阅读报告

——基于“A Parallel Framework for Constraint-Based Bayesian Network Learning via Markov Blanket Discovery“

1 相关背景

文章背景涉及到的主要是主要是贝叶斯网络学习框架。

贝叶斯网络是一种常用在对多个变量之间的关系进行建模的概率图模型。它可以用来表示变量之间的依赖关系，并且可以通过概率计算来推断变量之间的关系。近年来，基于并行计算的贝叶斯网络学习方法得到了广泛关注。其中，局部到全局约束方法是一种基于约束的贝叶斯网络学习方法，它将贝叶斯网络的学习过程分解为多个局部学习任务（，并将它们组合起来得到全局的贝叶斯网络即，将任务分成多分，进而利用并行计算）。这种方法可以提高贝叶斯网络的学习效率，同时也可以充分利用现代计算机的多核处理能力和分布式计算能力。

文章指出：全局搜索方法遍历DAG的全局空间以确定最佳结构，而局部到全局方法则首先发现每个变量的局部邻域，然后组合这些局部邻域以获得全局结构。启发式方法的并行化可以提高学习效率，但需要考虑负载均衡的问题。在基于约束的方法中，PC算法是一个重要的算法，已经有多种并行化策略被提出。局部到全局约束方法的并行化也已经被尝试，但需要解决负载均衡的问题。

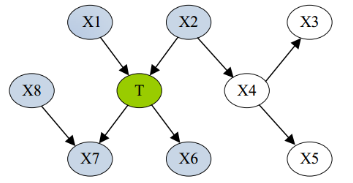
2 问题是什么？

贝叶斯网络 （BN） 是机器学习中广泛使用的图形模型。但学习BN的结构是NP困难的，其学习过程需要大量的计算资源与时间，因此高性能计算方法对于构建大规模网络是必要的。传统的贝叶斯网络学习方法通常是串行的，不能充分利用现代计算机的多核处理能力和分布式计算能力。因此，需要一种并行的贝叶斯网络学习方法来提高学习效率。局部到全局约束方法是一种基于约束的贝叶斯网络学习方法，它将贝叶斯网络的学习过程分解为多个局部学习任务，并将它们组合起来得到全局的贝叶斯网络。这种方法可以提高贝叶斯网络的学习效率，同时也可以充分利用现代计算机的多核处理能力和分布式计算能力。因此，问题是如何提高贝叶斯网络的学习效率，并如何使用局部到全局约束方法来解决这个问题。

局部到全局约束方法是一种基于约束的贝叶斯网络学习方法，它将贝叶斯网络的学习过程分解为多个局部学习任务，并将它们组合起来得到全局的贝叶斯网络。在这个过程中，每个变量的马尔可夫毯被用来确定该变量的父节点和子节点。因此，马尔可夫毯在局部到全局约束方法中起着重要的作用，它可以帮助我们确定每个变量的父节点和子节点，从而构建贝叶斯网络的结构，并提高贝叶斯网络的学习效率。

马尔可夫毯是贝叶斯网络中的一个重要概念，它指的是一个变量的所有父节点、所有子节点以及所有子节点的父节点。马尔可夫毯可以用来表示一个变量与其他变量之间的条件独立性，即给定马尔可夫毯中的变量，该变量与其他变量之间是条件独立的。在贝叶斯网络的学习过程中，马尔可夫毯是一个重要的概念，因为它可以帮助我们确定每个变量的父节点和子节点，从而构建贝叶斯网络的结构。

在可信的贝叶斯网络中，一个结点的马尔可夫毯即为该结点的父结子结点以及子结点的父结点。在如下图所示的贝叶斯网络中，结点X1、X2通过一条有向边指向结点T，那么X1，X2为结点T的父结点。结点T通过一条有向边指向结点X6、X7，那么X6、X7为T的子结点。由于T与其父结点、子结点通过一条有向边相连，那么也称T与这些结点是邻接的。T与结点X3、X4、X5、X8之间没有有向边相连，那么也称T与这些结点是不邻接的。结点X8通过一条有向边指向T的子结点X7，即T与X8有共同的子结点。那么称X8为T的配偶。结点T的马尔可夫毯由T的父结点X1、X2，T的子结点X6、X7，T的配偶X8组成，记为MB（T）={X1、X2、X6、X7、X8}.



3、现有解决方案

文章展示解决方案主要在SECTION III.Parallel Algorithm以及SECTION IV.Implementation。其中第三节是理论层面上的解决方案，第四节则落到了C++ MPI上进行框架的实现。

第三节提出了一个并行框架，用于加速贝叶斯网络的学习过程。单处理器上贝叶斯网络学习算法的四个阶段为：生长、收缩、对称矫正和构造PC（来自MB集）。然后，文章详细描述了三种学习算法（GS、IAMB和Inter-IAMB）的具体实现，包括候选集的选择和分数的更新等。拿GS算法举例，候选变量的选择是基于GS检验的p值。具体地，对于每个变量T，GS算法会按照变量列表的顺序依次考虑每个候选变量Z。对于每个候选变量Z，GS算法会计算I(T, Z|MB(T))的p值，然后选择第一个p值小于阈值α的候选变量Z。这个候选变量Z会被添加到MB(T)中，并从候选变量列表中删除。符号化的话，则可概括为T会被选取，当满足下面两个条件之一：



（1）



（2）

在这两种情况下，如果找到这样的元组，则 Z 被添加到MB(T)，并且⟨T，Z，⟩将从 c-scores列表中删除。在收缩阶段，我们检查所有 集并从中删掉来自MB(T)的X值（ 如果 I(T, X|MB(T)∖{X}) 中包含的话）。

在增长和收缩阶段的一次或多次迭代之后，MB 构造进入对称校正，其中我们验证是否T∈MB(Y)⇔Y∈MB(T)。当此断言对于（T,Y）对失败时，会从相应的 集合中删除有问题的变量。最后，PC通过验证每个子集的 CI 来学习MB集。

学习算法的四个阶段是我认为的作者的创新点，因此这部分会放在第四节。

下面是具体实现部分。作者团队基于GS算法开展对贝叶斯网络学习的优化。GS算法需要进行CI检验来确定变量之间的关系。在这一点上可以使用基于列联表的计数策略（Counting Strategies）来进行CI检验。具体来说，对于每个变量对(X,Y)和给定的变量集合Z，需要计算四个计数值：sabc，sac，sbc和sc。这些计数值可以用来计算G^2统计量并进行CI检验：

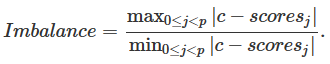


（3）

并且在测试的不同技术策略中，基于列联表的方法在实验中表现最好。

在优化GS算法这一点上，在每次Grow阶段的更新中，只计算第一个得分大于等于阈值的候选变量可以减少计算量。具体来说，对于每个目标变量X，算法会计算与X相关的所有候选变量的得分，并将它们存储在一个c-scores列表中。一旦找到第一个得分大于等于阈值的候选变量，就可以停止更新c-scores列表。这是因为根据公式（2），该候选变量将被选为MB(X)中的变量。

在如何解决负载不平衡的问题上，作者使用了一个指标来衡量负载不平衡的程度：



（4）

如果这个比率超过了一个用户指定的阈值，就重新分配剩余的候选变量，以平衡负载。可以使用MPIR库中的函数来实现这个功能。文中指出，将阈值设置为2.0可以获得最佳性能。

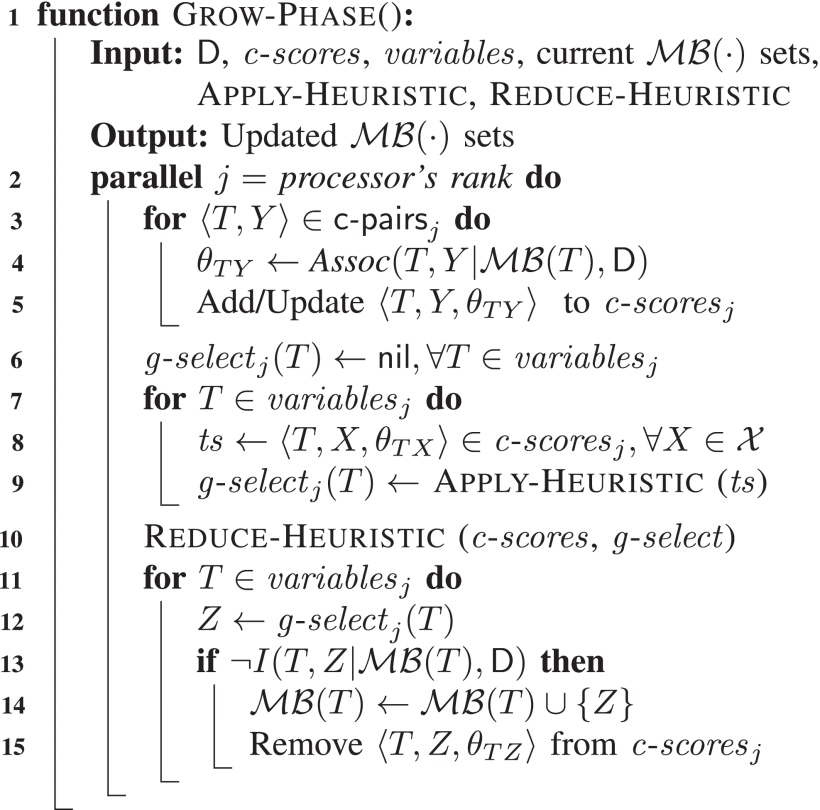
总之，通过使用不同的计数策略、优化策略和重新分配策略可以实现并行贝叶斯网络学习算法。这些策略包括使用基于列联表的计数策略、优化GS算法以减少计算量，并使用重新分配策略来平衡负载。

4、作者的核心思想、创新点是在哪里

第三节C并行框架组成部分详细介绍了生长、收缩、对称矫正和构造PC的工作，这四部分也是并行实现贝叶斯网络学习的关键。

1. Grow phase

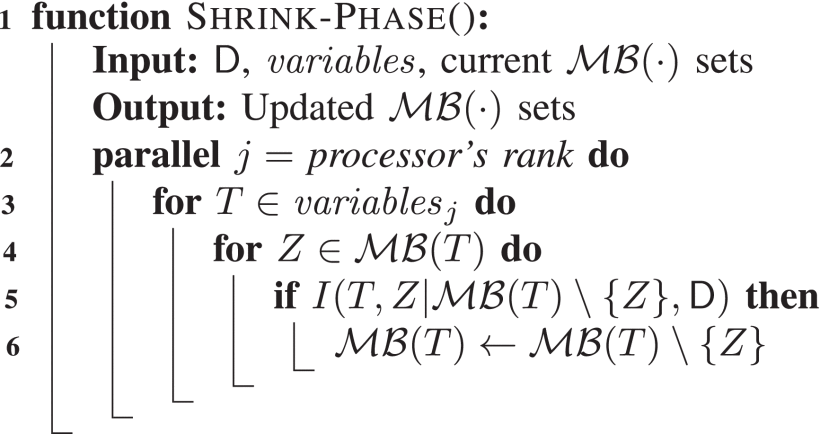
Grow phase有两个关键部分：(i)构建骨架需要所有变量的MB集合，而对于一个变量的MB集合的添加，所有其他变量都被视为候选变量。(ii)进行CI检验（或计算Assoc(⋅)）所需的时间与条件集大小成正比。算法伪代码如下所示：



在并行Grow算法的伪代码中包括两个函数：APPLY-HEURISTIC和REDUCE-HEURISTIC。APPLY-HEURISTIC函数接受一个c-scores列表的切片，该列表对应于一个变量T，其中包含所有候选变量X的⟨T,X,⟩，并返回最适合添加到MB(T)的候选变量。例如，对于GS算法，根据公式2选择候选变量。REDUCE-HEURISTIC函数将所有处理器的变量选择结果累积起来，以确定对于每个变量T，最佳的候选变量添加到其MB(T)中。

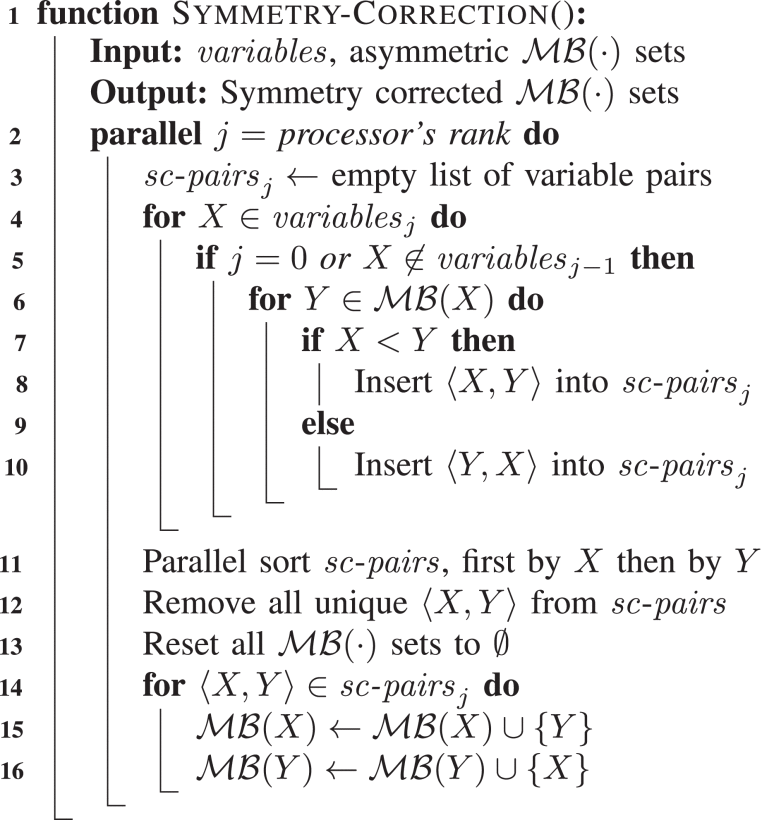
1. Shrink Phase

在Shrink Phase中，唯一的任务是删除在给定其余MB的情况下独立的MI3中的变量，这是通过循环遍历所有MI3集合来完成的（第3-6行）。并行Shrink阶段的运行时间与处理器上所有变量的MB集合大小成正比，这个时间复杂度被限制在n×O(np)=O(n2p)。这个组件不需要通信，只完成收缩任务。



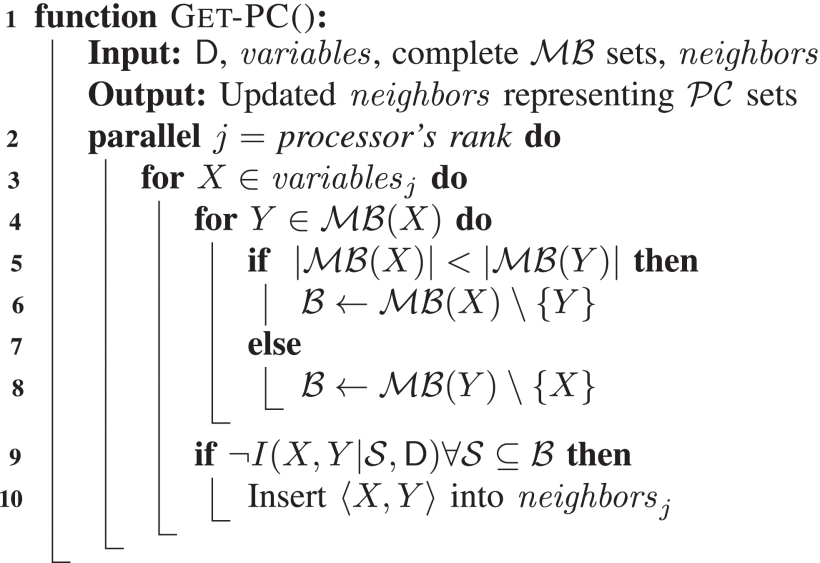
1. Symmetry Correction

这部分介绍了作者提出的并行MI3集合对称性检查算法的伪代码。在算法3中，首先为每个MI3集合成员创建一个有序元组列表sc-pairs（第3-10行），然后进行并行排序以识别不对称的MI3成员（第11-12行）。然后更新MI3集合以反映对称性修正（第13-16行）。构建sc-pairs、删除唯一元组和更新MI3集合的时间复杂度被限制在O()。并行排序可以通过任何基于比较的排序算法实现，例如并行比特排序，它需要O()和O()的计算和通信时间。在删除唯一元组时，还需要集体通信来识别跨处理器边界的元组对。这可以通过一对移位置换来完成，需要O(τ+μ)的时间。总之，这个组件的运行时间主要由并行排序的运行时间决定，它需要O()的计算时间和O()的通信时间。



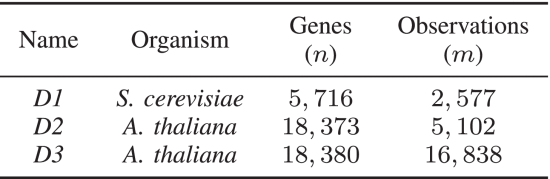
1. Construct PC from MB

这部分介绍了作者提出的并行构建BN骨架的算法的伪代码。在算法4中，这个组件尝试为MB(X)中的每个元素Y识别一个条件集，该条件集可以使X在给定Y的情况下条件独立（第3-10行）。如果对于X,Y这对变量找不到这样的条件集，算法将⟨X,Y⟩插入到neighbors中，这是一个分布式的元组列表（第10行）。请注意，这个组件需要X和Y的完整MI3集合，因此在调用GET-PC()之前，所有处理器上都应该提供完整的MI3集合。在最坏的情况下，这个组件的运行时间复杂度为O()，其中。由于实际网络中r的最大值很低（在我们的实验中小于6），因此相应的指数项可以视为常数。因此，这个阶段在实践中需要的时间最短（少于0.1%）。这个组件不需要集体通信。

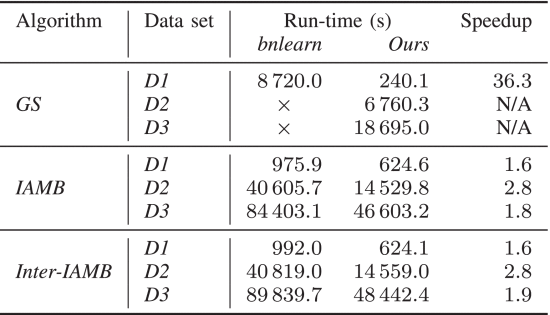


5、通过什么样的实验进行验证

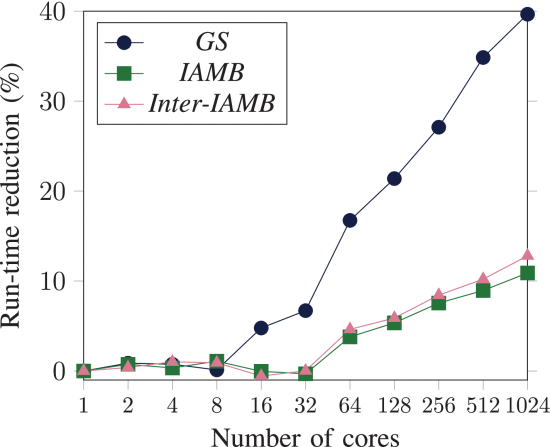
在实验之前首先要准备实验所用的数据集。作者选择了基因网络构建作为应用领域，因为这是一个具有大规模数据集和构建大规模网络需求的丰富应用领域。在实验中选择了三个不同大小的真实基因表达数据集和三个模拟数据集用于研究算法的性能和可扩展性。



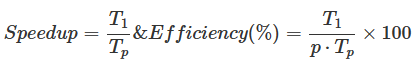
1. 关于数据预处理，作者对数据集进行离散化，并使用了0.05的显著性阈值进行BN结构学习，然后利用这些数据生成三个具有不同边密度的随机有向无环图（DAG），并使用这些DAG生成了三个模拟数据集。这些数据集以纯文本格式存储在GPFS存储器上，可以从集群中的所有节点访问。
2. 在下一阶段，作者进行了对比实验，比较了bnlearn软件包和作者自己优化的顺序实现在三个基因表达数据集上学习BN结构的运行时间。作者使用了Dl、D2和D3三个真实基因表达数据集，以及Sl、S2和S3三个模拟数据集。作者使用了三种算法：IAMB、Inter-IAMB和GS。作者发现，bnlearn的GS算法实现比其他两种算法慢了近一个数量级，而且在两个较大的数据集上无法完成学习。作者还发现，他们自己的GS算法实现比其他两种算法快2-3倍。作者还发现了bnlearn实现中的一个bug，并对其进行了修复。作者使用修复后的bnlearn实现进行了验证，发现他们自己的实现和修复后的bnlearn实现在学习BN结构时具有相似的性能。作者还比较了bnlearn在多核环境下的可扩展性，发现随着核数的增加，bnlearn的性能提升逐渐减弱。作者最后介绍了他们自己实现的算法在可扩展性方面的优势。下面是实验结果的具体数据：



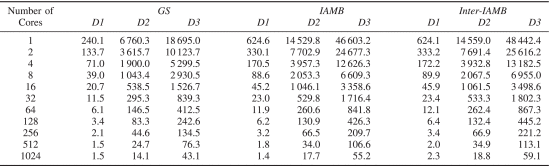
1. 下一阶段作者详细研究了负载平衡。作者在没有应用负载平衡策略的情况下，使用三种算法从数据集D2中学习BN结构，并记录了每个迭代结束时的负载不平衡情况。在少于16个核的情况下，负载不平衡保持在1左右。但是，当算法在更多的核上执行时，负载不平衡会增加，因为越来越多的进程在算法进行过程中空闲。此外，随着算法迭代次数的增加，负载不平衡也会增加。作者还通过实验验证了应用负载平衡策略对于算法运行时间的影响。作者发现，在少数核上运行时，负载平衡几乎没有改善运行时间。但是在多核上运行时，所有的算法效率都显著提高，其中GS算法在1024个核上的改善接近40%。这是因为GS算法的优化使得候选变量的选择更快，因此该算法更容易从更好的工作负载分布中获益。

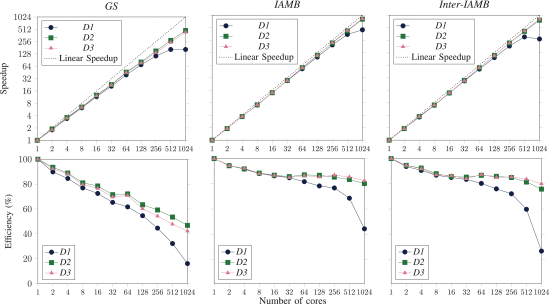


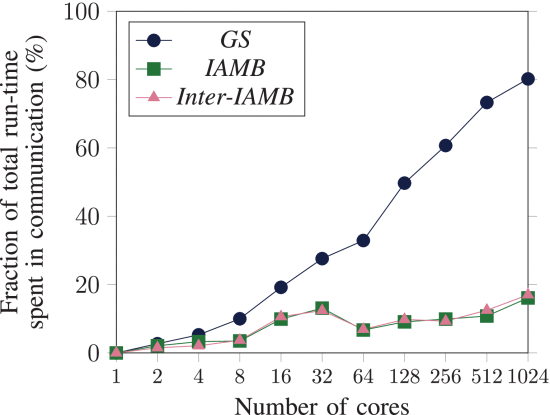
1. 在强扩展性实验中，作者使用了Dl、D2和D3三个数据集，并使用三种算法：IAMB、Inter-IAMB和GS。作者发现，在D2和D3数据集上，所有三种算法的实现在最多1024个核上都表现出了近线性的扩展性，其加速比如下所示：



而在Dl数据集上，扩展性在256个核以上就开始变差。作者还发现，GS算法在较大的核数上的效率较低，这是因为GS算法的优化减少了算法所需的总工作量，从而导致每个处理器的计算负载降低，通信时间占据了算法的运行时间。在强扩展性实验中，并行算法能够在1024个核上学习BN结构，并将顺序运行时间从数小时缩短到不到一分钟，同时在超过1024个核上也展示了可扩展性。







1. 对你的启发

从文章专业性的角度看，我初步了解了在BN学习的过程中如何依靠并行算法来提高效率，以及优化算法的过程。并行算法是一种可以显著缩短学习BN结构时间的方法，特别是在大规模数据集上。通过将计算任务分配给多个处理器，可以加速学习BN结构的过程，从而将运行时间缩短到几分钟或几秒钟，这一算法需要处理器之间进行通信，以便协调计算任务和收集计算结果。而通信过程虽然优化了BN学习的效率，但同样可能成为其瓶颈，因此需要进行优化，例如减少通信次数、减少通信数据量等。此外，不同的算法（GS\IAMB\INTER-IAMN）在不同的数据集（D1\D2\D3）和核数下表现不同，需要根据具体情况选择合适的算法。优化算法的工作负载分配可以确保每个处理器都有足够的计算任务，从而避免某些处理器处于空闲状态(负载平衡)。这可以提高算法的可扩展性，使其能够在更多的处理器上运行。

参考文献：

[马尔可夫毯（Markov Blanket） - 夕月一弯 - 博客园 (cnblogs.com)](https://www.cnblogs.com/wt869054461/p/9899929.html)

[A Parallel Framework for Constraint-Based Bayesian Network Learning via Markov Blanket Discovery | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/9355320)