高级算法设计与分析

全息归约

夏盟佶
Xia, Mingji

中科院软件所
计算机科学国家重点实验室

2020.4.20

斐波那契门

- 张量网络、函数，也可被叫做线路、门。
- 如果$f_{i+2} = f_{i+1} + f_i$，则称$[f_0, f_1, \dots, f_k]$斐波那契门。

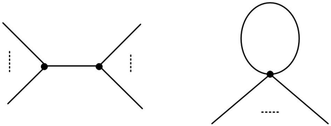
例

$$
\begin{aligned}
F &= [a_0, a_1] = [0, 1] \\
H &= [a_0, a_1, a_2] = [0, 1, 1] \\
K &= [a_0, a_1, a_2, a_3] = [0, 1, 1, 2] \\
&\dots
\end{aligned}
$$

- 斐波那契门构成的张量网络求值，即$\#\{F, H, K, \dots\}|\{=_2\}$问题，有多项式时间算法。

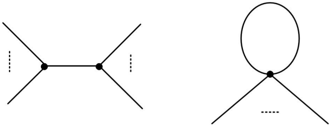
算法一：利用封闭性质

- 连通的张量网络可以通过两种运算（表示成开放的张量网络）生成。



算法一：利用封闭性质

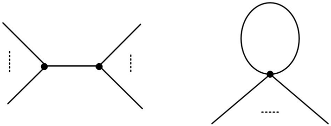
- 连通的张量网络可以通过两种运算（表示成开放的张量网络）生成。



- 斐波那契门关于这两种运算封闭。

算法一：利用封闭性质

- 连通的张量网络可以通过两种运算（表示成开放的张量网络）生成。



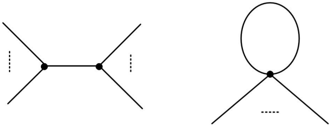
- 斐波那契门关于这两种运算封闭。

例

第二个运算，如果$F=[f_0,f_1,\ldots,f_k]$是斐波那契门，新得到的函数$[f_0+f_2,f_1+f_3,\ldots,f_{k-2}+f_k]$也是。

算法一：利用封闭性质

- 连通的张量网络可以通过两种运算（表示成开放的张量网络）生成。



- 斐波那契门关于这两种运算封闭。

例

第二个运算，如果$F = [f_0, f_1, \ldots, f_k]$是斐波那契门，新得到的函数$[f_0 + f_2, f_1 + f_3, \ldots, f_{k-2} + f_k]$也是。

- 斐波那契门可用多项式长度的串表示，并且作为每种运算的输入和输出是多项式时间可以计算的。

关于第一种运算封闭性的证明

- $$f_{i+2} = f_{i+1} + f_i$$

关于第一种运算封闭性的证明

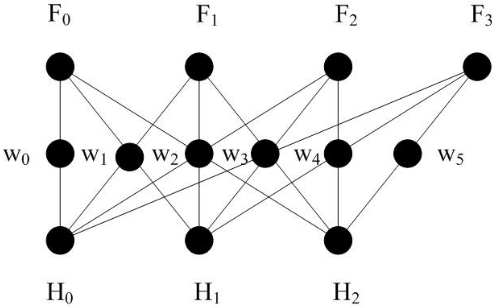
- $$f_{i+2} = f_{i+1} + f_i$$
- 记$F_i = (f_i, f_{i+1})'$，则$F_i = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} F_{i-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^i F_0$。

关于第一种运算封闭性的证明

- $$f_{i+2} = f_{i+1} + f_i$$
- 记$F_i = (f_i, f_{i+1})'$，则$F_i = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} F_{i-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^i F_0$。
- 对函数H，以及F和H通过第一种运算得到的W引入类似的记号。

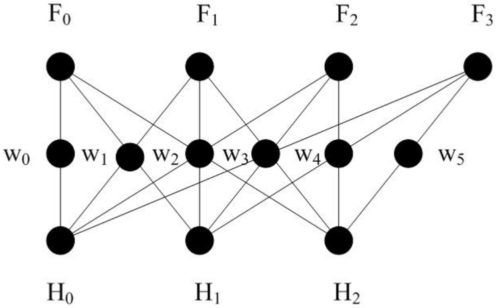
关于第一种运算封闭性的证明

- $$f_{i+2} = f_{i+1} + f_i$$
- 记$F_i = (f_i, f_{i+1})'$，则$F_i = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} F_{i-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^i F_0$。
- 对函数H，以及F和H通过第一种运算得到的W引入类似的记号。



关于第一种运算封闭性的证明

- $$f_{i+2} = f_{i+1} + f_i$$
- 记$F_i = (f_i, f_{i+1})'$，则$F_i = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} F_{i-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^i F_0$。
- 对函数H，以及F和H通过第一种运算得到的W引入类似的记号。



- $$W_i = F_0 H_i = F_1 H_{i-1} = \cdots = F_i H_0$$

算法二：全息算法

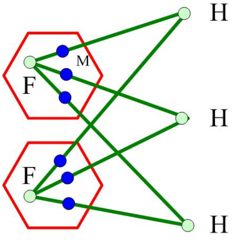
- $$f_{i+2} = f_{i+1} + f_i$$
- 设方程$x^2 = x + 1$的两个根是a, b。（显然$ab = -1$。）
- $[1, a, a^2, \ldots, a^n]$满足递推关系$a^{i+2} = a^{i+1} + a^i$是斐波那契门。
- 此函数的指数长的函数值表的向量形式：$(1, a)^{\otimes n}$。
- 类似的，$(1, a)^{\otimes n}$也是斐波那契门。
- $s(1, a)^{\otimes n} + t(1, b)^{\otimes n}$也是。

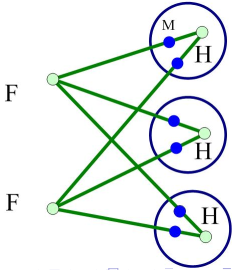
斐波那契门表示成张量积之和

- $F_i = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^i F_0 = (M \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} M^{-1})^i \begin{pmatrix} f_0 \\ f_1 \end{pmatrix} = T \begin{pmatrix} a^i \\ b^i \end{pmatrix}$
- $$[s+t, sa+tb, \cdots, sa^i+tb^i, \cdots, sa^n+tb^n]$$
- $$\begin{aligned} & s(1,a)^{\otimes n} + t(1,b)^{\otimes n} \\ = & (s(1,0)^{\otimes n} + t(0,1)^{\otimes n}) \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix}^{\otimes n} \\ = & \text{“}[s, 0, \ldots, 0, t]\text{”} \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix}^{\otimes n} \end{aligned}$$

回顾：全息归约

类比特例$(AB)C = A(BC)$。





定理

$\#\{FM^{\otimes 3}\}|\{H\}$和$\#\{F\}|\{M^{\otimes 2}H\}$值相同。

全息算法

- $M=\begin{pmatrix}1 & a\\ 1 & b\end{pmatrix}$ （之前有说明$ab=-1$）

全息算法

- $M = \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix}$ （之前有说明$ab=-1$）
- 右侧的二元相等变成了

全息算法

- $M=\begin{pmatrix}1 & a\\ 1 & b\end{pmatrix}$ （之前有说明$ab=-1$）
- 右侧的二元相等变成了
- $$M^{\otimes 2}\text{“}[1,0,1]\text{”}$$（可直接算）

全息算法

- $M = \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix}$ （之前有说明$ab=-1$）
- 右侧的二元相等变成了
- $$M^{\otimes 2}\text{“}[1,0,1]\text{”}$$（可直接算）
- 也等价于

$$MEM' = \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ a & b \end{pmatrix} = \begin{pmatrix} 1+a^2 & 0 \\ 0 & 1+b^2 \end{pmatrix}$$

全息算法

- $M = \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix}$ （之前有说明$ab = -1$）
- 右侧的二元相等变成了
- $$M^{\otimes 2}\text{“}[1,0,1]\text{”}\text{（可直接算）}$$
- 也等价于
$$MEM' = \begin{pmatrix} 1 & a \\ 1 & b \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ a & b \end{pmatrix} = \begin{pmatrix} 1+a^2 & 0 \\ 0 & 1+b^2 \end{pmatrix}$$
- 调用product type即#CSP$\{\mathcal{P}\}$的算法。

推广

- $f_{i+2}=f_{i+1}+f_i$ 可以替换成$f_{i+2}=\lambda f_{i+1}+f_i$。
- 全息算法的基$\begin{pmatrix}1 & a\\ 1 & b\end{pmatrix}$可以替换成任何一个正交矩阵$H$。
- 把$s(1,0)^{\otimes n}+t(0,1)^{\otimes n}$即$[s,0,\ldots,0,t]$推广到集合$\mathcal{E}$。
- 使用易解问题$\#\{=_2\}|\langle\{\mathcal{E}\rangle$做起始。
- 使用H做基变换，$H``=_2"H' \quad = \quad ``=_2"$。
- 我们得到$\#\{=_2\}|\langle H\mathcal{E}\rangle$ 有多项式时间算法。

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。

$[x, y, -x, -y, \dots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。
- $$\begin{aligned} & s(1, i)^{\otimes n} + t(1, -i)^{\otimes n} \\ = & \left(s(1,0)^{\otimes n} + t(0,1)^{\otimes n}\right) \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}^{\otimes n} \end{aligned}$$

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。
- $$\begin{aligned} & s(1, i)^{\otimes n} + t(1, -i)^{\otimes n} \\ = & \left(s(1,0)^{\otimes n} + t(0,1)^{\otimes n}\right) \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}^{\otimes n} \end{aligned}$$
- $M = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。
- $$\begin{aligned} & s(1,i)^{\otimes n} + t(1,-i)^{\otimes n} \\ = & \left(s(1,0)^{\otimes n} + t(0,1)^{\otimes n}\right) \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}^{\otimes n} \end{aligned}$$
- $M = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$
- 右侧的二元相等变成了

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。
- $$\begin{aligned} & s(1, i)^{\otimes n} + t(1, -i)^{\otimes n} \\ = & \left(s(1,0)^{\otimes n} + t(0,1)^{\otimes n}\right) \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}^{\otimes n} \end{aligned}$$
- $M = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$
- 右侧的二元相等变成了
- $$M^{\otimes 2}\text{“}[1, 0, 1]\text{”}$$

$[x, y, -x, -y, \ldots]$

- 仿照斐波那契门的算法，递推关系是$f_{i+2} + f_i = 0$。
- 方程$x^2 + 1 = 0$的两个根是$i, -i$。
-
$$\begin{aligned} & s(1, i)^{\otimes n} + t(1, -i)^{\otimes n} \\ = & \left(s(1,0)^{\otimes n} + t(0,1)^{\otimes n}\right) \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}^{\otimes n} \end{aligned}$$
- $M = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}$
- 右侧的二元相等变成了
-
$$M^{\otimes 2} \text{``}[1, 0, 1]\text{''}$$
- 它等价于
$$MEM' = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix} = 2\text{``}[0, 1, 0]\text{''}$$

推广

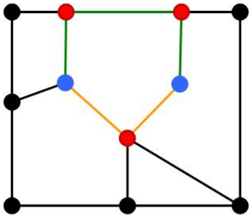
- 把$s(1,0)^{\otimes n}+t(0,1)^{\otimes n}$即$[s,0,\ldots,0,t]$推广到集合$\mathcal{E}$。
- 全息算法的基取$Z=\begin{pmatrix}1 & 1\\ i & -i\end{pmatrix}$。
- 等价的问题选$\#\{\neq_2\}|\langle\mathcal{E}\rangle$。
- 使用Z做基变换，我们得到$\#\{=_2\}|Z\langle\mathcal{E}\rangle$。
- 即$\#\langle Z\mathcal{E}\rangle$ 有多项式时间算法。

非偶图值为0

- 奇数长度的“$\neq_2$”环不能被满足。

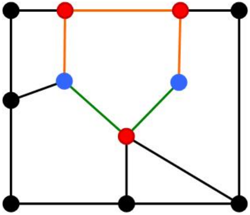
非偶图值为0

- 奇数长度的“$\neq_2$”环不能被满足。
- 在基变换之前的$[x, y, -x, -y, \ldots]$的张量网络中证明：有奇数环则值为零。



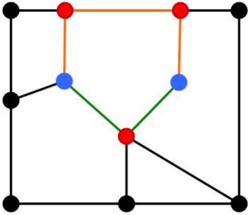
非偶图值为0

- 奇数长度的“$\neq_2$”环不能被满足。
- 在基变换之前的$[x, y, -x, -y, \ldots]$的张量网络中证明：有奇数环则值为零。



非偶图值为0

- 奇数长度的“$\neq_2$”环不能被满足。
- 在基变换之前的$[x, y, -x, -y, \dots]$的张量网络中证明：有奇数环则值为零。



- 历史上次序是反的。我们先发现了偶图时候（利于全息变换）有算法，非偶图的时候值为0；隔了很久，在利用了边其实就是二元相等函数之后，才有了统一的算法。

布尔定义域复数值域的$\#\mathcal{F}$

- 布尔定义域复数值域的$\#\mathcal{F}$的复杂性二分定理open。
- 用$\mathcal{U}$表示所有的一元函数。
- Holant$^*(\mathcal{F})$定义为$\#\mathcal{F} \cup \langle U \rangle$。
- Holant$^*(\mathcal{F})$的二分定理：
  1. $\#\langle H\mathcal{E} \rangle$ 有多项式时间算法。
  2. $\#\langle Z_j\mathcal{E} \rangle$ 有多项式时间算法，$j=1,2$。
     $Z_1 = \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}$, $Z_2 = \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$。
  3. $\#\langle Z_j\mathcal{M} \rangle$ 有多项式时间算法，，$j=1,2$。
  4. $\#\langle \mathcal{T} \rangle$ 有多项式时间算法。
  5. 除以上及其子集，其他集合$\mathcal{F}$定义的Holant$^*(\mathcal{F})$都是#P难的。

$\#\langle\mathcal{T}\rangle$

- 只需说明$\#\mathcal{T}$有多项式时间算法。
- $\mathcal{T}$表示所有一元和二元函数集合。
- 构成的张量网络，最大度不超过2。

$\#\langle Z_j \mathcal{M} \rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。

$\#\langle Z_j \mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。

$\#\langle Z_j \mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
  1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。

$\#\langle Z_j \mathcal{M} \rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
  1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  2. 如果右侧有度为2的顶点T，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。

$\#\langle Z_j\mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
  1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  2. 如果右侧有度为2的顶点T，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  3. 重复以上步骤，直至或者整个张量网络已经计算完成，或者右侧剩下的点的度至少为3。

$\#\langle Z_j\mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
  1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in\mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  2. 如果右侧有度为2的顶点T，它必然通过一个$\neq_2$接在一个$F(\in\mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  3. 重复以上步骤，直至或者整个张量网络已经计算完成，或者右侧剩下的点的度至少为3。
  4. 左边的顶点，给恰好半数的边赋值1，而右边顶点，给严格少于半数的边赋值1。

$\#\langle Z_j\mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
    1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in\mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
    2. 如果右侧有度为2的顶点T，它必然通过一个$\neq_2$接在一个$F(\in\mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
    3. 重复以上步骤，直至或者整个张量网络已经计算完成，或者右侧剩下的点的度至少为3。
    4. 左边的顶点，给恰好半数的边赋值1，而右边顶点，给严格少于半数的边赋值1。
- $\mathcal{M}$ 在被$\neq_2$函数连接成张量网络时，具有封闭性。

$\#\langle Z_j \mathcal{M}\rangle$

- 只需说明$\#\{\neq_2\}|\mathcal{M}$ 有多项式时间算法。
- 一个函数F在$\mathcal{M}$中，iff $F(X)$的函数值为0除非X的汉明权重为0或者1。
  1. 如果右侧有度为1的顶点U，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  2. 如果右侧有度为2的顶点T，它必然通过一个$\neq_2$接在一个$F(\in \mathcal{M})$上，计算这个局部，仍得到$\mathcal{M}$里的函数。
  3. 重复以上步骤，直至或者整个张量网络已经计算完成，或者右侧剩下的点的度至少为3。
  4. 左边的顶点，给恰好半数的边赋值1，而右边顶点，给严格少于半数的边赋值1。
- $\mathcal{M}$ 在被$\neq_2$函数连接成张量网络时，具有封闭性。
- 封闭性，加上$\mathcal{M}$有简洁表达，还因为基本运算易算，亦可得出算法（类似斐波那契门的第一算法）。

几个集合的关系

布尔定义域、复数值域的几个问题集合的关系

- #CSP($\mathcal{F}$) 和Holant*($\mathcal{F}$)，其实就是

几个集合的关系

布尔定义域、复数值域的几个问题集合的关系

- #CSP($\mathcal{F}$) 和Holant*($\mathcal{F}$)，其实就是
- $\#\mathcal{F} \cup \{=_d | d \in \mathbb{N}\}$ 和$\#\mathcal{F} \cup \mathcal{U}$

几个集合的关系

布尔定义域、复数值域的几个问题集合的关系

- #CSP($\mathcal{F}$) 和Holant*($\mathcal{F}$)，其实就是
- $\#\mathcal{F} \cup \{=_d | d \in \mathbb{N}\}$ 和$\#\mathcal{F} \cup \mathcal{U}$
- $\mathcal{F}$变动时，都成为计数问题集合，是$\{\#\mathcal{F} \cup \{[0,1],[1,0]\}\}$的子集。
  (#CSP框架下有一种特殊的方法实现$[0,1],[1,0]$。)

几个集合的关系

布尔定义域、复数值域的几个问题集合的关系

- #CSP($\mathcal{F}$) 和Holant*($\mathcal{F}$)，其实就是
- $\#\mathcal{F} \cup \{=_d | d \in \mathbb{N}\}$ 和$\#\mathcal{F} \cup \mathcal{U}$
- $\mathcal{F}$变动时，都成为计数问题集合，是$\{\#\mathcal{F} \cup \{[0,1],[1,0]\}\}$的子集。
  (#CSP框架下有一种特殊的方法实现$[0,1],[1,0]$。)
- 也是$\#\mathcal{F}$的子集。

几个集合的关系

布尔定义域、复数值域的几个问题集合的关系

- #CSP($\mathcal{F}$) 和Holant*($\mathcal{F}$)，其实就是
- $\#\mathcal{F} \cup \{=_d | d \in \mathbb{N}\}$ 和$\#\mathcal{F} \cup \mathcal{U}$
- $\mathcal{F}$变动时，都成为计数问题集合，是$\{\#\mathcal{F} \cup \{[0,1],[1,0]\}\}$的子集。
  (#CSP框架下有一种特殊的方法实现$[0,1],[1,0]$。)
- 也是$\#\mathcal{F}$的子集。

- 当定义域、输入图、是否要求对称函数等方面的设置不同时，会有很多不可比较的二分定理。
- 例如，图同态的二分定理，虽然它是#CSP框架的特殊情况，但它比上面的布尔#CSP二分定理，在函数定义域设置上更一般。

复杂性与最大度

- 有很多例子，当最大度限制从2到3时，发生从易解到难解的转变。
- 例如，顶点覆盖数目问题，当输入图最大度为2时，易解（练习题）；当输入图最大度为3时，#P困难。
- 有少量例子，这个转变发生在更大的数字之间。
- 本节构造一组人造的例子，对任意整数d，在最大度d和d+1之间可发生复杂性转变。
- $\#\{[1,1],[1,1,2],[1,1,2,3],\ldots,[a_0,a_1,\ldots,a_{d-1},a_d]\}$多项式时间可算。
- 可以证明$\#\{[1,1],[1,1,2],[1,1,2,3],\ldots,[a_0,a_1,\ldots,a_{d-1},a_d],[a_0,a_1,\ldots,a_{d-1},a_d,-2a_d]\}$是#P-难的。

转化为图同态权重和问题

- 方便记号，取$d=3$为例。
- 要把$\#\{[1,1],[1,1,2],[1,1,2,3]\}$和 $\#\{[1,1],[1,1,2],[1,1,2,3],[1,1,2,3,-6]\}$, 分别转化成输入图最大度为3和4的“$H-$同态权重和”问题，
即分别转化成大定义域下的 $\#\{=_1,=_2,=_3\}|H$ 和$\#\{=_1,=_2,=_3,=_4\}|H$。

函数也可强行被转化为变量

- 张量网络里边变量的本质是二元相等函数。
- 一个d元函数也可被强行被转化为一个出现d次的变量（即d元相等函数），代价是边也被同时全息变换了。
- 预热：
构造二元函数T，把$\#\{[1,1,2,3]\}|\{=_2\}$转化为$\#\{=_3\}|\{T\}$。
- $$\text{“}[1,1,2,3]\text{”} = c_1(1,1)^{\otimes 3} + c_2(1,2)^{\otimes 3} + c_3(1,3)^{\otimes 3} + c_4(1,4)^{\otimes 3}$$
$$= (c_1,0,\ldots,0,c_2,0,\ldots,0,c_3,0,\ldots,0,c_4)\begin{pmatrix}1 & 1\\ 1 & 2\\ 1 & 3\\ 1 & 4\end{pmatrix}^{\otimes 3}$$
- 通过把$c_i^{\frac{1}{3}}$送入张量幂中，把每个c_i变成1。

有解和同时转化

- $$\text{“}[1,1,2,3]\text{”} = c_1(1,1)^{\otimes 3} + c_2(1,2)^{\otimes 3} + c_3(1,3)^{\otimes 3} + c_4(1,4)^{\otimes 3}$$
  $$= c_1[1,1,1,1] + c_2[1,2,4,8] + c_3[1,3,9,27] + c_4[1,4,16,64]$$
- 满秩的范德蒙矩阵。一定有解c_1, c_2, c_3, c_4。
- 此解还能满足低元的函数，例如：
  $$\text{“}[1,1,2]\text{”} = c_1(1,1)^{\otimes 2} + c_2(1,2)^{\otimes 2} + c_3(1,3)^{\otimes 2} + c_4(1,4)^{\otimes 2}$$
- 我们只为最高的$d+1$元函数求解。
- 整系数的线性方程组的解一定是有理数解（根据克莱姆法则）。
- 乘以公分母D，“$[1,1,2,3]$”与“$D[1,1,2,3]$”带来相同的计算复杂性。

处理正整数c_i和负整数c_i

- 不妨假设解都是整数。
- 有不同的函数，不同的元数，不能再用$\sqrt[d+1]{c_i}$招。
- “赖皮招”：
- 假如c_i是非负整数，c_i个$(1,i)^{\otimes d+1}$相加代替$c_i(1,i)^{\otimes d+1}$。
- 假如c_i是负整数，只需看如何实现-1。
  设r是1的$d+2$次单位根$e^{\frac{2\pi}{d+2}\mathrm{i}}$，$b_1=r, b_2=r^2, \ldots, b_{d+1}=r^{d+1}$。
  对任意$1\le j\le d+1$，所有b_l的贡献为

$$\sum_{l=1}^{d+1} b_l^j = \sum_{l=1}^{d+1} r^{jl} = \sum_{l=0}^{d+1} r^{jl} - 1 = \frac{1-r^{j(d+2)}}{1-r^j} - 1 = -1。$$

布尔函数的傅里叶形式

- $F : \{0,1\}^n \to \{0,1\}$
- 线性函数：
  $F(X+Y) = F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

布尔函数的傅里叶形式

- $F : \{0,1\}^n \to \{0,1\}$
- 线性函数：
  $F(X+Y) = F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

- $F : \{1,-1\}^n \to \{1,-1\}$
- 线性函数：
  $F(X \odot Y) = F(X)F(Y)$
- $F(X) = \prod_{j\in s} x_j$
  记为$\chi_s(X)$

布尔函数的傅里叶形式

- $F : \{0,1\}^n \to \{0,1\}$
- 线性函数：$F(X+Y) = F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

- $F : \{1,-1\}^n \to \{1,-1\}$
- 线性函数：$F(X \odot Y) = F(X)F(Y)$
- $F(X) = \prod_{j\in s} x_j$ 记为$\chi_s(X)$

- F的真值表是一个2^n维向量。有理数域的2^n维向量空间。

布尔函数的傅里叶形式

- $F:\{0,1\}^n \to \{0,1\}$
- 线性函数：$F(X+Y)=F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

- $F:\{1,-1\}^n \to \{1,-1\}$
- 线性函数：$F(X \odot Y) = F(X)F(Y)$
- $F(X) = \prod_{j\in s} x_j$ 记为$\chi_s(X)$

- F的真值表是一个2^n维向量。有理数域的2^n维向量空间。
- 性质：线性函数集合$\{\chi_s | s \subseteq [n]\}$是一组基。

布尔函数的傅里叶形式

- $F:\{0,1\}^n \to \{0,1\}$
- 线性函数：$F(X+Y)=F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

- $F:\{1,-1\}^n \to \{1,-1\}$
- 线性函数：$F(X\odot Y)=F(X)F(Y)$
- $F(X)=\prod_{j\in s} x_j$ 记为$\chi_s(X)$

- F的真值表是一个2^n维向量。有理数域的2^n维向量空间。
- 性质：线性函数集合$\{\chi_s|s\subseteq[n]\}$是一组基。

Proof.

Hadmard矩阵$H=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$满秩。

$H^{\otimes n}$的行就是$\{\chi_s|s\subseteq[n]\}$。 □

布尔函数的傅里叶形式

- $F : \{0, 1\}^n \to \{0, 1\}$
- 线性函数：$F(X+Y) = F(X)+F(Y)$
- $F(X) = a_1x_1 + a_2x_2 + \cdots + c_nx_n$

- $F : \{1, -1\}^n \to \{1, -1\}$
- 线性函数：$F(X \odot Y) = F(X)F(Y)$
- $F(X) = \prod_{j\in s} x_j$ 记为$\chi_s(X)$

- F的真值表是一个2^n维向量。有理数域的2^n维向量空间。
- 性质：线性函数集合$\{\chi_s | s \subseteq [n]\}$是一组基。

Proof.

Hadmard矩阵$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$满秩。

$H^{\otimes n}$的行就是$\{\chi_s | s \subseteq [n]\}$。 □

- F的傅里叶形式，就是它在线性函数基下的坐标向量，

$$\hat{F} = H^{\otimes n} F, \quad \hat{F}(s) = \langle F, \chi_s \rangle$$

F与线性函数的距离

$\hat{F}$的长度:

- $\langle F, F\rangle = 2^n$
- $\hat{F} = H^{\otimes n}F$
- $\langle \hat{F}, \hat{F}\rangle = 2^{2n}$

F与线性函数的距离:

F与线性函数的距离

$\hat{F}$的长度:

- $\langle F, F\rangle = 2^n$
- $\hat{F} = H^{\otimes n}F$
- $\langle \hat{F}, \hat{F}\rangle = 2^{2n}$

F与线性函数的距离:

- $\hat{F}(s) = \langle F, \chi_s\rangle$
- $\hat{F}(s)$等于F和χ_s的相同点数目减去不同点数目。
- “F与所有χ_s的最小相对距离是ρ”，（相对距离：不同点数除以总点数）等价于，“$\forall s$, $\hat{F}(s) \le (1-2\rho)2^n$”。

线性检测

- 目的：检测F是否接近一个线性函数。
- 随机检测算法：随机抽取X, Y，如果$F(X \odot Y) = F(X)F(Y)$，就通过检测；否则，拒绝。
- 用p表示F被检测拒绝的概率。
- 如果F是线性函数，$p = 0$。

线性检测

- 目的：检测F是否接近一个线性函数。
- 随机检测算法：随机抽取X, Y，如果$F(X \odot Y) = F(X)F(Y)$，就通过检测；否则，拒绝。
- 用p表示F被检测拒绝的概率。
- 如果F是线性函数，$p = 0$。

- 定理

如果F与所有χ_s的最小相对距离是ρ，拒绝概率$p \geq \rho$。

线性检测

- 目的：检测F是否接近一个线性函数。
- 随机检测算法：随机抽取X,Y，如果$F(X\odot Y)=F(X)F(Y)$，就通过检测；否则，拒绝。
- 用p表示F被检测拒绝的概率。
- 如果F是线性函数，$p=0$。

- **定理**
如果F与所有χ_s的最小相对距离是ρ，拒绝概率$p\geq\rho$。
  - 证明：

$$\sum_s \hat{F}^3(s) \leq \max_s \hat{F}(s)\ \langle \hat{F}, \hat{F}\rangle = (1-2\rho)2^{3n}$$

$(1-2p)2^{2n}=\sum_{X,Y}F(X\odot Y)F(X)F(Y)$是线性检测抽样的$2^{2n}$组$(X,Y)$中，通过的数目减去被拒绝的数目，只需再证明这个数目是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$。

先看一下H^2的傅里叶形式。

先看一下H^2的傅里叶形式。

$$\hat{H}^2(S) = \sum_X H(X)H(X)\chi_S(X)$$

$$= \sum_X (\sum_U \hat{H}(U)\chi_X(U))(\sum_V \hat{H}(V)\chi_X(V))\chi_S(X)$$

$$= \sum_U \sum_V \hat{H}(U)\hat{H}(V) \sum_X \chi_X(U)\chi_X(V)\chi_S(X)$$

先看一下H^2的傅里叶形式。

$$\hat{H}^2(S) = \sum_X H(X)H(X)\chi_S(X)$$

$$= \sum_X (\sum_U \hat{H}(U)\chi_X(U))(\sum_V \hat{H}(V)\chi_X(V))\chi_S(X)$$

$$= \sum_U \sum_V \hat{H}(U)\hat{H}(V) \sum_X \chi_X(U)\chi_X(V)\chi_S(X)$$

$$\chi_X(U)\chi_X(V) = \chi_X(U \triangle V) = \chi_{U \triangle V}(X)$$

$$\sum_X \chi_X(U)\chi_X(V)\chi_S(X) = \langle \chi_{U\triangle V}, \chi_S \rangle$$

先看一下H^2的傅里叶形式。

$$\hat{H}^2(S) = \sum_X H(X)H(X)\chi_S(X)$$

$$= \sum_X (\sum_U \hat{H}(U)\chi_X(U))(\sum_V \hat{H}(V)\chi_X(V))\chi_S(X)$$

$$= \sum_U \sum_V \hat{H}(U)\hat{H}(V) \sum_X \chi_X(U)\chi_X(V)\chi_S(X)$$

$$= \sum_{U,V:U\triangle V=S} \hat{H}(U)\hat{H}(V)$$

$$\chi_X(U)\chi_X(V) = \chi_X(U\triangle V) = \chi_{U\triangle V}(X)$$

$$\sum_X \chi_X(U)\chi_X(V)\chi_S(X) = \langle \chi_{U\triangle V}, \chi_S \rangle$$

$\sum_s \hat{F}^3(s)$与$\sum_{X,Y} F(X\odot Y)F(X)F(Y)$

- $$\hat{H}^2(S) = \sum_{U,V:U\triangle V=S} \hat{H}(U)\hat{H}(V)$$

$\sum_s \hat{F}^3(s)$与$\sum_{X,Y} F(X\odot Y)F(X)F(Y)$

- $$\hat{H}^2(S) = \sum_{U,V:U\triangle V=S} \hat{H}(U)\hat{H}(V)$$
- $$\sum_s \hat{F}^3(S) = \langle \hat{F}, \hat{F}^2 \rangle$$

$\sum_s \hat{F}^3(s)$与$\sum_{X,Y} F(X\odot Y)F(X)F(Y)$

- $$\hat{H}^2(S) = \sum_{U,V:U\triangle V=S} \hat{H}(U)\hat{H}(V)$$

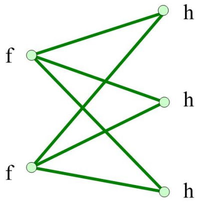
- $$\sum_s \hat{F}^3(S) = \langle \hat{F}, \hat{F}^2\rangle$$
$$= \langle F, \hat{\hat{F}^2}\rangle$$

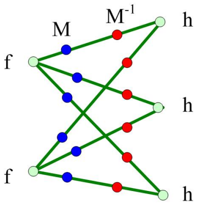
$\sum_s \hat{F}^3(s)$与$\sum_{X,Y} F(X \odot Y)F(X)F(Y)$

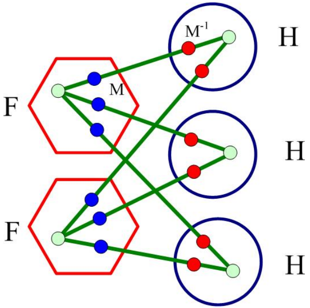
- $$\hat{H}^2(S) = \sum_{U,V:U \triangle V = S} \hat{H}(U)\hat{H}(V)$$
- $$\sum_s \hat{F}^3(S) = \langle \hat{F}, \hat{F}^2 \rangle$$
$$= \langle F, \hat{\hat{F}^2} \rangle$$
$$= \sum_S F(S) \sum_{U,V:S=U\triangle V} F(U)F(V)$$
$$= \sum_{U,V,S:S=U\odot V} F(S)F(U)F(V)$$

回顾：全息归约

$AB = AEB = AMM^{-1}B = (AM)(M^{-1}B)$。($E$是单位阵)







定理

$\#\{F\}|\{H\}$和$\#\{f\}|\{h\}$在相同的图上的值相等。其中，

$$F = fM^{\otimes 3},$$

$$(M^{-1})^{\otimes 2}g = G。$$

$$
\begin{aligned}
&\text{“}[1,0,0,1]\text{”}H^{\otimes 3}\\
&=(1,0,0,0,0,0,0,1)H^{\otimes 3}\\
&=((1,0)^{\otimes 3}+(0,1)^{\otimes 3})H^{\otimes 3}\\
&=((1,0)^{\otimes 3}+(0,1)^{\otimes 3})H^{\otimes 3}\\
&=(1,1)^{\otimes 3}+(1,-1)^{\otimes 3}\\
&=\text{“}[2020]\text{”}
\end{aligned}
$$

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X \odot Y)$。

全息归约：

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。

全息归约：

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1,x_2,\ldots,x_n$，第二个函数F的n条边是$y_1,y_2,\ldots,y_n$，第二个函数F的n条边是$z_1,z_2,\ldots,z_n$。

全息归约：

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X \odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1, x_2, \ldots, x_n$，第二个函数F的n条边是$y_1, y_2, \ldots, y_n$，第二个函数F的n条边是$z_1, z_2, \ldots, z_n$。
- 希望对任何$j = 1, 2, \ldots, n$，$z_j = x_j \odot y_j$。

全息归约：

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1,x_2,\ldots,x_n$，第二个函数F的n条边是$y_1,y_2,\ldots,y_n$，第二个函数F的n条边是$z_1,z_2,\ldots,z_n$。
- 希望对任何$j=1,2,\ldots,n$，$z_j=x_j\odot y_j$。
- 左侧：第j个函数是$[1,0,1,0]$（记为$\oplus_3$），作用于x_j,y_j,z_j。

全息归约：

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1, x_2, \ldots, x_n$，第二个函数F的n条边是$y_1, y_2, \ldots, y_n$，第二个函数F的n条边是$z_1, z_2, \ldots, z_n$。
- 希望对任何$j = 1, 2, \ldots, n$，$z_j = x_j \odot y_j$。
- 左侧：第j个函数是$[1, 0, 1, 0]$（记为$\oplus_3$），作用于x_j, y_j, z_j。

全息归约：

- 用H作用于右侧三个函数F。

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1, x_2, \ldots, x_n$，第二个函数F的n条边是$y_1, y_2, \ldots, y_n$，第二个函数F的n条边是$z_1, z_2, \ldots, z_n$。
- 希望对任何$j = 1, 2, \ldots, n$，$z_j = x_j \odot y_j$。
- 左侧：第j个函数是$[1, 0, 1, 0]$（记为$\oplus_3$），作用于x_j, y_j, z_j。

全息归约：

- 用H作用于右侧三个函数F。
- 用$\frac{1}{2}H$作用于左侧n个函数$\oplus_3$。

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X \odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1, x_2, \ldots, x_n$，第二个函数F的n条边是$y_1, y_2, \ldots, y_n$，第二个函数F的n条边是$z_1, z_2, \ldots, z_n$。
- 希望对任何$j = 1, 2, \ldots, n$，$z_j = x_j \odot y_j$。
- 左侧：第j个函数是$[1, 0, 1, 0]$（记为$\oplus_3$），作用于x_j, y_j, z_j。

全息归约：

- 用H作用于右侧三个函数F。
- 用$\frac{1}{2}H$作用于左侧n个函数$\oplus_3$。
- 右侧变成$\hat{F}$。

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X\odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1,x_2,\ldots,x_n$，第二个函数F的n条边是$y_1,y_2,\ldots,y_n$，第二个函数F的n条边是$z_1,z_2,\ldots,z_n$。
- 希望对任何$j=1,2,\ldots,n$，$z_j=x_j\odot y_j$。
- 左侧：第j个函数是$[1,0,1,0]$（记为$\oplus_3$），作用于x_j,y_j,z_j。

全息归约：

- 用H作用于右侧三个函数F。
- 用$\frac{1}{2}H$作用于左侧n个函数$\oplus_3$。
- 右侧变成$\hat{F}$。
- 左侧变成$\frac{1}{8}$“$[1,0,1,0]$”$H^{\otimes 3}=\frac{1}{2}$“$[1,0,0,1]$”。

全息归约：线性检测的张量网络值是$\frac{1}{2^n}\sum_s \hat{F}^3(s)$

线性检测的张量网络：

- 其值应是线性检测中，通过的数目减去被拒绝的数目，即$\sum_{X,Y} F(X)F(Y)F(X \odot Y)$。
- 右侧：三个函数F。
- 第一个函数F的n条边是$x_1, x_2, \ldots, x_n$，第二个函数F的n条边是$y_1, y_2, \ldots, y_n$，第二个函数F的n条边是$z_1, z_2, \ldots, z_n$。
- 希望对任何$j = 1, 2, \ldots, n$，$z_j = x_j \odot y_j$。
- 左侧：第j个函数是$[1, 0, 1, 0]$（记为$\oplus_3$），作用于x_j, y_j, z_j。

全息归约：

- 用H作用于右侧三个函数F。
- 用$\frac{1}{2}H$作用于左侧n个函数$\oplus_3$。
- 右侧变成$\hat{F}$。
- 左侧变成$\frac{1}{8}$“$[1, 0, 1, 0]$”$H^{\otimes 3} = \frac{1}{2}$“$[1, 0, 0, 1]$”。
- 新左侧要求新三组边X', Y', Z'相同。

线性检测的全息归约看法

- 以Hadmard矩阵为基的全息归约变换，把线性检测中的按概率求$F(X)F(Y)F(Z)$和，其中的X, Y, Z之间的奇偶性约束，转化成了相等约束$X = Y = Z$的同时，也把F转化成了傅里叶形式$\hat{F}$，变成了求$\hat{F}^3(X)$和，进而通过$\max_X \hat{F}(X)$把检测拒绝的概率和与线性函数的距离联系起来。

- Hadmard矩阵$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- k元相等函数$=_k$，$[1, 0, \dots, 0, 1]$。
- k元奇偶性函数$\oplus_k$，$[1, 0, 1, 0 \dots]$。
- H变$=_k$为$\oplus_k$，变$\oplus_k$为$=_k$。
- 布尔定义域的线性子空间，是一个齐次线性方程组的解集合。

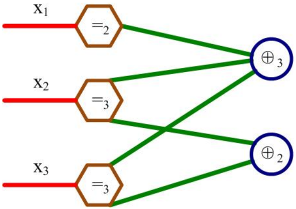
例

$$\begin{cases} x_1 + x_2 + x_3 & = & 0 \\ x_2 + x_3 & = & 0 \end{cases}$$

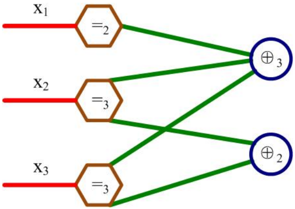
χ_L的张量网络

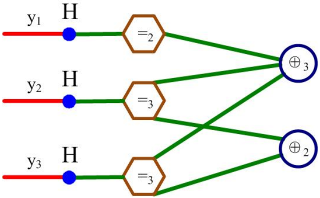
χ_L的张量网络

χ_L的张量网络

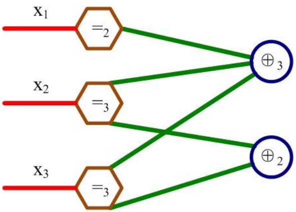


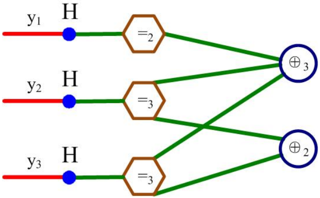
χ_L的张量网络

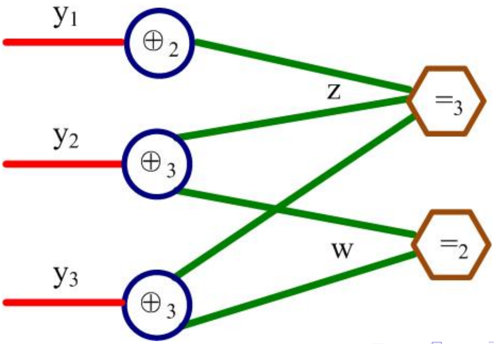




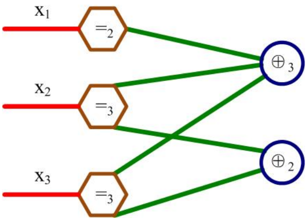
χ_L的张量网络



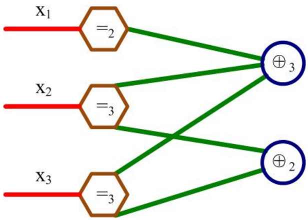




“正交补空间”

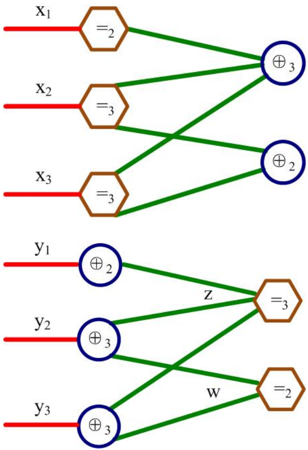


“正交补空间”



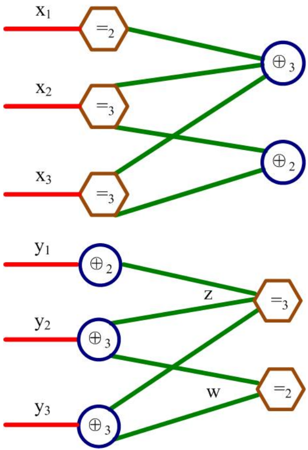
$$\begin{cases} x_1 + x_2 + x_3 & = & 0 \\ x_2 + x_3 & = & 0 \end{cases}$$

“正交补空间”



$$\begin{cases} x_1 + x_2 + x_3 & = & 0 \\ x_2 + x_3 & = & 0 \end{cases}$$

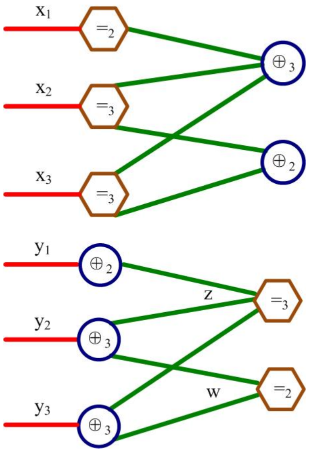
“正交补空间”



$$\begin{cases} x_1 + x_2 + x_3 &= 0 \\ x_2 + x_3 &= 0 \end{cases}$$

$$\begin{cases} y_1 &= z \\ y_2 &= z + w \\ y_3 &= z + w \end{cases}$$

“正交补空间”



$$\begin{cases} x_1 + x_2 + x_3 & = & 0 \\ x_2 + x_3 & = & 0 \end{cases}$$

$$\begin{cases} y_1 & = & z \\ y_2 & = & z + w \\ y_3 & = & z + w \end{cases}$$

《Analysis of Boolean Functions》Ryan O'donnell
Proposition 3.11
Proposition 3.12 generalizes it to affine subspace.

容斥原理

- $$\left|\bigcap_{i=1}^{n} \bar{A}_i\right| = \left|S - \bigcup_{i=1}^{n} A_i\right| =$$
$$|S| - \sum_{i=1}^{n} |A_i| + \sum_{1 \le i < j \le n} |A_i \cap A_j| - \ldots + (-1)^n |A_1 \cap \cdots \cap A_n|.$$

容斥原理

- $$\left|\bigcap_{i=1}^{n}\bar{A}_i\right| = \left|S - \bigcup_{i=1}^{n}A_i\right| =$$
$$|S| - \sum_{i=1}^{n}|A_i| + \sum_{1\le i<j\le n}|A_i\cap A_j| - \ldots + (-1)^n|A_1\cap\cdots\cap A_n|.$$
- 假设一个元素出现在$k>0$个集合中，在右边它被计数
$$1-\binom{k}{1}+\binom{k}{2}-\binom{k}{3}-\ldots+(-1)^k\binom{k}{k}=0$$

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。
- n个集合$\{A_1, \ldots, A_n\}$，把全集划分成2^n部分。

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。
- n个集合$\{A_1, \ldots, A_n\}$，把全集划分成2^n部分。
- 第$X = (x_1, \ldots, x_n)$部分的大小，记为$F(X)$。$x_i \in \{0, 1\}$。

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。
- n个集合$\{A_1,\ldots,A_n\}$，把全集划分成2^n部分。
- 第$X=(x_1,\ldots,x_n)$部分的大小，记为$F(X)$。$x_i\in\{0,1\}$。
- 记$A_i(0)=A_i$，$A_i(1)=\bar{A}_i$。

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。
- n个集合$\{A_1,\ldots,A_n\}$，把全集划分成2^n部分。
- 第$X=(x_1,\ldots,x_n)$部分的大小，记为$F(X)$。$x_i\in\{0,1\}$。
- 记$A_i(0)=A_i$，$A_i(1)=\bar{A}_i$。
- $$F(X)=|\bigcap_{i=1}^{n}A_i(x_i)|$$

集合与布尔函数

- 一个集合A，把全集划分成A和$\bar{A}$两部分。
- n个集合$\{A_1, \ldots, A_n\}$，把全集划分成2^n部分。
- 第$X = (x_1, \ldots, x_n)$部分的大小，记为$F(X)$。$x_i \in \{0, 1\}$。
- 记$A_i(0) = A_i$，$A_i(1) = \bar{A}_i$。
- $$F(X) = |\bigcap_{i=1}^{n} A_i(x_i)|$$

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。
- 第$Y=(y_1,\ldots,y_n)$部分，记为$H(Y)$。$x_i\in\{*,1\}$。

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。
- 第$Y = (y_1, \dots, y_n)$部分，记为$H(Y)$。$x_i \in \{*, 1\}$。
- 记$A_i(*) = S$ ，$A_i(1) = \bar{A}_i$。

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。
- 第$Y = (y_1, \ldots, y_n)$部分，记为$H(Y)$。$x_i \in \{*, 1\}$。
- 记$A_i(*) = S$，$A_i(1) = \bar{A}_i$。
- $$H(Y) = |\bigcap_{i=1}^{n} A_i(y_i)|$$

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。
- 第$Y=(y_1,\dots,y_n)$部分，记为$H(Y)$。$x_i\in\{*,1\}$。
- 记$A_i(*)=S$，$A_i(1)=\bar{A}_i$。
- $$H(Y)=|\bigcap_{i=1}^{n}A_i(y_i)|$$
- $$H(*1\cdots1)=|S\cap\bigcap_{i=2}^{n}A_i(y_i)|$$

$$=|\bar{A}_1\cap\bigcap_{i=2}^{n}A_i(y_i)|+|A_1\cap\bigcap_{i=2}^{n}A_i(y_i)|=F(01\cdots1)+F(11\cdots1)$$

集合与布尔函数

- 有时，在第i个维度不想使用$\bar{A}_i$或A_i，而是想用全集S或A_i。
- 第$Y = (y_1, \ldots, y_n)$部分，记为$H(Y)$。$x_i \in \{*, 1\}$。
- 记$A_i(*) = S$，$A_i(1) = \bar{A}_i$。
- $$H(Y) = |\bigcap_{i=1}^{n} A_i(y_i)|$$
- $$H(*1\cdots1) = |S \cap \bigcap_{i=2}^{n} A_i(y_i)|$$

$$= |\bar{A}_1 \cap \bigcap_{i=2}^{n} A_i(y_i)| + |A_1 \cap \bigcap_{i=2}^{n} A_i(y_i)| = F(01\cdots1) + F(11\cdots1)$$

- $$H(*1\cdots1) = \langle (1,1) \otimes (0,1)^{\otimes n-1}, F \rangle$$

集合与布尔函数

- $$H(*1\cdots 1) = \langle (1,1)\otimes(0,1)^{\otimes n-1}, F\rangle$$

集合与布尔函数

- $$H(*1\cdots 1) = \langle (1,1)\otimes(0,1)^{\otimes n-1}, F\rangle$$
- $$H = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes n} F$$

集合与布尔函数

- $$H(*1\cdots1) = \langle (1,1)\otimes(0,1)^{\otimes n-1}, F\rangle$$
- $$H = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes n} F$$
- $$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}^{\otimes n} H = F$$

集合与布尔函数

- $$H(*1\cdots1)=\langle(1,1)\otimes(0,1)^{\otimes n-1},F\rangle$$
- $$H=\begin{pmatrix}1&1\\0&1\end{pmatrix}^{\otimes n}F$$
- $$\begin{pmatrix}1&-1\\0&1\end{pmatrix}^{\otimes n}H=F$$
- $$F(0\cdots0)=(1,-1)^{\otimes n}H$$

集合与布尔函数

- $$H(*1\cdots 1) = \langle (1,1) \otimes (0,1)^{\otimes n-1}, F\rangle$$
- $$H = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{\otimes n} F$$
- $$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}^{\otimes n} H = F$$
- $$F(0\cdots 0) = (1,-1)^{\otimes n} H$$
- $$\left|\bigcap_{i=1}^{n} \bar{A}_i\right|$$

$$= |S| - \sum_{i=1}^{n} |A_i| + \sum_{1\le i<j\le n} |A_i \cap A_j| - \ldots + (-1)^n |A_1 \cap \cdots \cap A_n|.$$

这两个变换有名字：Zeta和Möbius变换

- $$(\zeta f)X = \sum_{Y \subseteq X} f(Y)$$
  $$(\mu f)X = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y)$$

这两个变换有名字：Zeta和Möbius变换

- $$(\zeta f)X = \sum_{Y \subseteq X} f(Y)$$

$$(\mu f)X = \sum_{Y \subseteq X} (-1)^{|X \setminus Y|} f(Y)$$

- http://resources.mpi-inf.mpg.de/conferences/adfocs13/material.htm ADFOCS 2013 Lukasz Kowalik's talk slides, page 22

回顾积和式——偶图的带权完美匹配数目

- M是$n \times n$矩阵。

回顾积和式——偶图的带权完美匹配数目

- M是$n \times n$矩阵。

定义

$$\mathit{Perm}(M) = \sum_{\pi \in S_n} \prod_{j=1}^{n} M_{j,\pi(j)}$$

回顾积和式——偶图的带权完美匹配数目

- M是$n \times n$矩阵。

定义

$$\mathit{Perm}(M) = \sum_{\pi \in S_n} \prod_{j=1}^{n} M_{j,\pi(j)}$$

- 按照此定义式计算，需要$\mathcal{O}((n-1)\ n!)$次乘法，$2^{\Omega(n \log n)}$量级。

Ryser公式

- $$\mathbf{M} = \begin{pmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$
- $\phi(M) = (a_{11} + a_{12} + \cdots + a_{1n})(a_{21} + a_{22} + \cdots + a_{2n}) \cdots (a_{n1} + a_{n2} + \cdots + a_{nn})$ 定义为矩阵（可以非方阵）列元素和的积。
- $\phi(M)$展开的项作为全集，即每行出一项的n项的乘积。
- Perm(M)所求和的，是每行出一项并且每列出一项的n项的乘积。
- 事件A_i表示，乘积中的n项，满足每行有一项，第i列没有项。
- $$\text{Perm}(M) = \phi(M) - A_1 - A_2 \cdots - A_n + A_1 \cap A_2 + \cdots$$

Ryser公式

- 令
$$S_k = \sum_{B\text{是}M\text{的}n\times(n-k)\text{子式}} \phi(B)$$
-
$$\text{Perm}(M) = \sum_{k=0}^{n-1} (-1)^k S_k$$

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V=\{1,2,\ldots,n\}$，$U=\{1',2',\ldots,n'\}$。
  无向偶图$H(V,U,E,W)$中，边(j,k')的权重$W(j,k)=A_{j,k}$。

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V=\{1,2,\ldots,n\}$，$U=\{1',2',\ldots,n'\}$。
  无向偶图$H(V,U,E,W)$中，边(j,k')的权重$W(j,k)=A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$，$U = \{1', 2', \ldots, n'\}$。
  无向偶图$H(V, U, E, W)$中，边(j, k')的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。
- U, V的点都用函数$[0, 1, 0, \ldots, 0]$。

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$，$U = \{1', 2', \ldots, n'\}$。
  无向偶图$H(V, U, E, W)$中，边(j, k')的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。
- U, V的点都用函数$[0, 1, 0, \ldots, 0]$。
- Permanent(A)是张量网络H的值。

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V = \{1, 2, \ldots, n\}$，$U = \{1', 2', \ldots, n'\}$。
  无向偶图$H(V, U, E, W)$中，边(j, k')的权重$W(j, k) = A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。
- U, V的点都用函数$[0, 1, 0, \ldots, 0]$。
- Permanent(A)是张量网络H的值。
- 考虑张量网络H'，把V的点的函数换成$[0, 1, \ldots, 1]$，值不变。（想想原因）

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V=\{1,2,\ldots,n\}$，$U=\{1',2',\ldots,n'\}$。
  无向偶图$H(V,U,E,W)$中，边(j,k')的权重$W(j,k)=A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。
- U,V的点都用函数$[0,1,0,\ldots,0]$。
- Permanent(A)是张量网络H的值。
- 考虑张量网络H'，把V的点的函数换成$[0,1,\ldots,1]$，值不变。（想想原因）
- $[0,1,\ldots,1]=(1,1)^{\otimes n}-(1,0)^{\otimes n}$

全息归约解释

- A有两种图表示：n个点的有向图，和$2n$个点的偶图。
- $V=\{1,2,\ldots,n\}$，$U=\{1',2',\ldots,n'\}$。
  无向偶图$H(V,U,E,W)$中，边(j,k')的权重$W(j,k)=A_{j,k}$。
- Permanent(A)是H的所有完美匹配权重之和。
- U,V的点都用函数$[0,1,0,\ldots,0]$。
- Permanent(A)是张量网络H的值。
- 考虑张量网络H'，把V的点的函数换成$[0,1,\ldots,1]$，值不变。（想想原因）
- $[0,1,\ldots,1]=(1,1)^{\otimes n}-(1,0)^{\otimes n}$

全息归约解释

- $[0,1,\ldots,1] = (1,1)^{\otimes n} - (1,0)^{\otimes n}$

全息归约解释

- $[0,1,\ldots,1]=(1,1)^{\otimes n}-(1,0)^{\otimes n}$
- V的n个点，从函数$\{(1,1)^{\otimes n},-(1,0)^{\otimes n}\}$中选一个，$2^n$种选法，选完后是$n$个星，易算。

全息归约解释

- $[0,1,\ldots,1]=(1,1)^{\otimes n}-(1,0)^{\otimes n}$
- V的n个点，从函数$\{(1,1)^{\otimes n},-(1,0)^{\otimes n}\}$中选一个，$2^n$种选法，选完后是$n$个星，易算。
- （对应容斥原理中的2^n个项，例如，若都选$(1,1)^{\otimes n}$，值是$\phi(A)=(a_{11}+a_{12}+\cdots+a_{1n})(a_{21}+a_{22}+\cdots+a_{2n})\cdots(a_{n1}+a_{n2}+\cdots+a_{nn})$。）

全息归约解释

- $[0, 1, \ldots, 1] = (1, 1)^{\otimes n} - (1, 0)^{\otimes n}$
- V的n个点，从函数$\{(1, 1)^{\otimes n}, -(1, 0)^{\otimes n}\}$中选一个，$2^n$种选法，选完后是$n$个星，易算。
- （对应容斥原理中的2^n个项，例如，若都选$(1, 1)^{\otimes n}$，值是$\phi(A) = (a_{11}+a_{12}+\cdots+a_{1n})(a_{21}+a_{22}+\cdots+a_{2n})\cdots(a_{n1}+a_{n2}+\cdots+a_{nn})$。）
- （函数也可看作变量。）

- 一般图的完美匹配数目也有2^npoly(n)算法，有没有全息归约解释?
- 张量网络H和H'的值相等，是全息归约定理逆命题不成立的情况之一。
- Jin-Yi Cai, Heng Guo, Tyson Williams:
  A complete dichotomy rises from the capture of vanishing signatures: extended abstract. STOC 2013: 635-644

参考文献

- 斐波那契门算法和$\#([x, y, -x, -y])$的算法见：
Jin-Yi Cai, Pinyan Lu, Mingji Xia:
Computational Complexity of Holant Problems. SIAM J. Comput. 40(4): 1101-1132 (2011)
- 线性检测和线性子空间指标函数的傅里叶形式见：
《Analysis of Boolean Functions》Ryan O'Donnell
（张量网络和全息归约还可解释此书其他一些内容）
- 积和式的Ryser公式算法见：
https://en.wikipedia.org/wiki/Computing_the_permanent