基于双重约束的最优BN结构学习算法

陈艺薇[1],邸若海[1*],王 鹏[1],张新兰[2],张 欢[2],许 文[2]
(1. 西安工业大学电子信息工程学院,陕西西安 710021;2. 航天材料及工艺研究所,北京 100076)

摘 要: 针对现有基于动态规划的贝叶斯网络结构学习算法复杂度高、无法在合理时间内学习大规模网络的问题,提出基于双重约束的最优贝叶斯网络(Bayesian Network,BN)结构学习算法. 首先,利用最大信息系数和马尔科夫毯限制条件独立性(Conditional Independence,CI)测试的候选节点集合和约束集,得到邻居节点集合;其次,利用邻居节点集合约束父节点图的搜索过程,得到候选父节点集合,从候选父节点集合中取出每个节点的最优父集构造初始有向图;再次,利用Tarjan算法计算初始有向图中的强连通分量,得到节点块序;最后,利用节点块序约束节点序图的搜索过程,获得最优的BN结构. 实验表明,相比于现有的5种基于动态规划的结构学习算法,本文提出的算法在精度稍微降低的前提下,极大幅度提高了算法的学习效率,如Sachs网络,本文提出的算法相对DPCMB(Dynamic Programming Constrained with Markov Blanket)算法降低了40.3%的时耗,算法精度下降了12.1%.

关键词: 贝叶斯网络;最大信息系数;条件独立性测试;马尔科夫毯

基金项目: 西安市科技计划项目(No.2023JH-QCYJQ-0086);陕西省杰出青年科学基金(No.2024JC-JCQN-57);2023陕西省高校工程研究中心项目(No.202301);陕西省电子设备智能测试与可靠性评估工程技术研究中心项目(No.2023-ZC-GCZX-0047);2022陕西省高校青年创新团队(No.202201);陕西省秦创原“科学家+工程师”队伍建设项目(No.2023KXJ-026)

中图分类号: TP181 **文献标识码:** A **文章编号:** 0372-2112(2024)07-2477-14

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230268

Optimal BN Structure Learning Algorithm Based on Double Constraints

CHEN Yi-wei[1], DI Ruo-hai[1*], WANG Peng[1], ZHANG Xin-lan[2], ZHANG Huan[2], XU Wen[2]
(1. *School of Electronics and Information Engineering, Xi' an Technological University, Xi' an, Shannxi* 710021, *China*;
2. *Aerospace Research Institute of Materials & Processing Technology, Beijing* 100076, *China*)

Abstract: Aiming at the problem that existing Bayesian network (BN) structure learning algorithms based on Dynamic programming are too complex to learn large-scale networks within a reasonable time, a Bayesian network structure learning algorithm based on double constraints is proposed. Firstly, the set of neighbor nodes is obtained by using the set of candidate nodes and constraint set for conditional independence (CI) tests based on the maximum information coefficient and Markov blanket. Secondly, the neighbor node set is used to constrain the search process of the parent node graph, so as to obtain the candidate parent node set. On this basis, the optimal parent set of each node is extracted from the candidate parent node set to construct the initial directed graph. Thirdly, the strongly connected components of the initial digraph are calculated using the Tarjan algorithm to get the node block order. Finally, the optimal BN structure is obtained by using node block order to constrain the search process of node order graph. Experiments show that, compared with the existing five structural learning algorithms based on dynamic programming, the algorithm proposed in this paper greatly improves the learning efficiency of the algorithm under the premise of slightly reduced accuracy. For Sachs network, the proposed algorithm reduces the time consumption by 40.3% and the accuracy by 12.1% compared with DPCMB (Dynamic Programming Constrained with Markov Blanket) algorithm.

Key words: Bayesian network; maximum information coefficient; conditional independence test; Markov blanket

Foundation Item(s): Xi' an Science and Technology Plan Project (No.2023JH-QCYJQ-0086); Outstanding Youth Science Fund Project of Shaanxi Province (No.2024JC-JCQN-57); Engineering Research Center Project of Shaanxi Universities in 2023 (No.202301); Electronic Equipment Intelligent Testing and Reliability Evaluation Engineering Technology

收稿日期:2023-03-23;修回日期:2023-08-06;责任编辑:梅志强

*通讯作者:邸若海

Research Center Project of Shaanxi Province (No.2023-ZC-GCZX-0047); Youth Innovation Team of Shaanxi Universities in 2022 (No.202201); "Scientist + Engineer" Team of Qinchuangyuan in Shaanxi Province (No.2023KXJ-026)

1 引言

贝叶斯网络(Bayesian Network,BN)是概率论与图论知识的结合,具备概率统计和数据分析能力,以图形化的方式表示变量间的因果关系,可以有效降低概率推理的复杂度,因此,BN成为处理不确定性问题的重要理论模型.近年来,贝叶斯网络成功地应用在疾病诊断[1]、人脸检测[2,3]、网络安全[4,5]、故障诊断[6,7]、因果分析[8]、可靠性分析[9,10]、风险评估[11]等领域.

BN学习分为参数学习和结构学习.由于构建正确的网络结构是进行参数学习的前提,故BN结构学习是BN学习研究的重点.BN结构学习是通过已知数据样本确定1个最符合数据样本变量之间连接关系的有向无环图(Directed Acyclic Graph,DAG),从有限的观测数据中学习最优的BN结构已被确认是1个非确定性多项式(Non-deterministic Polynomial,NP)问题[12].贝叶斯网络结构学习算法共分为3类:基于约束的方法、基于评分搜索(Score and Search,SS)的方法和混合搜索方法[13].其中,基于SS的方法包括近似搜索算法和精确搜索算法.近似搜索算法只能学习到接近于全局最优的BN结构[14],并不能获得最优的BN结构,但近似算法的效率较高;精确搜索算法可以得到最优的BN结构,但效率低于近似算法.精确算法分为动态规划[15](Dynamic Programming,DP)、整数线性规划[16]和分支定界法[17]等.在上述的精确算法中,本文主要研究基于DP的结构学习算法.

在DP算法中增加各种约束,可大幅度提高DP算法的效率.2018年,文献[18]提出了融合先验知识的DP算法,该方法通过边先验和路径先验对动态规划的搜索过程进行约束,可降低DP算法的时间和空间复杂度,但该方法的精度受到先验正确性影响.2019年,文献[19]提出了基于马尔科夫毯约束的DP算法,该方法通过马尔科夫毯约束DP算法的父节点图,提高算法效率,但算法精度取决于重要性阈值ε.文献[20]提出了基于分层思想的DP算法,该方法通过CI测试对节点进行分层,可降低DP算法的时耗,但该方法的精度依赖于CI测试的正确性.2020年,文献[21]提出了基于改进K均值分块的DP算法,首先,利用改进的K均值对网络分块;其次,利用DP算法学习分块后的结构;最后,将子图结构合并为完整的BN结构.该算法在一定范围内解决了DP算法无法在合理时间内处理大规模网络的问题.

综上所述,在DP算法中加入各种约束可降低算法的评分计算次数,降低算法的复杂度.但上述提出的方法仍具有存储空间容量大及运行时间长的问题.本文提出了基于双重约束的最优BN结构学习算法(Optimal BN structure Learning Algorithm based on Double Constraints,OLADC).首先,利用邻居节点集合约束父节点图的搜索过程,得到候选父节点集合;其次,从候选父节点集合中取出每个节点的最优父集构造初始有向图;再次,通过Tarjan算法计算初始有向图中的强连通分量(Strongly Connected Components,SCC),得到节点块序;最后,利用节点块序约束节点序图的搜索过程,从而得到最优的BN结构.本文提出的算法可降低DP算法的时耗,同时使DP算法在合理时间内处理大规模网络.

2 贝叶斯网络定义

2.1 贝叶斯网络定义

贝叶斯网络表示为$\mathrm{BN}=\langle G,\Theta\rangle$,$G=(V,E)$代表了DAG,其中,$V=\{x_1,x_2,\cdots,x_n\}$为随机变量;$E$为有向边集,代表了节点间的因果关系.$x_i$和$x_j$是DAG中的2个节点变量,$x_i\to x_j$代表节点$x_i$到节点$x_j$有1条有向边,其中,$x_i$为$x_j$的父节点,$x_j$为$x_i$的子节点.$\mathrm{Pa}(x_i)$为节点$x_i$的所有父节点集合,$\mathrm{Ch}(x_i)$为节点$x_i$的所有子节点集合.若1个节点没有任何的父节点,称该节点为根节点;若1个节点没有任何的子节点,称该节点为叶节点.

定义1 节点序[22]:1个节点序o是指一些变量的线性排列,$x_i\prec x_j$表示x_i排在x_j的前面.本文使用的符号及其代表的具体意义如表1所示.

2.2 基于动态规划BN结构学习算法

由于BN结构的无环性,得到的最优BN结构中至

表1 符号代表的意义

符号	表示意义
x,y,v	节点
n	网络节点个数
p,q	网格分割数
$\mathrm{MIC}(x_i,x_j)$	节点x_i和节点x_j之间的MIC
$\mathrm{MB}(x)$	节点x的马尔科夫毯
$\mathrm{nb\_cand}(x)$	节点x的候选节点集合
$x\perp y\|Z$	给定Z,x和y条件独立
$\mathrm{Sep\_cand}(x,y)$	节点x和y的约束集
$\mathrm{nb}(x)$	节点x的邻居节点集合
$\|\cdot\|$	集合的大小
$C_1,C_2,\cdots,C_m$	节点块
$\mathrm{order}=\{C_1,C_2,\cdots,C_m\}$	节点块序

少有1个节点为叶节点. 采用动态规划学习BN结构时,BN结构中至少含有1个叶节点,且算法所用的评分是可分解的. 动态规划的状态转移方程为

$$\max \text{Score}(V)=\max_{x\in V}\left\{\begin{matrix}\max \text{Score}(V\backslash x)\\ +\max \text{Score}(x,V\backslash x)\end{matrix}\right\} \tag{1}$$

$$\begin{aligned}\text{BestScore}(x,V\backslash x)&=\max \text{Score}(x,V\backslash x)\\&=\max_{\text{Pa}(x)\in V\backslash\{x\}}\text{Score}(x,\text{Pa}(x))\end{aligned} \tag{2}$$

其中,x是最优结构中的1个叶节点;Score(·)为可分解的评分函数[23]. 动态规划通过式(1)和式(2)将寻找最优网络结构与寻找最优子网络结构相结合,算法思想为:从空集出发寻找单个节点的最佳网络结构,逐步加入叶节点,直到找到所有节点的最优BN结构. 动态规划求解贝叶斯网络结构的整个过程可由每个节点的父节点图和节点序图表示.

3　基于双重约束的最优BN结构学习算法

在引出基于双重约束的最优BN结构学习算法之前,需要给出最大信息系数和马尔科夫毯的相关概念. Reshef[24]提出了最大信息系数(Maximal Information Coefficient,MIC)的概念,MIC值可以衡量节点x_i和x_j之间的依赖程度,它的值域在0~1之间,值越高表示依赖关系越强. MIC的计算是基于互信息(Mutual Information,MI)和网格划分的思想. 节点间的MIC值越大,代表节点间依赖关系越强,在BN结构中表示节点间可能存在间接或者直接的连接关系;节点间的MIC值越小,代表节点间依赖关系越弱,在BN结构中表示节点间存在连接关系的可能性较小.

定义2　互信息(MI):2个变量x_i和x_j之间的MI可以定义为

$$I(x_i,x_j)=\sum_{x_i\in V}\sum_{x_j\in V}p(x_i,x_j)\log\left(\frac{p(x_i,x_j)}{p(x_i)p(x_j)}\right) \tag{3}$$

将坐标平面进行p行q列的网格划分,通过调整p和q的值,得到不同的MI值,将其中最大的MI值记为$I^*(D,x_i,x_j)$. 为了将不同维数下的数据集进行比较,需将该值归一化. $m_{p,q}$被定义为由任何$p\times q$网格实现的最大归一化MI,定义如下:

$$m_{p,q}=\frac{I^*(D,x_i,x_j)}{\log\min(p,q)} \tag{4}$$

定义3　最大信息系数(MIC):给定数据集D,2个变量x_i和x_j之间的MIC定义为所有可能的$p\times q$网格上$m_{p,q}$的最大值,其中,$p,q<B$,B是样本大小n的函数. 最大信息系数可以通过式(5)计算:

$$\text{MIC}(x_i,x_j)=\max_{p,q<B}\{m_{p,q}\} \tag{5}$$

其中,B是1个变量,$B\approx n^{0.6}$. 此外,由于MI的对称特性,MIC也是对称的,即$\text{MIC}(x_i,x_j)=\text{MIC}(x_j,x_i)$.

考虑到仅使用式(5)没有办法判断节点对之间是否有连接边存在,因此,引入了约束阈值α_{MIC}. 利用式(6)的限制条件,判断节点对之间是否存在连接边,其表达式如下:

$$\begin{aligned}\text{MIC}(x_i,x_j)&<\alpha_{\text{MIC}}\text{MMIC}(x_i)\\ \text{MIC}(x_i,x_j)&<\alpha_{\text{MIC}}\text{MMIC}(x_j)\end{aligned} \tag{6}$$

其中,$\text{MMIC}(x_i)$,$\text{MMIC}(x_j)$分别表示节点x_i和x_j与其他节点间的最大MIC值;α_{MIC}是判断网络中节点间是否有连接边的约束阈值,若节点间的MIC值满足式(6),则证明节点间的依赖关系较弱,应从完全无向图中删除该节点间的连接边. 当式(6)中的约束阈值α_{MIC}取值较大时,许多节点间的依赖关系较弱,完全无向图中应删除的连接边较多,得到无向图中存在的连接边较少;α_{MIC}取值较小时,许多节点间的依赖关系较强,完全无向图中应删除的连接边较少,得到无向图中存在的连接边较多,即无向图包含标准网络中大部分连接边. 为保证得到的无向图包含标准网络中更多的连接边,α_{MIC}=0.1[25].

定义4　马尔科夫毯(MB(x)):MB(x)直观表现为x的父子节点集与x的配偶节点集,表达式如下:

$$\text{MB}(x)=\text{Pa}(x)\cup\text{Ch}(x)\cup\bigcup_{y\in\text{Ch}(x)}\text{Pa}(y) \tag{7}$$

基于双重约束的动态规划算法分为2个步骤:(1)基于邻居节点集合约束父节点图的搜索过程,得到候选父节点集合. 首先,利用MIC限制参与CI测试的候选节点集合;其次,通过结合MIC和马尔科夫毯限制CI测试的约束集;再次,通过方差对CI测试结果进行修剪得到邻居节点集合;最后,通过邻居节点集合约束DP算法的父节点图,得到候选父节点集合.(2)基于节点块序约束节点序图的搜索过程,得到最优的BN结构. 首先,从候选父节点集合中取出每个节点的最优父集,通过连接最优父集,得到初始有向图;然后,通过Tarjan算法计算初始有向图中的强连通分量SCC,得到节点块序;最后,利用节点块序约束DP算法的节点序图,得到最优BN结构. 本文提出的算法可降低算法的复杂度和运行时间,使算法在合理时间内学习大规模网络.

3.1　基于MIC和马尔科夫毯的CI测试方法

3.1.1　基于MIC限制CI测试的候选节点集合

MIC值可以衡量节点x_i和x_j之间的依赖程度,节点间的MIC值与其依赖程度呈正相关关系,即节点间的MIC值越大其依赖程度越强,反之,其依赖程度越低,所以可以利用MIC值限制CI测试的候选节点集合. 首先,初始化1个完全无向图G_0;其次,计算所有节点对

的 MIC 值，得到 $\mathrm{MIC}\left(x_i,x_j\right),i,j=1,2,\cdots,n$；再次，通过式(6)判断是否需要删除节点间的连接边，若 $\mathrm{MIC}\left(x_i,x_j\right)$ 满足式(6)，则认为节点 x_i 和 x_j 之间的依赖程度低，则在 G_0 中删除 2 个节点间的连接边，否则不删除；最后，G_0 中还存在的连接边便是 CI 测试的候选节点集合 nb_cand. 具体的伪代码如算法 1 所示.

算法 1　基于 MIC 限制 CI 测试的候选节点集合

基于MIC删除依赖关系较弱的节点,得到候选节点

Function1$\left[\mathrm{nb_cand},\mathrm{MIC}\right]=\mathrm{candidatenode}\left(D,\mathrm{ns},\alpha_{\mathrm{MIC}}\right)$

输入:样本数据集D,节点状态数ns,阈值α_{MIC}

输出:候选节点nb_cand,最大信息系数MIC

1.G_0 //初始化1个完全无向图

2.计算每个节点对的MIC:$\mathrm{MIC}\left(x_i,x_j\right),i,j=1,2,\cdots,n$

3.将计算得到的MIC值按行进行排列降序

4.FOR $i\leftarrow 1$ to n DO

5. FOR $j\leftarrow 1$ to n DO

6. IF $\mathrm{MIC}\left(x_i,x_j\right)<\alpha_{\mathrm{MIC}},\mathrm{MMIC}\left(x_i\right)\&\&\mathrm{MIC}\left(x_i,x_j\right)<\alpha_{\mathrm{MIC}}\mathrm{MMIC}\left(x_j\right)$ THEN

7. $G_0(x_i,x_j)=0$, $G_0(x_j,x_i)=0$

8. END IF

9. END FOR

10. 寻找G_0第i行不为0的下标,并将下标放入节点集$\mathrm{nb_cand}\left(x_i\right)$中

11.END FOR

3.1.2　基于马尔科夫毯限制 CI 测试的约束集

当节点 x 和节点 y 进行 CI 测试时，节点 y 可以从候选节点集合 $\mathrm{nb\_cand}(x)$ 中选择，该方法可有效减少 CI 测试的执行次数. 但 CI 测试还有随机性检测的缺陷，为了执行更有效的 CI 测试，需获得潜在的约束集.

考虑到高阶 CI 测试的计算结果可能会不正确，因此，利用 CI 测试判断节点 x 与 y 之间是否独立时，约束集 Z 从空集开始. 若 Z 为空集时节点 x 与 y 并未独立，接下来需要逐步增加 Z 的规模. Z 是在 $\mathrm{MB}(x)$ 所有可能的子集中进行选择，$\mathrm{MB}(x)$ 是节点 x 的配偶节点集和父子节点集，因此，可以利用 $\mathrm{MB}(x)$ 减小约束集 Z 的规模. 为了降低执行无效检验概率，从 $\mathrm{MB}(x)$ 相同规模的子集中选择约束集 Z 时，可选择与 y 更接近的节点，因为与 y 接近的节点更可能使 $x\perp y|Z$ 成立. 从上述对 MIC 的介绍得知，节点间 MIC 值越高，其依赖程度越高，可利用 y 的 MIC 值对 $\mathrm{MB}(x)$ 中对应的节点进行降序排列，并将其放入到节点集合 S 中；接着，将 S 的所有子集存储在 Sep_cand(x,y) 中. 在使用 CI 测试检验节点 x 和 y 之间的条件独立性关系时，可依次取出 Sep_cand(x,y) 中的值作为约束集 Z，当所有的约束集 Z 都不能使 $x\perp y|Z$ 成立时，将节点 y 放入邻居节点集合 $\mathrm{nb}(x)$ 中. 考虑到当 $\mathrm{nb}(x)$ 的值过多时，会增大 DP 算法的运行时间，所以本文在 $\mathrm{nb}(x)$ 的基础上删除了方差较小的节点，取得最终的 $\mathrm{nb}(x)$.

为了更清晰描述上述约束集 Z 的选择，下面使用无向图来说明，如图 1 所示.

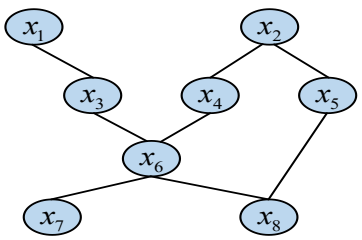


图 1　无向图

例如对节点 x_2 和节点 x_8 进行 CI 测试，若 Z 为空集时，$x_2\perp x_8|Z$ 不成立，假设 $\mathrm{MB}\left(x_2\right)=\left\{x_4,x_5\right\}$，利用 x_8 的 MIC 值对 $\mathrm{MB}\left(x_2\right)$ 中对应的节点进行降序排列后得到 $S=\left\{x_5,x_4\right\}$，取 S 的子集得到 Sep_cand$(x_2,x_8)=\left\{\left\{x_5\right\},\left\{x_4\right\},\left\{x_5,x_4\right\}\right\}$. 因此，在选择约束集 Z 时，优先选择 $\left\{x_5\right\}$ 作为约束集来执行 CI 测试，验证 $x_2\perp x_8|Z$ 是否成立. 用此方法进行 CI 测试时，发现只需要执行 2 次 CI 测试，便可以验证出 $x_2\perp x_8|Z$ 成立，所以该方法降低了 CI 测试的次数和阶数，提高 CI 测试的效率.

在对节点 x 与 $\mathrm{nb\_cand}(x)$ 中的每个节点 y 进行 CI 测试时，首先，计算节点 x 的马尔科夫毯 $\mathrm{MB}(x)$；其次，根据 y 的 MIC 值对 $\mathrm{MB}(x)$ 降序排列；再次，依次选取 $Z\in$ Sep_cand(x,y)；最后，检测节点 x 和节点 y 的条件独立性关系. 若不存在任何约束集使 x 与 y 条件独立，则将该节点加入到 x 的邻居节点集合 $\mathrm{nb}(x)$ 中. 考虑到当得到的 $\mathrm{nb}(x)$ 规模过大，会降低 DP 算法的效率，因此，本文利用方差对得到的 $\mathrm{nb}(x)$ 进行降序排列，再删除 $\mathrm{nb}(x)$ 后面的 b 个节点，得到节点 x 的邻居节点集合. 基于马尔科夫毯限制 CI 测试的约束集伪代码如算法 2 所示.

3.2　基于邻居节点集合约束父节点图的搜索过程

本文通过邻居节点集合约束 DP 算法的父节点图，可减少局部评分次数，降低算法时耗. 假设 $\mathrm{nb}\left(x_3\right)=\left\{x_1,x_4\right\}$，利用该集合约束节点 x_3 父节点图的具体过程如图 2 所示.

已知条件 $\mathrm{nb}\left(x_3\right)=\left\{x_1,x_4\right\}$，代表认为 $\left\{x_1,x_4\right\}$ 可能成为节点 x_3 的父集组合，而 x_2 不可能是 x_3 的父节点，则

算法2　基于马尔科夫毯限制CI测试的约束集

基于CI测试,得到邻居节点集合

Function2[nb] = candidateparentset(D, ns, nb_cand, MIC, b)

输入:样本数据集D,候选节点nb_cand,最大信息系数MIC,删除系数b

输出:邻居节点集合nb

1. MB(x) ←计算每个节点的马尔科夫毯
2.FOR　$x \leftarrow 1$ to n DO
3.　FOR　$y \in$ nb_cand(x) DO
4.　　$Z=\varnothing$,判断$x \perp y|Z$是否成立
5.　　IF　$x \perp y|Z$ THEN
6.　　nb_cand(x)中删除节点y
7.　　ELSE
8.　　$S \leftarrow$基于MIC($y,\cdot$)对MB(x)进行降序排列
9.　　Sep_cand(x,y)←取出S的所有子集
10.　　FOR　$Z \in$ Sep_cand(x,y) DO
11.　　判断$x \perp y|Z$是否成立
12.　　IF　$x \perp y|Z$成立 THEN
13.　　从nb_cand(x)中删除节点y
14.　　END IF
15.　　END FOR
16.　END IF
17.　IF　不存在约束集Z使$x \perp y|Z$成立 THEN
18.　　nb(x) ←y
19.　END IF
20.　END FOR
21. END FOR
22. 计算每个节点的方差,并对其进行降序排列
23. nb(x) ←根据方差对nb(x)进行降序排列
24. nb(x,(b:end)) ←∅//若nb(x)的数目>b,删除b之后的节点数

包含节点x_2的集合可以不计算,即图2(b)中灰色部分可以不用求解. 通过约束前和约束后的父节点图对比,发现基于nb(x)约束的父节点图可减少DP算法所需要的局部评分次数,提高DP算法的效率.

基于上述思路,利用邻居节点集合约束父节点图的搜索过程伪代码如算法3所示,第4~12行利用循环得到节点v可能的父集对应评分,其中,第4行通过集合nb(v)的数目r约束父节点集的最大父节点数,将父节点数量上限从$n-1$减少至r,且父集组合是从nb(v)中选择,以此降低局部评分次数.

算法3　基于邻居节点集合约束父节点图的搜索过程

基于nb得到候选父节点集合

Function3[Score, parents] = generatesparseparentgraph (D, n, ns, nb)

输入:样本数据集D,节点数n,节点状态数ns,邻居节点集合nb

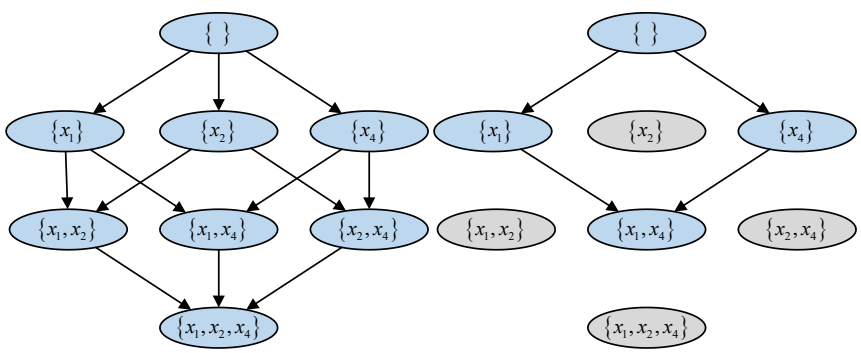
输出:父节点集合的评分Score,父节点集合的二进制编码parents

1.FOR　$v \leftarrow 1$ to n DO
2.　[Score$_v$, parent s$_v$] ←∅
3.　$r \leftarrow$ length(nb(v))
4.　FOR　layer ← 0 to r DO
5.　　FOR　each node U such that $U \in$ nb(v) & $|U|$=layer DO
6.　　BestScore(v, U) = $\max\limits_{Y \in U}$ BestScore($v, U \backslash \{Y\}$)
7.　　IF　Score(v, U) > BestScore(v, U) THEN
8.　　BestScore(v, U) ← Score(v, U)
9.　　Append [Score$_v$, parent s$_v$]
　　with [BestScore(v, U), bitnarize(U)]
10.　　END IF
11.　　END FOR
12.　END FOR
13.　Sort [Score$_v$, parent s$_v$] with Score$_v$ in descending
14. END FOR
15. RETURN [Score, parents]

从上述算法中得知,本文算法将父节点图中的评分计算次数从$n2^{n-1}$减少至$\sum_{v=1}^{n} 2^{|\mathrm{nb}(v)|}$.

3.3　基于节点块序约束节点序图的搜索过程

若有向图是强连通的,则该图的最大强连通子图称为SCC. 有向图的SCC是1个最大的顶点集,其中,有



(*a*) 约束前　　(*b*) 约束后

图2　约束x_3父节点图的示意图

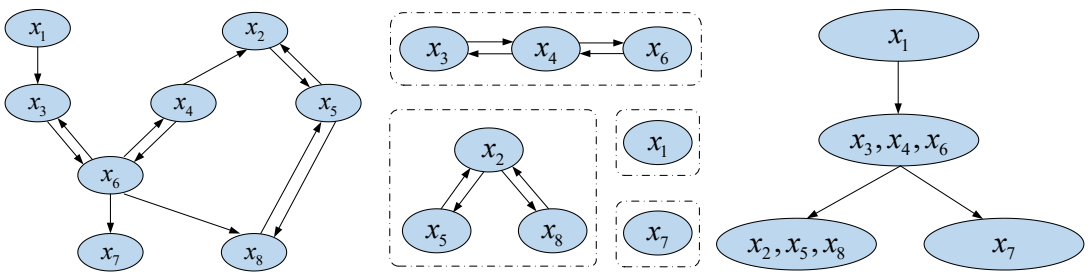
1条从集合中任何顶点到集合中任何其他顶点的路径. 如果有向图中的每个SCC收缩到单个顶点,则有向图就是广义的DAG,称为SCC图.

本文通过节点块序约束节点序图的搜索过程,可降低算法的运行时间,使算法能够在合理时间内学习大规模网络. 具体思路为:首先,通过父节点图得到每个节点的候选父集及评分,从评分中选出评分最高的集合作为每个节点的最优父集;其次,通过连接最优父集得到初始有向图,在初始有向图上寻找SCC,得到节点块 $C_1,C_2,\cdots,C_m$,其中,$i,j(i\neq j),C_i\cap C_j=\varnothing,\bigcup_{i=1}^{m}C_i=V$;再次,通过将节点块收缩为单个顶点,并进行排序,得到节点块序 $C_1\prec,C_2\prec,\cdots,C_i\prec,\cdots,C_m$,即 order=$\{C_1,C_2\prec,\cdots,C_i,\cdots,C_m\}$;最后,利用节点块序约束节点序图,同时结合上述候选父节点集合,得到最优的贝叶斯网络结构.

为了更清晰描述上述得到节点块序的过程,下面用Asia网络来说明. 假设,经过得分计算和修剪父节点图后,Asia网络中每个节点的最优父集如表2所示. Asia网络得到节点块序的过程如图3所示.

表2 Asia网络中每个节点的最优父集

节点	最优父集
x_1	$\{\ \}$
x_2	$\{x_4,x_5\}$
x_3	$\{x_1,x_6\}$
x_4	$\{x_6\}$
x_5	$\{x_2,x_8\}$
x_6	$\{x_3,x_4\}$
x_7	$\{x_6\}$
x_8	$\{x_5,x_6\}$



(a) 初始有向图 (b) 强连通分量 (c) SCC图

图3 Asia网络的节点块序

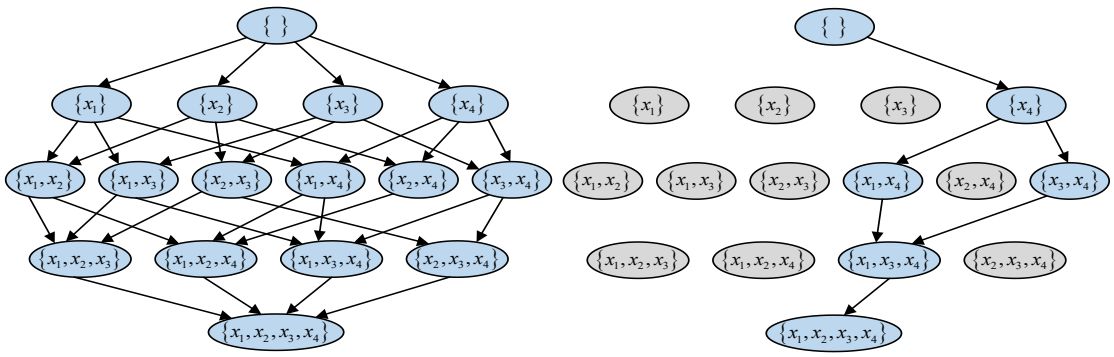
如图3(a)所示,通过连接每个节点的最优父集可得到初始有向图,该图中含有向环,故不是有向无环图. 利用Tarjan算法计算图3(a)中的4个SCC,分别是 $\{x_1\},\{x_2,x_5,x_8\},\{x_3,x_4,x_6\},\{x_7\}$,结果如图3(b)所示. 如果将每个SCC收缩到单个顶点,并学习每个块之间的顺序,可得到SCC图,如图3(c)所示. SCC图的固有拓扑关系便是节点块序,可以从图3(c)中得到Asia网络的节点块序为 $\{x_1\}\prec\{x_3,x_4,x_6\}\prec\{x_2,x_5,x_8\}\prec\{x_7\}$.

本文利用节点块序约束节点序图的具体思路为:对于节点序图中的任意节点 U 来说,若该节点输入的边为 k 条,则输出的边为 $n-k$ 条. 在节点序图中每个节点 U 输出的边对应下一步可扩展路径,通过节点块序约束节点序图的搜索过程时,明显观察到节点 U 的可扩展路径被限制,只有按节点块序进行扩展才能得到最优路径,其余均是次优路径,所以可以省略遍历其他路径.

为了更加清楚说明上述约束过程,使用节点数 $n=4$ 的节点序图说明约束过程,设节点块序 order=$\{\{x_4\},\{x_1,x_3\},\{x_2\}\}$,其中,每个节点块的数目为 order_num=$\{1,2,1\}$,具体约束过程如图4所示.

图4(a)为4节点无约束时节点序图的搜索过程,路径数为 $n!$,图4(b)是利用节点块序约束节点序图的搜索过程,路径数为 $1\times2\times1=2$. 图4(b)的具体搜索过程如下:从order中提取第1个节点块 $C_1=\{x_4\}$,该节点块的节点数目为1,需要向下扩展1层,因此,只需遍历节点序图的第1层,遍历的节点集合 $U=\{x_4\}$;由于第2个节点块为 $C_2=\{x_1,x_3\}$,该节点块的节点数目为2,所以需要向下扩展2层,第2层可通过添加节点 $X=x_1$ 或 $X=x_3$ 来扩展节点集合 U;第3层通过添加节点块 C_2 中剩余的节点来扩展节点集合 U,此时向下扩展了2层. 因此,接下来第4层通过添加节点块 $C_3=\{x_2\}$ 来扩展节点集合 U,此时节点集合 U 包含了所有的节点,故搜索过程结束.

若某网络的节点块序为order=$\{C_1,C_2,\cdots,C_i,\cdots,C_m\}$ 则该网络需要搜索的路径数为 $\prod_{i=1}^{m}|C_i|!$. 需要注意的是,DP算法的精度与初始有向图计算得到的节点块序是否正确有关,当节点块序正确时,DP的精度较高,反



(a) 无约束节点序图　　(b) order = $\{\{x_4\},\{x_1,x_3\},\{x_2\}\}$

图4　$n=4$的节点序图

之,精度较低.

3.4　算法流程

本论文提出的算法利用邻居节点集合约束父节点图的搜索过程,再利用节点块序约束节点序图的搜索过程,具有"压缩"动态规划搜索空间的效果. OLADC算法的伪代码如算法4所示,第2行通过父节点图的计算,得到每个节点的父节点集合及其评分;第3~8行取出节点评分最高的父节点集合,并用该集合构造有向图G;第9行通过Tarjan算法计算有向图G的SCC,即$C_1,C_2,\cdots,C_m$;第10~14行通过将SCC收缩为单个顶点,得到SCC图,SCC图的拓扑排序便是节点块序order,按序计算order中每个节点块的数目得到order_num;第15~17行通过节点块序order约束节点序图的搜索过程,输出最优的贝叶斯网络结构G^*.

3.5　算法时间复杂度分析

设n为网络节点个数,计算最大信息系数的时间复杂度[26]为$O(n^{2.4})$. 马尔科夫毯的时间复杂度为$O(n^2)$. 通过MIC限制参与CI测试的候选节点集合,可将节点x的CI测试的候选节点数从$n-1$减小为t_x($t_x=|\mathrm{nb\_cand}(x)|$),故可以选出$nt_x/2$种不同的变量对;通过结合MIC和马尔科夫毯限制CI测试的约束集,将每对变量的约束集从2^{n-2}减小为k_{xy}($k_{xy}=|\mathrm{Sep\_cand}(x,y)|$),因此,CI测试算法的时间复杂度为$O(k_{xy}t_xn)$. Tarjan算法的时间复杂度为$O(n+e)$,其中,$e$是有向图中的边数. 利用邻居节点集合和Tarjan算法约束后的DP算法时间复杂度为$O(n2^{r_x})$($r_x=|\mathrm{nb}(x)|$);综上所述,OLADC算法的时间复杂度为$O\left(n\left(n^{1.4}+n+k_{xy}t_x+1+2^{r_x}\right)\right)$.

使用MIC和马尔科夫毯限制CI测试的候选节点集合和约束集,可以将CI测试的候选节点集合数从$n(n-1)$降低为nt_x,将CI测试的约束集数从$\left(2^{n-2}n(n-1)\right)/2$降低为$\left(nt_xk_{xy}\right)/2$. 使用邻居节点集合和Tarjan算法约束DP算法,可以将DP算法的局部评分次数从$n2^{n-1}$降低为$n2^{r_x}$. 为了更加清楚描述约束过程前后数目的对比,以Sachs网络为例,CI测试的候选节点数目、约束集数目和DP算法局部评分次数在约束前与约束后的对比如表3所示.

算法4　基于双重约束的最优BN结构学习算法

基于双重约束的动态规划BN结构学习算法
输入:样本数据集D,节点状态数ns
输出:最优BN结构G^*
1.// 遍历父节点图,计算评分并存储
2.$[\mathrm{Score},\mathrm{parents}]=\mathrm{generatesparseparentgraph}(D,n,\mathrm{ns},\mathrm{nb})$
3.FOR　$x_i\leftarrow 1$ to n DO
　　//得到节点x_i评分最高的父集集合
4.　　$\mathrm{Bestparents}(x_i)=\arg\max_{P\in\mathrm{parents}(x_i)}\mathrm{Score}(P)$
5.　　FOR　$x_j\in\mathrm{Bestparents}(x_i)$ DO
6.　　$G(x_i,x_j)=1$ //利用最优父集得到有向图G
7.　　END FOR
8.END FOR
9.$C_1,C_2,\cdots,C_m\leftarrow\mathrm{Tarjan}(G)$//利用Tarjan算法得到强连通分量
10. $G^{\mathrm{SCC}}\leftarrow C_1,C_2,\cdots,C_i,\cdots,C_m$　//得到SCC图
11. $\mathrm{order}=\{C_1,C_2,\cdots,C_i,\cdots,C_m\}$　//对图G^{SCC}进行拓扑排序
12. FOR　$i\leftarrow 1$ to m DO
13.　　$\mathrm{order\_num}(k_i)\leftarrow\mathrm{length}(\mathrm{order}_i)$//得到节点块的数目
14. END FOR
15. //节点块序order约束节点序图
16. parents_nodes =
　　$\mathrm{GenerateOrderGraph}(D,\mathrm{Score},\mathrm{parents},\mathrm{order},\mathrm{order\_num})$
17. $G^*=\mathrm{parentstoday}(\mathrm{parents\_nodes})$　//得到最优BN结构

表3　Sachs网络约束前后的数目对比

数目	约束前	约束后
CI测试的候选节点数目	110	50
CI测试的约束集数目	28 160	856
DP算法的局部评分次数	11 264	150

由表3可知,约束后CI测试的候选节点数目、约束集数目和DP算法局部评分次数比约束前分别下降了54.5%、96.9%、98.6%,随着节点个数增加,这些数目会下降得愈加明显.

4 实验结果及分析

4.1 数据集

仿真实验在MATLAB R2019a环境下运行,操作系统为Windows10,硬件环境为Intel(R) Core(TM)i5-7300HQ CPU @ 2.50 GHz 2.50 GHz. 为验证OLADC算法性能,使用下列6种标准网络进行仿真实验,网络的具体情况说明如表4所示.

表4 标准网络的参数

网络	网络类型	节点数	边数	最大父节点数	最大节点出入度
Asia	小规模	8	8	2	4
Sachs	小规模	11	17	3	7
Child	中规模	20	25	2	8
Alarm	中规模	37	46	4	6
Hailfinder	大规模	56	66	4	17
Win95pts	大规模	76	112	7	10

4.2 评价指标

为检验本文算法的性能,利用上述每个网络生成样本容量分别为1 000、3 000、5 000的实验数据. 为降低实验数据对算法性能造成的影响,分别对上述每种标准网络的样本数量随机生成10组数据,将算法单独运行10次,最终算法的性能取平均值. 为了验证算法性能,本文采用运行时间、贝叶斯信息准则(Bayesian Information Criterion,BIC)评分、准确边数和汉明距离作为评价指标.

运行时间(T):算法学习BN结构所需时间.

BIC评分(BIC):算法学习得到BN结构的BIC评分,BIC评分越高,算法学习得到的BN结构越好.

$$\mathrm{BIC}(G|D)=\sum_{i=1}^{n}\sum_{j=1}^{q_j}\sum_{k=1}^{r_i}m_{ijk}\log\frac{m_{ijk}}{m_{ij}}-\frac{1}{2}\sum_{i=1}^{n}q_i(r_i-1)\log m \tag{8}$$

其中,G为网络拓扑结构,D为数据集,q_i表示x_i的父节点所取得的所有可能的组合状态,r_i为节点x_i状态取值个数,m_{ijk}为数据集D中满足$x_i=k$且$\mathrm{Pa}(x_i)=j$的样本个数,m为训练集中样本的数目.

准确边数(C):与标准网络相比,算法所得网络的正确边数量;添加边(A):标准网络中不存在但算法学习得到BN结构中存在的边数;反转边(R):标准网络和算法学习得到的BN结构中同时存在的边,但2个网络中边的方向相反;丢失边(M):标准网络中存在但算法学习得到BN结构中不存在的边数;汉明距离(HM):算法学习得到BN结构与真实网络结构之间不相同的边数之和. HM越小,代表学习得到的网络越好,HM=A+R+M. 6种标准网络的平均BIC评分如表5所示.

表5 6种标准网络的平均BIC评分

网络	1 000	3 000	5 000
Asia	−2 299.1	−6 757.7	−11 258.2
Sachs	−7 710.5	−21 995.4	−36 318.8
Child	−13 205.8	−38 388.1	−63 513.1
Alarm	−12 601.6	−34 688.4	−56 817.1
Hailfinder	−57 244.5	−156 642.3	−255 504.8
Win95pts	−10 985.9	−29 443.6	−47 893.8

4.3 实验结果分析

4.3.1 OLADC算法与精确算法的仿真对比

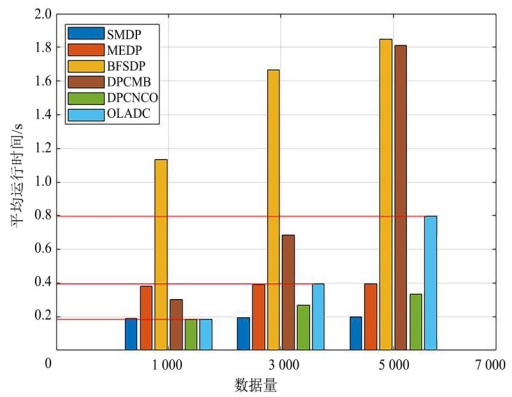
OLADC算法本质上是对DP算法进行改进的算法,为分析OLADC算法的性能,将算法与其他5种基于DP的结构学习算法如:SMDP(SM-Dynamic Programming)、MEDP(Memory-Efficient Dynamic Programming algorithm)、BFSDP(Breadth-First Search Dynamic Programming algorithm)、DPCMB(Dynamic Programming Constrained with Markov Blanket)和DPCNCO(Dynamic Programming Bayesian network structure learning algorithm based on Node Chunk Order Constraint)算法进行仿真对比. 在6种标准网络中分别运行OLADC和上述对比算法,算法的平均运行时间结果如表6所示. 其中,“—”表示网络学习在规定时间(4天)内没有得到结果或算法所需存储空间超出本机内存,加粗数据为最优结果.

为更加清楚地对算法运行时间进行对比,本文将OLADC和上述对比算法的运行时间进行比较,如图5所示. 由于上述对比算法在学习Hailfinder和Win95pts网络时,均超出时间限制或内存限制,因此,图5中只绘制了4个网络的运行时间对比图. Asia和Sachs网络中,算法运行时间相差并不太大,因此,没有对这2个网络的数据进行改进,对比结果如图5(a)和图5(b)所示,由于Child和Alarm网络中的部分算法运行时间过长,因此,将这2个网络的所有数据均取以10为底的对数,对比结果如图5(c)和图5(d)所示.

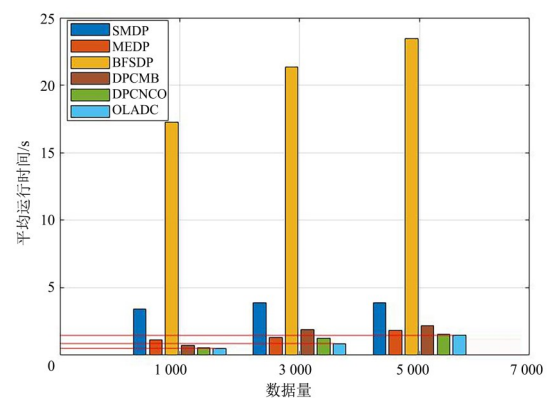
对比图5,在相同的标准网络中,相比于对比算法,OLADC算法运行时间明显下降. SMDP、MEDP、BFSDP、DPCMB、DPCNCO算法的平均运行时间,在Asia网络中分别是OLADC算法的1.02倍、2.07倍、6.17倍、1.64倍、0.99倍;在Sachs网络中上述5种算法分别是OLADC算法的6.65倍、2.16倍、

表6　6种算法在标准网络中的平均运行时间　　单位:s

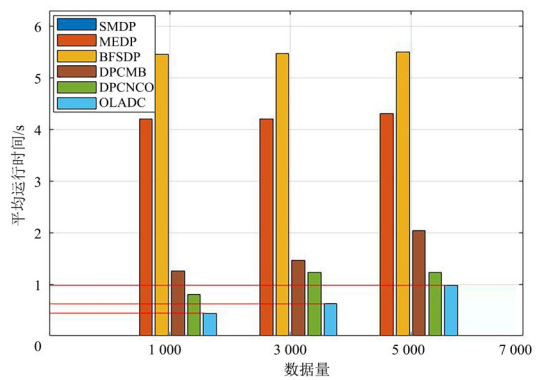
网络	样本数量	算法名称					
		SMDP	MEDP	BFSDP	DPCMB	DPCNCO	**OLADC**
Asia	1 000	0.187 2	0.380 3	1.133 1	0.301 5	0.182 5	**0.183 4**
	3 000	0.195 7	0.388 8	1.667 7	0.683 2	0.270 3	**0.394 1**
	5 000	0.198 7	0.393 3	1.847 7	1.808 8	0.333 8	**0.794 7**
Sachs	1 000	3.420 0	1.112 1	17.272 1	0.721 0	0.561 5	**0.514 2**
	3 000	3.859 9	1.327 7	21.341 4	1.873 7	1.245 1	**0.870 5**
	5 000	3.882 5	1.813 6	23.468 6	2.188 4	1.566 2	**1.465 5**
Child	1 000	—	17 563.400 0	282 720	18.085 9	6.265 6	**2.758 1**
	3 000	—	17 684.600 0	297 694	28.844 4	16.967 7	**4.175 1**
	5 000	—	18 276.100 0	313 673	109.384 1	33.827 1	**9.642 8**
Alarm	1 000	—	—	—	—	72.758 4	**10.592 9**
	3 000	—	—	—	—	202.292 9	**27.385 7**
	5 000	—	—	—	—	4 872.290 0	**1 305.500 0**
Hailfinder	1 000	—	—	—	—	—	**131.556 6**
	3 000	—	—	—	—	—	**194.168 3**
	5 000	—	—	—	—	—	**289.705 8**
Win95pts	1 000	—	—	—	—	—	**79.206 2**
	3 000	—	—	—	—	—	**1 039.927 0**
	5 000	—	—	—	—	—	**12 901.300 0**



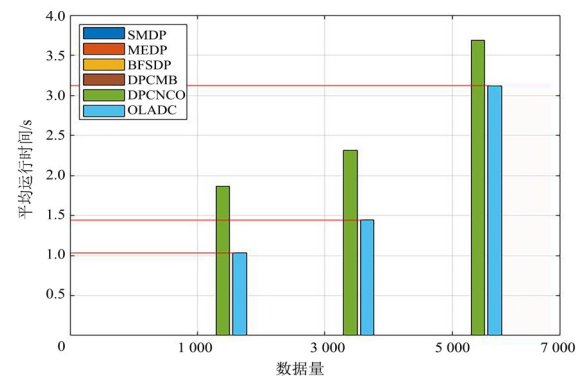
(*a*) Asia



(*b*) Sachs



(*c*) Child



(*d*) Alarm

图5　OLADC算法与对比算法的运行时间对比

33.59 倍、1.4 倍、1.09 倍. SMDP、BFSDP、DPCMB 算法在 Hailfinder 和 Win95pts 网络中需要储存的局部评分过多，MEDP、DPCNCO 算法在 Hailfinder 和 Win95pts 网络中需要计算的评分次数太多，因此，上述对比算法均没有在本文规定的时间内运行出结果. 上述对比算法对大于 37 个节点的标准网络已经很难在规定时间内得到结果，而本文提出的算法可以学习得到网络结构. 证明 OLADC 算法相比于上述对比算法有很大时间优势，可将其应用于大规模网络.

本文算法与上述对比算法在 Asia、Sachs、Child、Alarm 网络上的汉明距离、BIC 评分、正确边数对比结果如表 7~10 所示，加粗数据表示最优结果.

表 7 不同算法在 Asia 网络中的运行结果

样本数量	算法名称	BIC	*C*	HM
1 000	SMDP	−2 290.8	60.2	3.8
	MEDP	−2 294.3	60.4	3.6
	BFSDP	−2 301.5	60.7	3.3
	DPCMB	−2 270.3	57.8	6.2
	DPCNCO	−2 322.1	59.1	4.9
	OLADC	**−2 292.1**	**60.5**	**3.5**
3 000	SMDP	−6 774.2	60.5	3.5
	MEDP	−6 806.6	61.1	2.9
	BFSDP	−6 766.4	60.9	3.1
	DPCMB	−6 788.0	60.9	3.1
	DPCNCO	−6 766.9	61.2	2.8
	OLADC	**−6 849.8**	**61.0**	**3.0**
5 000	SMDP	−11 262.3	61.1	2.9
	MEDP	−11 280.4	61.3	2.7
	BFSDP	−11 257.7	61.0	3.0
	DPCMB	−11 150.0	61.2	2.8
	DPCNCO	−11 246.2	61.3	2.7
	OLADC	**−11 242.8**	**60.9**	**3.1**

6 种算法学习得到标准网络的 HM 和 BIC 评分的仿真对比，如图 6~9 所示. 由于上述算法得到的 HM 和 BIC 评分相差不大，因此，图中的数据并没有进行任何改进.

对比表 7~10，图 6~9，相同的标准网络中，相较于对比算法，本文提出的算法平均汉明距离偏大，平均 BIC 评分没有明显差异. OLADC 算法与 SMDP 算法、MEDP 算法、BFSDP 算法、DPCMB 算法、DPCNCO 算法相比，平均汉明距离在 Asia 网络中分别降低 5%、提升 4.2%、提升 2.1%、降低 20.6%、降低 7.8%；在 Sachs 网络中分别提升 17.5%、降低 4.5%、提升 16%、提升 12.1%、提升 18.9%，故 OLADC 算法学习精度有所下降. 以 Sachs 网络为例，平均 HM 为算法精度的评价指标，OLADC 算法相对 DPCMB 算法精度降低了 12.1%.

表 8 不同算法在 Sachs 网络中的运行结果

样本数量	算法名称	BIC	*C*	HM
1 000	SMDP	−7 678.1	109.5	11.5
	MEDP	−7 621.4	107.7	13.3
	BFSDP	−7 640.6	110.7	10.3
	DPCMB	−7 584.5	108.2	12.8
	DPCNCO	−7 666.0	111.5	9.5
	OLADC	**−7 656.6**	**108.1**	**12.9**
3 000	SMDP	−22 088.1	111.6	9.4
	MEDP	−22 016.0	109.0	12.0
	BFSDP	−22 078.1	111.4	9.6
	DPCMB	−22 198.4	113.2	7.8
	DPCNCO	−21 984.4	112.8	8.2
	OLADC	**−22 082.5**	**109.4**	**11.6**
5 000	SMDP	−36 433.1	112.9	8.1
	MEDP	−36 457.5	110.6	10.4
	BFSDP	−36 407.6	111.6	9.4
	DPCMB	−36 506.7	111.2	9.8
	DPCNCO	−36 386.9	110.3	10.7
	OLADC	**−36 355.0**	**111.4**	**9.6**

表 9 不同算法在 Child 网络中的运行结果

样本数量	算法名称	BIC	*C*	HM
1 000	SMDP	—	—	—
	MEDP	−13 132.6	395.0	5.0
	BFSDP	−13 178.6	394.0	6.0
	DPCMB	−13 279.0	395.7	8.0
	DPCNCO	−13 163.9	391.5	8.5
	OLADC	**−13 318.9**	**391.3**	**8.7**
3 000	SMDP	—	—	—
	MEDP	−38 392.7	396.0	4.0
	BFSDP	−38 803.7	395.0	5.0
	DPCMB	−38 394.4	396.6	3.4
	DPCNCO	−38 813.2	392.9	7.1
	OLADC	**−38 615.9**	**392.5**	**7.5**
5 000	SMDP	—	—	—
	MEDP	−63 059.3	394.0	6.0
	BFSDP	−63 214.1	394.0	6.0
	DPCMB	−63 598.5	393.8	6.2
	DPCNCO	−63 883.4	393.2	6.8
	OLADC	**−63 880.7**	**393.5**	**6.5**

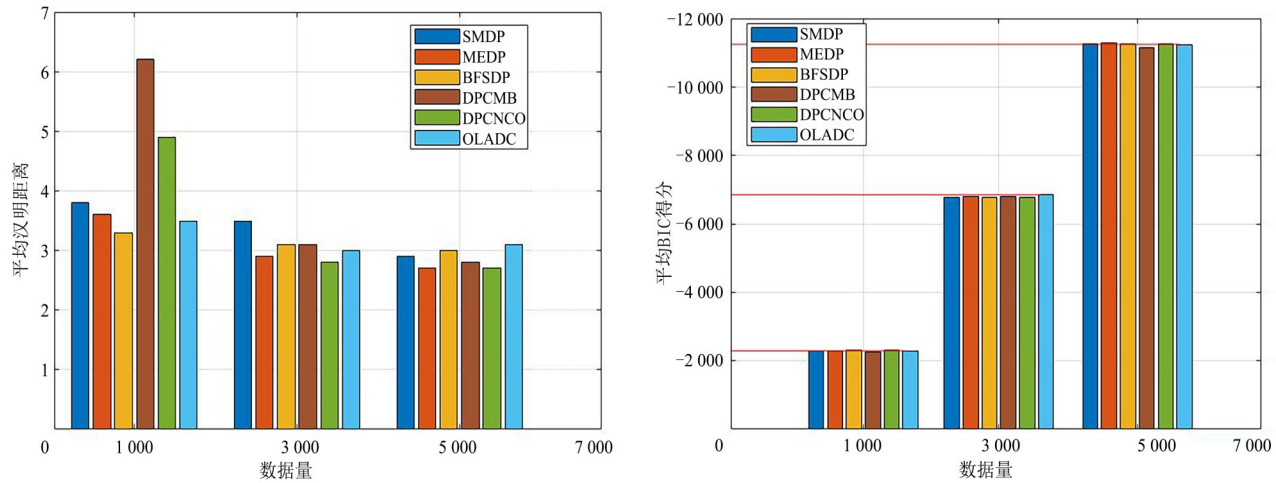
本文算法出现精度下降的原因是由于利用 SCC 图的拓扑关系得到的节点块序并不一定正确，若其中含有最优节点序，则 OLADC 算法精度不会有明显下降；若其中不包含最优节点序，则 OLADC 算法精度会有明显下降. 由于算法的运行时间与精度是难以均衡的 2 个评价指标，因此，本文算法允许以稍微降低算法精

表10　不同算法在Alarm网络中的运行结果

样本数量	算法名称	BIC	C	HM
1 000	DPCNCO	−12 392.3	1 334.0	35.0
	OLADC	**−12 959.1**	**1 332.3**	**36.7**
3 000	DPCNCO	−34 779.5	1 336.5	32.5
	OLADC	−35 368.7	1 336.0	33.0
5 000	DPCNCO	−57 552.9	1 334.0	35.0
	OLADC	**−57 889.9**	**1 339.2**	**29.8**

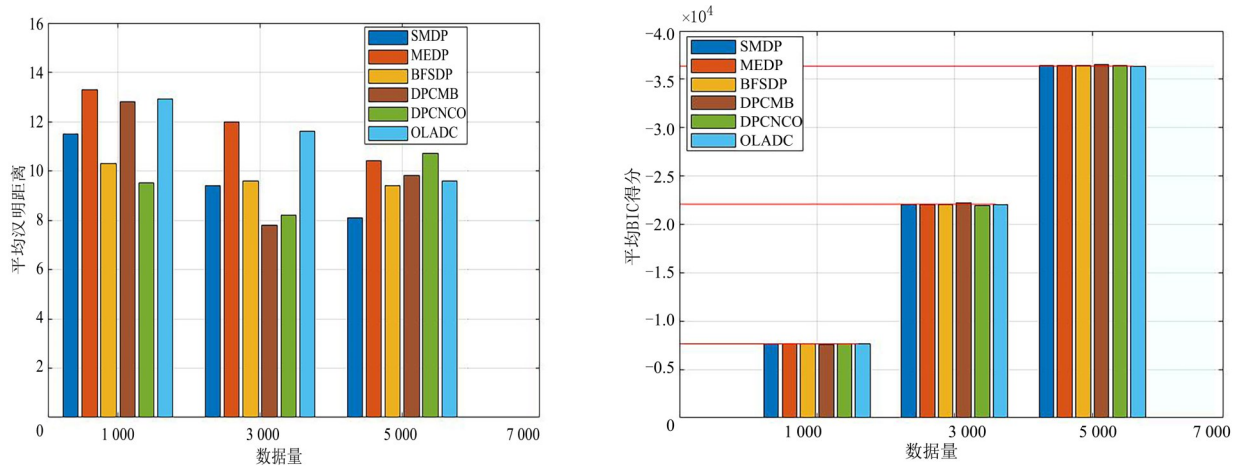
度的代价换取算法运行时间的优化.

综上所述,OLADC算法相比于对比算法虽然在精度上有一定范围下降,但在运行时间上有明显优势,如对Sachs网络,OLADC算法相对DPCMB算法时耗降低了40.3%. 以平均HM为算法精度的评价指标,OLADC算法相对DPCMB算法精度降低了12.1%. OLADC算法在允许稍微降低算法精度的前提下,大幅度提高算法效率.



(*a*) 平均HM　　(*b*) 平均BIC得分

图6　Asia网络中6种算法的精度对比

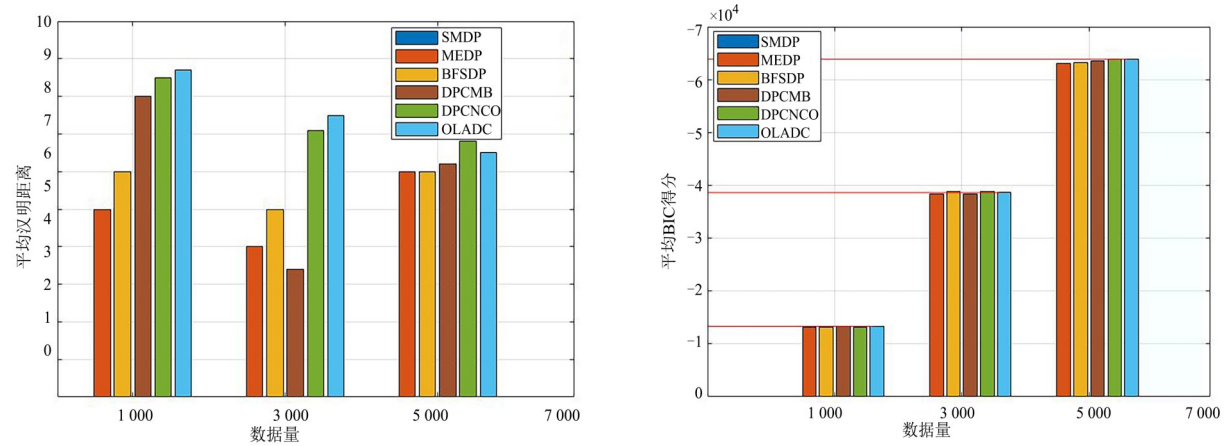


(*a*) 平均HM　　(*b*) 平均BIC得分

图7　Sachs网络中6种算法的精度对比

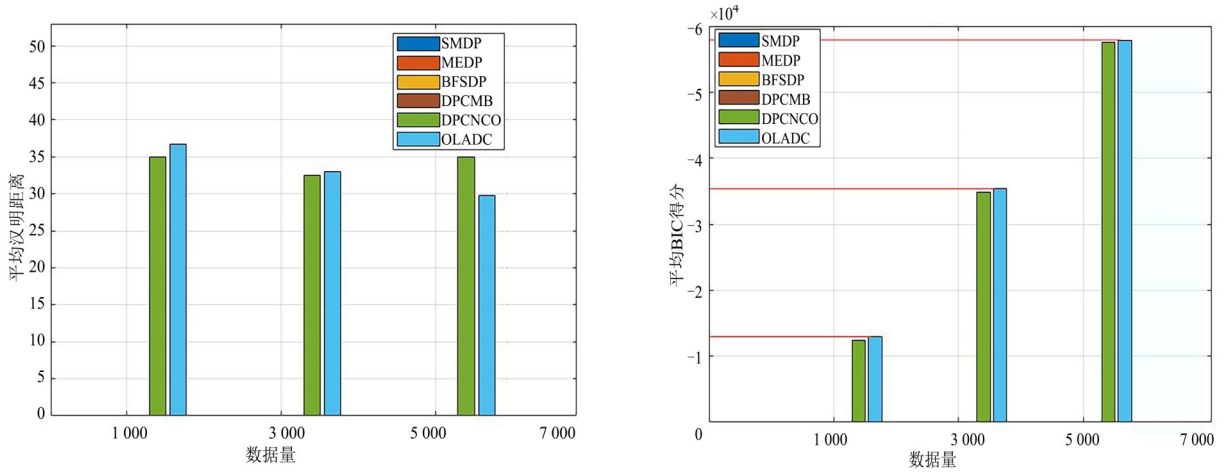
4.3.2　OLADC算法与近似算法的仿真对比

为验证OLADC算法性能,在标准的Sachs小型网络、Child中型网络和Hailfinder大型网络上进行仿真实验,样本容量分别为1 000、3 000和5 000,对比算法为PSO(Particle Swarm Optimization)和DFA-B(Bayesian network structure learning based on Discrete Firefly optimization Algorithm). 在本次实验中,种群规模均为20,Sachs、Child和Hailfinder网络的迭代次数分别为50、100、200. PSO和DFA-B算法的相关参数设置如表11所示,实验结果如表12~14所示,加粗数据表示最优结果.



(*a*) 平均HM (*b*) 平均BIC得分

图8 Child网络中6种算法的精度对比



(*a*) 平均HM (*b*) 平均BIC得分

图9 Alarm网络中6种算法的精度对比

表11 算法的相关参数设置

算法	参数
PSO	$c_1=c_2=2,\omega=0.6$
DFA-B	$\gamma=1,\beta_0=1$

表12 不同算法在Sachs网络中的运行结果

样本数量	算法名称	*T*/s	BIC	*C*	HM
1 000	PSO	358.828 0	−8 422.12	100.3	20.7
	DFA-B	123.095 2	−7 841.70	103.7	17.3
	OLADC	**0.514 2**	**−7 656.60**	**108.1**	**12.9**
3 000	PSO	381.197 0	−24 266.80	102.0	19.0
	DFA-B	233.318 0	−22 083.60	106.7	14.3
	OLADC	**0.870 5**	**−22 082.50**	**109.4**	**11.6**
5 000	PSO	527.488 0	−39 050.80	102.3	18.7
	DFA-B	293.642 7	−36 582.70	108.6	12.4
	OLADC	**1.465 5**	**−36 355.00**	**111.4**	**9.6**

表13 不同算法在Child网络中的运行结果

样本数量	算法名称	*T*/s	BIC	*C*	HM
1 000	PSO	2 088.130 0	−16 261.5	355.9	44.1
	DFA-B	539.971 0	−13 412.8	379.6	20.4
	OLADC	**2.758 1**	**−13 163.9**	**391.5**	**8.5**
3 000	PSO	2 967.640 0	−47 141 0	359.6	40.4
	DFA-B	949.320 6	−39 179.3	384.5	15.5
	OLADC	**4.175 1**	**−38 615.9**	**392.5**	**7.5**
5 000	PSO	4 130.070 0	−76 761.4	361.3	38.7
	DFA-B	1 335.470 0	−64 035.1	388.3	11.7
	OLADC	**9.642 8**	**−63 880.7**	**393.5**	**6.5**

对比表12~14,在相同的标准网络中,相比于PSO和DFA-B算法,本文提出的算法平均汉明距离减少,平均BIC评分增大,平均运行时间下降.OLADC算法与PSO算法、DFA-B算法相比,平均汉明距离在Sachs

表14　不同算法在Hailfinder网络中的运行结果

样本数量	算法名称	*T*/s	BIC	*C*	HM
1 000	PSO	30 844.100 0	−68 377.2	3 027.3	108.7
	DFA-B	7 501.620 0	−55 205.2	3 053.5	82.5
	OLADC	**131.556 6**	**−53 813.1**	**3 066.1**	**69.9**
3 000	PSO	71 661.400 0	−203 723.0	3 029.6	106.4
	DFA-B	14 783.300 0	−162 034.0	3 056.0	80.0
	OLADC	**194.168 3**	**−155 510.0**	**3 075.4**	**60.6**
5 000	PSO	96 321.100 0	−332 608.0	3 035.3	100.7
	DFA-B	20 963.700 0	−262 063.0	3 059.5	76.5
	OLADC	**289.705 8**	**−258 555.0**	**3 089.5**	**46.5**

网络中分别降低41.6%、22.5%;在Child网络中,分别降低81.7%、52.7%;在Hailfinder网络中,分别降低43.9%、25.9%.

观察表12~14,当样本容量不断增加时,OLADC算法学习得到的BN结构平均HM减少. 在样本容量相同时,相比于PSO和DFA-B算法,OLADC算法学习得到的BN结构最接近标准网络结构,说明了OLADC算法的有效性.

5　结论

针对现有基于动态规划的贝叶斯网络结构学习算法复杂度高、无法在合理时间内学习大规模网络的问题,提出基于双重约束的最优BN结构学习算法. 首先,通过基于MIC和马尔科夫毯的CI测试方法生成邻居节点集合;其次,通过该集合约束DP算法的父节点图,达到减少局部评分次数,降低算法时耗;再次,利用SCC图的拓扑关系得到节点块序;最后,通过节点块序约束DP算法的节点序图,达到扩大算法可处理网络规模的目的. 实验结果表明,OLADC算法相比现有的5种基于DP的结构学习算法在精度稍微降低的前提下,大幅度提高了算法的效率,如对Sachs网络,OLADC算法相对DPCMB算法时耗降低了40.3%,算法精度下降了12.1%.

尽管本文算法在运行时间上取得了良好结果,但仍存在进一步提升的空间,比如,本文的节点块序与算法的精度有很大关系,但目前并没有1个可以准确得到节点块序的算法,这将是后续研究的方向.

参考文献

[1] MIGLIORINI F, MAFFULLI N, COLAROSSI G, et al. Effect of drugs on bone mineral density in postmenopausal osteoporosis: A Bayesian network meta-analysis[J]. Journal of Orthopaedic Surgery and Research, 2021, 16(1): 533.

[2] LI Y Q, MAVADATI S M, MAHOOR M H, et al. Measuring the intensity of spontaneous facial action units with dynamic Bayesian network[J]. Pattern Recognition, 2015, 48(11): 3417-3427.

[3] HE J, YU X C, YU L J, et al. Facial emotion and action unit recognition based on Bayesian network[C]//2019 8th International Conference on Computing and Pattern Recognition. New York: ACM, 2019: 363-368.

[4] KOC L, CARSWELL A. Application of an AODE based classifier to detect DOS attacks[J]. International Journal of Computer Science and Network Security, 2015, 15(2): 24.

[5] SUN X Y, DAI J, LIU P, et al. Using Bayesian networks for probabilistic identification of zero-day attack paths[J]. IEEE Transactions on Information Forensics and Security, 2018, 13(10): 2506-2521.

[6] DON M G, KHAN F. Dynamic process fault detection and diagnosis based on a combined approach of hidden Markov and Bayesian network model[J]. Chemical Engineering Science, 2019, 201: 82-96.

[7] AMIN M T, KHAN F, AHMED S, et al. A data-driven Bayesian network learning method for process fault diagnosis[J]. Process Safety and Environmental Protection, 2021, 150: 110-122.

[8] CHEN C, ZHANG L, TIONG R. A novel learning cloud Bayesian network for risk measurement[J]. Applied Soft Computing, 2020, 87: 105947.

[9] BOUDALI H, DUGAN J B. A continuous-time Bayesian network reliability modeling and analysis framework[J]. IEEE Transactions on Reliability, 2006, 55(1): 86-97.

[10] ZHOU Q J, WONG Y D, LOH H S, et al. A fuzzy and Bayesian network CREAM model for human reliability analysis—The case of tanker shipping[J]. Safety Science, 2018, 105 : 149-157.

[11] LI M, WANG H T, WANG D M, et al. Risk assessment of gas explosion in coal mines based on fuzzy AHP and Bayesian network[J]. Process Safety and Environmental Protection, 2020, 135: 207-218.

[12] CHICKERING D M, HECKERMAN D, MEEK C. Large-sample learning of Bayesian networks is NP-hard[J]. Journal of Machine Learning Research, 2004, 5: 1287-1330.

[13] 吕志刚, 李叶, 王洪喜,等. 贝叶斯网络的结构学习综述[J]. 西安工业大学学报, 2021, 41(1): 1-17.
LVU Z G, LI Y, WANG H X, et al. Overview of Bayesian network structure learning[J]. Journal of Xi'an Technological University, 2021, 41(1): 1-17. (in Chinese)

[14] 朱明敏, 刘三阳, 杨有龙. 基于混合方式的贝叶斯网络等价类学习算法[J]. 电子学报, 2013, 41(1): 98-104.
ZHU M M, LIU S Y, YANG Y L. Structural learning Bayesian network equivalence classes based on a hybrid method[J]. Acta Electronica Sinica, 2013, 41(1): 98-104. (in Chinese)

[15] KOIVISTO M, SOOD K. Exact Bayesian structure discovery in Bayesian networks[J]. Journal of Machine Learning Research, 2004, 5: 549-573.

[16] JAAKKOLA T, SONTAG D, GLOBERSON A, et al. Learning Bayesian network structure using LP relaxations[C]// International Conference on Artificial Intelligence and Statistics. Brookline: Microtome Publishing, 2010: 358-365.

[17] CAMPOS D P C, JI Q. Efficient structure learning of Bayesian networks using constraints[J]. Journal of Machine Learning Research, 2011, 12: 663-689.

[18] 叶思懋. 融合先验的贝叶斯网络结构学习及其在智能决策中的应用[D]. 西安: 西北工业大学, 2018.
YE S M. Learning Bayesian Network Structure Learning with Prior Fusion and its Application in Intelligent Decision-Making[D]. Xi'an: Northwestern Polytechnical University, 2018. (in Chinese)

[19] 谭翔元, 高晓光, 贺楚超. 基于马尔科夫毯约束的最优贝叶斯网络结构学习算法[J]. 电子学报, 2019, 47(9): 1898-1904.
TAN X Y, GAO X G, HE C C. Learning optimal Bayesian network structure constrained with Markov blanket constraint[J]. Acta Electronica Sinica, 2019, 47(9): 1898-1904. (in Chinese)

[20] YANG Y, GAO X G, GUO Z G. Finding optimal Bayesian networks by a layered learning method[J]. Journal of Systems Engineering and Electronics, 2019, 30(5): 946-958.

[21] LI X, GAO X , WANG C. A novel BN learning algorithm based on block learning strategy[J]. Sensors (Basel, Switzerland), 2020, 20(21): E6357.

[22] 张连文, 郭海鹏. 贝叶斯网引论[M]. 北京: 科学出版社, 2006.
ZHANG L W, GUO H P. Introduction to Bayesian Networks[M]. Beijing: Science Press, 2006. (in Chinese)

[23] MALONE B, YUAN C, HANSEN E A. Memory-efficient dynamic programming for learning optimal Bayesian networks[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2011, 25(1): 1057-1062.

[24] RESHEF D N, RESHEF Y A, FINUCANE H K. Detecting novel associations in large data sets[J]. Science, 2011, 334(6062): 1518-1524.

[25] ZHANG Y H, ZHANG W S, XIE Y. Improved heuristic equivalent search algorithm based on maximal information coefficient for Bayesian network structure learning[J]. Neurocomputing, 2013, 117: 186-195.

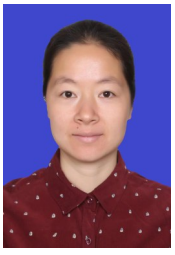
[26] SHAO F B, LI K P, XU X M. Railway accidents analysis based on the improved algorithm of the maximal information coefficient[J]. Intelligent Data Analysis, 2016, 20(3): 597-613.

作者简介

陈艺薇 女, 1999 年 2 月出生, 陕西宝鸡人. 西安工业大学电子信息工程学院研究生. 主要研究方向为复杂系统建模与评估.
E-mail: 1793692717@qq.com

邸若海 男, 1986 年 9 月出生, 陕西西安人. 西安工业大学电子信息工程学院副教授. 主要研究方向为复杂系统建模、机器学习理论方法、作战系统仿真.
E-mail: xfwtdrh@163.com

王 鹏 男, 1978 年 10 月出生, 山东泰安人. 西安工业大学电子信息工程学院教授. 主要研究方向为复杂系统建模、图像处理研究.
E-mail: wp-xatu@163.com



张新兰 女, 1986 年 9 月出生, 江西南康人. 航天材料及工艺研究所高级工程师. 主要研究方向为非金属材料与环境适应性.
E-mail: xinlan70@sohu.com

张 欢 女, 1988 年 7 月出生, 陕西宝鸡人. 航天材料及工艺研究所高级工程师. 主要研究方向为材料贮存与环境适应性.
E-mail: zhanghuan701@163.com