计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191200176

哈密顿图判定问题的多项式时间算法

姜新文
湘潭大学计算机学院·网络空间安全学院　湖南 湘潭 411105
国防科技大学计算机学院　长沙 410073
智能计算和信息处理教育部重点实验室　湖南 湘潭 411105

摘　要　NP=?P(即NP是否等于P)的问题是计算机科学和数学中的重要问题。美国克雷数学研究院将其列为新千年七大困难问题之首,2005年*Science*将其列为25个困难问题之19。*Science*最近列出的125个亟待解决的重要问题中,第19个问题实质上就是NP=?P的问题。如果NP=P,对于很多困扰科学研究的困难计算问题,理论上就存在多项式时间算法来迅速求解它们。而现代密码学建立在NP≠P的假设之上。人们希望存在难解问题,希望基于难解问题构造加密算法,希望能够利用难解问题的求解复杂性对抗分析和攻击。如果NP=P,所有在NP≠P假定之上开展的计算研究都至少需要重新审视其意义。NP完全问题的求解复杂性决定NP=P是否成立。针对一个被称为MSP问题的新问题,文中提出了一个关于MSP问题的多项式时间算法,并给出了该算法的证明和时间复杂性分析。由于已经发表了十多个经典的NP完全问题到MSP问题的归结以及MSP问题到SAT问题的归结,因此MSP问题存在多项式时间算法这样一个研究结果对于研究NP=P有重要和积极的意义。

关键词:MSP问题;HC问题;NP完全问题;多项式时间算法
中图法分类号　TP301.6

Polynomial Time Algorithm for Hamilton Circuit Problem

JIANG Xin-wen
School of Computer Science & School of Cyberspace Science, Xiangtan University, Xiangtan, Hunan 411105, China
School of Computer, National University of Defense Technology, Changsha 410073, China
The MOE Laboratory of Intelligent Computing & Information Processing, Xiangtan, Hunan 410005, China

Abstract　The 'P vs. NP problem' is the most famous and difficult problem in math. and computer science. Among the seven most important and difficult problems that the American Clay Mathematics Institute declared in 2000, this problem ranks the first one. Collecting 25 difficult problems that human being urgently want to solve, a list given by *Science* in 2005 contains the 'P vs. NP problem', ranking 19th. In the latest list of the most important 125 questions given by *Science*, the 'P vs. NP problem' ranks 19th too. Modern cryptography is based on the assumption that NP≠P. Whether NP=P or not depends on the complexity to solve a NPC problem. A new NP problem which is called MSP is proposed and a polynomial time algorithm to solve MSP is designed in this paper. To prove that the MSP problem is a NP complete problem, several papers, that reduced more than 10 NP complete problems to the MSP problem and reversely reduced the MSP problem to the SAT problem, were published in the last several years. Hence the result maybe helpful for proving NP=P.

Keywords　MSP problem, HC problem, NP-comlete problem, Polynomial time algorithm

NP=?P的问题是计算机科学和数学中的重要问题。美国克雷数学研究院将其列为新千年七大亟需解决的困难问题之首[1],2005年*Science*将其列为25个困难问题之19。*Science*最近列出的125个亟待解决的重要问题中,第19个问题实质上就是NP=?P的问题[2]。NP完全问题的求解复杂性决定NP=P是否成立。如果能够找到求解NP完全问题的多项式时间算法,即能够证明NP=P。

至今,仍然没有证明NP≠P或者NP=P的结果被公认。既具说服力又不具任何说服力的一个说法是:多年来找不到NP=P的证明,就是NP≠P的最好证明;多年来找不到NP≠P的证明,就是NP=P的最好证明。围绕证明NP≠P或者NP=P,人类做过许多尝试和努力[3-5]。总结起来,四十多年的所有努力包括4个方面。

1)在研究过程中产生了许多重要的相关成果。例如,在NP完全性研究中,找到了数以几千计的NP完全问题;为了求解一些复杂问题,开展了近似算法的研究;对计算模型的研

到稿日期:2019-12-30　返修日期:2020-04-10　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61272010)
This work was supported by the National Natural Science Foundation of China (61272010).
通信作者:姜新文(xinwenjiang@sina.com)

究使得人们对问题的难解性有了更深层次的认识,导致了密码学的革命性突破。Diffie 和 Hellman 提出了公钥密码体制的思想;Rivest,Shamir 和 Adlernan 提出了 RSA 公钥密码体制。作为现代密码学先驱,Diffie 和 Hellman 的工作在 2015 年获得了图灵奖。

2)围绕证明 NP≠P,研究者尝试了许多方法[4,6],例如泛对角化方法、电路复杂性方法、证明复杂性方法、切割平面方法以及基于多项式的代数证明系统等证明方法。有些方法,如电路复杂性方法曾经在很长时间被认为最有可能证明 NP≠P,其得到了很多关注,也产生了许多相关结果。遗憾的是,所有这些努力都没能给出令人信服的证明,并且无一例外地得到失败的结果[4]。

最近的关于 NP≠P 的轰动一时的努力,由惠普公司 Deolalikar 于 2010 年 8 月宣布完成[7-8],世界众多媒体顷刻之间进行了欢呼雀跃的报道。不幸的是,随之而来的审稿迅速找到了证明中的致命错误[8]。

ACM Transaction on Algorithms 开设有 NP 完全问题专栏,Johnson 长期主持该专栏。Deolalikar 失败之后,又一篇声称 NP≠P 的论文投稿 *ACM Transaction on Algorithms*。我们在审稿环节指出了其中的重要错误,受到高度赞赏。

3)围绕 NP=P 进行证明。证明 NP=P 的努力基本上都体现为多项式时间算法的寻找。随着一些长期没有被发现多项式时间算法的问题相继告破(如 Agrawal 等提出素性判定的 AKS 算法[9]),以及其他相近领域研究的突破(如 Valiant[10] 提出的全息算法解决了大量人们从前认为很难的计数问题),研究者们似乎看到了 NP=P 的可能。Knuth 是比较相信 NP=P 的,只是他悲观地认为就算给出了证明,也未必是构造性的;就算能找到算法,也会因复杂性过高而没有实际意义[11]。

与证明 NP≠P 的努力一样,证明 NP=P 的努力至今也没有产生被接受的证明。

4)围绕该问题不可判定进行。不断有研究机构就对 P vs. NP 问题的认识、NP≠P 或者 NP=P 的信念、该问题可能被解决的时间等进行问卷调查[12]。调查显示,多数人相信 NP≠P。

著名的 P-versus-NP page[1] 收集了全世界至今为了解决 P vs. NP 问题的成果和所做出的各种尝试。所有文章大致分为 3 类。极少研究认为该问题不可判定不可解决,其余围绕 NP≠P 或者 NP=P 进行。与许多机构进行的问卷调查结果不同,这个网站上列出的所有研究结果中,支持 NP=P 的明显多于支持 NP≠P 的。

本文的研究支持 NP=P 的信念,是对 NP≠P 的反动。哈密顿图判定问题是 NP 完全问题[13]。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,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[14-19]。本文的注意力集中在 MSP 问题的多项式时间求解。本文从此往后直至结尾全部内容完全属于简单图论问题最基本的算法设计与分析范畴。

(由于算法设计属于计算机、数学、密码学、软件工程等的基本能力,从这些专业的学者阅读本文需要的知识看,本文大致相当于数学教授眼中的中学数学题。但是下棋人人会,棋力各不同,阅读本文有一定难度。所以本文中常常夹带一些解释、导引、思想介绍、增加特殊标记之类的内容,写法上偏近教科书。很多人建议这样写,毕竟有助于阅读理解是比恪守形式更加重要的事情。不喜欢那些解释、导引、思想介绍之类的内容,可以将它们统统忽略或者去掉,让研究论文完全纯粹)

1 问题的引入及若干定义

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所设计的算法是否实现了计算目标,这需要证明。MSP 问题是一个人工构造的问题。直到最近两年,人们才发现 MSP 问题在基于 DAG 的区块链特别是在一个称为 hashgraph 区块链项目中的应用。本文的讨论从 MSP 问题的定义[14-18]开始。

定义 1 称 $G=\langle V,E,S,D,L,\lambda\rangle$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。$V=V_0\cup V_1\cup V_2\cup\cdots\cup V_L$,$V_i\cap V_j=\emptyset$,$0\leqslant i,j\leqslant L$,$i\neq j$。如果 $u\in V_i$($0\leqslant i\leqslant L$),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,G 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$\langle u,v,l\rangle\in E$ ($1\leqslant l\leqslant L$),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$\langle u,v,l\rangle$而不是通常的二元组$\langle u,v\rangle$表示边,这是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,用三元组表示更为直观,而且使得人们对处理与这些边相关的操作的复杂性的把握更加直接。

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取值定义如下。对于图 1(b),$\lambda(1)=\{e_1\}$,$\lambda(2)=\{e_2\}$,$\lambda(3)=\{e_1,e_2,e_3,e_4\}$,$\lambda(4)=\{e_1,e_3,e_5\}$,$\lambda(5)=\{e_2,e_4,e_6\}$,$\lambda(6)=\{e_1,e_3,e_5,e_{10}\}$,$\lambda(7)=\{e_{12}\}$,$\lambda(8)=\{e_1,e_3,e_6,e_8\}$,$\lambda(D)=\{e_1,e_3,e_5,e_{10},e_{12}\}$。对于图 1(a),$\lambda(1)=\emptyset$,$\lambda(2)=\emptyset$,$\lambda(3)=\emptyset$,$\lambda(4)=\{e_1,e_3,e_5\}$,$\lambda(5)=\{e_2,e_4,e_6\}$,$\lambda(6)=\{e_1,e_3,e_5\}$,$\lambda(7)=\lambda(7')=\{e_1,e_3,e_6,e_8\}$,$\lambda(8)=\{e_1,e_3,e_6,e_8\}$,$\lambda(D)=\emptyset$。

[1] http://www.win.tue.nl/~gwoegi/P-versus-NP.html

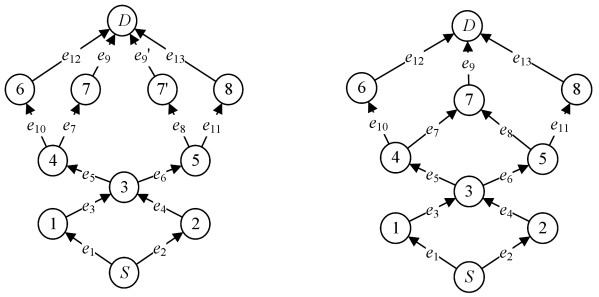

图 1 加标多级图的两个实例

Fig. 1 Two examples of labeled multistage graph

定义 2 设 $G=\langle V,E,S,D,L,\lambda\rangle$ 是一个加标多级图，P 是 G 中的一条路径，ES 是一个边集，$ES\subseteq E$。如果 P 的所有边都属于边集 ES，称 P 属于 ES，或者称 ES 包含 P，记为 $P\in ES$。如果 P 上有边不属于边集 ES，称 P 不属于 ES，记为 $P\notin ES$。

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

这里定义的简单路径不是图论中定义的简单路径。图论中简单路径的传统含义是：若一条路径称为简单路径，则该路径上的顶点不能重复。本文定义的加标多级图中，一条路径上的顶点肯定是不重复的。本文中的简单路径不仅顶点不重复，而且该路径必须满足该路径上所有顶点的顶点边集的约束，即 $S-\cdots-u_l\in\lambda(u_l)$，其中 $l\in\{1,2,\cdots,L\}$。

显然，根据定义 3，如果一条路径是简单路径，它必为半简单路径；反之不然。

设 $G=\langle V,E,S,D,L,\lambda\rangle$ 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 $S-u_1-\cdots-u_l-\cdots-u_L$ $(1\leqslant l\leqslant L, u_L=D)$，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，被称为多级图简单路径(Multistage-graph Simple Path，MSP)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(b)，$S-1-3-4-6-D$ 是一条简单路径，而图 1(a)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是否存在是容易的，这是一个简单的连通性判定问题。然而，判定简单路径的存在性，不是一件容易的事情。由于我们关注的路径需要满足该条路径上所有顶点边集的限制，即使当前寻找的部分路径满足当前的全部限制条件，我们也不知道未来能否满足更多限制。也就是说，MSP 问题的求解过程中遇到的困难，与很多难解问题的求解是类似的：我们发现任何局部的努力都可能是错误的和徒劳的。

下面讨论 MSP 问题的求解算法，称之为 ZH 算法。对于一个给定的加标多级图，ZH 算法能判定简单路径的存在性。

2 求解 MSP 问题的 ZH 算法

2.1 4 个基本算子的定义

基本算子 1：$[ES]_u^v$。

设 $G=\langle V,E,S,D,L,\lambda\rangle$ 是一个加标多级图。ES 是边集 E 的一个子集，$u,v\in V$。定义 $[ES]_u^v=\{e\mid e\in ES, e$ is on a path $u-\cdots-v$, and all the edges on $u-\cdots-v$ 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 $[ES]_u^v=\varnothing$。

$[ES]_u^v$ 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，所以从 ES 中去掉。$[ES]_u^v$ 的结果是 ES 的子集，只留下了 ES 中从 u 到 v 的路径上的那些边。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 $G=\langle V,E,S,D,L,\lambda\rangle$ 是一个加标多级图。如果 $G=\langle V,E,S,D,L,\lambda\rangle$ 中存在一条路径 $v-v_{l+1}-\cdots-v_{L-1}-D$，使得边 $\langle u,v,l\rangle\in\lambda(v)\bigcap\lambda(v_{l+1})\bigcap\cdots\bigcap\lambda(v_{L-1})\bigcap\lambda(D)$，则称路径 $v-v_{l+1}-\cdots-v_{L-1}-D$ 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。一般的算法在处理和描述路径时，都是将路径逐条描述出来，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。

下面的讨论中除了用 $R(u,v,l)$ 表示边 $\langle u,v,l\rangle$ 的可达路径集边集，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和终点时，我们选择用 $R(u,v,l)$ 表示；否则就简单地用 $R(e)$ 表示。

下面的算子 2 可以计算出任意一条边 $\langle u,v,l\rangle$ 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 $G=\langle V,E,S,D,L,\lambda\rangle$ 中边 $\langle u,v,l\rangle$ 的 $R(u,v,l)$ 值，计算结果放在 $R(u,v,l)$ 中。

1) $ES\leftarrow\{\langle a,b,k\rangle \mid \langle a,b,k\rangle\in E, l<k\leqslant L, \langle u,v,l\rangle\in\lambda(a)\bigcap\lambda(b)\}$ //Collecting edges

2) $R(u,v,l)\leftarrow[ES]_v^D$ //Linking edges together

其中，$R(u,v,l)$ 是一个变量符号，表示边 $\langle u,v,l\rangle$ 的可达路径集边集，所以计算结果可以放在 $R(u,v,l)$ 中。

按照定义，算子 2 计算的结果是，$R(u,v,l)$ 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，则可以设计 $O(|E|)$ 的算法计算 $Init(R(u,v,l))$。

注意，$R(e)$ 虽然称作可达路径集，但其仅仅是边的集合。本文讨论中所涉及的路径集合都是边集，因此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 和算子 4 适宜于被当作子程序看待(调用

子程序的结果是参数变量发生改变),算子 3 可被当作函数看待,函数值作为返回值。

设 $G=\langle V,E,S,D,L,\lambda\rangle$ 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←ES
2. For all e=⟨a,b,k⟩∈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←ES_temp−{e}.
   If k=l, l<L, and R(a,b,k) contains no path from v to D, ES_temp←ES_temp−{e}.
3. ES_temp←$[ES\_temp]_S^v$, where S is the unique vertex of V_0.
4. Repeat step 2 and step 3 until ES_temp will not change any more.

简单地说,上述步骤 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 $Comp(ES,v,R(E))$ 计算过程中需要使用的一组值,也可以作为全局变量定义,不需要作为参数变量代入算子中。有很多人建议在 $Comp(ES,v,R(E))$ 算子中明显给出 $R(E)$,也有很多人认为不需要给出 $R(E)$。这里在 $Comp(ES,v,R(E))$ 中明显给出 $R(E)$,使得我们可以明确看到 $R(E)$ 对 $Comp(ES,v,R(E))$ 的影响。事实上,ZH 算法就是一个反复利用 $R(E)$ 限制各个 $Comp(\lambda(v),v,R(E))$,又反过来利用所有的这些 $Comp(\lambda(v),v,R(E))$ 限制 $R(E)$ 的过程。

因为我们需要调用 $Comp(ES,v,R(E))$ 计算 $Comp(\lambda(D),D,R(E))$ 以及其他的 $Comp(\lambda(v),v,R(E))$,所以分 $k=l$ 或者 $k<l$ 两种情况讨论。因为假定了 v 是 l 级的顶点,$k<l$ 说明 $\langle a,b,k\rangle$ 在 l 级以下,顶点 b 距离 v 至少一级。这种情况下我们根据 $R(a,b,k)\cap ES\_temp$ 是否含 b 到 v 的路径决定 $\langle a,b,k\rangle$ 的去留。$k=l$ 说明顶点 b 与 v 同级。因为算子 3 的计算依赖于 $R(E)$,$Comp(\lambda(D),D,R(E))$ 的计算需要用到 $R(*,D,L)$,而 $R(*,D,L)$ 只能为空,所以 $k=l$ 的情况下,只有在 $l<L$ 时才依据 $[R(a,b,k)\cap ES\_temp]_v^D$ 是否为空来决定 $\langle a,b,k\rangle$ 的去留。

显然,对于任意 $\lambda(v)$,有 $Comp(\lambda(v),v,R(E))\subseteq\lambda(v)$。

可以设计 $O(|E|^2)$ 的算法计算步骤 2。步骤 2 和步骤 3 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 $Comp(ES,v,R(E))$ 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 $R(E)=\{R(e)\mid e\in E\}$ 中的 $R(e)$ 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 ⟨a,b,k⟩∈R(u,v,l), 1<l<k≤L−1
   if Comp($[\{e\mid e=\langle c,d,kk\rangle\in E, kk<l, [R(c,d,kk)\cap$ **Comp**$([\{\langle a,b,k\rangle\}\cup\{e\mid e=\langle x,y,ll\rangle\in E, ll<k, [R(x,y,ll)\cap\lambda(b)]_y^b$ **contains a path that contains** $\langle a,b,k\rangle\}]_S^b, b, R(E))]_d^b$ contains a path that contains ⟨u,v,l⟩ and ⟨a,b,k⟩$\}]_S^u$, u, R(E))≠∅
   then ⟨a,b,k⟩ is kept in R(u,v,l)
   else ⟨a,b,k⟩ is deleted from R(u,v,l).
2. R(u,v,l)←$[R(u,v,l)]_v^D$.
3. Repeat step 1 and step 2 until R(u,v,l) will not change any more.

上述计算过程中,$Comp([\{\langle a,b,k\rangle\}\cup\{e\mid e=\langle x,y,ll\rangle\in E, ll<k, [R(x,y,ll)\cap\lambda(b)]_y^b$ *contains a path that contains* $\langle a,b,k\rangle\}]_S^b, b, R(E))$ 是 $\lambda(b)$ 的子集。其中包含的边 e 都是从 $\langle a,b,k\rangle$ 进入 $\lambda(b)$ 的(即 $R(e)$ 包含经过 $\langle a,b,k\rangle$ 的路径)。为了后面描述方便,我们记这个集合为 A。可将 A 的形状想象为:将 $\langle a,b,k\rangle$ 看成一个柄,A 是一个柄下挂一个葫芦。

集合 $[\{e\mid e=\langle c,d,kk\rangle\in E, kk<l, [R(c,d,kk)\cap Comp([\{\langle a,b,k\rangle\}\cup\{e\mid e=\langle x,y,ll\rangle\in E, ll<k, [R(x,y,ll)\cap\lambda(b)]_y^b$ contains a path that contains $\langle a,b,k\rangle\}]_S^b, b, R(E))]_d^b$ contains a path that contains $\langle u,v,l\rangle$ and $\langle a,b,k\rangle\}]_S^u$ 是 A 的子集。为了后面描述方便,记这个集合为 C,记集合 $Comp(C,u,R(E))$ 为 B。B 包含在 A 中,可将 B 的形状想象为:A 是一个柄下挂一个葫芦,葫芦里面又有一条边 $\langle u,v,l\rangle$,将 $\langle u,v,l\rangle$ 看成一个柄,B 也是一个柄下挂一个葫芦。

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,则 $Comp(ES,u,R(E))$ 必须非空。

如果将算子 1−算子 3 的提出视为一种直觉的产生,算子 4 则主要出于一种逻辑证明的需要。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我们关心的性质。我们之所以能够完成算法证明,一个最有价值的发现便在于算子 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 $|E|*O(|E|^3)$ 的时间内计算 A 集并进一步计算 B 集,从而在 $|E|*|E|*O(|E|^3)$ 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 $R(u,v,l)$ 的子集,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。一般地,算法开始时变量 $R(u,v,l)$ 包含的是定义 4 定义的可达路径集边集,计算过程中 $R(u,v,l)$ 中包含的边的数量会单调减少。

2.2 ZH 算法的复杂性分析和必要性证明

算法 1 给出 ZH 算法的具体过程。

算法 1 ZH Algorithm

1. For all e∈E, we call Init(R(e)) to generate R(e) directly.
2. For l=2 to L−1
   2.1. For all ⟨u,v,l⟩ of stage l, call Change(R(u,v,l)) to modify R(u,v,l)
   2.2. For all v of stage l, λ(v)←Comp(λ(v),v,R(E))
3. Repeat step 2 until no R(e) and no λ(v) will change any more.
4. If Comp(λ(D),D,R(E))≠∅, we claim the existence of a simple path in G. Otherwise, we claim that there is no simple path in G.

ZH 算法的输入是 $G=\langle V,E,S,D,L,\lambda\rangle$,其中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,以及除源点 S 外的每个点 v 的边集 $\lambda(v)$。

我们将这些量都当成相应变量的初值。算法中的变量可以被修改，ZH 算法中 $\lambda(v)\leftarrow Comp(\lambda(v),v,R(E))$ 表示将修改的值放入 $\lambda(v)$。$R(e)$是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 1 的步骤 1 计算出了所有边的可达路径集边集；然后开始用 $R(E)$绑定 $R(u,v,l)$(步骤 2.1)，用 $R(E)$修改 $\lambda(v)$(步骤 2.2)，这个过程反复执行，直到所有 $R(e)$以及 $\lambda(v)$不再变化；算法的步骤 3 之后，基于 $R(E)$计算 $Comp(\lambda(D),D,R(E))$。

显然，$Comp(\lambda(v),v,R(E))$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，因为对每次迭代，至少有一个 $R(e)$中至少减少一条边，或者至少有一个 $\lambda(v)$中至少减少一条边，而 $R(e)$中和 $\lambda(v)$中边的总数受限于 $|E|$。

算法执行中会修改 $\lambda(v)$，$\lambda(v)$包含的边的数量单调减少，$\lambda(v)$中保留的值与初始值一般是不同的。简单路径的定义是基于初始值而不是计算值。由于计算过程中简单路径的存在性决不会被破坏(定理 2 将证明这一点)，如果 G 中有一条路径 $S-u_1-\cdots-u_l-\cdots-u_L(1\leqslant l\leqslant L,u_L=D)$满足条件 $S-\cdots-u_l\in\lambda(u_l)$，其中 $\lambda(u_l)$为初始值，那么 G 中有一条路径 $S-u_1-\cdots-u_l-\cdots-u_L(1\leqslant l\leqslant L,u_L=D)$满足条件 $S-\cdots-u_l\in\lambda(u_l)$，其中 $\lambda(u_l)$为计算值。我们约定：除非特别声明，下文的 $\lambda(v)$都是指变量 $\lambda(v)$中保留的值，即 $\lambda(v)$的计算值。

算法 1 的结论极为简洁：G 中存在简单路径，当且仅当 $Comp(\lambda(D),D,R(E))\neq\emptyset$。

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

证明：首先指出，任意边集和任意可达路径集中包含的边的条数$\leqslant|E|$，因为它们都是边集 E 的子集；顶点边集的个数$\leqslant|V|$；可达路径集的个数$\leqslant|E|$。

根据前文描述，计算 $Comp(ES,v,R(E))$的复杂性为 $O(|E|^3)$；计算 $Change(R(u,v,l))$的复杂性为 $O(|E|^6)$；对步骤 3 的每次迭代，某个 $R(u,v,l)$至少减少一条边，或者某个$\lambda(v)$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 $|E|*|E|*O(|E|^6)$。定理 1 得证。

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

证明：设 $v_0-v_1-v_2-\cdots-v_L$ 是 G 中的一条简单路径，$v_0=S,v_L=D$。根据简单路径的定义，有 $v_0-v_1-v_2-\cdots-v_l\in\lambda(v_l)(1\leqslant l\leqslant L)$，并且对于路径 $v_0-v_1-v_2-\cdots-v_L$ 上的所有$\langle v_{l-1},v_l,l\rangle(1\leqslant l\leqslant L)$，有$\langle v_{l-1},v_l,l\rangle\in\lambda(v_l)\bigcap\cdots\bigcap\lambda(v_{L-1})\bigcap\lambda(v_L)$。因此，ZH 算法的步骤 1 执行完毕后，$R(v_{l-1},v_l,l)$将包含 $v_l-v_{l+1}-\cdots-v_L(1\leqslant l\leqslant L)$；步骤 2 之后，$R(v_{l-1},v_l,l)$仍然包含 $v_l-v_{l+1}-\cdots-v_L(1\leqslant l\leqslant L)$；步骤 3 不会截断 $R(v_{l-1},v_l,l)$中的任何路径。因此，$Comp(\lambda(D),D,R(E))$包含 $v_0-v_1-v_2-\cdots-v_L$。

2.3 充分性证明准备

下面证明如果 $Comp(\lambda(D),D,R(E))\neq\emptyset$，则 G 中存在简单路径。

2.3.1 两个函数的定义

首先引进两个函数，即换名函数 I_y^x 和撕裂函数 $I_v^{v_1,v_2}$。两个函数都用于处理三元组，一个三元组在我们定义的加标多级图中表示一条边。

1)换名函数 I_y^x。设 EL 是一个集合，$ET=\{\langle a,b,k\rangle|a,b\in EL,k$ 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：

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

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

2)撕裂函数 $I_v^{v_1,v_2}$。设 EL 是一个集合；$ET=\{\langle a,b,k\rangle|a,b\in EL,k$ 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，$ES_1\neq\emptyset,ES_2\neq\emptyset,ES_1\bigcap ES_2=\emptyset$，$ES_1\bigcup ES_2=\{e|e\in ES,e=\langle c,v,l\rangle,c\in EL\}$。$I_v^{v_1,v_2}$ 定义如下：

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

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

下面介绍两个符号。

ZH\step4：表示 ZH 算法除步骤 4 之外的其余步骤。其只计算不判断。

$ES[i:j]$：设 ES 是个边集，$ES[i:j]$表示边集 ES 中从第 i 级到第 j 级的边，其中 $1\leqslant i\leqslant j\leqslant L$。如果 $i>j$，$ES[i:j]=\emptyset$。

还需要对算子 4 稍做修改。对于每条边$\langle a,b,k\rangle$在 $R(u,v,l)$中的去留，我们指定一条 L 级的边$\langle f,D,L\rangle$与$\langle a,b,k\rangle$一起绑定计算。

修改的算子 4：*V-Change*$(R(u,v,l),binding)$。

V-Change$(R(u,v,l),binding)$是用 $R(E)=\{R(e)|e\in E\}$中的 $R(e)$限制和绑定 $R(u,v,l)$。该算子将修改 $R(u,v,l)$中的值，修改后的值仍然放在 $R(u,v,l)$中。这里，*binding* 是$\{\langle\langle a,b,k\rangle,\langle f,D,L\rangle\rangle|\langle a,b,k\rangle,\langle f,D,L\rangle\in E,k\leqslant L-1\}$的子集，并且如果$\langle e_1,e_2\rangle\in binding$ 且$\langle e_1,e_3\rangle\in binding$，则 $e_2=e_3$。

1. For all ⟨a,b,k⟩∈R(u,v,l),1<l<k≤L−1
   1.1. 设⟨⟨a,b,k⟩,⟨f,D,L⟩⟩∈binding.
   1.2. If λ(b)⊈λ(f),we delete all ⟨ * ,b,k⟩ from R(u,v,l) and then goto step 2.
   1.3. if Comp ([{e|e=⟨c,d,kk⟩∈E,kk<l,[R(c,d,kk)⋂Comp

($[\{\langle \mathbf{a},\mathbf{b},\mathbf{k}\rangle\}\cup\{\mathbf{e}\mid \mathbf{e}=\langle \mathbf{x},\mathbf{y},\mathbf{ll}\rangle\in \mathbf{E},\mathbf{ll}<\mathbf{k},[\mathbf{R}(\mathbf{x},\mathbf{y},\mathbf{ll})\cap\boldsymbol{\lambda}(\mathbf{b})\cap\boldsymbol{\lambda}(\mathbf{f})]_{\mathbf{y}}^{\mathbf{b}}$ **contains a path that contains** $\langle \mathbf{a},\mathbf{b},\mathbf{k}\rangle\}]_{\mathbf{S}}^{\mathbf{b}},\mathbf{b},\mathbf{R}(\mathbf{E}))]_{\mathbf{d}}^{\mathbf{b}}$

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.

步骤 1.2 中"delete all $\langle *,b,k\rangle$ from $R(u,v,l)$"意味着所有终止于 b 的边都从 $R(u,v,l)$ 中被删除。修改的算子 4 与算子 4 的差别仅仅在于 A 集的计算。修改的算子 4 在确认 $\langle a,b,k\rangle$ 在 $R(u,v,l)$ 中的去留时,与 L 级的某条边 $\langle f,D,L\rangle$ "绑定",计算出来的 A 集是 $\lambda(b)\cap\lambda(f)$ 的子集;而算子 4 计算出来的 A 集仅仅是 $\lambda(b)$ 的子集。

显然,如果 $\langle a,b,k\rangle$ 在 $R(u,v,l)$ 中保留,则有 $\langle f,D,L\rangle\in[E]_b^D$。$[E]_b^D$ 是 G 中顶点 b 往上的边构成的集合。

ZH\step4 中只调用 *V-Change*($R(u,v,l)$,*binding*),不调用 *Change*($R(u,v,l)$)。

为了完成 ZH 算法的充分性证明,我们构造 Proving Algorithm(算法 2)。ZH 算法被镶嵌在算法 2 中(见步骤 2),构成其中的一个"实质性"部分。Proving Algorithm 的输入包括图 $G=\langle V,E,S,D,L,\lambda\rangle$ 以及一个边集 ESS ($ESS\subseteq E$)。为了讨论的简便性,我们要求 Proving Algorithm 的输入满足一些性质(见步骤 1)。

算法 2 Proving Algorithm

1. 如果输入的图 $G=\langle V,E,S,D,L,\lambda\rangle$ 不具备以下性质,则算法停机:

1.1. $\lambda(D)=E$;

1.2. 对第 L−1 级的所有点 v,d(v)=1,其中 d(v)是点 v 的入度;

1.3. $ESS[L-1:L-1]=E[L-1:L-1]$。

2. 产生 binding 集合:对于每条边 $\langle a,b,t\rangle$,$t\leqslant L-1$,我们指定一条 L 级的边 $\langle f,D,L\rangle$ 与 $\langle a,b,t\rangle$ 一起绑定计算。$\langle f,D,L\rangle$ 必须满足以下条件,否则算法停机:若 $Q=S-a_1-\cdots-a_{L-1}-D\in\{P_1\mid P_1$ 是一条简单路径,$P_1[1:k]\in ESS$,$P_1[1:k+1]\notin ESS$,$1\leqslant k\leqslant L-1$,没有简单路径 P_2 使得 $P_2[1:k]=P_1[1:k]$ 并且 $P_2[1:k+1]\in ESS\}$,则有 $\langle f,D,L\rangle\notin[E]_{a_k}^D$。

3. Apply ZH\step4 on G.

4. $ESS1\leftarrow ESS\cap Comp(\lambda(D),D,R(E))\cup E[L:L]$.

5. If $Comp(ESS1,D,R(E))\neq\emptyset$, there exists a simple path SP in G such that ESS contains SP.

为了证明的需要,对于所有多级图,我们定义一个"大小"关系。对于在这个"大小"关系下最小的图,我们直接证明算法 2 正确;而对于其他任意的图,我们通过"撕裂变换"或者"压缩变换"构造"更小"的图,从而进一步形成反驳。因为反驳中需要有新的、"更小"的算法输入实例构造,而新的构造必须同样具备这些性质,为了讨论中不遗漏,我们将这些性质全部列在算法 2 中。

任意输入的图如果不满足这些性质,算法 2 将停机。算法 2 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。

算法 2 中步骤 1.2 限制了图的"形状和结构"。按照步骤 1.2,输入的图的 $L-1$ 级都是单入度点的图。步骤 1.1 和步骤 1.3 限制了顶点边集以及 ESS 的取值。

步骤 3 的 ZH\step4 中调用 *V-Change*($R(u,v,l)$,*binding*),所以步骤 2 产生 *binding* 集合。对于每条边 $\langle a,b,t\rangle$,$t\leqslant L-1$,指定一条 L 级的边 $\langle f,D,L\rangle$ 与 $\langle a,b,t\rangle$ 一起绑定计算。

$Q=S-a_1-\cdots-a_{L-1}-D\in\{P_1\mid P_1$ 是一条简单路径,$P_1[1:k]\in ESS$,$P_1[1:k+1]\notin ESS$,$k\geqslant 1$,没有简单路径 P_2 使得 $P_2[1:k]=P_1[1:k]$ 并且 $P_2[1:k+1]\in ESS\}$,表示 Q 是满足如下条件的简单路径:Q 有"极大"的 k 使得 $Q[1:k]\in ESS$。

Q 有"极大"的 k 使得 $Q[1:k]\in ESS$ 是我们借用的数学概念。k 是个"极大值",不是"最大值"。给定的图中可能有简单路径 P 满足条件 $P[1:k+1]\in ESS$,但不可能有简单路径 P 满足条件 $P[1:k+1]\in ESS$ 并且 $P[1:k]=Q[1:k]$。

我们称 Q 为 Max-k 路径。若 $Q=S-a_1-\cdots-a_{L-1}-D$ 是一条 Max-k 路径,则顶点 a_k 往上的所有边构成集合 $[E]_{a_k}^D$。集合 $[E]_{a_k}^D$ 被称为 Q 的辖域,可以将其想象成一个扇形。$\langle f,D,L\rangle\notin[E]_{a_k}^D$ 说明 $\langle f,D,L\rangle$ 不在 Q 的辖域中。Q 的任意性,使得 $\langle f,D,L\rangle$ 不在任何 Max-k 路径的辖域中。这是 $\langle f,D,L\rangle$ 必须满足的一个条件,如果产生的 *binding* 不满足这个条件,算法就停机了。

显然,不同的指定会导致 *V-Change*($R(u,v,l)$,*binding*) 的计算结果不同,并最终导致步骤 5 中 $Comp(ESS1,D,R(E))$ 的计算结果不同;甚至不同的指定会导致算法停机。本文只对本算法做充分性判定。如果 $Comp(ESS1,D,R(E))$ 为空,或者算法停机了,算法 2 不做任何推断。只要有一组这样的指定使得 $Comp(ESS1,D,R(E))\neq\emptyset$,算法就推断 G 有简单路径 SP 且 ESS 包含 SP。

步骤 3 即为 ZH 算法。步骤 4 得到的 $ESS1$ 中的 $ESS1[1:L-1]$ 是 ESS 的子集。

算法 2 可能漏判,但我们证明它绝不误判。

步骤 1 容易实现,其中 1.1 和 1.3 只需要进行集合的比较;1.2 只需要对 $L-1$ 级的每个点 v,检查 $d(v)$ 是否等于 1 即可。

步骤 2 产生 binding 集合也容易实现。判定 $\langle f,D,L\rangle$ 是否满足条件可以这样进行:穷举每一条 Max-k 路径 $S-a_1-\cdots-a_{L-1}-D$,并计算 $[E]_{a_k}^D$,合并所有 $[E]_{a_k}^D$ 得到一个集合,然后判断 $\langle f,D,L\rangle$ 是否属于该集合。

实际上,算法 2 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的存在性问题;而算法 2 解决另外一个问题:给定 $G=\langle V,E,S,D,L,\lambda\rangle$ 以及一个边集 ESS,$ESS\subseteq E$,G 中有简单路径包含于 ESS 吗?

算法执行中会修改 $\lambda(v)$,$\lambda(v)$ 中保留的值与初始值一般是不同的。$\lambda(v)$ 中保留的值被称为计算值。算法 2 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 中。

2.4 *αβ* 定理及其证明

下面开始归纳证明,对于任意输入的 G 和 ESS,算法 2 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 $G=\langle V,E,S,D,L,\lambda\rangle$ 和 ESS 是算法 2 的输入，G 中第 3 级到第 $L-1(L\geqslant 4)$ 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。

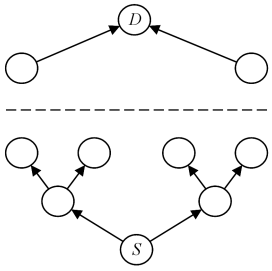

图 2 引理 1 的典型图例

Fig. 2 Typical graph of lemma 1

证明：G 中第 3 级到 $L-1$ 级没有多入度点，第 2 级可能有多入度点。

因为 $Comp(ESS1,D,R(E))\neq\emptyset$，所以一定有边 $\langle a_1,a_2,2\rangle\in Comp(ESS1,D,R(E))$，并且 $R(a_1,a_2,2)\cap Comp(ESS1,D,R(E))$ 包含 $a_2-\cdots-a_{L-1}-D$。由于 $R(a_1,a_2,2)\cap Comp(ESS1,D,R(E))$ 包含 $a_2-\cdots-a_{L-1}-D$，依据 $V\text{-}Change(R(a_1,a_2,2),binding)$ 的计算过程可知 $S-a_1\in\lambda(a_1)$，$S-a_1-a_2\in\lambda(a_2)\cap\cdots\cap\lambda(a_{L-1})\cap\lambda(D)$。

由 G 中第 3 级到 $L-1$ 级没有多入度点，且 $ESS1=ESS\cap Comp(\lambda(D),D,R(E))\cup E[L:L]$ 包含 $S-a_1-\cdots-a_{L-1}-D$ 可知，$S-a_1-\cdots-a_{L-1}-D$ 为 G 中的简单路径，并且 $S-a_1-\cdots-a_{L-1}\in ESS$。

如果 $\langle a_{L-1},D,L\rangle\in ESS$，则 G 有简单路径 SP 且 ESS 包含 SP。

如果 $\langle a_{L-1},D,L\rangle\notin ESS$，由于 $S-a_1-\cdots-a_{L-1}-D$ 是一条 Max-k 路径，$\langle a_{L-1},D,L\rangle$ 不可能被指定与 $\langle a_{L-2},a_{L-1},L-1\rangle$ 一起绑定计算，因此不可能有 $\langle a_1,a_2,2\rangle\in Comp(ESS1,D,R(E))$ 并且 $R(a_1,a_2,2)\cap Comp(ESS1,D,R(E))$ 包含 $a_2-\cdots-a_{L-1}-D$。这与 $Comp(ESS1,D,R(E))\neq\emptyset$ 矛盾。证毕。

现在给每个 $G=\langle V,E,S,D,L,\lambda\rangle$ 定义一个度量 $f(G)$：

$$f(G)=L+\sum_{v\in(V-\{S,D\}-V_2)}(d(v)-1)$$

其中，$d(v)$ 是 v 的入度，V_2 是第 2 级所有顶点的集合，$V-\{S,D\}-V_2$ 是 G 中除 D 和 S 以及 V_2 中的顶点之外的所有顶点的集合。

说明：符号 V_2 在后文中有其他的使用意义。仅在 $f(G)$ 的定义中，V_2 表示 G 中第 2 级所有顶点的集合。

因为 G 中入度为 0 的点肯定不在简单路径上，我们可以简单地将这些点从 G 中删去。为了讨论的简便性，假定对 G 中所有顶点 v，有 $d(v)-1\geqslant 0$。因此，对任意的 G，$f(G)\geqslant L$。

显然，对于任意输入的图 G，若 $f(G)=4$ 且 $Comp(ESS1,D,R(E))\neq\emptyset$，则必然有 $L=4$。根据引理 1，算法 2 能做出正确推断。

假定对于任意输入的图 G，若 $f(G)<m$ $(m>4)$，则算法 2 都能做出正确推断。我们下面考虑 $f(G)=m$ 的情况。注意，此时必然有 $L\geqslant 5$。

此时，如果 G 中第 3 级到 $L-1$ 级没有多入度点，根据引理 1，算法 2 能做出正确推断。如果 G 中第 3 级到 $L-1$ 级有多入度点，则分引理 2(有多入度点在 $L-2$ 级)和引理 3(有多入度点但没有多入度点在 $L-2$ 级)两种情况讨论。

引理 2 设 $G=\langle V,E,S,D,L,\lambda\rangle$ 和 ESS 是算法 2 的输入，$f(G)=m$，顶点 v 是 G 中 l 级的一个多入度点，$l=L-2$ 且 $l>2$，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。

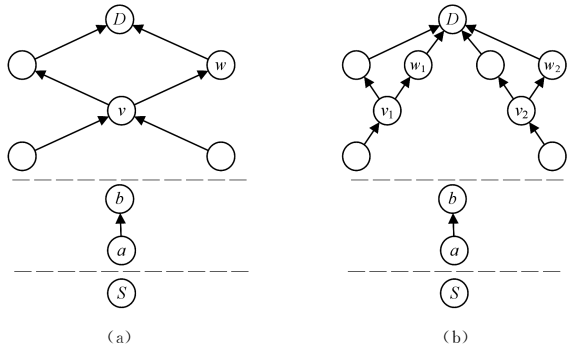

图 3 引理 2 的典型图例

Fig. 3 Typical graph of lemma 2

证明：引理 2 的证明思想是，构造一个新的 L 级图 G_1，使其满足 3 个条件，从而完成反驳。1)G_1 具备算法 2 中步骤 1 描述的性质，并且 $f(G_1)<f(G)$；2)如果 G 作为输入时 $Comp(ESS1,D,R(E))\neq\emptyset$，则 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，则 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。要构造另外一个加标多级图，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，并且新图中的一条包含于新的 ESS 的简单路径“实质上”就是原图中的一条包含于原来的 ESS 的简单路径？

我们对 $\lambda(v_1)$ 和 $\lambda(v_2)$ 进行重复设置，除了边被换名，其本质上都等于 $\lambda(v)$。同样地，对 $\lambda(w_1)$ 和 $\lambda(w_2)$ 进行重复设置，除了边被换名，其本质上都等于 $\lambda(w)$。新的 ESS，除了边被换名，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。图 3(b)中的一条包含于新的 ESS 的简单路径“实质上”就是图 3(a)中的一条包含于原来的 ESS 的简单路径。

以上初始设置是容易的，但是如果原来 $Comp(\lambda(v),v,R(E))$ 非空，现在以新的图为输入，$Comp(\lambda(v_1),v_1,R(E))$ 和 $Comp(\lambda(v_2),v_2,R(E))$ 会非空吗？或者说，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 为这个目的而产生，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，算法 2 能够得到本质上相同的结果，就是算子 4 设计的方向指导。我们之所以能够完成算法证明，是将问题(MSP 问

题)设计、算法(ZH 算法及算子)设计及证明框架(撕裂得到更小的图以及算法 2)一起考虑,慢慢调出来的。我们的方向非常明确和坚定,就是寻找一种算法实现以下目标:如果对于给定的图 G,算法能够得到一个计算结果,那么,希望算法在一个"更小"的图上,能够得到"本质"相同的计算结果。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!

证明开始。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 和 v_2,然后逐个撕裂 $L-1$ 级的多入度点,由此得到一个 $L-1$ 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 的定义如下。

令 $V_1=(V-\{v,w,w'\})\cup\{v_1,v_2\}\cup\{w_1,w_2,w_3,w_4\}$。$E_1=I_w^{w_1,w_2}(I_w^{w_3,w_4}(I_v^{v_1,v_2}(E\text{ of }G,group_1,group_2),\{\langle v_1,w',L-1\rangle\},\{\langle v_2,w',L-1\rangle\}),\{\langle v_1,w,L-1\rangle\},\{\langle v_2,w,L-1\rangle\})$(其中($E$ of G)表示 G 为输入的 E,这里就指 E。本文中经常要比较 G 为输入的某个量和 G_1 为输入的某个量,为了方便,我们用(*something* of G)表示 G 为输入的某个量,用(*something* of G_1)表示 G_1 为输入的某个量)。$(ESS\text{ of }G_1)=I_w^{w_1,w_2}((I_w^{w_3,w_4}(I_v^{v_1,v_2}(ESS\text{ of }G,group_1,group_2),\{\langle v_1,w',L-1\rangle\},\{\langle v_2,w',L-1\rangle\}),\{\langle v_1,w,L-1\rangle\},\{\langle v_2,w,L-1\rangle\})$。

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

对于 $x=v_1,v_2,w_1,w_2,w_3,w_4$,给($\lambda(x)$ of G_1)赋一个值,使得:

$$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(v_1))=I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(v_2))=\lambda(v)$$

$$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(w_1))=I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(w_2))=\lambda(w)$$

$$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(w_3))=I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(w_4))=\lambda(w')$$

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,按照构造,对于 G_1 的所有 x,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(\lambda(x)\text{ of }G_1)\subseteq(\lambda(x)\text{ of }G)$。如果 P 是 G_1 的简单路径,则 $I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(P)$是 G 的简单路径。

(特别构建 1) 对所有边$\langle a,b,k\rangle$,$2\leqslant k\leqslant L-4$,在 G_1 中新增加 b 到 D 的路径$b-\cdots-D$,将该路径上所有顶点的顶点边集设置成等于 E_1。但(ESS of G_1)仅仅增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边。如果有简单路径 P 经过 b 并且 $P[1:k]\in(ESS\text{ of }G_1)$,再新增加简单路径 Q 使得 $Q[1:k]=P[1:k]$,$Q[k+1:L]$是全新增加,(ESS of G_1)增加 $Q[k+1:k+1]$并且增加 $Q[L-1:L]$。

说明:(ESS of G_1)增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边,以及(ESS of G_1)增加 $Q[L-1:L]$,使得 G_1 仍然具备语句 1.3 定义的性质。

增加路径 $b-\cdots-D$ 后,G_1 没有增加包含于(ESS of G_1)的简单路径。增加 b 到 D 的路径,是使得原来经过 $\lambda(b)$到达 $\lambda(D)$的边,在 G_1 中可以经过新的路径 $b-\cdots-D$ 到达 $\lambda(D)$。这个增加有助于我们确认两件事情:1)对所有以 b 为终点的边,如果指定路径 $b-\cdots-D$ 上的 L 级的边与之绑定计算,对修改的算子 4 确认以 b 为终点的边的去留时,不产生任何限制意义,因为该路径上所有的顶点边集都等于 E_1;2)如果撕裂前 $R(e)$有可达路径经过 b,撕裂后一定有可达路径经过 b,因为路径 $b-\cdots-D$ 上所有顶点边集都等于E_1,e 到达b 之后不再有任何因素阻止 e。

增加简单路径 Q,使得 $Q[1:k]=P[1:k]$;同时(ESS of G_1)增加 $Q[k+1:k+1]$,使得路径 $b-\cdots-D$ 不在一条 Max-k 路径的辖域中。因为 $k\leqslant L-4$,所以 $Q[k+2:L-2]\notin(ESS\text{ of }G_1)$,又因为路径 $b-\cdots-D$ 上的第 $k+1$ 级的边也不在(ESS of G_1)中,所以 G_1 没有增加包含于(ESS of G_1)的简单路径。

(特别构建 2) 对所有边$\langle a,b,k\rangle$,$k=L-3$,$\langle a,b,k\rangle\in(ESS1\text{ of }G)$,在 G_1 中新增加 b 到 D 的路径$b-\cdots-D$。将该路径上所有顶点的顶点边集设置成等于 E_1。但(ESS of G_1)仅仅增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边。

说明:此时不会有简单路径 P 经过 b,并且 $P[1:k]\in(ESS\text{ of }G_1)$,$P[1:k+1]\notin(ESS\text{ of }G_1)$,否则 G 中有简单路径 P'经过 b 并且 $P'[1:k]\in(ESS\text{ of }G)$,$P'$是 Max-$k$ 路径,$\langle a,b,k\rangle\notin(ESS1\text{ of }G)$。

至此,已完成 G_1 的构造以及顶点边集和 ESS 的定义,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。

将 G_1 的构造总结如下: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"本质上"是一样的(G_1 中通过特别构建增加了一些"旁路"路径,不影响图的"大小",但可以辅助计算)。

现在,将构造的 G_1 和(ESS of G_1)作为算法 2 的输入进行计算。可以证明以下结论(1)-(5)。

(1)$f(G_1)<f(G)$。

设 v 是出现在 l 级的多入度点,$l=L-2$。

$$\begin{aligned}&\sum_{u\in V_l\text{ of }G_1}(d(u)-1)\\&=\sum_{u\in(V_l-\{v_1,v_2\})\text{ of }G_1}(d(u)-1)+(d(v_1)-1)+(d(v_2)-1)\\&=\sum_{u\in(V_l-\{v\})\text{ of }G}(d(u)-1)+(d(v)-1)-1\\&=\sum_{u\in V_l\text{ of }G}(d(u)-1)-1\end{aligned}$$

所以 $f(G_1)<f(G)$。

(2)对 G 中 $L-2$ 级的每个 $\lambda(v)$,如果算法 2 步骤 3 之后 $\lambda(v)$包含边 $e=\langle a,b,k\rangle$,$R(a,b,k)$包含边$\langle u,v,L-2\rangle$,$k<L-2$,G 一定有经过边$\langle a,b,k\rangle$和边$\langle u,v,L-2\rangle$,$\langle v,w,L-1\rangle$的简单路径 P,并且 $R(a,b,k)$包含 $P[k+1:L]$。

后文将给出结论(2)的证明。

结论(2)非常重要。因为 G 有简单路径经过边$\langle a,b,k\rangle$,

$\langle u,v,L-2\rangle$和点 w，撕裂之后，这条简单路径必然经过 w_1 或者 w_2，所以如果$\langle a,b,k\rangle$曾经进入($ESS1$ of G)，那么$\langle a,b,k\rangle$也必将进入($ESS1$ of G_1)，由此导致 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($ESS1$ of G_1)$\supseteq$($ESS1$ of G)。另外，如果($R(a,b,k)$ of G)包含某条经过$\langle u,v,L-2\rangle$的路径，则($R(a,b,k)$ of G_1)必然包含某条经过$\langle u,v_i,L-2\rangle$的路径（因为有结论(2)，G 一定包含经过边$\langle a,b,k\rangle$和边$\langle u,v,L-2\rangle$，$\langle v,w,L-1\rangle$的简单路径，所以($R(a,b,k)$ of G_1)必然包含该简单路径中从 b 到 D 的部分路径)。这将进一步导致 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($Comp(ESS1,D,R(E))$ of G_1)$\supseteq$($Comp(ESS1,D,R(E))$ of G)。

(3)G_1 具备算法 2 中步骤 1 要求的全部性质。G_1 具备步骤 1.1－步骤 1.3 定义的性质容易逐条验证。

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

先考虑算法 2 步骤 2 中“对于每条边$\langle a,b,k\rangle$，$k\leqslant L-1$，我们指定一条 L 级的边$\langle f,D,L\rangle$与$\langle a,b,k\rangle$一起绑定计算 $V\text{-}Change(R(u,v,l),binding)$”的问题。

对于$\langle a,b,k\rangle$，$k\leqslant L-3$，因为增加了 $b-b_{k+1}-\cdots-b_{L-1}-D$，该路径上所有顶点边集等于 E_1，所以可为$\langle a,b,k\rangle$指定$\langle b_{L-1},D,L\rangle$与之绑定计算，同时为路径 $b-b_{k+1}-\cdots-b_{L-1}$上的边指定$\langle b_{L-1},D,L\rangle$与之绑定计算。理由如下：若原来以 G 为输入时$\langle b,*,k+1\rangle$在某条 Max-k 路径辖域中，则被指定与$\langle a,b,k\rangle$绑定计算的任何边不可能在$[E]_b^D$ 中，于是$\langle a,b,k\rangle$必然不属于 $R(u,v,l)$。若原来以 G 为输入时$\langle b,*,k+1\rangle$不在任何 Max-k 路径辖域中，即使增加 $b-b_{k+1}-\cdots-b_{L-1}-D$ 导致增加简单路径 P，P 经过 b 并且 $P[1:k]\in$(ESS of G_1)，P 也不可能是 Max-k 路径，只有新增加的简单路径 Q 才是 Max-k 路径，而$\langle b_{L-1},D,L\rangle$不可能在 Q 的辖域中，故可为$\langle a,b,k\rangle$指定$\langle b_{L-1},D,L\rangle$与之绑定计算。如此指定之后，以 G_1 为输入时，修改的算子 4 在确认$\langle a,b,k\rangle$在 $R(u,v,l)$中的去留时，计算出来的 A 集是$\lambda(b)\bigcap\lambda(b_{L-1})=\lambda(b)$的子集。

对于路径 $b-b_{k+1}-\cdots-b_{L-1}-D$ 上的边，可指定$\langle b_{L-1},D,L\rangle$与之绑定计算。

对于路径 $Q[k+1:L-1]$上的边，也可指定$\langle b_{L-1},D,L\rangle$与之绑定计算(只是这样指定，任何 $R(e)$才不会包含 $Q[k+1:L-1]$上的边)。

对于$\langle a,b,k\rangle$，$k=L-2$，根据结论(2)，G 有简单路径经过边$\langle a,b,k\rangle$，$\langle u,v,L-2\rangle$和点 w，撕裂之后，这条简单路径必然经过 w_1 或者 w_2。$\lambda(v_i)(i=1,2)$必然是某个 $\lambda(w_j)$的子集，故可以继续原来指定的 L 级的边与$\langle a,b,k\rangle$绑定计算。由于原来被指定与$\langle a,b,k\rangle$绑定计算的 L 级的边可能变成两条边，因此如果与$\langle a,b,k\rangle$绑定计算的 L 级的边变成两条边，则指定其中之一与$\langle a,b,k\rangle$绑定计算即可。这样，以 G_1 为输入时修改的算子 4 在确认$\langle a,b,k\rangle$在 $R(u,v,l)$中的去留时，计算出来的 A 集必然非空，B 集也必然非空。

对于$\langle a,b,k\rangle$，$k=L-1$，由于原来被指定与$\langle a,b,k\rangle$绑定计算的 L 级的边$\langle f,D,L\rangle$可能变成两条边$\langle f_1,D,L\rangle$和$\langle f_2,D,L\rangle$，$\langle a,b,k\rangle$本身也可能变成两条边 e_1 和 e_2，因此如果 e_i 在简单路径上，指定$\langle f_i,D,L\rangle$与 e_i 绑定计算即可。这里，$i=1,2$。

下面分析，将证明算法(即算法 2)作用于 G_1 的情况下 $R(e)$的构成。

对每条边$\langle r,s,k\rangle$，若($R(r,s,k)$ of G)的初始值包含$\langle o,p,kk\rangle$，则必然有 e_1 和 e_2，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_1)=\langle r,s,k\rangle$，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_2)=\langle o,p,kk\rangle$，($R(e_1)$ of G_2)的初始值必包含 e_2。

因为，对每条边$\langle o,p,kk\rangle$，$kk<L-2$，有 p 到 D 的路径 $p-\cdots-D$，该路径上所有节点的顶点边集等于 E_1，对每条边$\langle r,s,k\rangle$，若($R(r,s,k)$ of G)的计算值包含$\langle o,p,kk\rangle$，所以，G_1 中必然有边 e_1 和 e_2，其中 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_1)=\langle r,s,k\rangle$，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_2)=\langle o,p,kk\rangle$，($R(e_1)$ of G_1)的计算值包含 e_2。

对每条边$\langle o,p,kk\rangle$，$kk\geqslant L-2$，对每条边$\langle r,s,k\rangle$，若($R(r,s,k)$ of G)的计算值包含$\langle o,p,kk\rangle$，则根据结论(2)，G_1 中必然有边 e_1 和 e_2，其中 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_1)=\langle r,s,k\rangle$，$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(e_2)=\langle o,p,kk\rangle$，($R(e_1)$ of G_1)的计算值包含 e_2。

接着分析将证明算法作用于 G_1 的情况下($Comp(\lambda(D),D,R(E))$ of G_1)的构成。

设 $e\in$($Comp(\lambda(D),D,R(E))$ of G)，必有 $L-2$ 级的某个($Comp(\lambda(x),x,R(E))$ of G)包含 e。根据结论(2)，e 在某条简单路径上。除了在撕裂点处被换名，换名后的这条简单路径也在 G_1 中。算子 3 不会破坏这条简单路径，所以换名后的这条简单路径最终也在($Comp(\lambda(D),D,R(E))$ of G_1)中。

于是：

$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($Comp(\lambda(D),D,R(E))$ of G_1)$\supseteq$($Comp(\lambda(D),D,R(E))$ of G)

$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($ESS\bigcap Comp(\lambda(D),D,R(E))$ of G_1)$\supseteq$($ESS\bigcap Comp(\lambda(D),D,R(E))$ of G)

$I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($Comp(ESS1,D,R(E))$ of G_1)$\supseteq$($Comp(ESS1,D,R(E))$ of G)$\neq\emptyset$

严格地说，一定有 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($Comp(ESS1,D,R(E))$ of G_1)$[1:L-2]\supseteq$($Comp(ESS1,D,R(E))$ of G)$[1:L-2]\neq\emptyset$，因为顶点 v 往上的所有路径中，至少有一条(可能只有一条)路径在 $I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}$($Comp(ESS1,D,R(E))$ of G_1)中留下。

(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) $\neq\emptyset$，所以 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，则 $P'=I_{w'}^{w_4}I_{w'}^{w_3}I_{w}^{w_2}I_{w}^{w_1}I_{v}^{v_2}I_{v}^{v_1}(SP)$必为 G 中的简单路径。证毕。

对于引理 2 的结论(2)，我们证明：对 G 中 $L-2$ 级的每个 $\lambda(v)$，如果步骤 3 之后 $\lambda(v)$ 包含边 $e=\langle a,b,k\rangle$，$R(a,b,k)$包含边$\langle u,v,L-2\rangle$，$k<L-2$，则 G 一定有经过边$\langle a,b,k\rangle$和边$\langle u,v,L-2\rangle$，$\langle v,w,L-1\rangle$的简单路径 P，并且 $R(a,b,k)$包含 $P[k+1:L]$。

步骤 3 之后的 $\lambda(v)$不是算法输入时的初值，而是算法 2 中步骤 3 结束后 $\lambda(v)$中留下的值。

算子 4 以及本结论的证明是我们能够完成算法证明的关键。

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' 作为算法 2 的输入,如果 $f(G')<m$ 并且得到 $Comp(ESS1,D,R(E))\neq\emptyset$,则 G' 中必有简单路径包含于 ESS'。

因此,我们基于 G_1 构造一个 G_2,G_2 的形状与 G_1 完全相同,以确保 $f(G_2)<m$。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$\langle u,v,L-2\rangle$,$\langle v,w,L-1\rangle$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 级、$L-1$ 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 级、$L-1$ 级和 $L-2$ 级的所有顶点 x,令($\lambda(x)$ of G_2)=E_2。

假定被指定与$\langle u,v,L-2\rangle$绑定计算的边为$\langle w,D,L\rangle$,并且$\langle v,w,L-1\rangle\in E$(否则,若$\langle v,w,L-1\rangle\notin E$,则必然有 $\lambda(v)\subseteq\lambda(w)$,故不可能有 $R(a,b,k)$包含$\langle u,v,L-2\rangle$)。令 (ESS of G_2)=$I_v^{v_1,v_2}$(([$\lambda(w)\cap\lambda(v)$ of G]$_S^v-\{\langle a_1,b_1,k\rangle|\langle a_1,b_1,k\rangle\in E,\langle a_1,b_1,k\rangle\neq\langle a,b,k\rangle\}-\{\langle a_1,b_1,L-2\rangle|\langle a_1,b_1,L-2\rangle\in E,\langle a_1,b_1,L-2\rangle\neq\langle u,v,L-2\rangle\}$),$group_1$,$group_2$)$\cup E_2[L-1:L]$。

注意:对于第 k 级边和第 $L-2$ 级边,(ESS of G_2)中都只包含有一条边。

(特别构建) 对所有边$\langle a,b,k\rangle$,$2\leqslant k\leqslant L-4$,增加 b 到 D 的路径 $b-\cdots-D$。将该路径上所有节点的顶点边集设置成等于 E_2。但(ESS of G_2)仅仅增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边。如果有简单路径 P 经过 b 并且 $P[1:k]\in$(ESS of G_2),增加简单路径 Q 使得 $Q[1:k]=P[1:k]$,(ESS of G_2) 增加 $Q[k+1:k+1]$以及 $Q[L-1:L]$。

说明:(ESS of G_2)仅仅增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边,以及(ESS of G_2)仅仅增加 $Q[k+1:k+1]$和 $Q[L-1:L]$使得 G_2 仍然具备语句 1.3 定义的性质,因为(ESS of G_2)仅仅增加 $b-\cdots-D$ 的 $L-1$ 级和 L 级的边,又 $k\leqslant L-4$,$Q[k+2:L-2]\notin$(ESS of G_2),所以 G_2 没有增加包含于(ESS of G_2)的简单路径。

增加 b 到 D 的路径,是使得原来经过 $\lambda(b)$到达 $\lambda(D)$的边,在 G_2 中可以经过新的路径 $b-\cdots-D$ 到达$\lambda(D)$。这个增加可以有助于我们确认两件事情:1)对所有以 b 为终点的边,如果指定路径 $b-\cdots-D$ 上的 L 级的边与之绑定计算,对修改的算子 4 确认以 b 为终点的边的去留时,不产生任何限制意义,因为该路径上所有顶点边集都等于 E_2;2)如果 G 中 $R(e)$有可达路径经过 b,G_2 中一定有可达路径经过 b,因为路径 $b-\cdots-D$ 上所有顶点边集都等于 E_2,e 到达 b 之后不再有任何因素阻止 e。

增加简单路径 Q 使得 $Q[1:k]=P[1:k]$,同时(ESS of G_1)增加 $Q[k+1:k+1]$以及 $Q[L-1:L]$,使得路径 $b-\cdots-D$ 不在任何一条 Max-k 路径的辖域中。

与 G_1 的构造略有不同,$k=L-3$ 时,我们没有增加路径。这是因为 G_2 中对于 L 级、$L-1$ 级和 $L-2$ 级的所有顶点 x,有($\lambda(x)$ of G_2)=E_2,不需要增加了。

由此得到的图如图 3(b)所示。G_2 的"形状"与 G_1 相同。

将算法 2 应用于 G_2 后,可以得到以下 3 个结果。

(1)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 1.1—步骤 1.3 所列性质。

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

$Comp$([{$e|e=\langle c,d,kk\rangle\in E,kk<k$,[$R(c,d,kk)\cap Comp$([{$\langle u,v,L-2\rangle$}$\cup${$e|e=\langle x,y,ll\rangle\in E,ll<L-2$,[$R(x,y,ll)\cap\lambda(v)\cap\lambda(w)$]$_y^v$ contains a path that contains $\langle u,v,L-2\rangle$}]$_S^v$,v,$R(E)$)]$_d^v$ contains a path that contains $\langle u,v,L-2\rangle$ and $\langle a,b,k\rangle$}]$_S^a$,a,$R(E)$)$\neq\emptyset$,导致($Comp(ESS1,D,R(E))$ of G_2) $\neq\emptyset$(注意:由于变量命名和使用的原因与算子 4 定义中不同,这里 $\langle u,v,L-2\rangle$ 和 $\langle a,b,k\rangle$交换了位置)。

下面说明($Comp(ESS1,D,R(E))$ of G_2) $\neq\emptyset$ 的原因。

1)对每条边$\langle r,s,t\rangle$,若($R(r,s,t)$ of G)的初始值包含$\langle o,p,kk\rangle$,则必然有 e_1 和 e_2,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_1)=\langle r,s,t\rangle$,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_2)=\langle o,p,kk\rangle$,($R(e_1)$ of G_2)的初始值必包含 e_2。

2)在特别构建中增加简单路径 Q 之前,G_2 中没有简单路径 P,$P[1:L-3]\in$(ESS of G_2),否则本结论(2)已经成立。特别构建中增加的简单路径 Q 构成 G_2 中的全部 Max-k 路径,处在 Max-k 路径辖域中的属于 L 级的边是 $Q[L:L]$。除这些 $Q[L:L]$之外的 L 级的边都可以被指定与其他边绑定计算。

3)对每条边$\langle r,s,t\rangle$,若($R(r,s,t)$ of G)的计算值包含$\langle o,p,kk\rangle$,$kk<L-3$,因为 G_2 中对每条边$\langle o,p,kk\rangle$,有 p 到 D 的路径 $p-\cdots-D$,该路径上所有节点的顶点边集等于 E_2,如果指定路径 $p-\cdots-D$ 的属于 L 级的边与$\langle o,p,kk\rangle$绑定计算,那么 G_2 中必然有 e_1 和 e_2,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_1)=\langle r,s,t\rangle$,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_2)=\langle o,p,kk\rangle$,($R(e_1)$ of G_2)的计算值包含 e_2。

对每条边$\langle r,s,t\rangle$,若($R(r,s,t)$ of G)的计算值包含$\langle o,p,kk\rangle$,$kk\geqslant L-3$,必然有 e_1 和 e_2,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_1)=\langle r,s,t\rangle$,$I_{w'}^{w_4}I_{w'}^{w_3}I_w^{w_2}I_w^{w_1}I_v^{v_2}I_v^{v_1}(e_2)=\langle o,p,kk\rangle$,($R(e_1)$ of G_2)的计算值包含 e_2。因为 G_2 中 $L-2$ 级、$L-1$ 级和 L 级顶点边集都等于 E_2,指定 e_2 的辖域(设 $e_2=\langle x,y,kk\rangle$,称集合[$E_2$]$_y^D$ 为 e_2 的辖域)内的任何 L 级的边与 e_2 绑定计算,必然有($R(e_1)$ of G_2)的计算值包含 e_2。

4)对第 m 级中的每个点 x,$m<L-2$,若有($Comp(\lambda(x),x,R(E))$ of G)包含边 e,必然有($Comp(\lambda(x),x,R(E))$ of G_2)包含边 e,因为对每条边$\langle o,p,t\rangle$,$t<L-3$,有特别构建增加的 p 到 D 的路径 $p-\cdots-D$,该路径上所有节点的顶点边集等于 E_2。对每条边$\langle o,p,t\rangle$,$t=L-3$,根据 G_2 的构造,同样有 p 到 D 的路径 $p-\cdots-D$,该路径上所有节点的顶点边集等于 E_2。

对第 m($m\geqslant L-2$)级的每个点 x,若有($Comp(\lambda(x),x,R(E))$ of G)包含边 e,x 不是撕裂点,则必然有($Comp(\lambda(x),$

$x, R(E))$ of G_2)包含边 e,因为 G_2 中 $L-2$ 级和 $L-1$ 级的顶点边集都等于 E_2。

对第 $m(m \geqslant L-2)$ 级的每个点 x,若有($Comp(\lambda(x), x, R(E))$ of G)包含边 e,x 是撕裂点,由于 G_2 中 $L-2$ 级和 $L-1$ 级顶点边集都等于 E_2,假设 u 是 $L-3$ 级的顶点,x 是 G 中路径 $u-\cdots-D$ 上的顶点,($Comp(\lambda(x), x, R(E))$ of G)包含边 e,则必有($Comp(\lambda(u), u, R(E))$ of G)包含边 e,必有 G_2 中路径 $u-\cdots-D$ 上的顶点 x',有($Comp(\lambda(x'), x', R(E))$ of G_2)包含($Comp(\lambda(u), u, R(E))$ of G_2)。说明若有($Comp(\lambda(u), u, R(E))$ of G_2)包含边 e,则必然有($Comp(\lambda(x'), x', R(E))$ of G_2)包含边 e。

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

证明如下:由 G 中 $Comp([\{e \mid e=\langle c,d,kk\rangle \in E, kk<k, [R(c,d,kk) \cap Comp([\{\langle u,v,L-2\rangle\} \cup \{e \mid e=\langle x,y,ll\rangle \in E, ll<L-2, [R(x,y,ll) \cap \lambda(v) \cap \lambda(w)]_y^v$ contains a path that contains $\langle u,v,L-2\rangle\}]_S^y, v, R(E))]_d^v$ contains a path that contains $\langle u,v,L-2\rangle$ and $\langle a,b,k\rangle\}]_S^v, a, R(E)) \neq \emptyset$,可知 A 集非空,B 集非空。假定 G_2 中所有经过 $u-v-w-D$ 的边 e(即 $R(e)$包含 $u-v-w-D$),连同 $u-v-w-D$ 一起构成集合 X,A 集非空等同于($Comp(X, D, R(E))$ of G_2)非空,B 集非空等同于($Comp(ESS1, D, R(E))$ of G_2)非空。

(3)$f(G_2) < f(G)$。

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

基于结果(1)、结果(2)以及结果(3),我们可以推断 G_2 中存在简单路径 SP 且(ESS of G_2)包含 SP。由于 $\langle u,v,L-2\rangle$ 和 $\langle a,b,k\rangle$ 是(ESS of G_2)中第 $L-2$ 级和第 k 级的唯一边,因此 $\langle u,v,L-2\rangle$ 和 $\langle a,b,k\rangle$ 必在 SP 上。

又根据(ESS of G_2)的取值,$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}$ ($SP[1:L-2]$) $\in \lambda(v)$,$I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}$ ($SP[1:L-1]$) $\in \lambda(w)$,故 $I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}$ (SP)就是我们关心的简单路径。

只要(ESS of G_2)中的$[\lambda(w) \cap \lambda(v)$ of $G]_S^v$ 取计算值,就必然有($R(a,b,k)$ of G)包含 $I_{w'}^{w_4} I_{w'}^{w_3} I_w^{w_2} I_w^{w_1} I_v^{v_2} I_v^{v_1}$ (SP)从 b 到 D 的部分。

结论(2)因此得证。

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

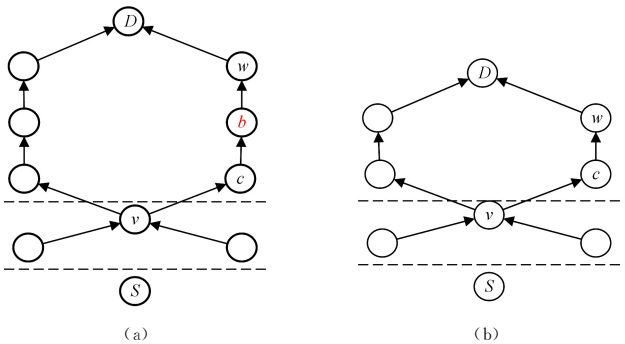

图 4 引理 3 的典型图例

Fig. 4 Typical graph of lemma 3

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

1)G_1 具备算法 2 中步骤 1 定义的性质,并且 G_1 是一个 $L-1$ 级图;

2)如果 G 作为输入时 $Comp(ESS1, D, R(E)) \neq \emptyset$,则 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,则 P 本质上是 G 的简单路径且包含于 ESS。

经典的归纳证明是通过去点或者去边构造“更小”的图。证明引理 2 时,我们没有找到构造“更小”图的方法,所以我们采用了撕裂,即通过复制更多点而非去点完成了引理 2 的证明。引理 3 的证明可以回归经典证明方法。

去掉 G 中 $L-2$ 级的所有点,构造一个 $L-1$ 级图 $G_1=\langle V_1, E_1, S, D, L-1, \lambda\rangle$,如图 4(b)所示。

(1)G_1 的顶点:$V_1 = (V$ of $G) - (V_{L-2}$ 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,那么 $\langle c,w,(L-1)-1\rangle$ 是 G_1 的边。

(3)G_1 的顶点边集,以及(ESS of G_1)。

对所有 $w \in V_1$,如果 w 在 k 级,$k=(L-1)-1$:令($\lambda(w)$ of G_1) = (($Comp(\lambda(w), w, R(E))$ of G) $- E[L-2:L-1]) \cup \{e \mid e=\langle c,w,(L-1)-1\rangle$,存在 b 使得 $\langle c,b,L-2\rangle \in (\lambda(w)$ of $G)$ 并且 $\langle b,w,L-1\rangle \in (\lambda(w)$ of $G)\}$。

对所有 $x \in V_1$,如果 x 在 k 级,$k=L-1$:令($\lambda(x)$ of G_1) $= E_1$。

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

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

(4)修改边集。

对 G 中所有 $\langle g_1, g_2, k\rangle$,如果($\lambda(x)$ 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($\lambda(x)$ of G_1)中的 $\langle g_1, g_2, k\rangle$。

(特别构建) 根据一些特殊情况,需要对上面 4 步得到的图 G_1 进行有条件的修改,使得构造的图满足算法 2 中步骤 1 定义的性质。下面的证明中再讲解如何做有条件修改。

至此,便完成了 G_1 的构造以及顶点边集和 ESS 的定义。G_1 $L-1$ 级的图,G_1 中各个顶点的入度与 G 中各个顶点的入度相同,因此有 $f(G_1) < f(G) = m$。

将构造的 G_1 和(ESS of G_1)作为算法 2 的输入进行计算,可以证明以下结论(1)-(7)。

(1)$f(G_1) < f(G)$。

(2)G_1 具备算法 2 中步骤 1 所列性质。

根据 G_1 的构造,步骤 1.1 和 1.2 的性质容易验证。

若要 G_1 具备算法 2 中步骤 1.3 所定义的性质,需要进一步对其进行改造(即前文提及的特别构建)。注意,改造后的图满足步骤 1 的所有性质。

压缩之后得到的 G_1 要满足步骤 1.3 的要求,(ESS of G_1)中必须扩展增加 $E_1[(L-1)-1:(L-1)-1]$中的所有$\langle c,w,(L-1)-1\rangle$。如果 G_1 因此增加了包含于(ESS of G_1)的简单路径,可以从(ESS of G_1)中去掉$\langle w,D,L-1\rangle$。

下面证明,以 G 为算法输入时产生的 *binding*,可以作为以 G_1 为算法输入时产生的 *binding*。

对于任意边$\langle a,b,k\rangle$,$k\leqslant L-3$,以 G 为算法输入时指定$\langle f,D,L\rangle$与$\langle a,b,k\rangle$一起绑定计算,以 G_1 为算法输入时仍然可以指定$\langle f,D,L-1\rangle$与$\langle a,b,k\rangle$一起绑定计算;对于任意边$\langle a,b,k\rangle$,$k=L-2$,以 G 为算法输入时指定$\langle f,D,L\rangle$与$\langle a,b,k\rangle$一起绑定计算,以 G_1 为算法输入时仍然可以指定$\langle f,D,L-1\rangle$与$\langle a,w,k\rangle$一起绑定计算。这里,$\langle a,b,k\rangle$和$\langle b,w,k+1\rangle$是 G 中的边。如果$\langle b,f,L-1\rangle$是 G 中的边,则 $w=f$。

于是我们得到以下结论:

(3)图 G 作为算法 2 的输入时的 *binding*,可以作为压缩得到的图 G_1 作为算法 2 的输入时的 *binding*。

(4)将证明算法作用于 G_1,($Comp(ESS1,D,R(E))$ 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(如$\langle c,b,L-2\rangle$和$\langle b,w,L-1\rangle$变成$\langle c,w,(L-1)-1\rangle$),所以,考虑到结论(2)和结论(3),以 G_1 为输入得到的所有计算结果,包括所有 $\lambda(x)$以及所有 $R(e)$,都与 G 为输入得到的相应的计算结果"本质上"相同。因此,由($Comp(ESS1,D,R(E))$ of $G)\neq\emptyset$ 可知,($Comp(ESS1,D,R(E))$ of $G_1)\neq\emptyset$。

(5)G_1 有简单路径 SP 且(ESS 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($Comp(ESS1,D,R(E))$ of $G_1)\neq\emptyset$,所以 G_1 有简单路径 SP 且(ESS of G_1)包含 SP。

(6)如果 G_1 有简单路径 $P=S-a_1-\cdots-a_{L-3}-a_{L-1}-D$ 并且(ESS 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 of G)包含 P'。

证明:

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

($\lambda(a_{L-1})$ of G_1)包含 $P[1:(L-1)-1]$,即($\lambda(a_{L-1})$ of G)包含 $P'[1:L-1]$,以及($\lambda(a_(L-2))$ of G)包含 $P'[1:L-2]$,因为对于 G_1 的$(L-1)-1$ 级的顶点 w,($\lambda(w)$ of G_1)=((($Comp(\lambda(w),w,R(E))$ of $G)-E[L-2:L-1]$)$\cup\{e|e=\langle c,w,(L-1)-1\rangle$,存在 b 使得$\langle c,b,L-2\rangle$,$\langle b,w,L-1\rangle\in(\lambda(w)$ of $G)\}$。

(7)G 有简单路径 SP 且 ESS 包含 SP。

根据结论(6),结论(5)中提到的简单路径 SP 实际上就是 G 中一条包含于(ESS of G)的简单路径。证毕。

根据上面的讨论,由引理 1、引理 2、引理 3,得到 $\alpha\beta$ 定理。

$\alpha\beta$ 定理:设 $G=\langle V,E,S,D,L,\lambda\rangle$ 和 ESS 是算法 2 的输入,如果 $Comp(ESS1,D,R(E))\neq\emptyset$,则 G 有简单路径 SP 且 ESS 包含 SP。

2.5 ZH 算法的充分性证明

定理 3 设 $G=\langle V,E,S,D,L,\lambda\rangle$是一个多级图,以 G 作为 ZH 算法的输入,如果 $Comp(\lambda(D),D,R(E))\neq\emptyset$,则 G 中存在简单路径。

证明:在 G 中增加路径 $D-w_1-D_{new}$,令 $\lambda(w_1)=\lambda(D_{new})=E\cup\{D-w_1-D_{new}\}$,我们得到一个新的 $L+2$ 级的图,称为 G'。然后,令 $ESS=E\cup\{D-w_1-D_{new}\}$。显然,$G'$ 具备算法 2 中步骤 1.1—步骤 1.3 所定义的性质。

以 G'和 ESS 作为算法 2 的输入。

可以假定 G'中没有简单路径,否则定理 3 已经成立,所以$\langle w_1,D_{new},L+2\rangle$不会在任何 Max-$k$ 路径辖域中,可以成为所有边的唯一指定。

G'的特殊结构,使得除 $\lambda(D_{new})$外的任意 $\lambda(x)$必然是 $\lambda(w_1)$的子集。于是,对任意边$\langle a,b,k\rangle$ ($k\leqslant(L+2)-2$),V-$Change(R(u,v,l),binding)$的计算结果与 $Change(R(u,v,l))$的计算结果完全相同。因为 ZH 算法的执行结果有 $Comp(\lambda(D),D,R(E))\neq\emptyset$,所以以 G'和 ESS 为算法 2 的输入,必有 $Comp(ESS1,D,R(E))\neq\emptyset$。

依据 $\alpha\beta$ 定理,G'有简单路径 SP 且 ESS 包含 SP,这意味着 $SP[1:L]$是 G 中的简单路径。

3 MSP∈NPC 的证明以及本文结论

MSP∈NPC 的证明请参看文献[16-20]。目前已有研究将哈密顿图判定问题多项式归结到 MSP 问题[15-17]。事实上,为了回答人们对 MSP 问题 NP 完全性质的怀疑,我们已经发表了十多个 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[18]。

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 3,如果归结正确,我们得到定理 4。

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。

结束语 本文给出了 MSP 问题的多项式时间算法,这个结果可能对于证明 NP=P 有重要意义。但是本文算法的复杂性仍然太高,不具备实用性。下一步将着力于降低算法的时间复杂性和空间复杂性。

致谢 很长时间以来,刘万伟、魏登萍、樊硕、李鹏坤、王琪、吴添君、周泰杨、彭立宏、姜子恒、盖方宇、邵成成、马兰先后对本文研究做出贡献。刘万伟教授还给出了 MSP 问题到 SAT 问题的归结。

提出 MSP 问题是 John Edward Hopcroft 教授 2006 年的建议。陈建二教授高度评价 MSP 问题的表达能力,给出了独立集问题到 MSP 问题的归结,并在之后长期支持我们,以及在学术和技术上指导我们关于 MSP 问题多项式时间算法的研究。

$f(G)=L+\sum_{v\in(V-\{S,D\}-V_2)}(d(v)-1)$的定义得益于张家琳博士的启示。之前我们是为每个多级图定义了一个向量并且定义了一个线性序,这容易带来漏洞。2014 年在北京举行的一个为期一周的专题讨论会上,她建议我们将所有分量加起来。

感谢卢锡城院士、李晓梅教授、窦文华教授、杨学军院士、

王怀民院士、彭宇行教授的经费、人员、条件、激励、方向、学术、技术的支持和帮助。感谢许可教授、李未院士、刘任任教授的方向性指导和帮助。姚期智院士托孙家广院士通过李思昆教授转达指导性意见。数学家侯振挺教授长期关注支持本文研究并驱动前进。

参 考 文 献

[1] THE CLAY MATHEMATICS INSTITUTE. P vs. NP Problem [EB/OL]. https://www.claymath.org/millennium-problems/p-vs-np-problem.

[2] SCIENCE (AAAS). 125 Questions of Science [EB/OL]. http://www.sciencemag.org/site/feature/misc/webfeat/125th/.

[3] FORTNOW L. The status of the P versus NP problem[J]. Communications of the ACM, 2009, 52(9): 78-86.

[4] COOK S. The importance of the P versus NP question[J]. Journal of the ACM (JACM), 2003, 50(1): 27-29.

[5] THE CLAY MATHEMATICS INSTITUTE. Millennium Problems [EB/OL]. https://www.claymath.org/millennium-problems/.

[6] KAYAL N. The complexity of the annihilating polynomial[C]// 2009 24th Annual IEEE Conference on Computational Complexity. IEEE, 2009: 184-193.

[7] DEOLALIKAR V. P≠NP [EB/OL]. https://www.win.tue.nl/~gwoegi/P-versus-NP/Deolalikar.pdf.

[8] VARDI M Y. on P, nP, and computational complexity[J]. Communications of the ACM, 2010, 53(11): 5.

[9] AGRAWAL M, KAYAL N, SAXENA N. PRIMES is in P[J]. Annals of mathematics, 2004, 160(2): 781-793.

[10] VALIANT L G. Quantum circuits that can be simulated classically in polynomial time[J]. SIAM Journal on Computing, 2002, 31(4): 1229-1254.

[11] KNUTH D. All questions answered[J]. Notices of the AMS, 2002, 49(3): 318-324.

[12] GASARCH W I. The p=?np poll[J]. Sigact News, 2002, 33(2): 34-47.

[13] GAREY M R, JOHNSON D S. Computers and intractability: A guide to the theory of NP-completeness [M]. San Francisco: freeman, 1979.

[14] JIANG X W. Determining the H Property of A Graph by Changing It into Multistage Graph[J]. Computing Technology and Automation, 2004 23(2): 52-54.

[15] JIANG X W, Wang Q, JIANG Z H. The fourth version of the Proof for ZH Algorithm to Solve MSP Problem[J]. Computing Technology and Automation, 2010, 29(3): 35-48.

[16] JIANG X W, PENG L H, WANG Q. MSP Problem: Its NP-Completeness and its Algorithm[C]// 2010 Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications. IEEE, 2010: 1-5.

[17] JIANG X W. A Polynomial Time Algorithm for the Hamilton Circuit Problem[J]. arXiv preprint arXiv: 1305.5976, 2013.

[18] JIANG X W, LIU W W, WU T J, et al. Reductions from MSP to SAT and from SUBSET SUM to MSP[J]. Journal of Computational Information Systems, 2014, 10(3): 1287-1295.

[19] FAN S, JIANG X W, PENG L H. Polynomial-time heuristic algorithms for several NP-complete optimization problems[J]. Journal of Computational Information Systems, 2014, 10(22): 9707-9721.

[20] JIANG X W, WU T J, LI P K, et al. A New Algorithm for the MSP Problem [J]. Computing Technology and Automation, 2016, 35(1): 60-70.

JIANG Xin-wen, born in 1962, professor, is a member of China Computer Federation and a member of China Society for Industrial and Applied Mathematics (CSIAM), serving for their blockchain committee bothly. His main research interests include algorithm and computational complexity, information security and blockchain technology. He is a teacher who received several science and technology progress awards, including one national-level prize and six ministerial-level prizes. He was earned Army Yucai Award twice due to his distinguished contribution in talents cultivation.