由SAT问题展开说(1) (转)

**由SAT问题展开说**

[**XML**](http://amyz.itpub.net/post/34151/leibie-12-1.html)**:namespace prefix = o ns = "urn:schemas-**[**microsoft**](http://amyz.itpub.net/post/34151/tag-126-1.html)**-com:**[**Office**](http://amyz.itpub.net/post/34151/tag-100-1.html)**:office" />**

  作者： 李连华  崔涛

**摘要**：本文试图以活泼的笔调，讨论一个[计算机](http://amyz.itpub.net/post/34151/tag-223-1.html)科学难题，SAT问题。并由此展开，展示计算机算法的魅力，与计算机算法分析和设计的一些基本思想。最后，给出一个SAT问题的演化计算的算法和[程序](http://amyz.itpub.net/post/34151/tag-254-1.html)实现。

**关键字**：SAT问题，组合爆炸，爬山法，A算法，演化计算

  由SAT问题展开说（1）

**序言**

计算机是一门科学。科学离不开问题；计算机科学也面临着一大批问题，尤其以NPC问题称著。它们目前都没有得到很好的解决。如果你仔细审视它们，就会发现科学发展的急迫与艰难，及美好。

在众多NPC问题中，有一个问题被称为其它问题的“种子”，那就是SAT问题。SAT问题是Cook提出的著名计算机科学难题，依据

  Cook’s Theorem  SATI[sf](http://amyz.itpub.net/post/34151/tag-161-1.html)IBILITY is NP-complete,

 SAT问题是一类问题的难度标准，所以又可称为NPC问题的种子。

我们不证明Cook定理（见注释1），叙述SAT问题如下。

**问题的叙述**

SAT问题或称为可满足性问题，我们要了解它，先看一些概念。

句子：为有限个逻辑变量（或其非¬）的或（∨）式，例如句子

C1=x1 ∨¬x2 ∨x3 ∨ ¬x4，

每一个逻辑变量可取值为真或假，分别用1和0表示。则对于一个句子的值，可用逻辑变量的一组指派来求得。例如x1 、x2 、x3 、x4的一个指派为：0001，则C1的值为1（真），该指派为成真指派。

对于逻辑变量的集合 A={ x1,x2,…,xn },所形成的有限句子集合为F={ c1,c2,…,cm }。如何判断是否有一个A上的指派，使得F中所有句子皆为真？则这就是SAT问题。例如

F1={ (x1 ∨x2 ∨x3),(¬x1∨x2),(¬x2∨x3), (¬x3∨x1),(¬x1 ∨¬x2∨ ¬x3)  }，

可以证明F1是不可满足的。

但对于一个一般性的集合F，就不是那么容易判断了。

**问题的初步分析**

  不妨设逻辑变量的集合A={ x1,x2,…,xn }，即 |A|=n ，有限句子集合为F={ c1,c2,…,cm }，即 |F|=m。我们来讨论如何用计算机算法来解决这个问题。关于算法，这里须是有限时间停机的（见注释2） 。

这是一个有数理逻辑背景的问题，问题解空间的描述形式，使用长度为n的0/1串的集合是十分自然的。设这个集合为Ω，则若 s∈Ω，就有s是长度为n的0/1串，记作 s=（0|1）^n 。

显然我们要判断能否找出一个s\*∈Ω，其中s\*是F的成真指派。

集合Ω并不是无穷集，显然有|Ω|=2^n 。如果使用穷举法，是可解的，但算法的时间[效率](http://amyz.itpub.net/post/34151/tag-246-1.html)是O（2^n），指数级增长的，这个规模使得我们的计算机无法承担问题规模增长所带来的消耗。如此大的搜索空间，计算机在可等待的时间里算不完了，产生了组合爆炸问题。如下图：

[ASP](http://amyz.itpub.net/post/34151/leibie-6-1.html)ectratio="t">

2003-8-261022020.gif" align=baseline border=0>

这种搜索路径，只怕未至问题的解，大家都不知“今昔是何年”了。我们该如何办呢？如何才能得到尽可能快的搜索路径，接近我们的问题的解呢？

依据直觉，我们需要某种限制，使得我们的搜索活动在一个较小的空间里接近问题的解，而走一个较直的搜索路径。当然，最好是直线走到问题的解，甚至一下子跳到目标那里，可是你我都知道，多数时候是办不到的。一般的，我们希望能如此：

可是，这个搜索的限制，对于每个问题，如何找到？图1和图2中我们还给出两个参数α（n）和β（n），都是关于问题的规模n的[函数](http://amyz.itpub.net/post/34151/tag-266-1.html)，分别表示解空间增长的速度和搜索限制的好坏，一般的，都不象是图中的边界那样为线性的。

这是本文的核心问题。我们以SAT问题为例展开，一起来领略计算机算法世界的神奇与艰辛。

**问题的进一步分析**

虽然SAT问题有数理逻辑的背景，可是数学模型也完全可以用图来描述。想一下，每个指派s都可视为一个状态，作为图中的结点。则集合Ω就是图中的结点集。考虑到集合Ω的势，我们可想象出这个图，而不全部画出它，如：

至于边，则可以自己因方便定义。

例如，在超立方体[网络](http://amyz.itpub.net/post/34151/tag-166-1.html)中，相临的结点的编码（0/1串）都是只有一位相异的，[路由](http://amyz.itpub.net/post/34151/tag-334-1.html)可以很方便的选择的算法为e立方体路由算法，受其启发，我们可以定义图中边（Ri，Rj）存在当且仅当Ri与Rj的编码只有一位不同。等等。

不论如何定义边，我们都可以方便的想到，我们的算法，都是由某个（些）结点出发，走一条尽可能短的路径，达到某个（些）特殊的结点----问题的解！

在这个数学模型上，有些措施在搜索解时是可以方便的加以利用的，如深度优先搜索和宽度优先搜索。可是，我们更希望的是，搜索可以有所限制，而避免一些路径的出现！我们猜想着这样可以对我们的问题有帮助。

最简单的一个策略是，每当我们想走向下一个结点的时候，都要给它作出一个判断，一个评价值。如果这个值小于当前所在的结点，则我们不考虑延伸这条路径致该结点。

其实这是爬山法的基本思想。所谓的“爬山”，是形象的说法。其中关键的，就是这个评价函数，另外，爬山法的基本思想是好的，但也有一些问题。如下图：

首先，当连续许多结点的评价值都是相同的时候，意味着对于该评价函数，我们到了一个“平地”，如何走出“平地”，或判断该地就是最高点？其次，对于 不是单峰的爬山函数，常常会有许多“局部山峰”，它们代表了问题的局部最优解，可是，如何识别这个“局部山峰”的陷阱，而不丢失“顶峰”？

为了避免这些问题，我们的思路可以这样进行。既然只考虑下一结点与当前结点的比较还是不够的，我们就再加一些限制，以期望能包含更多的信息，搜索起来更方便。例如，走到一个“平地”或“局部山峰”，我们就往“顶峰”望一望，看看差距是多少。然后再决定下面的步骤。

这种思路，是人工智能中常用的一种策略，一些带启发信息的算法，比如A算法，A\*算法，都是这一类。我们具体的看SAT问题。

对于F，任意s∈Ω，我们设

F’={ ci | ci 对于s成真，ci∈F ,1<=i<=m} ，

一个简单的爬山函数为

  f(s)= | F’ | ,

设定回朔及算法返回出口，可以写一个带回朔的爬山法来求解。

同样，我们可以考虑加入启发信息。如下，设集合

  F1’={ ci | ci 对于s 成真，ci∈F ,1<=i<=m} ，

  F2’={ xi | xi∈A且xi出现在cj中而不出现在ck中，其中cj∈F-F1’，ck∈F1’ ,

  1<=i<=n,1<=j,k<=m} ，

则   耗散函数为  f ’(s)= | F1’ |,

  启发函数为  h(s)= |F2’ |,

  评价函数为  f(s)= f ‘(s) + h(s),

据此，可以写出A算法求解（见注释3） 。

事实上，这些算法的关键都在于评价函数的选择。寻找好的评价函数，则是一件十分困难的事情。我们这里所给出的评价函数，都未必是最优的。可是通过这 样的一步步分析，希望的是读者能理解，计算机算法是如何来解决问题的。我们倾向于认为，一个好算法的设计，需要形象思维和数学知识的结合，需要理论分析和 实验的结合。我们也试图向你展示这一点。

我们还要讨论更为快捷的搜索策略。

**问题的再分析**

对于s=（0|1）^n 本身，我们可以使用图来描述，那样使用图中的搜索策略会方便的多。可是串本身也是很好的数学[对象](http://amyz.itpub.net/post/34151/tag-249-1.html)，尤其是在演化计算中，这次我们就由“串”出发来进行讨论。

我们先不给出演化计算的定义，而是接着上面的思路，来看这个SAT问题；尽可能的，我们需要一切都“水到渠成”。

在非常大的搜索空间里寻找问题的解，我们说我们希望能提供一个限制，以加快搜索速度。上面的讨论中，本质都是在图论的模型上，一个结点、一个结点的 接近目标。似乎有句戏言，“甚至一下子跳到目标那里”，我们的潜意识里给屏蔽掉了。呜呼哀哉，潜意识里躲避不理智的东西，有时反而是个枷锁。

想一想，采用随机搜索策略，我们搜索到一个s，如果不是解，依据某种随机控制机制，把s变到另一个s’，如此进行。若这种随机策略是收敛的，则我们可以尽可能快的且有保证的“跳”到问题的解。

这种“变”的过程，是策略的关键。如何用数学来表述它？作为形象的和有效的，事实选择了大自然的进化特点，我们来看一幅图：

是不是很有意思？这样我们的“变”，可以借用生物的术语，是杂交和变异。例如对于

  s1=010-100-11，

  s2=110-000-10，

杂交后可以是：

s1’=010-000-11，

s2’=110-100-10；

而变异后可能是：

s1’’=010-101-11。

  作为数学的理论分析，我们使用[模式](http://amyz.itpub.net/post/34151/tag-270-1.html)理论等来分析这种“变”的收敛性和[性能](http://amyz.itpub.net/post/34151/tag-245-1.html)。但这个思路，的确是个非常好的思路。作者总是惊奇那些优美而有效的解决方案，当初产生在人脑之初的时候，会是个什么样子！总是忍不住想尽力去模拟，我们也有意提醒读者要养成这样的习惯。

这个思路，就是演化计算的思路。演化计算，是模拟大自然的演化特点，而使用于计算机算法上的一种自组织智能计算。

 我们下次将继续深入讨论这种思路。

注释1：

该定理的证明，要用到数理逻辑的结论。事实上，只要证明任一属 NP的问题可以转化为可满足性问题，其时间有多项式的界，本定理即可证明。

注释2：

以前在csdn上写的文章《有趣的算法世界》里讨论算法的定义的时候，曾谈了算法可否不停机的问题。所以这里有此说，读者若感兴趣，可参阅：

.NET/develop/read\_article.asp?id=19193">http://www.csdn.net/develop/read\_article.asp?id=19193

<http://www.csdn.net/develop/read_article.asp?id=20317>

注释3：

可以证明 h(t) 和 h\*(t) 不具备关系： h(t)<=h\*(t)，故此算法为A算法。