**在有界度图中独立集的改进逼近**

Magntús M. Halldórsson, Jaikumar Radhakrishnan

**摘要：**在有界度图中寻找最大独立集是一个被充分研究的NP完全问题。我们研究了两种求近似解的方法，得到了集中改进的性能比。

第一种是在我们之前的论文中介绍的子图移除方案。利用更好的组件算法（component algirithms），我们得到了一个具有性能比的有效方法。然后我们给出了Ajtai等人的定理用于求无团图的独立集个数的一个实现，并利用这一实现在我们的方案中得到了 的性能比。这是第一个的比率。

第二种是Berman和Fürer的局部搜索方法，他们证明了一个很好的性能比率，但需要极多的时间。我们展示了在最大度为的图上，如何大大减少计算需求，同时保持相同的性能比大致在。然后我们展示了他们算法的缩小版本，有着的性能比，在以前合理有效的方法的界限实现了改进。

**1 前言**

图的独立集是一个顶点的集合，它的任意两个顶点都不相邻。在图论和组合优化中，寻找独立集的最大顶点数是一个非常重要的问题。

假设这个问题是NP困难的，那么最有希望的方法就是开发能够找到高质量近似解的启发式算法。算法的性能比的定义是最优解的大小与算法的大小之间的最坏比率。尽管付出了相当大的努力，还是没有一种针对独立集问题的算法的性能比小于[6]，其中n是输入图中的顶点数。近年来关于交互式证明系统的研究结果，在著名的Arora et al的论文[3]中达到了顶峰，表明不能期望有常数近似算法，事实上，的比率是无法到达的[4]。

考虑到一般情况下这个问题的明显难度，很自然地会问，什么限制能使这个问题更容易近似。也许最自然和最频繁发生的情况是最大顶点度以一个常数为界。正如考虑成对元素之间的冲突会联想到独立集(或团)问题，当问题的关键参数固定时，自然会想到有界度的变体。

对有界度版本（B-IS），求精确解仍是NP完全问题，但求近似解却变得相当容易。事实上，任何一个在最大度为Δ的图上寻找最大独立集的算法都有和Δ相关的性能比。这个问题也属于原始的最大SNP完全问题[11]，所以[3]的结果暗示存在一个常量，c近似是NP苦难的，自然意味着没有多项式时间近似模式存在（除非P=NP）。这自然导致我们会寻找B-IS可以近似的最佳常数。

我们通过研究两种最近的方法来处理这个问题: 前者是一种从图中去除小团的算法模式。这个想法起源于[6]，其可以追溯到Erdös[7]——来自于没有小团的图比一般图包含可证明的更大的独立集的观察。此外，这些更大的解可以被有效地找到。对于几乎没有不相交团的图，我们可以手动删除所有的团，并在其他部分找到承诺的改进解。另一方面，具有多个不相交团的图不能包含一个非常大的独立集，从而给出了最优解的上界。因此，无论在哪种情况下，我们的性能比率都将得到提高。

我们先前使用这个模式来改进最小度贪婪算法的逼近，从提升到[8]。这次，通过对Khanna等人的一个简单局部搜索算法[10]进行新的研究，我们得到了一个令人惊讶的强比率，。

我们也用这个模式来回答一个诱人的问题：迄今为止所有的B-IS近似结果仅仅有限改善前面的系统，的性能比有没有可能实现？答案是肯定的，我们提出了一个有性能比的算法。作为关键的一步，我们给出了Ajtai, Erdös, Kolmlós和Szemerédi[1]关于不含小团的系数图的独立集数量的存在定理的确定性实现。

我们考虑的后一种方法是基于Berman 和 Fürer[5]的，它可以被描述为一种局部搜索算法，其在当前解的补充中进行额外的搜索。对于任意固定常数，他们的算法在低最大度图有出色的性能比。当是偶数时为；当是奇数时为。

不幸的是，这个方法花费非常多的时间。局部搜索邻域是搜索一个改进的解集合，它包括所有与当前解的差(或距离)为具有σ个顶点的连通图的解。为了得到它们的结果，我们必须搜索一个大小为的邻域，这意味着搜索复杂度为 。特别地，复杂性对近似的大小的依赖是双指数的。例如，为了在时获得1.6的比率，这种方法需要的时间复杂度。

我们通过收紧分析来解决这个可行性问题，从而大大减少了对邻域大小的要求。特别是，我们得到了相同的性能比的同时，缩小邻域大小为，并消除了对h的指数依赖。虽然仍然不完全实用，但这接近这种方法的极限。

我们还观察到，小的邻域产生了惊人的良好性能比。在只有大小的邻域，可以很容易地在时间内实现并得到的比率。这是以前实用的有界度独立集算法的最佳比率的改进[8]。

**符号。**我们使用相当标准的图术语。 对于问题中的图，通常用表示，表示顶点的数量，表示最大度，表示平均度，表示独立集数目（或最大独立集的大小），表示独立分数（或独立集数目与顶点数的比率）。 对于一个顶点，表示的度数，表示v的邻居集合。对于独立集算法，是该算法在图上得到的解的大小。该问题中算法的性能比被定义为

**3 局部搜索方法**

**3. 1 算法**

本算法基于Berman 和 Fürer[5]的算法，是一种局部搜索方法——但带有变形。局部性，或者说邻域，其自然定义如下: 一个解的集合，它在很少的顶点上与当前解不同(就集合对称差而言)。从任意一个解开始(例如一个极大独立集)，我们搜索所有严格大于它的相邻的解。这个过程会一直持续下去，直到找不到任何改进为止。

我们的变形是这个方法的一个递归应用，其可以作为这个解的补充。一旦找到一个不可改进的解，就把它应用到中，其被定义为与至少两个中节点相邻的诱导子图(当时，一个相邻节点就足够了)。由于这个子图的最大度小于等于2，递归会结束于最优解的二度图。考虑到这个新的更简单的解，如果它比原来的解更大，我们就令其成为我们当前的解，并继续尝试改进它; 否则，我们退出并宣布在这个搜索过程中的最简单的解决方案。

LS()

if () return 最大独立集()

最大独立集()

repeat

尝试所有的-改进

if ()

until (找不到改进)

end

释义：在每个步骤中，我们都要寻找一个改进，它是一个与当前解的对称差较大的顶点集，且仍然保持独立。一个-改进从当前解中增加个顶点并删除个顶点。容易发现它能够寻找一个诱导连通图来改进，并且因为最大度被限制为，最多有个大小为的连通子图。因此用表示的局部搜索区域大小是影响复杂度的关键因素。我们说一个解是-最优（)的，如果它在上没有-改进或普通的改进，并称找到这个解的算法为。

**3. 2 分析工具**

我们分析了任意-最优解的保证大小。设 是这样一个解，是一个假设的最优解，中顶点的带子（lace）定义为中相邻顶点的集合。分析的要点是表明少数顶点可以有一个小的带子，从而限制了的大小，因为的邻接容量有限，这是由度约束导致的。

**符号。**令表示和的交集。让表示的顶点的子集，其带子是单位大小的（也就是与的恰好一个顶点相邻），并令表示中和相邻的顶点。令，。我们进一步将分成和，即带子大小分别为二，和三或更多。

最后，用上述集合各自的小写字母表示基数。用表示该算法在最大度为的图上的性能比。

**简单限制。**带子尺寸的总和最多可以达到A的容量。这给了我们第一个关键的限制。

(6)

第二个限制基于的大小。根据定义，。从另一个角度，中与的两个顶点相邻的顶点形成了一个2-改进。因此下列等式在2-opt下是严格的。

(7)

我们也可以以这个形式表示：。

**对数的限制。**根据引用文章[5]的引理3.3，-opt证明

(8)

通过构造一个图来说明这一点，其顶点集为 ，其边由中的带子节点构成。至少有一个带子的一个元素在内；只有一个端点在内的带子被建模为在那个节点上的一个自环。如果所得到的子图节点的边数超过了上述限制，则可以证明存在一个诱导子图有最多个节点，最多包含严格大于点数的边数。由于图可能被假定是连通的，所以它最多包含两个自环。这可以映射到中最多个节点的集合，中的2个节点和中相应的一组节点来说，它们共同构成了一个改进。

**递归得到的限制。**递归程序为我们提供了关于的限制。如果找不到更大的解，上的最大独立集就不能大于-度图的性能比率的因子。但是，这个补集，A中至少与两个顶点相邻的顶点集，闭幕包含（及，当时）。因此我们可得：

(9)

(10)

**衍生。**我们现在介绍了分割和派生的概念，类似于[5，lemma 3.5]的理念。从图分割和中除去，又分成两部分: 。这个过程可以继续，产生第二个和第三个衍生等等。

我们需要把‘单元’集合等等的大小联系起来。中的每个顶点都和中某个特定顶点相邻。此外，它必须与中的某个顶点相邻，否则它就属于。中的每个顶点都和中某个顶点相邻，因此最多有个顶点属于。由此，我们有一个有用的限制:

(11)

**3. 3 应用工具**

**快局部搜索。**我们首先给-opt的的性能比做一个简单证明，这是借助之前已知的限制通过使用方法的一个改进。

将(7)和(6)相加并使用衍生：

(12)

对，我们将半个(12)，半个（9）和一个(7)相加，得到：

通常的，我们将个(12)，个(9)和一个(7)相加：

这限制于，并略微比小偶数值的好。

**慢局部搜索，更快。**现在让我们推导出与[5]相同的性能界限.

将(6),(8)和两个(7)相加生成：

经过s次衍生得到

这意味着使用(7)和(11)衍生，我们得到

因此，只要，就有，我们得到基本的不等式：

(13)

我们需要次衍生，得到一个改进需要

现在对性能比证明的分析根据[5,sec.4]的论点来进行。将个(13)，个(9)（若，使用(10)），和一个(7)来得到循环：

产生所需的比率当为偶数（奇数）时

得到结论：

**定理9** LS在时间内得到的性能比，和在时间内得到

**参考文献**