国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第16卷第1期 1994年3月 Vol. 16 No. 1

一个NP－完全问题的求解复杂性剖析*

姜新文 王兵山

（国防科技大学电子计算机系 长沙 410073）

摘 要 本文提出一个构造的NP完全问题RHC并证明其NP完全性。在此基础上，通过分析通用图灵机带头移动的次数，讨论了通用图灵机上任一求解RHC的算法的复杂性。分析结果揭示了在简单计算模型（定义见正文）上寻找一个对满足RHC的任意输入，而不是对某些特殊实例都能正确求解的算法的困难性。根据本文的讨论，我们认为，给出本文分析的严格论证或许只是时间问题。

关键词 复杂性，算法，NP问题，NP－完全问题

分类号 TP301.5

NP－完全问题的求解复杂性决定NP＝P是否成立。1970年，COOK证明了SAT∈NPC[1]。此后，人们证明了3000个以上的问题的NP完全性[2]。关于NP完全问题的求解复杂性有许多猜测和分析，很多人倾向于NP完全问题不是多项式时间可解的，即NP≠P[2,3]. 1987年，图论六大难题之一的子图同胚问题终获解决[4,5]，更多的人开始倾向于NP＝P。本文是对上述倾向的反动。

究NP完全问题难获解决之因，我们认为，在于难以确定求解一个NP问题所必需的操作。通常讨论计算复杂性都和一个具体的模型有关。因此，每提到一个问题的计算复杂性下界，都要指出在什么模型下某种操作需要多少。这自然引起下述问题：

（1）不采用指定的某种操作，不能求该问题吗？

（2）如果取相同的耗费标准且不限制计算模型、不限制所采用的操作，该问题的计算复杂性下界又是多少？

如果确定了求解一个问题所必需的操作（与模型无关），就可以讨论问题的固有计算难度。一个操作必然涉及操作对象，模型M上一个算法的执行过程中全部操作所涉及的对象的个数和次数决定了算法的计算复杂性。这在通用图灵机上对应于带头移动到表示相应数据的代码段并在其上工作。带头的移动当然是最基本、最起码的工作。所以，一个算法涉及的对象的个数和次数的多少决定通用图灵机上带头移动的次数，因而决定了它的计算复杂性。求解一个问题的任一算法必需的复杂性即问题的复杂性下界。

因此，讨论NP完全问题的时间复杂性。首先要确定什么操作是求解该问题必需的操

* 1992年11月23日收稿

作。本文首先定义了简单计算模型和复杂计算模型，然后提出一个构造的NP完全问题RHC、证明RHC与OPRHC的NP等价性，最后分析在简单计算模型上设计一个对满足OPRHC的任意输入而不是对某些特殊实例都能正确求解的算法的困难性。

1 主要结论

定义 1 计算模型M称为简单计算模型当且仅当它满足以下两个条件：

1° M的计算能力与通用图灵机等价；

2° 对求解任意规模为 n 的问题，M上的每个基本动作（用图灵机模拟）的完成时间囿于 n 的多项式函数 $p(n)$。

定义 2 计算模型M称为复杂计算模型当且仅当它满足以下两个条件：

1° M的计算能力与通用图灵机等价；

2° 对求解任意规模为 n 的问题，M上的每个基本动作（用图灵机模拟）的完成时间不囿于 n 的任一给定的多项式函数 $p(n)$。

计算复杂性研究中采用的模型（如RAM模型等）都是简单计算模型。因为复杂计算模型上由于每个基本动作的“复杂性”可以导致算法的“简单性”，这将使复杂性讨论不能正确反映问题的固有计算难度。相反，若在简单计算模型上定义耗费标准、进行必要的简化和讨论复杂性，就复杂性量级而言，则能准确反映问题的计算难度。下面所提到的模型都指简单计算模型。

现在提出一个构造的NP一完全问题。

实例 设 $G=<V, E>$ 为一个简单无向图，各顶点的标号为1～n，边 $\{i, j\}$ 的权 $C(i, j)$ 为0或1，B 为一个正整数。

问：G 中是否存在一个以顶点1为起点的顶点序列 $\pi=<1, v_2, \cdots, v_n>$ 满足条件：

①$1-v_2-\cdots v_n-1$ 构成一条哈密顿回路；

②$f(\pi) \leqslant B$?

其中
$$f(\pi)=1*(n+1)^{n-1}*c'_1+\sum_{i=2}^{n}v_i*(n+1)^{n-i}*c'_i$$

$c'_1, c'_2, \cdots, c'_n$，为 $C(1, v_2), C(v_2, v_3), \cdots, C(v_{n-1}, v_n), C(v_n, 1)$ 的一个排列且满足条件：$c'_1 \leqslant c'_2 \leqslant \cdots \leqslant c'_n$。

为讨论方便，记这个问题为RHC。对于RHC，显然有以下结论成立。

定理 1 RHC∈NPC

证明 显然，RHC∈NP。对于非确定图灵机，只要猜想一个顶点序列并验证 $f(\pi) \leqslant B$ 即可。

其次，RHC∈NPC。因为RHC可由判定问题HC多项式归结得到。这只要给HC问题中的图的顶点编号，令各边的权为1，并取下式即可：

$$B=(n+1)^{n-1}+\sum_{i=2}^{n}a_i(n+1)^{n-i}, \text{其中}, a_i=n-i+2$$

（注：*HC* 为哈密顿图判定问题.）如果将RHC中“问是否存在一个顶点序列 π，其构成‘哈密顿’回路且使 $f(\pi) \leqslant B$”改成“在 G 中寻找一个顶点序列 π 使 $f(\pi)$ 最

小，或在这样的顶点序列不存在的情况下回答 'NO'" 就得到了另一个问题。为方便起见，我们称之为 OPRHC.

定理 2 RHC∈P 当且仅当 OPRHC 是多项式时间可解的。

证明 充分性显然。下证必要性。

设求解 RHC 的一个多项式算法为 Q. 利用 Q 可构造求解 OPRHC 的多项式算法如下：

step1. 令 $B=(n+1)^{n-1}+\sum_{i=2}^{n}a_i(n+1)^{n-i}$，其中 $a_i=n-i+2$。调 Q 判 RHC 有否解。无解则终止。有解转至下步。

step2. 令 $B_{min}=0$. $B_{max}=B$.

step3. 若 $B_{min}=B_{max}$，则终止并据 B 确定满足条件的顶点序列，否则转至下步。

step4. 取 $B=[(1/2)*(B_{min}+B_{max})]$，调 Q 判有否解。若有解 $B_{max}=B$. 否则 $B_{min}=B$.

step5. goto step3.

上述算法必然终止。因为若干步以后，$B_{min}=B_{max}$。step3 中根据 B 确定顶点序列可以这样进行：在图 G 中逐条边考虑是否可去掉。若去掉该边以后，对指定的 B，RHC 仍然有解，则继续考虑下一条边是否可去掉，否则恢复该边再考虑下一条边是否可去掉。这样做的结果是 G 中最终只剩下一条哈密顿回路。由这条回路我们可马上得到顶点序列。

算法的复杂性集中在 step3，step4 和 step5。这三个语句的循环次数由 B 确定。设算法 Q 的时间复杂性为 $p(n)$，则上述算法的时间复杂性为

$$\begin{aligned}&O(\log_2 B*n^2*p(n))\\&=O(\log_2(n*(n+1)^{n-1})*n^2*p(n))\\&=O(\log_2 n)*n^3*p(n))\end{aligned}$$

上述复杂性估计式中的 $\log_2 B$ 为 step3、step4 和 step5 的循环次数，n^2 是 step3 中逐条边考虑去掉时的循环次数，$p(n)$ 是考虑一条边是否可去时调 Q 判 RHC 有否解的时间。

根据定理 2，若 OPRHC 不是多项式时间可解的，则 RHC$not\in p$，于是

$$\text{NPC}\cap(\text{NP}-\text{P})=\text{NPC},\text{即 NP}\neq\text{P}$$

因此，证明了 OPRHC 不是多项式时间可解的也就证明了 NP≠P。下面讨论求解 OPRHC 的复杂性。目标是分析在简单计算模型上，寻找一个对 OPRHC 的任何输入而不是对某些特殊实例都能正确求解的算法的困难性。虽然这种分析或许不是一个严格的证明，但分析结果至少表明，寻找求解 RHC 的多项式算法不是一件容易的事情。

我们以操作涉及的对象的多少和次数来讨论复杂性。这相当于在通用确定图灵机上，带头要移动到表示相应数据的代码段并在其上工作。这是一个算法所必须的工作量。对于求解 OPRHC 的任一算法，以下几点显然是必须满足的：

(1) 算法实质上是从 $\{\pi|\pi$ 是 G 中的顶点序列且 π 构成一条哈密顿回路$\}$ 中寻找 $f(\pi)$最小者，因此，不论方式如何，一个算法必然唯一地确定了一个淘汰_选择机制。该机制以预先确定（由算法确定）的顺序和方式注视（扫描）着顶点序列，并能一步一步地按预先确定（由算法确定）的顺序、方式和操作决定某些序列的保留和另一些序列的

舍弃，直到仅留一条序列。这里的操作包括淘汰_选择操作。虽然我们不知道这些操作具体怎样实现，但是

a. 具有淘汰和选择功能的操作肯定存在，并且这些操作构成了淘汰_选择机制，否则，算法不可能从众多的序列中选出解来。

b. 客观上存在的（当然包括我们所有可能列举的）各种各样的可以在求解过程中帮助决定序列舍弃或保留，并能正确实现序列的淘汰和选择的操作、措施等，都属于淘汰_选择操作。

（2）由于淘汰、选择操作是一个算法必须具备的操作，因此，其中必有基本淘汰_选择操作。所谓基本淘汰_选择操作，这里是指满足以下三个条件的动作或动作群：

a. 该操作确定地以某种方式一次性引起一些序列的保留和另一些序列的舍弃。

b. 该操作包含为确定这些序列的保留和舍弃所需的辅助动作。这些动作的操作对象只能涉及序列已经确定的点，而不能为确定某个或某些序列存在性和大小性的目的涉及未确定的点，并且因此而对随后的淘汰和选择产生影响。

c. 该操作内部不能再分解出淘汰_选择操作。

基本淘汰_选择操作定义中的条件b和条件c是为了保证其基本性。如果为确定某个或某些序列的存在性和大小性涉及了未确定的 点并且由此而对随后的淘汰和选择产生影响，此时实际上已引起了一次淘汰_选择。当然，淘汰_选择操作不排除无目的使用未确定的点。这种情况表明算法有冗余操作。

基本淘汰_选择操作主要强调淘汰_选择的不可分解性。有淘汰_选择操作，当然也就有不可分解的基本淘汰_选择操作。

（3）在由一系列淘汰_选择操作构成的淘汰_选择机制中，有些操作的操作对象涉及到前面的淘汰_选择操作的结果。这些淘汰_选择操作称为非原始淘汰_选择操作，否则称为原始淘汰_选择操作。原始淘汰_选择操作的操作对象直接瞄准图中的顶点序列。任一淘汰_选择机制所确定的淘汰_选择过程应该从原始淘汰_选择操作开始。为简便计，以下简称原始的基本淘汰_选择操作为淘汰_选择操作。

（4）由于输入的对称性，任一顶点序列都可能是要求的解也可能不是要求的解。因此一个正确的算法应具备完备性，即任一顶点序列都应在算法的考虑之内，对于输入的任意一个实例，算法都必须能从 $\{\pi|\pi$ 是 G 中的顶点序列且 π 构成一条哈密顿回路$\}$ 确定出要求的解。

（5）无论是指明对一类序列的舍弃还是保留，都必须以某种确定的方式直接或间接地、显式或隐含地确定地涉及这类序列，否则可认为该操作与该类序列无关。所谓间接涉及，我们是指一个序列因另一序列被涉及而间接地被涉及。

（6）我们可以仅分析任一求解OPRHC的无冗余操作的算法中，由该算法确定的淘汰_选择机制所包含的原始的基本淘汰_选择操作的性质。这并不失一般性。如果算法有冗余操作，去掉这些冗余操作仍可得到一个正确的算法。

从 $\{\pi|\pi$ 是 G 中的顶点序列且 π 构成一条哈密顿回路$\}$ 中寻找 $f(\pi)$ 最小者需要不断淘汰不可能产生解的顶点序列。不论淘汰_选择具体怎样进行，对最终产生的结果必须能确认其最小性。因此，算法必须以某种方式扫描所有的顶点序列并最终认定其余序列

都小于等于最后产生的结果。这样，最终产生的作为解的顶点序列必然在一系列的淘汰_选择操作中与其它序列直接或间接地相遇并淘汰掉其余序列。再次指出，淘汰_选择操作是实质上具备淘汰和选择功能、并事实上引起了一次淘汰和选择过程的动作或动作群，它不一定是比较操作。例如，按语言识别的观点，它也可能是直接以某种方式产生并在产生的同时舍弃其余，等等。客观上存在的（当然包括我们所有可能列举的）各种各样的可以在求解过程中帮助决定序列舍弃或保留，并能正确实现序列的淘汰和选择的操作、措施等，都属于淘汰_选择操作。

下面分三种情况说明这种淘汰_选择过程的复杂性。

1° 一次淘汰_选择操作中，如果被选择的类和被淘汰的类都只确定了不足 1/2 的点并且被确定的点集 S_1 和 S_2 满足条件：

$$S_1 \cap S_2 \neq \varphi \text{ 且 } S_1 \neq S_2 \text{ 且 } S_1 \neq \varphi$$

本次淘汰_选择不可能正确完成。

因为淘汰_选择不一定逐条序列进行，这里使用类的概念。所谓类在这里是指组或集合。其次，正如前面我们指出的，这里的确定包括显式或隐含地确定。

我们首先考虑决定本次淘汰和选择之先必须完成的事情。由于淘汰_选择操作不能涉及未确定的点，要决定淘汰和选择操作怎样进行必须充分利用已经确定的信息来预测和判定 $f(\pi)$ 的变化趋势。这需要确定点在序列中的位置（或顺序），以及由此而来的联结这些点的边及其权。如果进行上述判断只需用到所确定信息的一部分，则说明算法有冗余操作，本次操作之先不产生这些冗余信息即是。又因为任何一条回路可能存在也可能不存在，所以必须预测已经确定的部分点将来发展成解的可能性，至少必须能断定，肯定存在某种方式，使目前已确定的点能联结成一条路径。

其次，我们考虑本次淘汰_选择操作之后必须完成的事情。由于序列并未完全确定，进一步的确定还将进行。根据哈密顿回路点不重复的性质，本次淘汰_选择之后的对序列的确定依赖于本次淘汰_选择之前的对序列的确定。这也需要确定点在序列中的位置（或顺序），以及由此而来的联结这些点的边及其权。

因此，不论以何种编码方式（为了降低计算复杂性，编码必然尽可能简单）来描述解，也不论按什么性质或规则来淘汰序列，以下几点必须满足：

（1）不能脱离输入的图的顶点间的联接关系及序列的大小关系。这将要涉及到图的顶点及联接点的边。因为我们寻求的解，根据问题的定义，仅决定于联接方式与大小关系。

（2）必须支持不可能产生解的序列，即被淘汰的序列从集合 $\{\pi|\pi$ 是 G 中顶点序列且 π 构成一条哈密顿回路$\}$ 中分裂出去。这至少使得被保留的序列必须被确定性地编码表示出来。

（3）如果编码表示的，在一次淘汰_选择操作中被保留的序列集合，在以后的淘汰_选择过程中要分 p 种情况使用，则可认为本次淘汰_选择操作可分解成 p 个淘汰_选择操作。由于这里的淘汰_选择操作为原始的基本淘汰_选择操作，若 p 不囿于 n 的多项式函数，则算法已不是多项式算法。若 p 囿于 n 的多项式函数，则一个原始的基本淘汰_选择操作的 p 次计数，不会从本质上改变算法的计算复杂性量级。

因此可认为，无论以何种方式描述回路，实质上都是显式地或隐含地、直接或间接地以某种方式通过确定点来实现。

最后，我们考虑本次淘汰和选择之先，序列已完全确定的情况。此时，淘汰_选择是逐条序列进行，下面我们将看到，其复杂性更高。

根据哈密顿回路点不重复的性质，注意到 S_1 和 S_2 不全相同也有相同部分因而所确定的边也不全相同，未确定的点和边，即那些将被逐步确定，以便和已经确定的点和边联接成一条哈密顿回路的点和边也可以不全相同。当图的联通度较高时，必存在序列 π_1 和 π_2，使类 A 中含序列 π_1，且不含序列 π_2，类 B 中必含序列 π_2，且不含序列 π_1.

调整各边的权，使已经确定的边的权均等于 1. 对未被确定的边，使 $f(\pi_1)$ 最小。得到实例 E_1.

再调整各边的权，使已经确定的边的权均等于 1。对未被确定的边，使 $f(\pi_2)$ 最小。得到实例 E_2.

E_1 和 E_2 的存在说明，除非能进一步断定 π_1 和 π_2 的存在性，仅根据已经确定的点，不能对 E_1 和 E_2 都能正确求解。

2° 一次淘汰_选择操作中，如果被选择的类和被淘汰的类中，有一类只确定了不足 1/2 的点，本次淘汰_选择操作不可能正确完成。

由前面的讨论，一个算法必须先产生一些已确定了 1/2 以上点的序列以构成淘汰_选择机制。这又产生了两种情况：

(1) 在一次淘汰_选择操作中，被选择的类 A 仅确定了不足 1/2 的点。

一次淘汰_选择操作应同时能使类 A 中不能产生解的序列从类 A 中分裂出去，即导致更多的点被确定。否则，算法不能从众多的序列中挑选出解来。如果类 A 仅确定了不足 1/2 的点，类 B 却确定了超过 1/2 的点，类 A 与类 B 的淘汰_选择不可能实现对 A 的分裂。因为这种操作不可能覆盖对类 A 的所有序列的考虑，这将使算法不具备完备性。

(2) 在一次淘汰_选择操作中，被淘汰的类 A 仅确定了不足 1/2 的点。

由输入的对称性，类 A 中任一序列有时存在，有时不存在。于是类 B 对类 A 的淘汰需要确定任一序列 π 的存在性及 $f(\pi)$ 的大小性，这将使类 A 进一步分细。说明对类 A 的淘汰要分解成一系列淘汰_选择操作。本次操作不是基本淘汰_选择操作。

于是，无论对情况 (1)，还是情况 (2)，都有实例使得除非进一步判断序列类中某些序列的存在性，算法不能正确求解。事实上，因为两类序列已被确定的点数不等，确定点数较少的一类中必含一序列 π，π 不在确定的点数较多的一类中，调整各边的权，使 $f(\pi)$ 最小，得到实例 E_3。对实例 E_3，不进一步判断序列 π 的存在性，便无法决定类 A 和类 B 的舍弃或保留。

3° 淘汰_选择过程中，如果对于所有的类(包括被选择的类和被淘汰的类)，都确定了 1/2 以上的点，则算法至少将从 $\binom{n}{n/2}$ 个不同类中选出最优解。否则，算法将不能覆盖对所有序列的考虑。根据输入的对称性，如果算法不能覆盖对所有序列的考虑，则总能找到一个实例 E_4，使算法对于该实例找不到解。这样的算法不是对所有实例都能正确求解的算法。前面分析过，任一确定的算法都是以预先确定的方式来搜索解，因此最坏

情况下，求解OPRHC的任一正确的算法，将在至少遍历上述（$\binom{n}{n/2}$）个不同类之后，找到问题的解。在意味着不论淘汰_选择操作具体怎样进行，也不论该操作一次可对多少个对象进行运算，在通用图灵机上实现该淘汰_选择过程，输入带带头将至少移动（$\binom{n}{n/2}$）次。因为通用图灵机必须在表示这些类的代码段上工作。显然，这种情况下算法的时间复杂性至少是0（$\binom{n}{n/2}$）。这不是多项式算法。

上面的分析过程中，并未指出淘汰_选择操作具体如何实现，而且分析是在简单模型上进行。简单模型是具有‘简单性’的模型类，而不是一个具体的模型。但是，不论模型上是否提供了淘汰_选择操作，任意算法应该实现淘汰_选择过程。而存在的各种各样的可以在求解过程中帮助决定序列舍弃或保留，并能正确实现序列的淘汰和选择的操作、措施等，都属于淘汰_选择操作。

2 结束语

上面的每一步分析，尽管不是很严格，但我们认为它至少有以下意义：

1. 揭示了在简单计算模型上，寻找一个对满足RHC的任意输入，而不是对某些特殊实例都能正确求解的算法的困难性。

2. 从某种程度上，分解了该问题并指出了研究该问题的今后的努力方向。

算法研究中，寻找实际运行效果好的算法，也是人们努力的一个方向，就很多问题人们找到了一些运行效果很好的算法。按上面的分析，寻找求解NP－完全问题的多项式算法至少非常困难。对NP－完全问题寻找实际运行效果好的算法是一个努力的方向。

文献[6]中介绍了一求解哈密顿图判定问题的高效算法。对2000多例64阶以上图作为输入的实际运行表明，该算法运行很快。虽然，本质上该算法是回溯算法，但极少出现回溯现象。

参 考 文 献

1 Cook S A. The Complexity of Theorem－proving Procedures. Proc 3rd Ann ACM Symp on Theory of Computing, Association for Computing Machinery, New York 1971a. 151～158.

2 Michael R, Garey David S, Johnson. Computers and Intractability, a Guide to the Theory of NP－Completeness. W. H. Freeman and Company , 1979

3 姜新文. 一个构造的NP－完全问题及其复杂性下界猜测. 计算技术与自动化，1991

4 D S Johnson. The NP－completeness Column: An Ongoing Guide, Journal of Algorithms, 1987, 8: 438－448

5 D S Johnson. An Ongoing Guide, Journalof Algorithms, 1987, 8: 285－300

6 姜新文. 简单无向图中H回路搜索的避圈算法. 计算技术与自动化，1991

Analysing the the complexity of A NP—complete problem

Jiang Xinwen Wang Bingshan

(Deparement of computer science, changsha, NUDT 410073)

Abstract

We introduce a NP—complete problem named RHC and prove its NP—complete property in this paper. Secondly, we discuss the complexity any algorithm solving RHC presents by counting the moving times of the tape head on turning machine. The conclution drawn from our analysis shows at least that it is difficult to find an algorithm to solve RHC though the analysis is not a mathematical one. Hence, maybe it needs only time we think, to confirm the analysis.

Key words complexity, algorithm, NP problem, NP—complete problem