交叉点位置；
  - 最后再相互交换配对染色体之间的部分基因。

交叉过程

编号	选择结果	配对	交叉点位置
1	01,1101	2	2
2	11,1001	1	2
3	1010,11	4	4
4	1110,01	3	4

交叉结果

011001
111101
101001
111011

变异运算

- 变异运算是对个体的某一个或某一些基因座上的基因值按某一较小的概率进行改变，它也是产生新个体的一种操作方法。
- 本例中，我们采用基本位变异的方法来进行变异运算，其具体操作过程是：
  - 首先确定出各个个体的基因变异位置，下表所示为随机产生的变异点位置，其中的数字表示变异点设置在该基因座处；
  - 然后依照某一概率将变异点的原有基因值取反。

变异过程

个体编号	交叉结果	变异点	变异结果
1	011001	4	011101
2	111101	5	111111
3	101001	2	111001
4	111011	6	111010

变异运算

- 变异运算是对个体的某一个或某一些基因座上的基因值按某一较小的概率进行改变，它也是产生新个体的一种操作方法。
- 本例中，我们采用基本位变异的方法来进行变异运算，其具体操作过程是：
  - 首先确定出各个个体的基因变异位置，下表所示为随机产生的变异点位置，其中的数字表示变异点设置在该基因座处；
  - 然后依照某一概率将变异点的原有基因值取反。

变异过程

个体编号	交叉结果	变异点	变异结果
1	011001	4	011101
2	111101	5	111111
3	101001	2	111001
4	111011	6	111010

变异运算

- 变异运算是对个体的某一个或某一些基因座上的基因值按某一较小的概率进行改变，它也是产生新个体的一种操作方法。
- 本例中，我们采用基本位变异的方法来进行变异运算，其具体操作过程是：
  - 首先确定出各个个体的基因变异位置，下表所示为随机产生的变异点位置，其中的数字表示变异点设置在该基因座处；
  - 然后依照某一概率将变异点的原有基因值取反。

变异过程

个体编号	交叉结果	变异点	变异结果
1	011001	4	011101
2	111101	5	111111
3	101001	2	111001
4	111011	6	111010

变异运算

- 变异运算是对个体的某一个或某一些基因座上的基因值按某一较小的概率进行改变，它也是产生新个体的一种操作方法。
- 本例中，我们采用基本位变异的方法来进行变异运算，其具体操作过程是：
  - 首先确定出各个个体的基因变异位置，下表所示为随机产生的变异点位置，其中的数字表示变异点设置在该基因座处；
  - 然后依照某一概率将变异点的原有基因值取反。

变异过程

个体编号	交叉结果	变异点	变异结果
1	011001	4	011101
2	111101	5	111111
3	101001	2	111001
4	111011	6	111010

新群体

- 对群体 P(t) 进行一轮选择、交叉、变异运算之后可得到新一代的群体 p(t+1)。
- 从表中可以看出，群体经过一代进化之后，其适应度的最大值、平均值都得到了明显的改进。事实上，这里已经找到了最佳个体“111111”。
- 表中有些数据是随机产生的。为了更好地说明问题，特意选择了一些数值以便能够得到较好的结果，而在实际运算过程中可能需要一定的循环次数才能达到最优结果。

新群体

个体编号	种群 $p(1)$	x_1	x_2	适应值	占总数比例
1	011101	3	5	34	0.14
2	111111	7	7	98	0.42
3	111001	7	1	50	0.21
4	111010	7	2	53	0.23

新群体

- 对群体 P(t) 进行一轮选择、交叉、变异运算之后可得到新一代的群体 p(t+1)。
- 从表中可以看出，群体经过一代进化之后，其适应度的最大值、平均值都得到了明显的改进。事实上，这里已经找到了最佳个体“111111”。
- 表中有些数据是随机产生的。为了更好地说明问题，特意选择了一些数值以便能够得到较好的结果，而在实际运算过程中可能需要一定的循环次数才能达到最优结果。

新群体

个体编号	种群 $p(1)$	x_1	x_2	适应值	占总数比例
1	011101	3	5	34	0.14
2	111111	7	7	98	0.42
3	111001	7	1	50	0.21
4	111010	7	2	53	0.23

新群体

- 对群体 P(t) 进行一轮选择、交叉、变异运算之后可得到新一代的群体 p(t+1)。
- 从表中可以看出，群体经过一代进化之后，其适应度的最大值、平均值都得到了明显的改进。事实上，这里已经找到了最佳个体“111111”。
- 表中有些数据是随机产生的。为了更好地说明问题，特意选择了一些数值以便能够得到较好的结果，而在实际运算过程中可能需要一定的循环次数才能达到最优结果。

新群体

个体编号	种群 $p(1)$	x_1	x_2	适应值	占总数比例
1	011101	3	5	34	0.14
2	111111	7	7	98	0.42
3	111001	7	1	50	0.21
4	111010	7	2	53	0.23

新群体

- 对群体 P(t) 进行一轮选择、交叉、变异运算之后可得到新一代的群体 p(t+1)。
- 从表中可以看出，群体经过一代进化之后，其适应度的最大值、平均值都得到了明显的改进。事实上，这里已经找到了最佳个体“111111”。
- 表中有些数据是随机产生的。为了更好地说明问题，特意选择了一些数值以便能够得到较好的结果，而在实际运算过程中可能需要一定的循环次数才能达到最优结果。

新群体

个体编号	种群 $p(1)$	x_1	x_2	适应值	占总数比例
1	011101	3	5	34	0.14
2	111111	7	7	98	0.42
3	111001	7	1	50	0.21
4	111010	7	2	53	0.23

新群体

- 对群体 P(t) 进行一轮选择、交叉、变异运算之后可得到新一代的群体 p(t+1)。
- 从表中可以看出，群体经过一代进化之后，其适应度的最大值、平均值都得到了明显的改进。事实上，这里已经找到了最佳个体“111111”。
- 表中有些数据是随机产生的。为了更好地说明问题，特意选择了一些数值以便能够得到较好的结果，而在实际运算过程中可能需要一定的循环次数才能达到最优结果。

新群体

个体编号	种群 $p(1)$	x_1	x_2	适应值	占总数比例
1	011101	3	5	34	0.14
2	111111	7	7	98	0.42
3	111001	7	1	50	0.21
4	111010	7	2	53	0.23

Topic

遗传算法描述

- GA 研究的问题是搜索一个候选假设的空间，以确定最佳的假设。
- 在 GA 中，“最佳假设”被定义为是使“适应度（fitness）”最优的假设，适应度是为当前问题预先定义的数字度量。
- 算法迭代更新一个假设池，这个假设池称为群体。在每一次迭代中，
  - 根据适应度函数评估群体中的所有成员。
  - 然后从当前群体中用概率方法选取适应度最高的个体产生新的一代。
  - 在这些被选中的个体中，
    - 一部分保持原样地进入下一代群体，
    - 其他的被用作产生后代个体的基础，其中应用象交叉和变异这样的遗传方法。

遗传算法（变量说明）

- *Fitness*：适应度评分函数，为给定假设赋予一个评估得分。
- *Fitness_threshold*：指定终止判据的阈值。
- p：群体中包含的假设数量。
- r：每一步中通过交叉取代群体成员的比例。
- m：变异率。

遗传算法（算法流程）

GA($Fitness, Fitness\_threshold, p, r, m$)

- 初始化群体：$P \leftarrow$ 随机产生 p 个假设
- 评估：对于 P 中的每一个 h ，计算 $Fitness(h)$
- 当 $[\max_h Fitness(h)] < Fitness\_threshold$ ，产生新的一代 P_S ：
  - 选择：用概率方法选择 P 的 $(1-r)p$ 个成员加入 P_S 。从 P 中选择假设 h_i 的概率 $\Pr(h_i)$ 通过下面公式计算：

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  - 交叉：根据上面给出的 $\Pr(h_i)$ ，从 P 中按概率选择 $\frac{r \cdot p}{2}$ 对假设。对于每一对假设 $\langle h_1, h_2 \rangle$ 应用交叉算子产生两个后代。把所有的后代加入 P_S 。
  - 变异：使用均匀的概率从 P_S 中按比例 m 选取成员。对于选出的每个成员，在它的表示中随机选择一个位取反。
  - 更新：$P \leftarrow P_s$ 。
  - 评估：对于 P 中的每一个 h 计算 $Fitness(h)$
- 从 P 中返回适应度最高的假设。

表示假设

$$(Outlook = Overcast \vee Rain) \wedge (Wind = Strong)$$

表示为

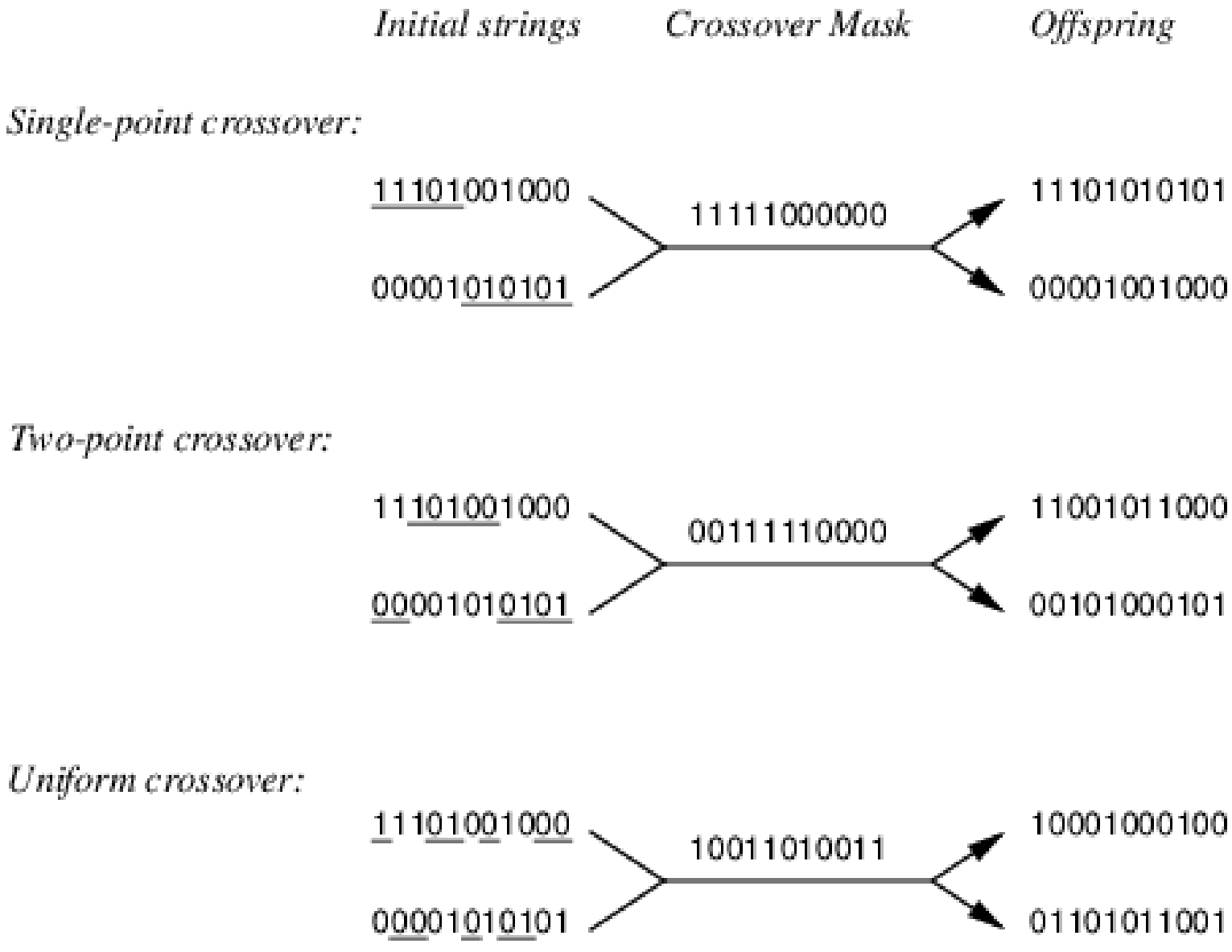
$Outlook$	$Wind$
011	10

$$\text{IF} \quad Wind = Strong \quad \text{THEN} \quad PlayTennis = yes$$

表示为

$Outlook$	$Wind$	$PlayTennis$
111	10	10

遗传算子


Initial strings
Crossover Mask
Offspring
Single-point crossover:
11101001000
00001010101
11111000000
11101010101
00001001000
Two-point crossover:
11101001000
00001010101
00111110000
11001011000
00101000101
Uniform crossover:
11101001000
00001010101
10011010011
10001000100
01101011001

适应度函数

适应度函数定义了候选假设的排名准则，并且是以概率方法选择下一代群体的准则。

- 如果任务是学习分类的规则，那么适应度函数中会有一项用来评价每个规则对训练样例集合的分类精度。
- 适应度函数中也可能包含其他的准则，
  - 例如规则的复杂度和一般性（generality）。
- 当位串被解释为复杂的过程时（例如，当位串表示一系列规则，这些规则要被链接在一起控制一个机器人设备），

适应度函数可以测量生成的过程总体性能而不是单个规则的性能。

假设选择

- 适应度比例选择（Fitness proportionate selection）

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

  (易导致拥挤 (**crowding**))
- 锦标赛选择（Tournament selection）
  - 按均匀分布随机选取两个假设 h_1, h_2
  - 按事先定义的概率 p, 选择适应度较高的假设。按 $1 - p$ 选择适应度较低的假设
- 排序选择（Rank selection）
  - 将当前群体的所有假设按适应度排序
  - 选择某假设的概率与此假设的排名成比例

Topic

GABIL [DeJong et al. 1993]

- 学习以命题规则的析取集合表示的布尔概念
- 适应度（{**Fitness**}）

$$Fitness(h) = (correct(h))^2$$

- 表示假设（{**Representation:**}）

IF $a_1 = T \wedge a_2 = F$ **THEN** $c = T$; **IF** $a_2 = T$ **THEN** $c = F$

表示为

a_1	a_2	c	a_1	a_2	c
10	**01**	**1**	**11**	**10**	**0**

- 遗传算子
  - 编码规则集位串长度可变
  - 产生的位串表示良定义的（**well-defined**）规则集

可变长度位串交叉操作 (Crossover with Variable-Length Bitstrings)

- 交叉前

	a_1	a_2	c		a_1	a_2	c
h_1 :	10	01	1		11	10	0
h_2 :	01	11	0		10	01	0

- 选择 h_1 的交叉点, 如，位于第 1, 第 8 位后
- 限制 h_2 中的交叉点以产生良定义的位串, 如, $\langle 1,3\rangle$, $\langle 1,8\rangle$, $\langle 6,8\rangle$. 若选择 $\langle 1,3\rangle$, 结果为

	a_1	a_2	c
h_3 :	11	10	0

和

	a_1	a_2	c		a_1	a_2	c		a_1	a_2	c
h_4 :	00	01	1		11	11	0		10	01	0

GABIL 扩展

- 增加两个新的遗传算子
  - AddAlternative
    - 泛化对某个特定属性的约束，方法是把这个属性对应的子串中的一个 0 改为 1。
    - 例如，如果一个属性的约束使用串 10010 表示，那么这个算子可能把它改为 10110。这个算子在每一代群体中对选定的成员按照 0.01 的概率应用。
  - DropCondition，
    - 采用一种更加极端的泛化措施，把一个特定属性的所有位都替换为 1。
    - 这个算子相当于通过完全撤销属性约束来泛化规则，它按照概率 0.60 在每一代中应用。
- 对假设的位串表示进行了扩展，使其包含另外两位以决定是否可以对该假设应用这两个算子。

a_1	a_2	c	a_1	a_2	c	AA	DC
01	11	0	10	01	0	1	0

GABIL Results

对人为设计的 12 个问题平均性能：

- GABIL 达到了 92.1% 的平均泛化精度，
- GABIL 扩展算法（with *AA* and *DC* operators）：达到了 95.2% 的平均泛化精度
- 符号规则/树学习方法 C4.5，ID5R，AQ14 的性能是在 91.2% 到 96.6% 之间。

Topic

模式（Schemas ）

是否能用数学的方法刻画 GA 中群体随时间进化的过程？

- 一个模式是由若干 0、1 和 * 组成的任意串。
- “*” 表示一个不关心的位
- 例如模式 0*10 表示的位串集合中只包含 0010 和 0110。

模式理论

根据每个模式的实例数量来刻画 GA 中群体的进化。

- $\bar{f}(t) =$ 群体中所有个体在时间 t 的平均适应度
- $\hat{u}(s,t) =$ 群体中模式 s 的实例在时间 t 的平均适应度
- $m(s,t)$ 表示群体中的模式 s 在时间 t（也就是在第 t 代）的实例数量。
- 模式理论根据 m(s, t) 和模式、群体及 GA 参数的其他属性，来描述 $m(s,t+1)$ 的期望值。

考虑选择算子

- 单次选中 h 的概率为：

$$\begin{aligned} \Pr(h) &= \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} \\ &= \frac{f(h)}{n\bar{f}(t)} \end{aligned}$$

- 单次选中 s 的实例的概率为：

$$\begin{aligned} \Pr(h \in s) &= \sum_{h \in s \cap p_t} \frac{f(h)}{n\bar{f}(t)} \\ &= \frac{\hat{u}(s,t)}{n\bar{f}(t)} m(s,t) \end{aligned}$$

- n 次选择后 s 的实例的期望数量

$$E[m(s,t+1)] = \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)$$

模式定理 (Schema Theorem)

$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t) \left(1 - p_c \frac{d(s)}{l-1}\right) (1-p_m)^{o(s)}$$

- $m(s,t) =$ 模式 s 在时间 t 的实例数
- $\bar{f}(t) =$ 在时间 t 的群体适应度
- $\hat{u}(s,t) =$ 在时间 t 模式 s 的实例平均适应度
- $p_c =$ 单点交叉算子概率
- $p_m =$ 变异概率
- $l =$ 个体位串长度
- $o(s) = s$ 中确定位的个数，(不包括 “*”)
- $d(s) =$ 模式 s 中最左边的确定位和最右边的确定位间的距离

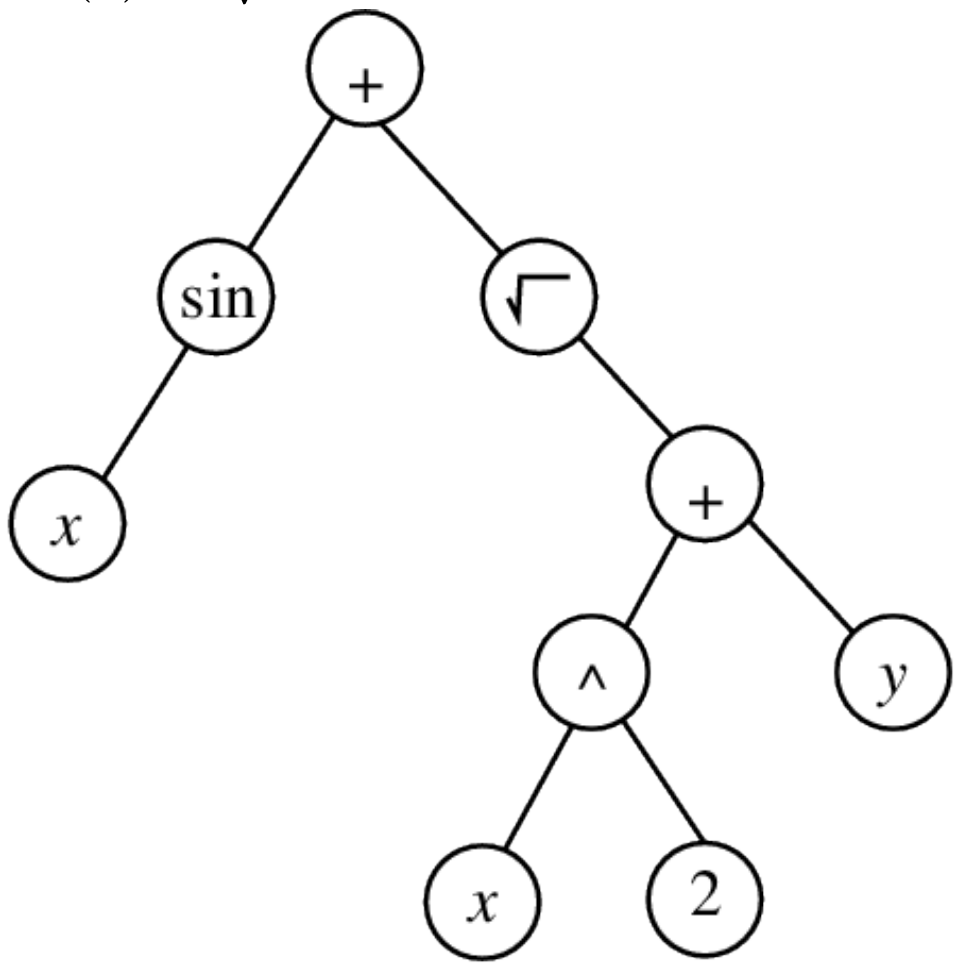
Topic

程序的种群表示为树

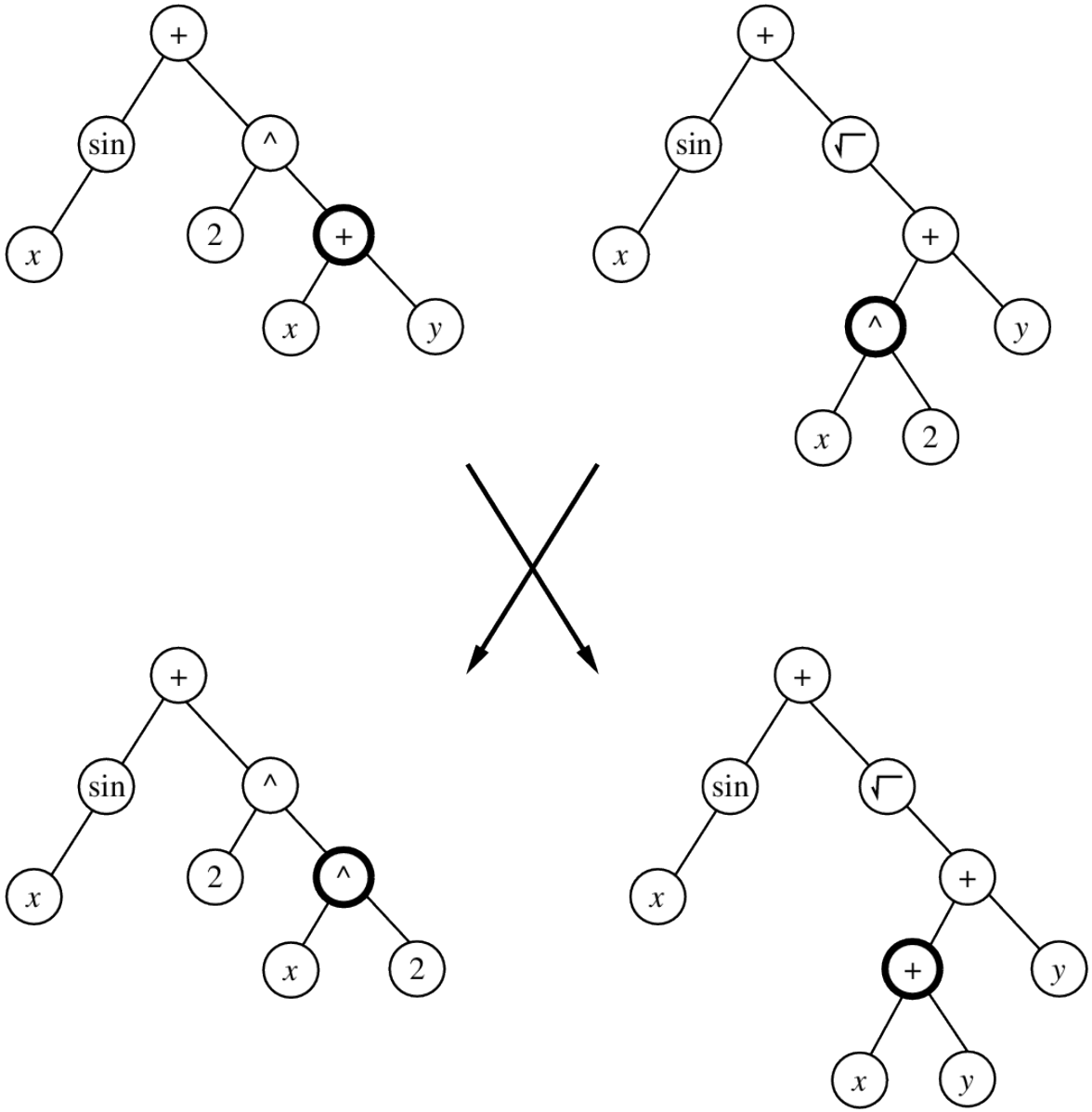
- 解析树

  GP 操作的程序一般被表示为程序的解析（parse）树。每个函数调用被表示为树的一个节点，函数的参数通过它的子结点给出。

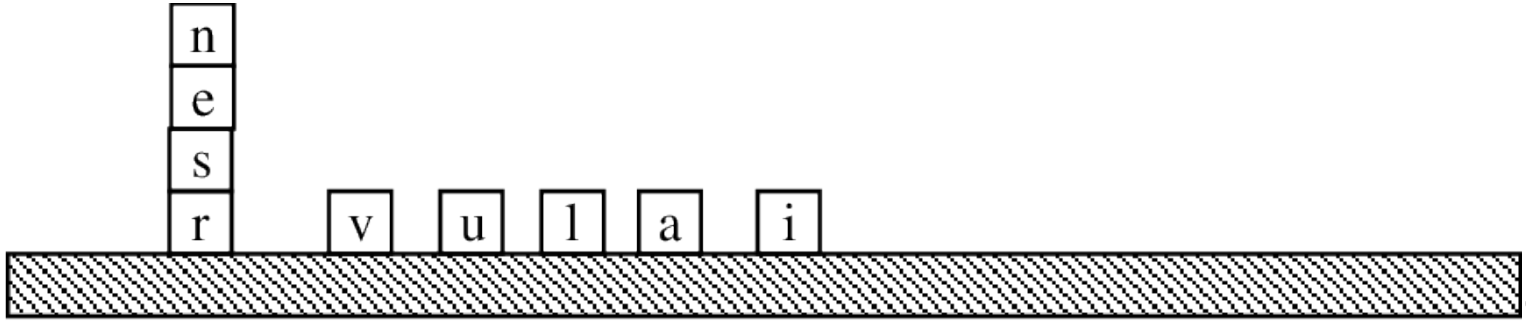
- $\sin(x) + \sqrt{x^2 + y}$



解析树交叉



Block Problem



Goal: spell UNIVERSAL

端点参数（Terminals）：

- CS (“current stack”) = name of the top block on stack, or *F*.
- TB (“top correct block”) = name of topmost correct block on stack
- NN (“next necessary”) = name of the next block needed above TB in the stack

原始函数（Primitive functions）：

- (MS x): ("move to stack"), if block x is on the table, moves x to the top of the stack and returns the value T. Otherwise, does nothing and returns the value F.
- (MT x): ("move to table"), if block x is somewhere in the stack, moves the block at the top of the stack to the table and returns the value T. Otherwise, returns F.
- (EQ x y): ("equal"), returns T if x equals y, and returns F otherwise.
- (NOT x): returns T if $x = F$, else returns F
- (DU x y): ("do until") executes the expression x repeatedly until expression y returns the value T

Learned Program

- Trained to fit 166 test problems
- Using population of 300 programs, found this after 10 generations:

```
(EQ (DU (MT CS) (NOT CS)) (DU (MS NN)(NOT NN)))
```

Genetic Programming

- More interesting example: design electronic filter circuits
- Individuals are programs that transform begining circuit to final circuit, by adding/subtracting components and connections
- Use population of 640,000, run on 64 node parallel processor
- Discovers circuits competitive with best human designs

GP for Classifying Images

- Teller and Veloso(1997)
- Fitness: based on coverage and accuracy
- Representation:
  - Primitives include Add, Sub, Mult, Div, Not, Max, Min, Read, Write, If-Then-Else, Either, Pixel, Least, Most, Ave, Variance,Difference, Mini, Library
  - Mini refers to a local subroutine that is separately co-evolved Library refers to a global library subroutine (evolved by selecting the most useful minis)
- Genetic operators:
  - Crossover, mutation
  - Create “mating pools” and use rank proportionate reproduction

Biological Evolution

- Lamark (19th century)
  - Believed individual genetic makeup was altered by lifetime experience
  - But current evidence contradicts this view
- What is the impact of individual learning on population evolution?

Baldwin Effect

- Assume
  - Individual learning has no direct influence on individual DNA
  - But ability to learn reduces need to “hard wire” traits in DNA
- Then
  - Ability of individuals to learn will support more diverse gene pool
  - Because learning allows individuals with various “hard wired” traits to be successful
  - More diverse gene pool will support faster evolution of gene pool

$\rightarrow$ individual learning (indirectly) increases rate of evolution

Baldwin Effect,Plausible example:

- New predator appears in environment
- Individuals who can learn (to avoid it) will be selected
- Increase in learning individuals will support more diverse gene pool
- resulting in faster evolution
- possibly resulting in new non-learned traits such as instintive fear of predator

Computer Experiments on Baldwin Effect

- Hinton and Nowlan(1987)
- Evolve simple neural networks:
  - Some network weights fixed during lifetime, others trainable
  - Genetic makeup determines which are fixed, and their weight values
- Results:
  - With no individual learning, population failed to improve over time
  - When individual learning allowed
    - Early generations: population contained many individuals with many trainable weights
    - Later generations: higher fitness, while number of trainable weights decreased

Summary: Evolutionary Programming

- Conduct randomized, parallel, hill-climbing search through H
- Approach learning as optimization problem (optimize fitness)
- Nice feature: evaluation of Fitness can be very indirect
  - consider learning rule set for multistep decision making
- no issue of assigning credit/blame to indiv. steps