# 哈密顿图判定问题的多项式时间算法

姜新文 **

**摘要** 本文给出一个称为$MSP$问题的定义及其多项式时间判定算法。已经证明包括哈密顿图判定问题在内的十多个$NP$完全问题可以多项式归结到$MSP$问题。本文结果暗示$NP=P$。

**关键词** 算法 $MSP$问题 $HC$问题 $NP$完全问题 多项式时间算法

# 1 问题引入及若干定义

哈密顿图判定问题是$NP$完全问题[1]。关于该问题的多项式时间算法研究，多年来未取得明显进展。最近几年值得一提的工作是惠普的 Vinay Deolalikar 宣布证明了$NP\neq P$。他的工作否定了哈密顿图判定问题多项式时间算法的存在性。然而不幸，人们很快找出了他证明中的错误[2]。本文给出该判定问题的一个多项式时间算法。

为了求解哈密顿图判定问题，我们需要对问题进行转换。为缩短证明长度，以分段确认，分割围歼，综合众多意见，我们提出$MSP$问题，并通过将哈密顿图判定问题等十多个$NP$完全问题，多项式归结到$MSP$问题，证明了$MSP$问题的$NP$完全性[3~8]。本文的注意力集中在$MSP$问题多项式时间判定上。

算法是一些动作的有序集合。为一个问题设计算法是容易的，将一些动作凑在一起即可。设计算法的困难性和严肃性在于，你的算法是否实现了你的计算目标？于是需要证明。而如何证明常常让人无从着手。

$MSP$问题是一个人工构造的问题，它的特别的结构特性，使我们找到了多项式时间算法并证明其正确性。为了本文的完整性，我们首先转述对$MSP$问题的定义[3~7]，同时采用比较直观的写法。

**定义 1** 称$G=<V,E,S,D,L,\lambda>$是一个加标多级图(labeled multistage graph)，如果满足以下条件：

1. $V$为顶点集合，$L$称为$G$的级。$V=V_0\cup V_1\cup V_2\cup\cdots\cup V_L$，$V_i\cap V_j=\emptyset$，$0\le i,j\le L$，$i\neq j$。如果$u\in V_i$ $(0\le i\le L)$，称$u$所在级为$i$级，也称$u$是$i$级的顶点。
2. $V_0$和$V_L$都只包含唯一顶点。称$V_0$中的唯一顶点为源点，记为$S$，称$V_L$中的唯一顶点为汇点，记为$D$。
3. $E$为边的集合，$E$中的边均为有向边。用三元组$\langle u,v,l\rangle$表示一条$u$到$v$的边。如果$\langle u,v,l\rangle\in E$ $(1\le l\le L)$，则$u\in V_{l-1}$，$v\in V_l$。称$\langle u,v,l\rangle$为$G$的第 $l$ 级的边。
4. λ是一个从$V-\{S\}$ 到$2^E$的映射。对每个顶点$v\in V-\{S\}$，$\lambda(v)\subseteq E$。称$\lambda(v)$为顶点$v$的边集。

上述定义中，用三元组$\langle u,v,l\rangle$表示边，而不是通常的采用二元组$\langle u,v\rangle$，我们有特殊的考虑。因为我们在算法处理过程中总是需要知道边的起点、终点以及所在的级，用三元组表示，处理起来更为直观，而且对于复杂性的把握变得更加直接。

看两个例子。图1所示的两个图，都是加标多级图。各个顶点的边集的一组可能的取值定义如下。对左边的图，$\lambda(1)=\{e_1\}$，$\lambda(2)=\{e_2\}$，$\lambda(3)=\{e_1,e_2,e_3,e_4\}$，$\lambda(4)=\{e_1,e_3,e_5\}$，$\lambda(5)=\{e_2,e_4,e_6\}$，$\lambda(6)=\{e_1,e_3,e_5,\ e_{10}\}$，$\lambda(7)=\{e_{12}\}$，$\lambda(8)=\{e_1,e_3,e_6,e_8\}$，$\lambda(D)=\{e_1,e_3,e_5,e_{10},e_{12}\}$。对右边的图，$\lambda(1)=\emptyset$，$\lambda(2)=\emptyset$，

**Please visit http://blog.sina.com.cn/u/1423845304 for revision information

$\lambda(3) = \emptyset$，$\lambda(4) = \{e_1, e_3, e_5\}$，$\lambda(5) = \{e_2, e_4, e_6\}$，$\lambda(6) = \{e_1, e_3, e_5\}$，$\lambda(7) = \lambda(7') = \{e_1, e_3, e_6, e_8\}$，$\lambda(8) = \{e_1, e_3, e_6, e_8\}$，$\lambda(D) = \emptyset$。

**定义 2** 设$G =< V, E, S, D, L, \lambda >$是一个加标多级图, $P$是$G$中一条路径。如果$P$上的所有边都属于边集$ES$，我们称$P$属于$ES$，或者称$ES$包含$P$，记为$P \in ES$。

**定义 3** 设$G =< V, E, S, D, L, \lambda >$是一个加标多级图，$S - u_1 - \cdots - u_l - \cdots - u_L$ $(1 \leq l \leq L$，$u_L = D)$是$G$中一条路径。如果对任意的顶点$u_l$，其中$l \in \{1,2,\cdots,L\}$，$S - \cdots - u_l \in \lambda(u_l)$，称$G$中的这条路径为简单路径。如果对任意的顶点$u_l$，其中$l \in \{1,2,\cdots,L-2\}$或者$l \in \{1, 2, \cdots, L-1\}$，$S - \cdots - u_l \in \lambda(u_l)$，称$G$中的这条路径为半简单路径。

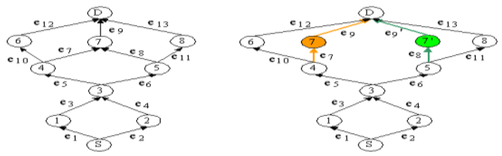


图 1 加标多级图的两个实例

显然，根据定义，如果一条路径是简单路径，它必为半简单路径。反之不然。

简单路径的概念在图论中已经有过。它的传统的含义是，一条路径称为简单路径，其中的顶点不能重复。对本文中定义的加标多级图，一条路径上的顶点肯定是不重复的。但是，路径上的顶点边集，实际上实现了一种“排它”性，所以，我们借用了简单路径这个概念。

现在提出一个问题。

**设$G =< V, E, S, D, L, \lambda >$是一个加标多级图。**

**问：$G$中是否存在一条简单路径，即是否存在路径$S - u_1 - \cdots - u_l - \cdots - u_L$ $(1 \leq l \leq L$，$u_L = D)$，使得$S - \cdots - u_l \in \lambda(u_l)$?**

这个问题就是要判定一个加标多级图中简单路径的存在性。我们称这个问题为$MSP$问题，即多级图简单路径（Multistage-graph Simple Path）问题。

不是所有的加标多级图都有简单路径。一个加标多级图是否包含简单路径，取决于图，同时也取决于各个顶点的边集的取值。例如，对于图 1 中左图，$S-1-3-4-6-D$ 是一条简单路径。而图 1 右图中没有简单路径。

在一个加标多级图中，判定从源点到汇点的路径的存在性是容易的。这是一个简单的连通性判定的问题。然而，判定简单路径的存在性，不是一件容易的事情。

下面开始讨论$MSP$问题的求解算法，称之为$ZH$算法。我们的算法只解决判定问题，对于一个给定的加标多级图，回答简单路径的存在性。

# 2 求解$MSP$问题的$ZH$算法

首先给出四个基本算子。

## 2.1 四个基本算子定义

**基本算子 1**：$[ES]_u^v$。

设$G=<V,E,S,D,L,\lambda>$是一个加标多级图。$ES$是边集$E$的一个子集，$u,v\in V$。定义$[ES]_u^v=\{e\mid e\in ES, e\text{ is on a path } u-\cdots-v, \text{and all the edges on } u-\cdots-v \text{ are contained in } ES\}$。如果$u,v$属于同一级或者$u$的级高于$v$的级，定义$[ES]_u^v=\emptyset$。

$[ES]_u^v$的目的是整理边集$ES$。$ES$中可能包含一些零零散散的边。这些边因为不在$u$到$v$路径上，因而从$ES$中去掉。$[ES]_u^v$的结果是$ES$的子集。如果用$|E|$表示$G$中边的数目，可以设计$O(|E|)$的算法计算$[ES]_u^v$。

**定义 4** 设$G=<V,E,S,D,L,\lambda>$是一个加标多级图。如果$G=<V,E,S,D,L,\lambda>$中存在一条路径$v-v_{l+1}-\cdots-v_{L-1}-D$，使得边$\langle u,v,l\rangle\in\lambda(v)\cap\lambda(v_{l+1})\cap\cdots\cap\lambda(v_{L-1})\cap\lambda(D)$，我们称路径$v-v_{l+1}-\cdots-v_{L-1}-D$为$\langle u,v,l\rangle$的可达路径。$\langle u,v,l\rangle$的所有可达路径构成$\langle u,v,l\rangle$的可达路径集。$\langle u,v,l\rangle$的所有可达路径经过的边构成可达路径集边集。

为了描述$\langle u,v,l\rangle$的可达路径集边集，特别是为了算法中处理可达路径，本文用变量符号$R(u,v,l)$表示$\langle u,v,l\rangle$的可达路径集边集。一般处理和描述路径都是将路径一条条描述出来，但本文的$R(u,v,l)$是边集$E$的子集，$R(u,v,l)$中包含的是$\langle u,v,l\rangle$的可达路径经过的边。因为$R(u,v,l)$是算法的变量符号，所以开始的时候，$R(u,v,l)$包含的是定义 4 定义的可达路径集边集，计算过程中会单调减少。

下面的讨论中，我们除了用$R(u,v,l)$表示边$\langle u,v,l\rangle$的可达路径集边集，有时候也用$R(e)$表示边$e$的可达路径集边集。当需要精确知道一条边的起点、终点以及所在级的时候，我们选择用$R(u,v,l)$表示。否则，就简单地用$R(e)$表示。

一个加标多级图给定之后，可以计算出所有边的可达路径集边集。

**基本算子 2：**$Init(R(u,v,l))$。

$Init(R(u,v,l))$用来计算加标多级图$G=<V,E,S,D,L,\lambda>$中边$\langle u,v,l\rangle$的$R(u,v,l)$的值，计算结果放在$R(u,v,l)$中：

1. $ES\leftarrow\{\langle a,b,k\rangle\mid\langle a,b,k\rangle\in E, l<k\le L, \langle u,v,l\rangle\in\lambda(a)\cap\lambda(b)\}$ //Collecting edges
2. $R(u,v,l)\leftarrow[ES]_v^D$ //Linking edges together

按照定义，算子2计算的结果，$R(u,v,l)$中包含$\langle u,v,l\rangle$的所有可达路径经过的边。如果用$|E|$表示$G$中边的数目，可以设计$O(|E|)$的算法计算$Init(R(u,v,l))$。

再次强调一下，虽然称作可达路径集，$R(e)$仅仅是边的集合。本文讨论中涉及路径的集合都是边集，因而，它们的规模是多项式的而不是指数的。如同英文单词很多，但字母表只有 26 个字符。

**基本算子 3：**$Comp(ES,v,R(E))$。

不同于算子2和下面的算子4适宜于被当做子程序看待（调用子程序的结果是，参数变量发生改变），请将算子3当做函数看待，函数值作为返回值。

设$G=<V,E,S,D,L,\lambda>$是一个加标多级图，$ES\subseteq E$，$v\in V$，$R(E)=\{R(e)\mid e\in E\}$。$Comp(ES,v,R(E))$等于以下迭代结束时$ES_temp$中的结果：

1. $ES\_temp\leftarrow ES$
2. For all $e=\langle a,b,k\rangle\in ES\_temp$:
   Let $v$ be a vertex at stage $l$.
   If $k<l$ and $[R(a,b,k)\cap ES\_temp]_b^v$ contains no path from $b$ to $v$, $ES\_temp\leftarrow ES\_temp-\{e\}$.
   If $k=l$, $l<L$, and $R(a,b,k)$ contains no path from $v$ to $D$, $ES\_temp\leftarrow ES\_temp-\{e\}$.
3. $ES\_temp\leftarrow[ES\_temp]_S^v$, where, $S$ is the unique vertex of $V_0$.

4. Repeat step 2 and step 3 until $ES\_temp$ will not change any more.

解释一下$Comp\big(ES, v, R(E)\big)$的计算过程。简单地说，步骤2是去掉$ES_temp$中的边$\langle a, b, k\rangle$，条件是$R(a, b, k) \cap ES_temp$不含$b$到$v$的路径。步骤3是整理边集$ES_temp$。这个去边和整理的过程反复实施，直到$ES_temp$不再变化。由于$ES$中包含的边的条数是一个定数，这个计算过程必然终止。

$R(E) = \{R(e) \mid e \in E\}$是$Comp\big(ES, v, R(E)\big)$计算过程中需要使用的一组值。也可以将$R(E) = \{R(e) \mid e \in E\}$作为全局变量定义，不需要作为参数变量带入到算子中来。有很多人建议在$Comp\big(ES, v, R(E)\big)$算子中明显给出$R(E)$，也有很多人认为不需要给出$R(E)$。这里还是在$Comp\big(ES, v, R(E)\big)$中明显给出$R(E)$，使我们可以明确看到$R(E)$对于$Comp\big(ES, v, R(E)\big)$的影响。事实上，后面将介绍的$ZH$算法，就是一个反复的利用$R(E)$限制各个$Comp\big(\lambda(v), v, R(E)\big)$，又反过来利用所有的这些$Comp\big(\lambda(v), v, R(E)\big)$限制$R(E)$的过程。

显然，对任意$\lambda(v)$，有$Comp\big(\lambda(v), v, R(E)\big) \subseteq \lambda(v)$。

可以设计$O(|E|^2)$的算法计算步骤2。步骤2和步骤3因而可以在$O(|E|^2)$内完成。每次迭代至少减少一条边，$ES_temp$最多有$|E|$条边，所以计算$Comp\big(ES, v, R(E)\big)$的时间复杂性为$O(|E|^3)$。

**基本算子 4：**$Change(R(u, v, l))$。

$Change(R(u, v, l))$是用$R(E) = \{R(e) | e \in E\}$中其它的$R(e)$限制和绑定$R(u, v, l)$。$Change(R(u, v, l))$将修改$R(u, v, l)$中的值，修改后的值仍然放在$R(u, v, l)$中。

1. For all $\langle a, b, k\rangle \in R(u, v, l), l < k \leq L$
   if $Comp([\{ e \mid e = \langle c, d, kk\rangle \in E, kk < l, \big[R(e) \cap Comp\big(\lambda(b), b, R(E)\big)\big]_d^b$ contains a path that contains $\langle u, v, l\rangle$ and $\langle a, b, k\rangle\}]_S^u, u, R(E)) \neq \emptyset$
   then $\langle a, b, k\rangle$ is kept in $R(u, v, l)$
   else $\langle a, b, k\rangle$ is deleted from $R(u, v, l)$.
2. $R(u, v, l) \leftarrow [R(u, v, l)]_v^D$.
3. Repeat step 1 and step 2 until $R(u, v, l)$ will not change any more.

解释一下$Change(R(u, v, l))$的计算过程。粗略地说，步骤1中，如果$\langle a, b, k\rangle$继续留在$R(u, v, l)$中，除非有路径$P = S - \cdots - u$陪着$\langle u, v, l\rangle$经过$\langle a, b, k\rangle$，而且那些路径$P = S - \cdots - u$还需要满足苛刻条件：设那些路径的所有边构成集合$ES$，$Comp\big(ES, u, R(E)\big)$必须非空。

算子 4 是有点复杂的。如果说算子 1、算子 2、算子 3 的提出还是出于一种直觉导致的产生，算子 4 主要出于一种逻辑证明的需要。我们在下面的2.4那个章节将进一步分析指出这一点。算子4完成的信息探测，使我们可以确认在输入的图中存在一种性质。算子 4 在我们完成算法证明中起着至关重要的作用。之所以我们能够证明$NP = P$，一个最有价值的发现之一在于算子 4。

$Change(R(u, v, l))$的复杂性依赖于算子 1、算子 2 和算子 3。我们可以在$|E| * O(|E|^3)$的时间内完成计算$Comp([\{ e \mid e = \langle c, d, kk\rangle \in E, kk < l, \big[R(e) \cap Comp\big(\lambda(b), b, R(E)\big)\big]_d^b$包含一条路径，该路径包含$\langle u, v, l\rangle$和$\langle a, b, k\rangle\}]_S^u, u, R(E))$，从而$|E| * |E| * O(|E|^3)$时间内完成步骤 1。每次迭代至少减少一条边，因此$Change(R(u, v, l))$的复杂性为$|E| * |E| * |E| * O(|E|^3) = O(|E|^6)$。

修改后的$R(u, v, l)$是修改前的$R(u, v, l)$子集。我们仍然不加区别地称其为$\langle u, v, l\rangle$的可达路径集边集。

### 2.2 $ZH$算法、复杂性分析、必要性证明

现在给出求解$MSP$问题的、由以下四个语句构成的$ZH$算法以及关于算法的分析证明。

$ZH$算法的输入是$G = < V, E, S, D, L, \lambda >$。算法中符号$ES[i{:}\, j]$表示边集$ES$中从第$i$级到第$j$级的边，其中$1 \leq$

$i \leq j \leq L$。如果$i > j$，$ES[i{:}j] = \emptyset$。$'R(a,b,k)[k+1{:}l] \leftarrow \bigcup_{v \in V_l}\left[R(a,b,k) \cap Comp\big(\lambda(v), v, R(E)\big)\right]_b^v{}'$ 表示对$R(a,b,k)$ 的一部分赋值。只有$R(a,b,k)$的从第$k+1$级到第$l$级的部分被修改，其余的部分不变。如果将$R(a,b,k)$看成数组，这个赋值表示数组的一部分被改变。如果将$R(a,b,k)$放在图灵机工作带上，这个赋值表示工作带上存放$R(a,b,k)$的那片区域的一部分被重新印刷。

因为算法输入包含了一个顶点集合$V$，一个边集$E$，一个源点$S$，一个汇点$D$，一个表示图的级的量$L$，以及对于除源点$S$外的每个点$v$，还有一个边集$\lambda(v)$。我们将这些量都当成相应变量的初值。算法中变量当然可以被修改，算法中$'\lambda(v) \leftarrow Comp\big(\lambda(v), v, R(E)\big)'$ 表示将修改的值放入$\lambda(v)$。$R(e)$是因为算法执行需要而新开辟的变量，所有的$R(e)$构成 $R(E) = \{R(e) |\ e \in E\}$。

算法执行中会修改$\lambda(v)$，$\lambda(v)$中保留的值与初始的值一般是不同的。现在开始一个约定：如不特别声明，今后说到$\lambda(v)$，我们是指图定义时给出的值，或者算法输入的初值。而用$Comp\big(\lambda(v), v, R(E)\big)$表达变量$\lambda(v)$中保留的值，称为$\lambda(v)$的计算值。简单路径的定义是基于初始值而不是计算值。

***ZH Algorithm***

1. For all $e \in E$, we call $Init(R(e))$ to generate $R(e)$ directly.
2. For $l = 2$ to $L-1$
   - 2.1 For all $\langle u, v, l\rangle$ of stage $l$, call $Change(R(u,v,l))$ to modify $R(u,v,l)$
   - 2.2 For all $v$ of stage $l$, $\lambda(v) \leftarrow Comp\big(\lambda(v), v, R(E)\big)$
   - 2.3 For all $\langle a,b,k\rangle \in E, k < l$, execute the following two steps:
     - $R(a,b,k)[k+1{:}l] \leftarrow \bigcup_{v \in V_l}\left[R(a,b,k) \cap Comp\big(\lambda(v), v, R(E)\big)\right]_b^v$ //Limit R(e)
     - $R(a,b,k) \leftarrow [R(a,b,k)]_b^D$ //Tidy R(e)
3. Repeat step 2 until no $R(u,v,l)$ in $R(E) = \{R(e) \mid e \in E\}$ will change any more.
4. If $Comp\big(\lambda(D), D, R(E)\big) \neq \emptyset$, we claim the existence of a simple path in $G$.
   Otherwise, we claim that there is no simple path in $G$.

解释一下算法的动作。‘$R(a,b,k)[k+1{:}l] \leftarrow \bigcup_{v \in V_l}\left[R(a,b,k) \cap Comp\big(\lambda(v), v, R(E)\big)\right]_b^v$’意味着$R(a,b,k)$的从$k+1$级到$l$级的边由$\bigcup_{v \in V_l}\left[R(a,b,k) \cap Comp\big(\lambda(v), v, R(E)\big)\right]_b^v$替换。替换之后，进一步执行‘$R(a,b,k) \leftarrow [R(a,b,k)]_b^D$’，以便保持$R(a,b,k) = [R(a,b,k)]_b^D$。由于算法中$\lambda(v)$最终会等于$Comp\big(\lambda(v), v, R(E)\big)$，因此，实际上也可以用$R(a,b,k) \cap \lambda(v)$替换$R(a,b,k) \cap Comp\big(\lambda(v), v, R(E)\big)$。

$ZH$算法的基本思想是用$R(E)$绑定$R(u,v,l)$（步骤 2.1），用$R(E)$修改$\lambda(v)$（步骤 2.2），同时，使用$Comp\big(\lambda(v), v, R(E)\big)$限制修改$R(E)$（步骤 2.3）。步骤 3 之后，用$R(E)$计算$Comp\big(\lambda(D), D, R(E)\big)$。

结论是令人惊异的简洁：**$\boldsymbol{G}$中存在简单路径，当且仅当$\boldsymbol{Comp\big(\lambda(D), D, R(E)\big) \neq \emptyset}$。**

讨论一下几个算子和$ZH$算法的一些性质。

显然，$Comp\big(\lambda(v), v, R(E)\big)$是$\lambda(v)$的子集，$Change(R(u,v,l))$后得到的$R(u,v,l)$是修改之前的$R(u,v,l)$的子集。$ZH$算法最终肯定会停止，因为每次迭代至少减少一条边，而边的总数受限于$|E|$。

但是计算是有顺序的。比如，点的编号顺序不同，边的编号顺序不同，会使得$\lambda(v)$和$R(u,v,l)$中的边以不同的顺序被去掉。不同的计算顺序会不会使计算结果不同？答案是否定的。即计算结果与顺序无关。

设$ZH$算法对于输入的$G = \langle V, E, S, D, L, \lambda \rangle$计算到了完全稳定。此时算法中所有$\lambda(v)$和$R(u,v,l)$都不能再改变。改变顶点编号和边的编号，重新按照新的顺序开始计算过程。开始时，新的$\lambda(v)$和$R(u,v,l)$都比原来

算到稳定时的$\lambda(v)$和$R(u,v,l)$包含更多边。如果新的计算过程第一次出现必须消去某条边，比如$R(u,v,l)$需要消去$\langle a,b,k\rangle$，而原来已经稳定的计算过程没有消去该边(形式上，可能$\langle a,b,k\rangle$原来有不同编号)，这是不可能的。因为消去$\langle a,b,k\rangle$之前，新的$\lambda(v)$和$R(u,v,l)$一直保持了比原来算到稳定时的$\lambda(v)$和$R(u,v,l)$包含更多边，因此，如果新的$\lambda(v)$和$R(u,v,l)$必须遭受消去$\langle a,b,k\rangle$，原来算到稳定时的$\lambda(v)$和$R(u,v,l)$更应该消去$\langle a,b,k\rangle$。

这就说明，按照不同的顺序计算，一定得到相同的结果。

**定理 1.** 设$|V|$表示$V$的顶点个数，$|E|$表示$E$中边的条数。$ZH$算法的时间复杂性是$|V| * |E|$的多项式函数。

**证明** 首先指出，任意边集，任意可达路径集中包含的边的条数$\leq |E|$，因为他们都是边集$E$的子集；顶点边集的个数$\leq |V|$；可达路径集的个数$\leq |E|$。

前面已述，计算$Comp\big(ES,v,R(E)\big)$的复杂性为$O(|E|^3)$。计算$Change(R(u,v,l))$的复杂性为$O(|E|^6)$。可以分析出来步骤 2.3 限制$R(e)$的复杂性为$O(|E|^3)$。对步骤 3 的每次迭代，$R(u,v,l)$至少减少一条边。

$ZH$算法的步骤 2 是$ZH$算法中最为复杂的语句。所以$ZH$算法的时间复杂性为$|E| * |E| * O(|E|^6)$。由此证明了$ZH$算法的时间复杂性为$|V| * |E|$的多项式函数。 ■

**定理 2.** 如果$G$中存在简单路径，则一定有$Comp\big(\lambda(D),D,R(E)\big) \neq \emptyset$。

**证明** 设$v_0 - v_1 - v_2 - \cdots - v_L$是$G$中一条简单路径，$v_0 = S$，$v_L = D$。根据简单路径定义，有$v_0 - v_1 - v_2 - \cdots - v_l \in \lambda(v_l)$ $(1 \leq l \leq L)$，并且，对路径$v_0 - v_1 - v_2 - \cdots - v_L$上所有$\langle v_{l-1}, v_l, l\rangle$ $(1 \leq l \leq L)$，有$\langle v_{l-1}, v_l, l\rangle \in \lambda(v_l) \cap \cdots \cap \lambda(v_{L-1}) \cap \lambda(v_L)$。所以，$ZH$算法步骤 1 执行完毕，$R(v_{l-1}, v_l, l)$将包含$v_l - v_{l+1} - \cdots - v_L$ $(1 \leq l \leq L)$。步骤 2 之后，$R(v_{l-1}, v_l, l)$仍然包含$v_l - v_{l+1} - \cdots - v_L$ $(1 \leq l \leq L)$。步骤 3 不会截断$R(v_{l-1}, v_l, l)$中的任何路径。我们因此知道$Comp\big(\lambda(D),D,R(E)\big)$包含$v_0 - v_1 - v_2 - \cdots - v_L$。 ■

### 2.3 充分性证明准备

我们现在开始证明，如果$Comp\big(\lambda(D),D,R(E)\big) \neq \emptyset$，则$G$中存在简单路径。

#### 2.3.1 两个函数

为了下面讨论需要，引进两个函数，一个是换名函数$I_y^x$，另一个是撕裂函数$I_v^{v_1,v_2}$。两个函数都是处理三元组的。一个三元组在我们定义的加标多级图中表示一条边。建议在后面引理 2 证明中介绍完撕裂，获得一些直观认识后，再细看这里的定义。

**换名函数$I_y^x$.** 设$EL$是一个集合，$ET = \{\langle a,b,k\rangle \mid a,b \in EL$，$k$ 是一个整数$\}$，$ES \subseteq ET$，$e \in ET$，并且$x,y \in EL$。$I_y^x$递归定义如下：

$$I_y^x(\{e\}) = \begin{cases} \{\langle b,y,k\rangle\}, & \text{if } e = \langle b,x,k\rangle \\ \{\langle y,b,k\rangle\}, & \text{if } e = \langle x,b,k\rangle \\ \{e\}, & \text{otherwise} \end{cases}$$

$$I_y^x(ES) = \bigcup\nolimits_{e \in ES} I_y^x(\{e\})$$

**撕裂函数$I_v^{v_1,v_2}$.** 设$EL$是一个集合；$ET = \{\langle a,b,k\rangle \mid a,b \in EL$，$k$ 是一个整数$\}$；$v$，$v_1$，$v_2 \in EL$，$l$是一个整数；$ES$，$ES_1$，$ES_2$是$ET$的子集，$ES_1 \neq \emptyset$，$ES_2 \neq \emptyset$，$ES_1 \cap ES_2 = \emptyset$，$ES_1 \cup ES_2 = \{e \mid e \in ES, e = \langle c,v,l\rangle, c \in EL\}$。$I_v^{v_1,v_2}$定义如下：

$$\begin{aligned} I_v^{v_1,v_2}(ES,ES_1,ES_2) = (ES &- \{e \mid e \in ES, e = \langle a,v,l\rangle \text{ or } e = \langle v,a,l+1\rangle, a \in EL\} \\ &\cup \{\, e \mid e = \langle a,v_1,l\rangle, \langle a,v,l\rangle \in ES_1, a \in EL\} \\ &\cup \{e \mid e = \langle a,v_2,l\rangle, \langle a,v,l\rangle \in ES_2, a \in EL\} \end{aligned}$$

$\cup \{e \mid e = \langle v_1, a, l+1\rangle \text{ or } e = \langle v_2, a, l+1\rangle, \langle v, a, l+1\rangle \in ES, a \in EL\}$

在$I_v^{v_1,v_2}(ES, ES_1, ES_2)$中，我们从$ES$中去掉所有三元组$\langle a, v, l\rangle$和$\langle v, a, l+1\rangle$，如果$\langle a, v, l\rangle \in ES_1$，用$\langle a, v_1, l\rangle$替换$\langle a, v, l\rangle$，如果$\langle a, v, l\rangle \in ES_2$，用$\langle a, v_2, l\rangle$替换$\langle a, v, l\rangle$，如果$\langle v, a, l+1\rangle \in ES$，用$\langle v_1, a, l+1\rangle$和$\langle v_2, a, l+1\rangle$替换$\langle v, a, l+1\rangle$。

### 2.3.2 构造证明框架

我们还需要一个符号。

$ZH\backslash step4$：$ZH\backslash step4$表示$ZH$算法除步骤 4 之外的其余步骤。

为了构造反驳，我们将$ZH$算法应用两次。一次将算法应用于$G$，得到一个中间结果。然后将算法应用于$G^+$，并结合刚才得到的中间结果，得到最终结果。根据最终结果，我们推断$G$中简单路径的存在性。为了方便提及，我们将$ZH$算法应用两次的这个过程称作 ***Proving Algorithm***。***Proving Algorithm*** 的输入包括图$G =< V, E, S, D, L, \lambda >$以及任意的一个边集$ESS$（$ESS \subseteq E$）。为了简化讨论，我们讨论的图需要满足一些性质。对不满足这些性质的图，算法将停机。因为反驳中必须有新的实例构造，而新的构造同样必须具备这些性质，为了讨论中不会遗漏的原因，我们将这些性质全部列在 ***Proving Algorithm*** 中。

请别纠结 ***Proving Algorithm*** 的复杂性。实际上，***Proving Algorithm*** 是一个不同于$ZH$算法的算法。这两个算法各自花费代价做各自的事情。$ZH$算法解决给定的加标多级图中简单路径存在性问题，而 ***Proving Algorithm*** 解决另外一个问题：给定$G =< V, E, S, D, L, \lambda >$以及任意的一个边集$ESS$，$ESS \subseteq E$，$G$中有简单路径包含于$ESS$吗？

***Proving Algorithm*** 只做充分性判断。算法依据它形成的判据$Comp(ESS1, D, R(E))$非空做出$G$中有简单路径包含于$ESS$的回答，否则，算法不做回答。就像你下水捕鱼，发现网中有鱼，你说水中有鱼，没有发现网中有鱼，你不对水中有鱼做任何推断。***Proving Algorithm*** 可能漏判，但我们证明它绝不误判。

| ***Proving Algorithm*** |
|---|
| 1. For the input graph $G =< V, E, S, D, L, \lambda >$ and $ESS$, check if it is: (1.1) If $P$ is a simple path in $G$ such that $P[1:L-2] \in ESS$, $P \in ESS$; (1.2) $\lambda(D) = E$; (1.3) For $ESS$ and for each $\lambda(v)$, it is a subset of $E$. Let it be $A$ and $V' \subseteq V_1 \cup V_2 \cup \cdots \cup V_{L-3}$. $A[1:L-2] = (\, E[1:L-2] - \{\langle a, b, k\rangle \mid \langle a, b, k\rangle \in E, a \in V' \text{ or } b \in V'\}$; (1.4) $d(v) = 1$ for all $v$ at stage $L-1$. (Where, $d(v)$ is the in-degree of $v$)<br>If not, we stop algorithm.<br>2. Apply $ZH\backslash step4$ on $G$.<br>3. $ESS1 \leftarrow ESS \cap Comp(\lambda(D), D, R(E))$.<br>4. For each $e \in E$, $R(e) \leftarrow E$.<br>5. If $Comp(ESS1, D, R(E)) \neq \emptyset$, there exists a simple path $SP$ in $G$ such that $ESS$ contains $SP$. |

算法中步骤 1 的(1.3)定义的性质，可以这么粗略地理解：对于$ESS$和$\lambda(v)$中$L-2$级及其以下的部分，如果它包含了一条从点$a$出发或者进入点$a$的边，那么它必须包含所有进入$a$和从$a$出发的边。其中，$a$在$G$中$L-3$级或者$L-3$级以下。

步骤 4 重新产生的$R(e)$已经不再是可达路径集边集。此时它就是个边集而已。至于为什么令所有$R(e) = E$，先存个疑。到引理 2 的结论(4)证明时就明白了。

几个概念再次明确一下。算法执行中会修改$\lambda(v)$，$\lambda(v)$中保留的值与初始的值一般是不同的。称$\lambda(v)$中保留的值为计算值。简单路径的定义基于初始值而不是计算值。***Proving Algorithm*** 步骤 5 中推断存在的简单路径，不仅仅是$G$中的简单路径，它还必须包含于$ESS$当中。

### 2.4 $\alpha\beta$ 定理及其证明

下面开始归纳证明，对于任意输入的$G$和$ESS$，***Proving Algorithm*** 都能做出正确推断，即，如果$Comp(ESS1,D,R(E)) \neq \emptyset$，则$G$有简单路径 $SP$且$ESS$包含$SP$。

**引理 1.** 设$G=<V,E,S,D,L,\lambda>$和$ESS$是 ***Proving Algorithm*** 的输入，$G$中第1级到$L-1$级没有多入度点，如图 2 所示。如果$Comp(ESS1,D,R(E)) \neq \emptyset$，则$G$有简单路径 $SP$且$ESS$包含$SP$。

**证明** 因为$Comp(ESS1,D,R(E)) \neq \emptyset$，因此，一定有$S-a-b \in Comp(ESS1,D,R(E))$并且$R(a,b,2) \cap Comp(ESS1,D,R(E))$包含$b-a_3-\cdots-a_{L-1}-D$。又因为$ESS1$必然包含$S-a-b-a_3-\cdots-a_{L-1}-D$，所以 ***Proving Algorithm*** 在步骤 2 计算$ESS1$时，$\lambda(a_{L-1})$的计算值必然非空，否则与$a_{L-1}$关联的边，不可能进入$ESS1$。这就说明$P=S-a-b-a_3-\cdots-a_{L-1}-D$即为$G$中简单路径且$P \in ESS$。 ■

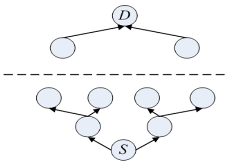
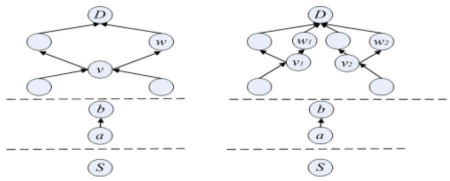



图 2 Typical graph of lemma 1




(a) (b)

图 3 Typical graph of lemma 2

现在给每个$G=<V,E,S,D,L,\lambda>$定义一个度量。

$f(G)=L+\sum_{v\in V-\{D\}}(d(v)-1)$。其中，$d(v)$是$v$的入度，$V-\{D\}$是$G$中除$D$之外的所有顶点的集合。

显然，对于任意输入的图$G$，若$f(G)=3$且$Comp(ESS1,D,R(E)) \neq \emptyset$，必然有$L=3$。根据引理 1，***Proving Algorithm*** 能做出正确推断。

**假定对于任意输入的图$G$，若$f(G)<m$，*Proving Algorithm* 都能做出正确推断。考虑$f(G)=m$的情况。**

此时，如果$G$没有多入度点，根据引理 1，*Proving Algorithm* 能做出正确推断。如果$G$有多入度点，我们分引理 2（有多入度点在$L-2$级）和引理 3（没有多入度点在$L-2$级）两种情况讨论。

**引理 2.** 设$G=<V,E,S,D,L,\lambda>$和$ESS$是 ***Proving Algorithm*** 的输入，$f(G)=m$，顶点$v$是$G$中$L-2$级的一个多入度点，$G$中$L-1$级没有多入度点，如图 3(a)所示。如果$Comp(ESS1,D,R(E)) \neq \emptyset$，则$G$有简单路径 $SP$且$ESS$包含$SP$。

**证明** 引理 2 的证明思想是，构造一个新的$L$级图$G_1$满足三个条件，从而完成反驳：

(1) $G_1$具备 ***Proving Algorithm*** 的步骤 1 描述的性质，并且$f(G_1)<f(G)$。
(2) 如果$G$作为输入时$Comp(ESS1,D,R(E)) \neq \emptyset$，$G_1$作为输入时同样有$Comp(ESS1,D,R(E)) \neq \emptyset$。
(3) 如果$P$是$G_1$的简单路径且包含于$ESS$，$P$（经过适当改变）是$G$的简单路径且包含于$ESS$。

为此，我们将图 3(a)撕裂成另外一个图，如图 3(b)所示。既然要构造另外一个加标多级图，当然，我们需要给每个点配置相应的边集，同时构造新的$ESS$，这样，当新的图和新的$ESS$作为算法输入时，我们同样会得到一个计算结果$Comp(ESS1,D,R(E))$。问题在于，如何确保新的$Comp(ESS1,D,R(E))$非空，并且新的图中的一条包含于新的$ESS$的简单路径，**“实质上”**就是原来的图中的一条包含于原来的$ESS$的简单路径？

我们将$\lambda(v_1)$和$\lambda(v_2)$重复设置，除了换名，本质上都等于$\lambda(v)$。同样地，$\lambda(w_1)$和$\lambda(w_2)$重复设置，除了换名，本质上都等于$\lambda(w)$。新的$ESS$，除了换名，本质上也等于原来的$ESS$。因为在$MSP$问题中，一个点的顶点边集实际上是一种控制，重复设置相当于相同的控制。设想中国曾经有两湖省，左路从湖南来，右路从湖北来，现在分置成湖南省和湖北省，湖南管左路，湖北管右路。这种控制关系实质上不是一回事吗？

图 3 右图中的一条包含于新的$ESS$的简单路径“**实质上**”就是图 3 左图中的一条包含于原来的$ESS$的简单路径。

困难来了。以上初始设置是容易的。但是，如果原来$Comp\big(\lambda(v), D, R(E)\big)$非空，现在，以新的图为输入，$Comp\big(\lambda(v_1), D, R(E)\big)$和$Comp\big(\lambda(v_2), D, R(E)\big)$会非空吗？精确地说，就算赋予本质上相同的初值，撕裂不会影响计算结果吗？

算子4为这个目的产生，它主要出于一种逻辑证明的需要，而不是简单地由直觉得到。如何通过一种计算，探测到图中的一种性质，然后使得我们可以确认，对于撕裂前后的图，***Proving Algorithm*** 能够得到本质上相同的结果，就是算子 4 设计的方向指导。我们为此尝试了很多次努力。之所以我们能够证明$NP = P$，我们是将问题($MSP$问题)设计，算法设计($ZH$算法及算子)，证明框架(撕裂得到更小的图以及 ***Proving Algorithm***)一起考虑，慢慢调出来的。我们的方向非常明确和坚定，就是寻找一个算法实现以下目标：如果对于给定的图$G$，算法能够得到一个结果，那么，希望算法在一个“更小”的图上，能够得到本质相同的结果。而“更小”的图中一个存在的答案，本质上就是原来图中的答案！

开始证明引理2。不失一般性，假定$d(v) = 2$并且$v$的出度也等于2。这意味着$G$中只有$v - w - D$和$v - w' - D$。

将以$v$为终点的边分成非空的两个部分，即$group_1$和$group_2$，自底向上将$v$撕裂成$v_1$,$v_2$。然后逐个撕裂$L - 1$级的多入度点。因此得到一个$L - 1$级没有多入度点、$L - 2$级$d(v_1) + d(v_2) = d(v)$的图$G_1$，如图 3(b)所示。

顶点边集和$ESS$定义如下：

令$V_1 = (V - \{v, w, w'\}) \cup \{v_1, v_2\} \cup \{w_1, w_2, w_3, w_4\}$。

令 $E_1 = I_w^{w_1,w_2}\big(I_{w'}^{w_3,w_4}\big(I_v^{v_1,v_2}(E \text{ of } G, group_1, group_2), \{\langle v_1, w', L-1\rangle\}, \{\langle v_2, w', L-1\rangle\}\big), \{\langle v_1, w, L-1\rangle\}, \{\langle v_2, w, L-1\rangle\}\big)$。

令( $ESS$ of $G_1$)=$I_w^{w_1,w_2}\big(I_{w'}^{w_3,w_4}\big(I_v^{v_1,v_2}(ESS \text{ of } G, group_1, group_2), \{\langle v_1, w', L-1\rangle\}, \{\langle v_2, w', L-1\rangle\}\big), \{\langle v_1, w, L-1\rangle\}, \{\langle v_2, w, L-1\rangle\}\big)$。

(记号说明：( $ESS$ of $G_1$)表示$G_1$为输入时的$ESS$。本论文中经常要比较$G$为输入时的某个量和$G_1$为输入时的某个量。为了方便和清晰的表示，我们用‘($something$ of $G$)’表示$G$为输入时的某个量，用‘( $something$ of $G_1$))’表示$G_1$为输入时的某个量。($ESS$ of $G$)就是这种表示形式的一个例子)

对于除$v_1$、$v_2$、$w_1$、$w_2$、$w_3$、$w_4$外的每个$x$，令( $\lambda(x)$ of $G_1$) $= I_w^{w_1,w_2}(I_{w'}^{w_3,w_4}(I_v^{v_1,v_2}(\lambda(x) \text{ of } G, group_1, group_2), \{\langle v_1, w', L-1\rangle\}, \{\langle v_2, w', L-1\rangle\}), \{\langle v_1, w, L-1\rangle\}, \{\langle v_2, w, L-1\rangle\})$。

对于$x = v_1$、$v_2$、$w_1$、$w_2$、$w_3$、$w_4$，給( $\lambda(x)$ of $G_1$)赋一个值，使得$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(v_1)) = I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(v_2)) = \lambda(v)$，$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(w_1)) = I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(w_2)) = \lambda(w)$，$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(w_3)) = I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(\lambda(w_4)) = \lambda(w')$。

(说明：比如，撕裂$\lambda(v)$，将结果赋给$\lambda(v_1)$和$\lambda(v_2)$，撕裂$\lambda(w)$，将结果赋给$\lambda(w_1)$和$\lambda(w_2)$，撕裂$\lambda(w')$，将结果赋给$\lambda(w_3)$和$\lambda(w_4)$。

显然，对于所有$x$，$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}\big(\lambda(x)\big) \subseteq \lambda(x)$。如果$P$是$G_1$的简单路径，$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(P)$是$G$的简单路径)

在( $ESS$ of $G_1$)中增加包含$E_1[L-1{:}\,L]$中所有边。(可以假定，扩展($ESS$ of $G_1$)以后，$G_1$不会增加包含于 ($ESS$ of $G_1$)的简单路径。如果有，那么$G$必有简单路径$P$且$P[1{:}\,L-2]$包含于($ESS$ of $G$)。这将违背图$G$满足的性质。下面讨论结论(4)的证明时会看到扩展($ESS$ of $G_1$)的目的和意义)

因此完成$G_1$构造以及顶点边集和$ESS$定义，得到一个$L - 1$级没有多入度点、$L - 2$级$d(v_1) + d(v_2) = d(v)$的图$G_1$，如图 3(b)所示。

总结一下$G_1$的构造：如果不考虑换名，$G_1$中包含于($ESS$ of $G_1$)的简单路径与$G$中包含于($ESS$ of $G$)的简单路径是一样的。

现在，将构造的$G_1$和($ESS$ of $G_1$)作为 ***Proving Algorithm*** 的输入进行计算。可以证明以下结论 (1)、(2)、(3)、 (4)、 (5)。

(1) $f(G_1) < f(G)$.

$v$是出现在$l$级的多入度点，$l = L - 2$。

$$\sum\nolimits_{u\in V_l \text{of } G_1}(d(u)-1)$$
$$=\sum\nolimits_{u\in (V_l-\{v_1,v_2\})\text{ of } G_1}(d(u)-1)+(d(v_1)-1)+(d(v_2)-1)$$
$$=\sum\nolimits_{u\in (V_l-\{v\})\text{ of } G}(d(u)-1)+(d(v)-1)-1$$
$$=\sum\nolimits_{u\in V_l \text{of } G}(d(u)-1)-1$$

因此$f(G_1) < f(G)$。

(2) 对$G$中$L-1$级的每个$\lambda(w)$，如果步骤 2 之后$\lambda(w)$包含边$e = \langle a,b,k\rangle$，$G$一定包含经过边$\langle a,b,k\rangle$和点$w$的半简单路径。

为不影响对整体思路的理解，将证明附在本引理 2 的尾部。

结论(2)非常重要。因为$G$有半简单路径经过边$\langle a,b,k\rangle$和点$w$，又因为$\lambda(D) = E$，这条半简单路径实际上是简单路径。撕裂之后，这条半简单路径必然在$\lambda(w_1)$或者$\lambda(w_2)$中。因此，当$G_1$作为 ***Proving Algorithm*** 输入，步骤 2 结束时，$\lambda(w_1)$或者$\lambda(w_2)$中必然包含这条半简单路径。如果$\langle a,b,k\rangle$曾经进入($ESS1$ of $G$)，$\langle a,b,k\rangle$必将进入($ESS1$ of $G_1$)。由此导致$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(ESS1 \text{ of } G_1) \supseteq (ESS1 \text{ of } G)$。

(3) $G_1$具备 ***Proving Algorithm*** 的步骤 1 要求的全部性质。

$G_1$具备步骤 1 的(1.2)，(1.3)和(1.4)定义的性质容易逐条验证。下面证明$G_1$具备(1.1)定义的性质。

若 $G_1$ 有简单路径 $P$， $P[1:L-2] \in$ ( $ESS$ of $G_1$)， 则 $G$ 有简单路径 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(P)$，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(P)[1:L-2] \in$ ($ESS$ of $G$)。因为$G$具备步骤 1 要求的性质，所以$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(P) \in$ ($ESS$ of $G$)，$P \in$ ( $ESS$ of $G_1$)。

(4) 将证明算法作用于$G_1$，($Comp\big(ESS1, D, R(E)\big)$ of $G_1$)$\neq \emptyset$。

首先，根据结论(2)，证明算法步骤 3 之前，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(ESS1 \text{ of } G_1) \supseteq (ESS1 \text{ of } G)$。

因此，对所有$\langle a,b,k\rangle$ ($k < L-2$)以及所有$\langle a,b,L-2\rangle$ ($b \neq v$)，如果($R(a,b,k)$ of $G$)包含一条路径$b - b_{k+1} - \cdots - b_{L-1} - D$，($R(a,b,k)$ of $G_1$) 包含 $b - c_{k+1} - \cdots - c_{L-1} - D$ 并且 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(b - c_{k+1} - \cdots - c_{L-2}) = b - b_{k+1} - \cdots - b_{L-2}$。(注意此时所有$R(e) = E$，否则我们不能推断($R(a,b,k)$ of $G_1$)包含$b - c_{k+1} - \cdots - c_{L-1} - D$并且$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(b - c_{k+1} - \cdots - c_{L-2}) = b - b_{k+1} - \cdots - b_{L-2}$。这是我们为什么令所有$R(e) = E$的原因)

对所有$\langle u,v,L-2\rangle$，如果 ($R(u,v,L-2)$ of $G$)包含$v - w - D$并且$\langle u,v,L-2\rangle \in group_1$，($R(u,v_1,L-1)$ of $G_1$) 包含$v_1 - w_1 - D$；如果 ($R(u,v,L-2)$ of $G$)包含$v - w - D$并且$\langle u,v,L-2\rangle \in group_2$，($R(u,v_2,L-1)$ of $G_1$)包含$v_2 - w_2 - D$。

因此，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}\big(Comp\big(ESS1, D, R(E)\big)\text{of } G_1\big) \supseteq \big(Comp\big(ESS1, D, R(E)\big) \text{ of } G\big)[1:L-2] \neq \emptyset$。(因为

($ESS$ of $G_1$)增加包含了$E_1[L-1:L]$的所有边)

(5) $G$有简单路径 $SP$且$ESS$包含$SP$。

因为$f(G_1) < f(G) = m$并且$(Comp(ESS1, D, R(E))$ of $G_1) \neq \emptyset$，$G_1$一定有简单路径$SP$且($ESS$ of $G_1$)包含$SP$。

如果$SP$是$G_1$中的简单路径，$P' = I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}(SP)$必为$G$中的简单路径。 ■

**引理 2 之结论(2)的证明**

我们证明，**对$G$中$L-1$级的每个$\lambda(w)$，如果步骤 2 之后$\lambda(w)$包含边$e = \langle a, b, k\rangle$，$G$一定包含经过边$\langle a, b, k\rangle$和点$w$的半简单路径。**

我们说的步骤 2 之后的$\lambda(w)$，不是算法输入时的初值，它是 ***Proving Algorithm*** 步骤 2 结束后$\lambda(w)$中留下的值。

算子 4 以及本结论的证明是我们能够完成$NP = P$证明的关键。

我们现在唯一知道的是归纳假设：将加标多级图$G'$以及边集$ESS'$作为 ***Proving Algorithm*** 的输入，如果$f(G') < m$并且$Comp(ESS1, D, R(E)) \neq \emptyset$，则$G'$中必有简单路径包含于$ESS'$。

所以，我们基于$G_1$构造一个$G_2$，确保$f(G_2) < m$，给$G_2$的各个顶点边集以及($ESS$ of $G_2$)，“挖空心思”地赋予恰当的值。如果 $(Comp(ESS1, D, R(E))$ of $G_2) \neq \emptyset$，那么 $G_2$一定有简单路径$SP$且($ESS$ of $G_2$)包含$SP$。而“挖空心思”设置的($ESS$ of $G_2$)，迫使$\langle a, b, k\rangle$和$w$都在简单路径$SP$上。

$G_2$构造如下：

令$V_2 = V_1$，$E_2 = E_1$。
对于$L-1$级和$L-2$级之外的所有顶点$x$，令$(\lambda(x)$ of $G_2) = (\lambda(x)$ of $G_1)$。
对于$L-1$级和$L-2$级的所有顶点$x$，令$(\lambda(x)$ of $G_2) = E_2$。
令 $(ESS$ of $G_2) = I_v^{v_1,v_2}((\lambda(w) \cap \lambda(v)[1:L-2]$ of $G) -$
$(\{e \mid e = \langle a_1, b_1, k-1\rangle, b_1 \neq a$，或者 $e = \langle a_1, b_1, k\rangle, a_1 \neq a\} \cup$
$\{e \mid e = \langle a_1, b_1, k+1\rangle, a_1 \neq b$，或者$e = \langle a_1, b_1, k\rangle, b_1 \neq b\})$,
$group_1, group_2) \cup E_2[L-1:L]$。

得到的图如图 3(b)所示。由于$G_2$的形状与$G_1$完全相同，$f(G_2) = f(G_1) < m$。

将 ***Proving Algorithm*** 作用于 $G_2$，我们有以下 (a)、(b)、(c) 3 个结果：

(a) $G_2$具备步骤 1 所列性质。

根据$G_2$的构造，可以逐条核对，$G_2$显然具备步骤(1.2)到(1.4)所列性质。可以假定$G_2$没有简单路径$P$且$P[1:L-2] \in (ESS$ of $G_2)$，否则，本结论(2)已经成立。因此，$G_2$具备步骤(1.1)所列性质。

(b) $(Comp(ESS1, D, R(E))$ of $G_2) \neq \emptyset$。

以下结果导致$(Comp(ESS1, D, R(E))$ of $G_2) \neq \emptyset$：

$(ESS$ of $G_2) \supseteq I_v^{v_1,v_2}((Comp(\lambda(w), w, R(E))$ of $G) - \{\langle a_1, b_1, k\rangle | \langle a_1, b_1, k\rangle \neq \langle a, b, k\rangle\}$, $group_1$, $group_2) \cup E_2[L-1:L]$；尤其重要的是，$(Comp([\{e \mid e = \langle c, d, kk\rangle \in E,\ kk < k,\ [R(e) \cap Comp(\lambda(w), w, R(E))]_d^w$ contains a path that contains $\langle v, w, L-1\rangle$ and $\langle a, b, k\rangle\}]_S^a,\ a,\ R(E))$ of $G) \neq \emptyset$。

(说明：$(Comp(\lambda(w), w, R(E))$ of $G) - \{\langle a_1, b_1, k\rangle \mid \langle a_1, b_1, k\rangle \neq \langle a, b, k\rangle\}$不满足步骤(1.3)定义的性质。($ESS$ of $G_2$)满足步骤(1.3)定义的性质)

我们详细说明一下为什么$\left(Comp\left(ESS1, D, R(E)\right) \text{ of } G_2\right) \neq \emptyset$。分 5 步看。

第一步，将眼光放在$G$上，因为步骤 2 的迭代已经稳定，如果将$\left(Comp\left(\lambda(w),\ w,\ R(E)\right) \text{ of } G\right)$再算一遍，$\left(Comp\left(\lambda(w),\ w,\ R(E)\right) \text{ of } G\right)$与$\lambda(w)$相同，不会有任何改变。

第二步，还是将眼光放在$G$上，将$\left(Comp\left(\lambda(w),\ w,\ R(E)\right) \text{ of } G\right)$中边$\langle a, b, k\rangle$所在级，除$\langle a, b, k\rangle$之外的其他边去掉，得到集合$A = \left(Comp\left(\lambda(w),\ w,\ R(E)\right) \text{ of } G\right) - \{\langle a_1, b_1, k\rangle \mid \langle a_1, b_1, k\rangle \neq \langle a, b, k\rangle\}$。基于当前的$R(E)$，再计算$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right)$，可以断定$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right) \neq \emptyset$，**这是算子 4 保证的**。因为，$(Comp([\{e \mid e = \langle c, d, kk\rangle \in E,\ kk < k,\ [R(e) \cap Comp(\lambda(w), w, R(E))]_d^w$ contains a path that contains $\langle v, w, L - 1\rangle$ and $\langle a, b, k\rangle\}]_S^a,\ a,\ R(E))$ of $G) \neq \emptyset$，而对于$\langle a, b, k\rangle$以上的边$e$，$R(e) \cap Comp(\lambda(w), w, R(E))$ 仍然包含从$e$的终点到$v$再到$w$的路径，计算$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right)$的过程中，他们仍然将被保留。

第三步，将$G$的$L-1$级和$L-2$级的顶点边集都换成全集，重复步骤 2。基于得到的$R(E)$（这个$R(E)$中的每个$R(e)$包含的边，不会比上面第一步说到的$R(e)$包含的边更少），还是对上一步的$A$，再算一次$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right)$，可以断定$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right) \neq \emptyset$。

第四步，将$G$撕裂成$G_2$。因为$G$的$L-1$级和$L-2$级的顶点边集都换成了全集，撕裂不会影响$R(E)$的计算。$L-1$级和$L-2$级所有$\lambda(x)$都等于$E_2$，所以，***Proving Algorithm*** 的步骤 4 后得到的$R(E)$与上面第三步得到的$R(E)$，除了换名，本质上一样。

第五步，此时，在$G_2$上计算$\left(Comp\left(ESS1, D, R(E)\right) \text{ of } G_2\right)$，相当于第三步计算$\left(Comp\left(A,\ w,\ R(E)\right) \text{ of } G\right)$。所以，$\left(Comp\left(ESS1, D, R(E)\right) \text{ of } G_2\right) \neq \emptyset$。

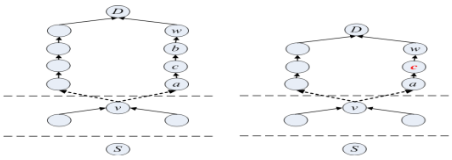
(c) $f(G_2) < f(G)$。

$f(G_2) = f(G_1)$而$f(G_1) < f(G)$，因此$f(G_2) < f(G) = m$。

基于 (a)、(b)、(c)，我们可以推断$G_2$中存在简单路径$SP$且$(ESS \text{ of } G_2)$包含$SP$。由于$\langle a, b, k\rangle$是$(ESS \text{ of } G_2)$中第$k$级的唯一边，因此$\langle a, b, k\rangle$必在$SP$上。根据$(ESS \text{ of } G_2)$的取值，$SP[1{:}L-2] \in \lambda(v)$，$SP[1{:}L-1] \in \lambda(w)$。

$I_{w'}^{w_4} I_{w'}^{w_3} I_{w}^{w_2} I_{w}^{w_1} I_{v}^{v_2} I_{v}^{v_1}(SP)[1{:}L-1]$就是我们关心的半简单路径。

结论(2)因此得证。 ■




(a) (b)

图 4 Typical graph of lemma 3

**引理 3.** 设$G = < V, E, S, D, L, \lambda >$和$ESS$是 ***Proving Algorithm*** 的输入，$f(G) = m$，$v$是$G$中$l$级的一个多入度点，$l < L-2$，$l+1$级到$L-1$级没有多入度点，如图 4(a)所示。若$Comp\left(ESS1, D, R(E)\right) \neq \emptyset$，则$G$有简单路径$SP$且$ESS$包含$SP$。

**证明** 引理 3 的证明思想是，构造一个新的图$G_1$满足三个条件，从而完成证明：

(1) $G_1$具备 ***Proving Algorithm*** 的步骤 1 描述的性质，并且$G_1$是一个$L-1$级图。
(2) 如果$G$作为输入时$Comp\left(ESS1, D, R(E)\right) \neq \emptyset$，$G_1$作为输入时同样有$Comp\left(ESS1, D, R(E)\right) \neq \emptyset$。
(3) 如果$P$是$G_1$的简单路径且包含于$ESS$，$P$本质上是$G$的简单路径且包含于$ESS$。

传统的归纳证明是通过去点或者去边构造“更小”的图。引理 2 证明的时候，我们没有找到这样的构造“更小”的图的方法。所以我们采用撕裂，不是去点，而是复制更多点，完成了引理 2 的证明。撕裂是

个创新意义的发现。引理 3 的证明可以回归传统证明方法了。

去掉$G$中$L-2$级所有点，我们构造一个$L-1$级图$G_1 =< V_1, E_1, S, D, L-1, \lambda >$，如图 4(b)所示：

1. $G_1$的顶点: $V_1 = (V \text{ of } G) - (V_{L-2} \text{ of } G)$。其中，$V_{L-2}$表示图$G$的$L-2$级顶点的集合。
2. $G_1$的边，即$E_1$：如果$k < L-2$，$G$中所有$\langle g_1, g_2, k\rangle$ 变成$G_1$的边。如果$k > L-1$，$\langle g_1, g_2, k-1\rangle$变成$G_1$的边。如果$\langle c, b, L-2\rangle$和$\langle b, w, L-1\rangle$是$G$的边，$\langle c, w, (L-1)-1\rangle$是$G_1$的边。
3. $G_1$的顶点边集，以及$(ESS \text{ of } G_1)$：

   对所有$w \in V_1$，如果$w$在$k$级，$k = (L-1)-1$：

   令 $(\lambda(w) \text{ of } G_1) = ((Comp(\lambda(w), w, R(E)) \text{ of } G) - E[L-2: L-1]) \cup \{e \mid e = \langle c, w, (L-1)-1\rangle$，存在$b$使得$\langle c, b, L-2\rangle \in (\lambda(w) \text{ of } G)$并且$\langle b, w, L-1\rangle \in (\lambda(w) \text{ of } G)\} \cup (\lambda(w) \text{ of } G)[1:2]$。

   (将两级控制合并成由$(\lambda(w) \text{ of } G_1)$统一控制，这样设置的$\lambda(w)$是我们实质上关心的内容。但是形式上可能不满足步骤(1.3)定义的性质。为了形式上满足步骤(1.3)定义的性质要求，我们可以这样设置$(\lambda(w) \text{ of } G_1)$的值：如果$((Comp(\lambda(w), w, R(E)) \text{ of } G)$为空，可以令$(\lambda(w) \text{ of } G_1)$为空。否则，设$\langle b, w, L-1\rangle$是$G$中的边，我们

   令 $(\lambda(w) \text{ of } G_1) = (\lambda(w) \text{ of } G) \cap (\lambda(b) \text{ of } G)) - E[L-2: L-1]) \cup \{ e \mid e = \langle c, w, (L-1)-1\rangle$，存在$b$使得$\langle c, b, L-2\rangle \in E$并且$\langle b, w, L-1\rangle \in E\}$

   这样就满足步骤(1.3)定义的性质了)

   对所有$x \in V_1$，如果$x$在$k$级，$k = L-1$：

   令$(\lambda(x) \text{ of } G_1) = E_1$。

   对所有$x \in V_1$，如果$x$在$k$级，$k < (L-1)-1$： 令$(\lambda(x) \text{ of } G_1) = (\lambda(x) \text{ of } G)[1:k]$。

   令 $(ESS \text{ of } G_1) = ((ESS \text{ of } G) - E[L-2: L-1]) \cup \{ e \mid e = \langle c, w, (L-1)-1\rangle$，存在$b$使得$\langle c, b, L-2\rangle \in (ESS \text{ of } G)$并且$\langle b, w, L-1\rangle \in (ESS \text{ of } G)\}$。
4. 修改边集：

   对$G$中所有$\langle g_1, g_2, k\rangle$，如果$(\lambda(x) \text{ of } G_1)$包含$\langle g_1, g_2, k\rangle$并且$\langle g_1, g_2, k\rangle$已经变成$G_1$的$\langle g_1, g_2, k-1\rangle$，用$\langle g_1, g_2, k-1\rangle$替换$(\lambda(x) \text{ of } G_1)$中的$\langle g_1, g_2, k\rangle$。

现在证明以下结论(1)、(2)、(3)、(4)、(5)。

(1) $G_1$具备 ***Proving Algorithm*** 的步骤 1 所列性质。

根据$G_1$的构造，步骤 1 的(1.2)，(1.3)和(1.4)的性质容易验证。下面证明$G_1$具备(1.1)定义的性质。

若$G_1$有简单路径$P$，$P[1: L-1-2]$包含于 $(ESS \text{ of } G_1)$，那么$G$必有简单路径$P$且$P[1: L-3]$包含于$(ESS \text{ of } G)$。于是，根据图$G$满足的性质(1.3)，$G$必有简单路径$P$且$P[1: L-2]$包含于$(ESS \text{ of } G)$。由于$G$满足性质(1.1)，说明$P$包含于$(ESS \text{ of } G)$，根据$G_1$的构造，$P$包含于 $(ESS \text{ of } G_1)$。

(2) 将证明算法作用于$G_1$，$(Comp(ESS1, D, R(E)) \text{ of } G_1) \neq \emptyset$。

因为顶点$v$以上没有多入度点，压缩的结果，只是将两条邻接的边合并成一条边(例如，$\langle c, b, L-2\rangle$、$\langle b, w, L-1\rangle$变成$\langle c, w, (L-1)-1\rangle$)，所以，以$G_1$中为输入得到的所有的计算结果，包括所有$\lambda(x)$，以及所有$R(e)$，都与$G$为输入得到相应的计算结果本质上相同。所以，由$(Comp(ESS1, D, R(E)) \text{ of } G) \neq \emptyset$，知道$(Comp(ESS1, D, R(E)) \text{ of } G_1) \neq \emptyset$。

(3) $G_1$有简单路径$SP$且$(ESS \text{ of } G_1)$包含$SP$。

因为$G_1$是$L-1$级图，$f(G_1) < m$，$G_1$具备步骤 1 所列性质，同时$(Comp(ESS1, D, R(E)) \text{ of } G_1) \neq \emptyset$，所以，$G_1$有简单路径$SP$且$(ESS \text{ of } G_1)$包含$SP$。

(4) 如果$G_1$有简单路径$P = S - a_1 - \cdots - a_{L-3} - a_{L-1} - D$并且$(ESS \text{ of } G_1)$包含$P$，那么，$P' = S - a_1 - \cdots - a_{L-3} - a_{L-2} - a_{L-1} - D$一定是$G$的简单路径并且$(ESS \text{ of } G)$包含$P'$。

理由如下：

$(ESS \text{ of } G_1) = \big((ESS \text{ of } G) - E[L-2{:}\,L-1]\big) \cup \{\, e | e = \langle c, w, (L-1)-1\rangle$，存在$b$使得$\langle c, b, L-2\rangle$、$\langle b, w, L-1\rangle \in (ESS \text{ of } G)\}$。因此，$(ESS \text{ of } G_1)$包含$P$意味着$(ESS \text{ of } G)$包含$P'$。

$(\lambda(a_{L-1}) \text{ of } G_1)$包含$P[1{:}\,(L-1)-1]$即$(\lambda(a_{L-1}) \text{ of } G) \cap (\lambda(a_{L-2}) \text{ of } G)$包含$P'[1{:}\,L-1]$，因为 $(\lambda(w) \text{ of } G_1) = \Big(\big(Comp(\lambda(w), w, R(E)) \text{ of } G\big) - E[L-2{:}\,L-1]\Big) \cup \{\, e \mid e = \langle c, w, (L-1)-1\rangle$，存在$b$使得$\langle c, b, L-2\rangle$、$\langle b, w, L-1\rangle \in (\lambda(w) \text{ of } G)\}$。

(5) $G$有简单路径 $SP$且$ESS$包含$SP$。

根据结论(4)，结论(3)中提到的简单路径$SP$实际上就是$G$中一条包含于$(ESS \text{ of } G)$的简单路径。 ■

总结上面的讨论，由引理 1、引理 2、引理 3，我们有下面的$\boldsymbol{\alpha\beta}$**定理。**

$\boldsymbol{\alpha\beta}$ **定理.** 设$G =< V, E, S, D, L, \lambda >$和$ESS$是 ***Proving Algorithm*** 的输入。如果$Comp\big(ESS1, D, R(E)\big) \neq \emptyset$，$G$有简单路径 $SP$且$ESS$包含$SP$。

### 2.5 证明$\boldsymbol{ZH}$算法充分性

**定理 3.** 设$G =< V, E, S, D, L, \lambda >$是一个多级图。以$G$作为$ZH$算法输入，如果$Comp\big(\lambda(D), D, R(E)\big) \neq \emptyset$，则$G$中存在简单路径。

**证明** 在$G$中增加路径$D - w_1 - D_{new}$，令$\lambda(w_1) = \lambda(D_{new}) = E \cup \{D - w_1 - D_{new}\}$，我们得到$G'$。然后，我们令$ESS = E \cup \{D - w_1 - D_{new}\}$。$G'$具备 ***Proving Algorithm*** 步骤 1 的(1.2)、(1.3)、(1.4)所定义的性质。可以假定$G'$具备步骤(1.1)定义的性质，否则定理 3 无需证明。(要满足步骤 1 的(1.3)定义的性质，加标多级图定义中也必须对每个$\lambda(v)$的取值加限制。这样定义的$MSP$问题仍然是$NP$完全问题。实际上，我们从哈密顿图判定问题，还有从$SAT$问题等问题归结到$MSP$问题的时候，构造的顶点边集可以满足这样的性质的：对与某个点关联的边（即进入该点或者从该点出发的边），要么全部拥有，要么全部去掉)

因为$ZH$算法执行结果有$Comp\big(\lambda(D), D, R(E)\big) \neq \emptyset$，以$G'$和$ESS$为 ***Proving Algorithm*** 输入，必有$Comp\big(ESS1, D, R(E)\big) \neq \emptyset$。

依据$\boldsymbol{\alpha\beta}$ **定理**，$G'$有简单路径$SP$。这意味着$SP[1{:}\,L]$是$G$中的简单路径。 ■

# 3 $\boldsymbol{MSP \in NPC}$的证明以及本文结论

关于$MSP \in NPC$的证明，请参看文献[5~9]。已经将哈密顿图判定问题多项式归结到$MSP$问题[4~6]。事实上为了回答人们对于$MSP$问题$NP$完全性质的怀疑，我们已经发表十多个$NP$完全问题到$MSP$问题的归结。同时，也将$MSP$问题归结到了$SAT$问题[5~9]。

因为哈密顿图判定问题可以多项式时间归结到$MSP$问题，结合定理 1、定理 2、定理 3，我们得到：

**定理 4.** 存在多项式时间算法求解$MSP$问题。存在多项式时间算法求解哈密顿图判定问题。

**致谢**

# Reference

1. Michael R. Garey and David S. Johnson (1979), “Computer and In tractability, a Guide to the Theory of NP-completeness”, W.H.Freeman and Company

2. Moshe Y. Vardi (2010), On P, NP, and Computational Complexity, Communications of the ACM, 53(11):5.
3. Xinwen Jiang (2004), "Determining the H Property of A Graph by Changing It into A Multistage Graph", Computer Technology And Automation 23(2): 52-54
4. Xinwen Jiang, Qi Wang, and Ziheng Jiang (2010), The fourth version of the Proof for Z-H Algorithm to Solve MSP Problem, Computer Technology And Automation 29(3): 35-48
5. Xinwen Jiang, Lihong Peng and Qi Wang (2010), MSP Problem**:** Its NP-Completeness and its Algorithm, The proceedings of CUTE' 2010
6. Jiang, X.W. A Polynomial Time Algorithm for the Hamilton Circuit Problem. arXiv preprint cs/1305.5976 (2013)
7. Jiang, X.W., Liu, W.W., Wu, T.J., Zhou L.T. Reductions from MSP to SAT and from SUBSET SUM to MSP. J. Journal of Computational Information Systems, 10(3): 1287--1295 (2014)
8. Fan, S., Jiang, X.W., Peng, L.H. Polynomial-time Heuristic Algorithms for Several NP-complete Optimization Problems. J. Journal of Computational Information Systems , 10(22): 9707--9723 (2014)
9. Xinwen Jiang (2016), A new algorithm for the MSP problem