左侧带权凸二分图动态权值匹配

祖 佺[1] 张苗苗[1] 刘 静[2]

[1](同济大学软件学院 上海 201804)

[2](华东师范大学上海市高可信计算重点实验室 上海 200062)

摘 要 动态匹配问题是指在图结构变更的情况下求解某特定匹配,包括添加和删除图中顶点和边的更新操作以及计算匹配信息的查询操作.凸二分图是一类特殊二分图,在其顶点二划分(X,Y)中,Y 顶点集为一个全序集,每个$x\in X$ 的邻点集在 Y 中形成一段连续区间.已有的凸二分图动态基数匹配算法不能求解权值匹配,因而该文研究左侧顶点带权凸二分图中动态最大权值匹配问题.文中提出一种问题求解的框架:在更新操作中维护参与匹配的顶点集合,继而在查询操作中计算相应的匹配信息.文中基于交错路定义了可替换集,并证明可通过计算可替换集来维护参与匹配的顶点集;提出紧致子图的概念,证明可替换集的求解等价于紧致子图的求解,从而将传统的通过寻找交替路求解匹配的方法改进为通过寻找子图结构来求解匹配.文中利用凸二分图的凸性质将紧致子图的计算转化为查找该子图中最大或最小 y 顶点操作,进而结合隐性表征技术在增广平衡二叉查找树数据结构中快速求解,继而设计动态匹配算法在 $O(\log^2|V|)$平摊时间下维护更新操作,在最坏线性时间下维护查询操作.较之于已知最好的解决不带权凸二分图动态基数匹配问题的方法,该文提出的方法能在与之相同的时间复杂度下解决难度更高的左侧带权问题.

关键词 凸二分图;动态匹配;交错路;紧致子图;隐性表征;平衡二叉查找树

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2016.02388

Dynamic Weight Matchings in Left Weighted Convex Bipartite Graphs

ZU Quan[1] ZHANG Miao-Miao[1] LIU Jing[2]

[1](*School of Software Engineering, Tongji University, Shanghai* 201804)

[2](*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai* 200062)

Abstract The dynamic matching problem is the problem of maintaining certain matchings in dynamically changing graphs under a set of update operations that includes inserting and deleting vertices and edges, and a set of query operations that includes obtaining the matching status of a vertex or finding the mate of a matched vertex. Convex bipartite graph is a bipartite graph with the vertex bipartition (X,Y) in which the neighborhood of each $x\in X$ forms an interval of Y where Y is linearly ordered. The existing algorithm for dynamic cardinality matching in convex bipartite graphs cannot solve dynamic maximum weighted matching. And this paper researches dynamic maximum weight matching in left weighted convex bipartite graphs. To solve this problem, we propose a framework that is maintaining the set of vertices participating in the matching in the update operations, and based on the set, computing the matching information in the query operations. We define the concept of replaceable set, i.e., the set of matched vertices reachable by alternating paths from an unmatched vertex, and prove that the computation for replaceable sets is sufficient to maintain the matched vertex set. We then define the concept of tight

收稿日期:2015-04-01;在线出版日期:2015-11-23.本课题得到国家自然科学基金(61472279,61332008,91318301)资助.**祖 佺**,男,1982 年生,博士研究生,主要研究方向为组合优化、模型检验.E-mail:7quanzu@tongji.edu.cn.**张苗苗**(通信作者),女,1971 年生,博士,教授,博士生导师,主要研究领域为时间和混合系统的分析与验证.E-mail:miaomiao@tongji.edu.cn.**刘 静**,女,1965 年生,博士,教授,博士生导师,主要研究领域为模型驱动软件开发和软件工程.

subgraph, and prove that computing a replaceable set is equivalent to finding a certain tight subgraph. Thus, the traditional way of computing matchings via finding alternating paths is reduced to our approach of computing matchings via finding a subgraph structure. Utilizing the convexity property of a convex bipartite graph, we further put forward a method to find a tight subgraph by searching for the maximum or minimum y vertices in the subgraph. The method can be efficiently implemented in an augmented balanced binary search tree data structure with the implicit representation of the subgraph. Then we design algorithms that maintain the update operations in $O(\log^2|V|)$ amortized time and the query operations in worst-case linear time, which achieves the same time bound as the best known solution to the dynamic cardinality matching problem in the unweighted situation.

Keywords　convex bipartite graph; dynamic matching; alternating path; tight subgraph; implicit representation; balanced binary search tree

1　引　言

二分图中的匹配问题[1]是一类重要的组合优化和图论问题,现实生活中很多实际问题可以基于二分图匹配问题来求解,如调度和指派问题[2]、模式识别和 Web 服务中的文本语义分析[3-4]、生物计算[5]和网络安全问题[6]等. 二分图中的动态匹配问题是在图结构动态变更的情况下维护某一特定匹配. 当图结构规模较大时,对于图的不断变更,若用已有的静态算法来维护特定匹配,则效率较低,故需相应的动态算法以解决这类问题.

二分图可形式化描述为图 $G=(X,Y;E)$,其中顶点集 $V=X\cup Y$,X 和 Y 是 V 的一个二划分,边集 $E\subseteq X\times Y$. 若 Y 上有一全序序列,每个 $x\in X$ 的邻居在 Y 的全序序列中形成一段连续区间,则称图 G 为凸二分图(convex bipartite graph);对于任一 $x\in X$,若 (x,y_1) 和 (x,y_2) 属于边集 E,其中 y_1 和 y_2 属于 Y 且 $y_1<y_2$,则对每个 $y\in Y$ 且 $y_1<y<y_2$,$(x,y)\in E$. 若 X 中每个 x 附属一权值,则称 G 为左侧带权凸二分图,其中每条边的权值等于其 x 端点的权值. 左侧带权凸二分图上的最大权值匹配问题对应着一类经典的调度问题[2]:给定任务集合和时间片集合,其中每个任务有释放时间、期限和权值,且执行时间为单位时间,其调度目标是寻找最大权值的可行调度任务子集. 凸二分图动态匹配问题在基因染色体比较[7]、频谱拍卖[8]和大规模传感器信道分配[9]等问题中都有广泛的应用.

凸二分图上静态匹配问题的研究相对完备. 文献[10]最早提出凸二分图上的最大基数匹配问题,并给出一种贪心算法:按递增序列遍历 Y 集合,对每个顶点 y,在与其相连且当前未匹配的 x 顶点集中选取结束期限最小的顶点与之相匹配;基于快速优先级队列[11],算法可在 $O(|Y|+|X|\log\log|X|)$ 时间复杂度下求解. 随后,文献[12]将该问题转换成为最小脱机问题[13](off-line minimum problem),进而基于并查集[14](Union-Find)数据结构在接近线性时间复杂度下求解. 最终,文献[15]通过将图划分为独立分支(distinct component)进而在 $O(|X|)$ 时间复杂度下求解该问题.

权值匹配问题一般比基数匹配问题难度更高. 根据边的权值的定义不同,凸二分图上的最大权值匹配问题有 3 种变型. 在右侧带权凸二分图中(每个 y 顶点关联一权值),文献[16]给出一贪心算法,在 $O(|E|+|Y|\log|X|)$ 时间复杂度下找到一个最大权值匹配. 在左侧带权凸二分图中(每个 x 顶点关联一权值),文献[12]基于拟阵理论给出了一种 $O(|X|^2+|X||Y|)$ 时间复杂度的解决方法. 文献[2]基于一种层次数据结构在 $O(|X|+k\log^2 k)$ 的平摊时间复杂度下求解,其中 $k\leq\min\{|X|,|Y|\}$. 近期,文献[8]指出在普通带权凸二分图中(每个顶点关联一权值),可在 $O(|V|\log^2|V|)$ 时间复杂度下求解最大权值匹配,并针对左侧带权问题,给出了一种 $O(|V|\log|V|)$ 时间复杂度的解决方法.

凸二分图动态匹配问题通过更新操作和查询操作维护某类特定的匹配,其中更新操作包括添加和删除顶点和边,查询操作包括查询某顶点是否已匹配,以及查询某已匹配顶点的相匹配顶点. 在此类问题中,一个顶点或一条边的更新可能会导致已有匹配中所有的边的改变,以至于维护动态匹配的难度

很高.文献[17]继而指出:“即使在平摊时间,也不可能在次线性(sub-linear)时间下维护最大匹配的显性表征”.通过隐性表征的技术,文献[17]设计了一种算法解决了凸二分图的最大基数动态匹配问题,该算法基于二叉计算树数据结构[18],在 $O(\log^2|V|)$ 平摊时间复杂度下维护更新操作,在最坏接近线性时间复杂度下维护查询操作.

当顶点带权时,不仅需维护匹配的最大边数,而且还需考虑匹配权值的可增广性,这使得动态维护权值匹配有更高的难度.本文研究左侧带权凸二分图中最大权值动态匹配问题,发现解决该问题的关键在于维护最大权值匹配所对应的已匹配顶点集合,而维护该集合的关键在于计算从某个未匹配的顶点通过交错路可达的已匹配顶点集合,或称为可替换集.进一步地,我们提出紧致子图的概念(该子图蕴含有完美匹配),并证明了可替换集的求解等价于紧致子图的求解,同时针对凸二分图结构找到了一种快速求解该子图的方法.在此基础之上改进二叉计算树数据结构,结合隐性表征设计算法,使其在 $O(\log^2|V|)$ 平摊时间下维护更新操作,在最坏线性时间复杂度下维护查询操作.进而与文献[17]所考察的不带权问题相比,在与其相同的时间复杂度下解决了难度更高的左侧带权问题.

本文第 2 节介绍基本定义和研究问题;第 3 节界定所提出的可替换集和紧致子图的概念,证明相关的性质和定理,并给出凸二分图中快速求解紧致子图的方法;后面三节设计并分析动态匹配算法和数据结构;最后第 7 节进行总结.

2 基本定义与问题描述

设 $G=(X,Y;E)$ 是一个二分图,$V=X\cup Y$,$X(G)=X$,$Y(G)=Y$;$N_G(v)$ 表示顶点 v 的邻集;$N_G(V')$ 表示顶点集 $V'\subseteq V$ 中所有顶点的邻集的并;$G[V']$ 表示顶点集 $V'\subseteq V$ 的导出子图.匹配是图中不相邻的边组成的集合.设 M 是 G 的一个匹配,若顶点 v 是 M 中某条边的端点,则称 v 是 M 饱和的,或称 M 饱和 v;用 $X(M)$ 和 $Y(M)$ 分别表示 G 中所有 M 饱和的 x 顶点和 y 顶点集合.M 交错路指其边在 E-M 和 M 中交替出现的路;若一条 M 交错路的起点和终点都是 M 非饱和的,则称其为 M 增广路[19].由 G 的某一匹配 M 中所有的边的 x 端点组成的集合称为 G 的一个 X 匹配集,用 MX 表示.给定二分图 G 的 X 集合上的一个全序关系,下面定义 G 的最优 X 匹配集:

定义 1. 二分图 G 的最优 X 匹配集是指 G 的 X 匹配集 $MX=\{x_1,\cdots,x_k\}$,$x_1>\cdots>x_k$,使得对任一 G 的 X 匹配集 $MX'=\{x'_1,\cdots,x'_\ell\}$,$x'_1>\cdots>x'_\ell$,满足以下性质:$k\geq\ell$,而且 $\forall i\in[1,\ell]$,$x_i\geq x'_i$.

设 $G=(X,Y;E)$ 是一个左侧带权凸二分图,G 中每个 y 顶点表示为一个唯一的整数.定义 Y 集合上一种全序关系为 $y_1<y_2$.对 Y 集合中的任意子集 Y',Y'.min 和 Y'.max 分别表示 Y' 中最小的和最大的 y 顶点.G 中每个 x 顶点用一个四元组 $(x.id,x.s,x.e,x.w)$ 表示,其中 $x.id$ 为一个唯一的整数,$x.s$ 和 $x.e$ 分别表示 x 的邻集中最小的和最大的 y 顶点,$x.w$ 是一个表示 x 的权值的整数.定义 X 集合上的 3 种全序关系:定义 start-id 序列为 $(x.s,x.id)$ 的字典序;定义 end-id 序列为 $(x.e,x.s,x.id)$ 的字典序;定义 weight-id 序列为 $(x.w,-x.e,-x.s,-x.id)$ 的字典序.在左侧带权凸二分图 G 中,我们根据 weight-id 的全序关系定义最优 X 匹配集.对 G 的任意 $x\in X$,$y\in Y$,若 $x.s\leq y\leq x.e$,则 $(x,y)\in E$,即给定 X 顶点集和 Y 顶点集,边集可由顶点集隐性表征,故左侧带权凸二分图可用二元组 (X,Y) 表示.对一个 x 顶点 $x'\notin X$,用 $G+x'$ 表示左侧带权凸二分图 $G'=(X',Y')$,其中 $X'=X+x'$,$Y'=Y$.同理,$G+y'$ 表示在 G 中加入顶点 $y'\notin Y$.

在左侧带权凸二分图动态匹配问题中,若某 x 顶点的权值为负,则该 x 必然不属于任一最大权值匹配.因带负权值的 x 顶点容易识别并去除,故本文假设 x 权值为非负.根据文献[20]中的引理 2 可推出,在权值非负的左侧带权凸二分图中,任一饱和最优 X 匹配集的匹配 M 都是一个最大权值匹配,故本文动态匹配算法在更新操作中维护最优 X 匹配集.算法维护的更新操作包括:

(1) *insert*$(\hat{x})$:添加顶点 $\hat{x}\notin X$,对于 Y 中每个满足 $\hat{x}.s<y<\hat{x}.e$ 的顶点 y,y 与 $\hat{x}$ 相连.

(2) *insert*$(\hat{y})$:添加顶点 $\hat{y}\notin Y$,对于 X 中任一满足 $x.s<\hat{y}<x.e$ 的顶点 x,x 与 $\hat{y}$ 相连.

(3) *delete*$(\hat{x})$:删除顶点 $\hat{x}\in X$;图中所有与 $\hat{x}$ 相连的边隐性删除.

(4) *delete*$(\hat{y})$:删除顶点 $\hat{y}\in Y$,$\hat{y}$ 不等于任一 $x\in X$ 的 $x.s$ 或 $x.e$;图中所有与 $\hat{y}$ 相连的边隐性删除.

删除任意边的操作将破坏凸性质,因而边的更新操作需加以限制.通过相应顶点的更新操作来完成边的加入或删除,即添加边 (x,y) 通过先删除 x 顶点,继而重新加入已修改 $x.s$ 或 $x.e$ 的 x 顶点来实

现；删除边的操作与之类似. 算法维护的查询操作包括：

(1) $status(v)$：查询顶点 v 的匹配状态.

(2) $pair(v)$：查询已匹配顶点 v 的相匹配顶点.

3　紧致子图与可替换集

本节界定二分图上的可替换集和紧致子图的概念，证明其性质以及它们之间的关系，并针对凸二分图结构给出一种快速求解紧致子图的方法.

3.1　可替换集

在二分图 G 中，当 x 顶点 $\hat{x}$ 不能增广 MX 时，即 $MX+\hat{x}$ 不是 G 的 X 匹配集时，我们定义 $\hat{x}$ 的 MX 可替换集如下.

定义 2.　在二分图 $G=(X,Y;E)$ 中，对于 X 匹配集 MX 和顶点 $\hat{x}\in X-MX$，设 M 为饱和 MX 的任一匹配，若 $\hat{x}$ 不能增广 M，则称从 $\hat{x}$ 出发的 M 交错路可达的 MX 中的顶点集 $R(\hat{x},MX)$ 为 $\hat{x}$ 的 MX 可替换集.

例如，图 1 中 $MX=\{x_1,x_2,x_4\}$，从 $\hat{x}$ 出发的 M 交错路可达 x_1 和 x_2，$\hat{x}$ 的 MX 可替换集 $R(\hat{x},MX)=\{x_1,x_2\}$.

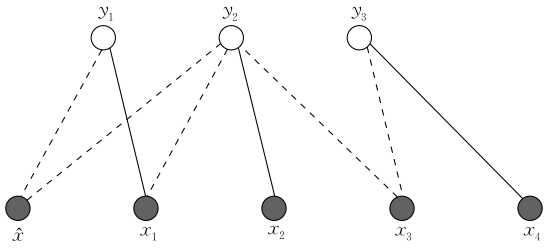

图 1　可替换集(实线边代表匹配)

由以下命题可知 $\hat{x}$ 的 MX 可替换集仅与 X 匹配集 MX 相关，与具体饱和 MX 的匹配无关.

命题 1.　对于二分图 $G=(X,Y;E)$ 中 X 匹配集 MX 和顶点 $\hat{x}\in X-MX$，若 $\hat{x}$ 不能增广 MX，即 $MX+\hat{x}$ 不是 G 中 X 匹配集，则对于任一饱和 MX 的匹配 M，$\hat{x}$ 的 MX 可替换集相同.

证明.　设 M_1,M_2 为饱和 MX 的任意两个匹配，即 $X(M_1)=X(M_2)=MX$. 由文献[1]中定理 1 可知对称差 $G'=M_1\oplus M_2$ 包含由 M_1 和 M_2 中边交替出现的圈或迹. 若可证明 $\hat{x}$ 所有的邻点都属于 G' 中的圈，则命题可证. 任取 $\hat{x}$ 的邻点 y_1，y_1 是 M_1 饱和且 M_2 饱和的，否则 $(\hat{x},y_1)$ 可增广 M_1 或 M_2，与前提相矛盾. 令 $(x_1,y_1)\in M_1$，$(x_2,y_1)\in M_2$. 若 $x_1=x_2$，得证. 若 $x_1\neq x_2$，从 y_1 出发经过 x_1 的路径必然可沿 M_1 和 M_2 中的边交替延伸，且：(1) 中间经过的 y 顶点必然是 M_1 饱和的，否则 $\hat{x}$ 增广 M_1；(2) 中间经过的 x 顶点必然是 M_2 饱和的，因为 M_2 饱和 MX. 易推知该路径必然最终延伸回到顶点 y_1. 故而，所有从 $\hat{x}$ 出发的 M_1 交错路可达的 x 顶点都可被从 $\hat{x}$ 出发的 M_2 交错路可达.　　证毕.

下面两个命题刻画了不同 X 匹配集之间的转换关系以及最优 X 匹配集的性质，为证明定理 1 做准备.

命题 2.　若 MX,MX' 在二分图 G 中是 X 匹配集，其中 $|MX|=s$，$|MX'|=r$，且 $s>r$，则存在 X 匹配集 MX''，满足 $|MX''|=s$，$MX'\subset MX''$，且 $MX''-MX'\subseteq MX-MX'$.

证明.　设 M_1,M_2 分别为 G' 中饱和 MX,MX' 的任意两个匹配，则在 $M_1\oplus M_2$ 中必然存在至少 $s-r$ 条顶点不相交的 M_2 增广路(文献[1]中定理 1). 故而 M_2 可增广至 M_3，其中 $|M_3|=s$，$X(M_2)\subset X(M_3)$ 且 $X(M_3)-X(M_2)\subseteq X(M_1)-X(M_2)$.　　证毕.

命题 3.　设 MX 在二分图 G 中是最优 X 匹配集，则对 $\forall x\in X-MX$，MX 在 $G'=G-x$ 中也是最优 X 匹配集.

证明.　(反证法)假设 MX 不是 $G-x$ 的最优 X 匹配集，令 $\overline{MX}\neq MX$ 为最优. 设 $\overline{MX}=\{\bar{x}_1,\cdots,\bar{x}_m\}$，$MX=\{x_1,\cdots,x_m\}$，则 $\exists i\leq m,\bar{x}_i>x_i$. 可知 $\overline{MX}$ 在 G 中是 X 匹配集，且 $\bar{x}_i>x_i$，与 MX 在 G 中是最优 X 匹配集矛盾.　　证毕.

本文的动态算法通过维护最优 X 匹配集以维护最大权值匹配，以下定理给出了一种通过可替换集维护最优 X 匹配集的方法.

定理 1.　设二分图 $G=(X,Y;E)$ 的最优 X 匹配集为 MX，对于顶点 $x'\notin X$，若 $MX'=MX+x'$ 为 $G'=G+x'$ 的 X 匹配集，则 MX' 为 G' 的最优 X 匹配集；否则，令 $R(x',MX)$ 为 G' 中 x' 的 MX 可替换集，x'' 为 $R(x',MX)\cup x'$ 中在 weight-id 序列下的最小元素，则 $MX+x'-x''$ 为 G' 的最优 X 匹配集.

证明.　(1) 先证明：若 $MX'=MX+x'$ 为 $G'=G+x'$ 的 X 匹配集，则 MX' 为 G' 的最优 X 匹配集. (反证法)假设 MX' 不是 G' 的最优 X 匹配集，令 $\overline{MX}\neq MX'$ 为最优. 若 $|\overline{MX}|>|MX'|$，则 $\overline{MX}-x'$ 在 G 中是 X 匹配集，且 $|\overline{MX}-x'|>|MX|$，与已知 MX 在 G 中是最优 X 匹配集相矛盾，故而 $|\overline{MX}|=|MX'|$. 若 $x'\notin\overline{MX}$，则 $\overline{MX}$ 在 G 中是 X 匹配集且 $|\overline{MX}|>|MX|$，矛盾，故而 $x'\in\overline{MX}$. 设 $\overline{MX}=$

$\{\bar{x}_1,\cdots,\bar{x}_m\}$,其中 $\bar{x}_1>\cdots>\bar{x}_m$,设 $MX'=\{x'_1,\cdots,x'_m\}$,其中 $x'_1>\cdots>x'_m$,由 $\overline{MX}$ 是 G' 的最优 X 匹配集,可知:$\forall i\in[1,m],\bar{x}_i\geq x'_i$.(集合示意关系参看图 2)令 $\bar{x}_k=x'$,$x'_j=x'$,可推知 $k\geq j$,否则 $\bar{x}_k=x'_j<x'_k$,与 $\overline{MX}$ 最优相矛盾.可推知 $\forall i\in[j,k-1],\bar{x}_i\geq x'_{i+1}$,即对于 $\overline{MX}[j,k)$ 和 $MX'(j,k]$ 中的相同序列元素,前者不小于后者.故对于序列 $\overline{MX}-x'=\overline{MX}[1,j)+\overline{MX}[j,k)+\overline{MX}(k,m)=\{\bar{x}'_1,\cdots,\bar{x}'_{m-1}\}$ 以及序列 $MX'-x'=MX'[1,j)+MX'(j,k]+MX'(k,m)=\{x''_1,\cdots,x''_{m-1}\}=MX$,满足 $\forall i\in[1,m-1],\bar{x}'_i\geq x''_i$,这与已知 MX 是 G 中最优 X 匹配集矛盾,故而 MX' 为 G' 的最优 X 匹配集.

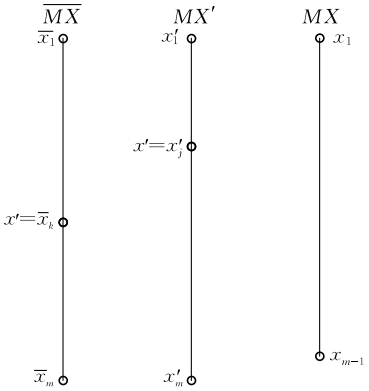

图 2 定理 1 证明中所涉及的集合间关系

证明.(2)若 $MX+x'$ 不是 $G'=G+x'$ 的 X 匹配集,则 $MX'=MX+x'-x''$ 为 G' 的最优 X 匹配集,其中 x'' 为 $R(x',MX)\cup x'$ 中按 weight-id 序列排序下的最小元素.(反证法)假设 MX' 不是 G' 的最优 X 匹配集,令 $\overline{MX}\neq MX'$ 为最优.假设 $x'\notin\overline{MX}$,由命题 3,$\overline{MX}$ 在 G 中是最优 X 匹配集,若 $\overline{MX}=MX$,则 $x'=x''$,$\overline{MX}=MX'$,矛盾;若 $\overline{MX}\neq MX$,与已知 MX 最优相矛盾,故可得 $x'\in\overline{MX}$.若 $|\overline{MX}|>|MX|$,根据命题 2,存在 X 匹配集 $\widetilde{MX}$ 使得 $|\widetilde{MX}|=|\overline{MX}|$ 且 $\widetilde{MX}-MX\subseteq\overline{MX}-MX$,设 $\bar{x}\in\widetilde{MX}-MX$.若 $\bar{x}=x'$,即 $MX+x'$ 是 G' 的 X 匹配集,与已知相矛盾.若 $\bar{x}\neq x'$,即 $MX+\bar{x}$ 是 G 的 X 匹配集,与 MX 在 G 中最优矛盾,故而 $|\overline{MX}|=|MX'|$.设 $\overline{MX}=\{\bar{x}_1,\cdots,\bar{x}_m\}$,其中 $\bar{x}_1\geq\cdots\geq\bar{x}_m$;设 $MX=\{x_1,\cdots,x_m\}$,其中 $x_1\geq\cdots\geq x_m$;设 $MX'=\{x'_1,\cdots,x'_m\}$,其中 $x'_1\geq\cdots\geq x'_m$.

可证明 $x''\notin\overline{MX}$.(反证法)假设 $x''\in\overline{MX}$,令 x' 和 x'' 在 $\overline{MX}$ 中序列分别为 l 和 k,即 $\bar{x}'_l=x'$,$\bar{x}_k=x''$.可推知 $\overline{MX}[1,k-1]=MX'[1,k-1]$;否则,$\exists i<k,\bar{x}_i>x'_i$,使得 $\overline{MX}[1,k-1]-x'$ 在 G 中是 X 匹配集,且与 MX 是 G 中最优相矛盾.对于 G' 中 X 匹配集 $\widetilde{MX}=\overline{MX}[1,k]$ 和 MX',根据命题 2,存在 X 匹配集 $\widetilde{MX'}$ 使得 $|\widetilde{MX'}|=|MX'|$ 且 $\widetilde{MX'}-\widetilde{MX}\subseteq MX'-\widetilde{MX}$,因此 $\widetilde{MX'}-MX'=x''$,$MX'-\widetilde{MX'}=\bar{x}$,其中 $\bar{x}\in MX'[k,m]$,即 $\bar{x}\in R(x',MX)\cup x'$ 且 $\bar{x}<x''$,与已知相矛盾.故而 $x''\notin\overline{MX}$.

若 $x'=x''$,由上述证明可知 $x'\in\overline{MX}$ 且 $x''\notin\overline{MX}$,矛盾.若 $x'>x''$,即 $MX'=MX+x'-x''$.设在 MX 中,$x_k=x''$.由 $\overline{MX}$ 最优可知,$\forall i\leq k,\bar{x}_i\geq x'_i$.若 $\exists i\leq k,\bar{x}_i>x'_i$,则 $\overline{MX}[1,i]-x'$ 在 G 中是 X 匹配集,且与 MX 是最优相矛盾,故而 $\forall i\leq k,\bar{x}_i>x'_i$.由 $\overline{MX}\neq MX'$,设 $r>k$ 为最小的整数使得 $\bar{x}_r>x'_r=x_r$.(如图 3 所示)考察 $MX_1=\overline{MX}[1,r]$ 和 $MX_2=MX[1,r-1]$,$\{x',\bar{x}_r\}\subset MX_1-MX_2$;根据命题 2,$MX_2$ 可与 x' 或 $\bar{x}_r$ 增广一个单位.若 $\bar{x}_r$ 可增广 MX_2,则 $MX_2+\bar{x}_r$ 在 G 中是 X 匹配集,且 $\bar{x}_r>x_r$,与 MX 在 G 中是最优相矛盾.若 x' 可增广 MX_2,考察 MX' 和 MX_2+x',根据命题 2,MX_2+x' 可增广至 MX_3,$|MX_3|=|MX|$,且 $MX_3-MX_2-x'\subseteq MX'-MX_2-x'=MX[r,m]$,可推知 $MX-MX_3=\tilde{x}<x''$,$MX_3-MX=x'$,即 $\tilde{x}\in R(x',MX)$ 且 $\tilde{x}<x''$,与已知相矛盾.故 MX' 是 G' 的最优 X 匹配集.

证毕.

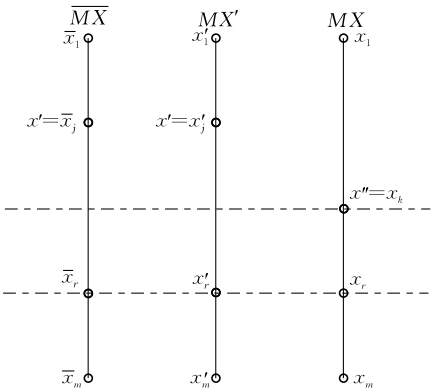

图 3 定理 1 证明中 MX 不可增广情况

3.2 紧致子图与可替换集

二分图中一类特殊的子图与可替换集有密切的联系,下面先给出紧致子图的定义.

定义 3. 二分图 G 中满足下列性质的子图 H 称为紧致子图:H 是一个蕴含完美匹配的点集非空的导出子图,且 $\forall x'\in X(H),N_G(x')\subseteq Y(H)$.

更为确切地说,二分图 $G=(X,Y;E)$ 中,若顶点子集 $X'\subseteq X$,其中 $|X'|=|N_G(X')|$,且对于任意 $X''\subseteq X'$,都满足 $|X''|\leq|N_G(X'')|$,则称导出子图

$G[X' \cup N_G(X')]$是 G 的紧致子图. 对于 G 的紧致子图 G'和 G'',若 $X(G'') \subset X(G')$,则称 G'覆盖 G''.

根据 Hall 定理[21],易推知满足上述关系的子图蕴含完美匹配. 我们有如下命题.

命题 4.　对于二分图 G 的匹配 M 和紧致子图 G',若 $X(G') \subseteq X(M)$,则 $Y(G') \subseteq Y(M)$.

证明.　假设 $Y(G') \not\subseteq Y(M)$,即 $Y_1 = Y(G') - Y(M) \neq \varnothing$,任选 $y_1 \in Y_1$. 令 Y_2 为 $X(G')$中的 x 顶点在 M 中相匹配的 y 顶点集合,易知 $|Y_2| = |X(G')|$ 且 $Y_2 \subseteq Y(M)$. 因此 $Y_2 + y_1 \subseteq N_G(X(G'))$,可推出 $|N_G(X(G'))| > |X(G')|$,与定义矛盾.　　证毕.

定义 4.　在二分图 $G=(X,Y;E)$中,对于 X 匹配集 MX,顶点 $\hat{x} \in X - MX$, $N_G(\hat{x}) \neq \varnothing$,若 G'是 G 的紧致子图,其中 $X(G') \subseteq MX$ 且 $N_G(\hat{x}) \subseteq Y(G')$,则称 G'是图 G 中 $\hat{x}$ 的 MX 紧致子图. 若不存在另一 $\hat{x}$ 的 MX 紧致子图 G''使得 G'覆盖 G'',则称 G'为图 G 中 $\hat{x}$ 的 MX 最小紧致子图.

例如图 4 所示,图 G 中 $X = \{x_1, x_2, x_3, x_4\}$, $Y = \{y_1, y_2, y_3\}$, $MX = \{x_1, x_2, x_3\}$,令 $X' = MX$,则 $G' = G[X' \cup Y]$是图 G 的紧致子图,G'也是图 G 中 $\hat{x}$ 的 MX 紧致子图;令 $X'' = \{x_1, x_2\}$, $Y'' = \{y_1, y_2\}$,则 $G'' = G[X'' \cup Y'']$是 $\hat{x}$ 的 MX 最小紧致子图.

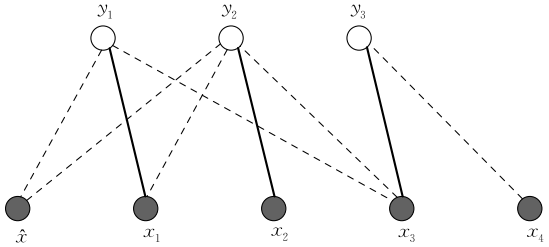

图 4　紧致子图(实线边代表匹配)

由命题 4 的证明可知,若 G'是 $\hat{x}$ 的 MX 紧致子图,对于饱和 MX 的任一匹配 M, $N_G(\hat{x}) \subseteq Y(G') \subseteq Y(M)$. 下面的命题刻画了最小紧致子图的唯一性.

命题 5.　设 MX 是二分图 G 中一个 X 匹配集,对于顶点 $\hat{x} \in X(G) - MX$,且 $N_G(\hat{x}) \neq \varnothing$,若图 G 中存在 $\hat{x}$ 的 MX 紧致子图 G',则存在唯一的 $\hat{x}$ 的 MX 最小紧致子图.

证明.　假设存在不止一个 $\hat{x}$ 的 MX 最小紧致子图,设其中任意两个为 G_1 和 G_2, $G_1 \neq G_2$. 令 $Y_1 = Y(G_1)$, $Y_2 = Y(G_2)$,易知 $Y_1 \neq Y_2$,否则 $X(G_1) = X(G_2)$, $G_1 = G_2$. 因此, $Y_1 \cap Y_2 \neq \varnothing$, $Y_1 \not\subseteq Y_2$, $Y_2 \not\subseteq Y_1$. 令 $Y_3 = Y_1 \cap Y_2$,可得 $Y_3 \subset Y_1$, $Y_3 \subset Y_2$. 根据 x 的 MX 紧致子图定义, $N_G(x) \subseteq Y_3$. 设 M 为任一饱和 MX 的匹配. 令 $X_3 \subseteq MX$ 包含 M 中与 Y_3 中 y 顶点相匹配的 x 顶点集合,可推知 $X_3 \subseteq X(G_1) \cap X(G_2)$,否则集合 $X(G_1) \oplus X(G_2)$中存在未匹配的 x 顶点. 令 $Y_3' = N_G(X_3)$, $\bar{Y} = Y_3' - Y_3$. 根据紧致子图定义, $\bar{Y} \subseteq Y_1 \cap Y_2$;根据 Y_3 定义, $\bar{Y} \not\subseteq Y_1 \cap Y_2$,故而 $\bar{Y} = \varnothing$,可得 $N_G(X_3) = Y_3$. 因为 $|X_3| = |Y_3|$,可得 $G_3 = G[X_3 \cup Y]$是 $\hat{x}$ 的 MX 紧致子图,且被 G_1 和 G_2 覆盖,与假设相矛盾.　　证毕.

下面的关键引理刻画了匹配的可增广性与紧致子图的存在性之间的关系.

引理 1.　对于二分图 $G=(X,Y;E)$中 X 匹配集 MX 和顶点 $\hat{x} \in X - MX$,且 $N_G(\hat{x}) \neq \varnothing$, $\hat{x}$ 不能增广 MX 当且仅当存在 $\hat{x}$ 的 MX 紧致子图.

证明.　(必要性)设 $\hat{x}$ 不能增广 MX, $R(\hat{x}, MX)$为 $\hat{x}$ 的 MX 可替换集,令 M 为饱和 MX 的任一匹配. 令 $X' = R(\hat{x}, MX)$, $Y' = N_G(X')$. 可推知 $N_G(\hat{x}) \subseteq Y'$成立,否则对于 $\forall y \in N_G(\hat{x}) - Y'$, $(\hat{x}, y)$是一条 M 增广路. 若 $|X'| < |Y'|$,则必有 $y' \in Y'$未匹配,且存在从 $\hat{x}$ 到 y'的 M 增广路,矛盾,因此 $|X'| \geq |Y'|$. 可知 $\forall MX' \subseteq MX$, $|MX'| \leq |N_G(MX')|$,而 $X' \subseteq MX$,故 $|X'| \leq |Y'|$. 因此, $|X'| = |Y'|$, $G[X' \cup Y']$是 $\hat{x}$ 的 MX 紧致子图.

(充分性)设 $G' = G[X' \cup Y']$是 $\hat{x}$ 的 MX 紧致子图,令 M 为饱和 MX 的任一匹配. 由反证法,假定 $\hat{x}$ 可增广 MX,即存在从 $\hat{x}$ 到某未匹配 y 顶点 y'的增广路 $\bar{P}$. 因为 Y'中所有顶点均匹配,因此 $y' \notin Y'$. 故 $\bar{P}$ 从 $\hat{x}$ 出发,进入 $N_G(\hat{x}) \subseteq Y'$,返回 X',如此前进直至最终从某个 $x' \in X'$达到某个 $y'' \notin Y'$,矛盾,因为 $\forall x' \in X'$, $N_G(x') \subseteq Y'$.　　证毕.

由引理 1 可得出下面关于可替换集和紧致子图之间的重要关系.

推论 1.　对于二分图 $G=(X,Y;E)$中 X 匹配集 MX 和顶点 $\hat{x} \in X - MX$,若 $\hat{x}$ 不能增广 MX, $\hat{x}$ 的 MX 可替换集 $R(\hat{x}, MX) = X(\hat{G})$,其中 $\hat{G}$为 G 中 $\hat{x}$ 的 MX 最小紧致子图.

证明.　根据引理 1 可知存在 $\hat{x}$ 的 MX 紧致子图,设 $\hat{G} = G[\hat{X} \cup \hat{Y}]$为 $\hat{x}$ 的 MX 最小紧致子图, M 为饱和 MX 的任一匹配. 令 $X' = R(\hat{x}, MX)$. $X' \subseteq \hat{X}$成立,因为任一从 $\hat{x}$ 出发的交错路不能延伸出 $\hat{G}$. 下面证明 $\hat{X} \subseteq X'$. 初始,令 $Y'' = N_G(\hat{x})$, $X'' = N_{G[M]}(Y'')$,其中 $G[M]$为 M 的边导出子图. 因为 $\hat{G}$是 $\hat{x}$ 的 MX 紧致子图, $Y'' \subseteq \hat{Y}$,可得 $X'' \subseteq \hat{X}$,因此 $|X''| \leq |N_G(X'')|$且 $N_G(X'') \subseteq \hat{Y}$. 可推知 $X'' \subseteq \hat{X}$始终满足,故可递归赋值: $Y'' = N_G(X'')$, $X'' = N_{G[M]}(Y'')$,直到 $|X''| =$

$|N_G(X'')|$. 此时,若 $X''=\hat{X}$,引理得证. 若 $X''\subset\hat{X}$,$G[X''\cup Y'']$是一个 $\hat{x}$ 的 MX 紧致子图且为 $\hat{G}$ 所覆盖,与 $\hat{G}$ 是 $\hat{x}$ 的 MX 最小紧致子图相矛盾. 证毕.

3.3 紧致子图在凸二分图中的求解

在凸二分图中,由于图的凸结构性质,图中的紧致子图含有一些特殊性质.

命题 6. 设 MX 是凸二分图 $G=(X,Y)$ 中一个 X 匹配集,对于顶点 $\hat{x}\in X-X(M)$,$N_G(\hat{x})\neq\varnothing$,若图 G 中存在 $\hat{x}$ 的 MX 紧致子图 G',则 $Y(G')$ 是 Y 上的一个连续区间.

证明. 设 M 为任一饱和 MX 的匹配. 令 $Y_0=N_G(\hat{x})$,$X_i=N_{G[M]}(Y_i)$,即 X_i 是 M 上与 Y_i 中 y 顶点相匹配的 x 顶点集合,$Y_{i+1}=N_G(X_i)$. 下面用归纳法:显然 Y_0 在 $Y(G)$ 上连续,假设 Y_i 在 $Y(G)$ 上连续,证明 Y_{i+1} 在 $Y(G)$ 上连续. 初始地令 $Y'=Y_i$. 遍历 X_i,对于每个顶点 $x'\in X_i$,扩充 $Y'=Y'\cup N_G(x')$. 根据凸性质以及已知 x' 在 M 中相匹配顶点 $y'\in Y'$,即 $N_G(x')\cap Y'\neq\varnothing$,可知在每次扩充后 Y' 在 $Y(G)$ 上连续. 因此 Y_{i+1} 在 $Y(G)$ 上连续. 当 $Y_i=Y(G')$ 时,定理得证. 证毕.

进而可利用凸性质在凸二分图中快速判断和求解紧致子图.

定义 5. 给定凸二分图 $G=(X[1,n],Y[1,m])$ 和某一 X 匹配集 MX,定义:

$$\alpha(i,MX)=|\{x\in MX:x.e\leq Y[i]\}|,$$

$$d_\alpha(i,MX)=i-\alpha(i,MX),i\in[1,m],$$

$$\beta(j,MX)=|\{x\in MX:x.s\geq Y[j]\}|,$$

$$d_\beta(j,MX)=m-j+1-\beta(j,MX),j\in[1,m];$$

若 $d_\alpha(k,MX)=0$,我们称 k 为 α-MX 紧致点,若 $d_\beta(k,MX)=0$,称 k 为 β-MX 紧致点.

以下引理刻画紧致点与紧致子图间的关系.

引理 2. 对于凸二分图 $G=(X,Y)$ 中 X 匹配集 MX,若 k 为 α-MX 紧致点,则 $G[V']$ 是紧致子图,其中 $V'=MX[1,k]\cup Y[1,k]$,MX 按 end-id 序列排序;若 k 为 β-MX 紧致点,则 $G[V'']$ 是紧致子图,其中 $V''=MX[|MX|-m+k,|MX|]\cup Y[k,m]$,$MX$ 按 start-id 序列排序.

证明. 由凸二分图的凸性质以及紧致点、紧致子图定义即可得证. 证毕.

下面界定一些特殊的紧致点,它们与求解最小紧致子图相关.

定义 6. 给定凸二分图 $G=(X,Y)$ 中 X 匹配集 MX 和顶点 $y\in Y$,若存在 α-MX 紧致点 k 且 $Y[k]\geq y$,则称其中最小的 k 为 $\alpha_{\text{Post}}(y,MX)$ 紧致点;若存在 α-MX 紧致点 k 且 $Y[k]<y$,则称其中最大的 k 为 $\alpha_{\text{Pre}}(y,MX)$ 紧致点. 对称地,若存在 β-MX 紧致点 k 且 $Y[k]\leq y$,则称其中最大的 k 为 $\beta_{\text{Pre}}(y,MX)$ 紧致点;若存在 β-MX 紧致点 k 且 $Y[k]>y$,则称其中最小的 k 为 $\beta_{\text{Post}}(y,MX)$ 紧致点.

引理 3 和引理 4 给出了在凸二分图中,对于几种特殊的 x 顶点,根据紧致点判断 X 匹配集的可增广性和求解可替换集的方法.

引理 3. 设 MX 为凸二分图 $G=(X,Y)$ 中 X 匹配集,对于顶点 $\hat{x}\notin X$,当 $\hat{x}.s\leq Y.\min$,或存在 $x'\in MX$,使得从 $\hat{x}$ 出发的 MX 交错路可达 x' 且 $x'.s\leq Y.\min$ 时,在图 $G+\hat{x}$ 中,若不存在 α-MX 紧致点 i 且 $Y[i]\geq\hat{x}.e$,则 $\hat{x}$ 可增广 MX;若存在这类紧致点,令 k 为 $\alpha_{\text{Post}}(\hat{x}.e,MX)$ 紧致点,则 $\hat{x}$ 的 MX 可替换集 $R(\hat{x},MX)=MX[1,k]$,其中,MX 按 end-id 序列排序.

证明. 令 $j=\min\{i\leq|Y|:Y[i]\geq\hat{x}.e\}$. 先考察情况:$\hat{x}.s\leq Y.\min$. 在图 $G+\hat{x}$ 中,MX 仍为 X 匹配集,若存在 $\hat{x}$ 的 MX 紧致子图 G',根据紧致子图的定义则有 $Y[1,j]\subseteq Y(G')$. 又根据命题 6 可知,$Y(G')$ 是 Y 上的连续区间,令 $Y[k]=Y(G').\max$,则 $Y(G')=Y[1,k]$. 再根据紧致子图的定义,可知 $|X(G')|=k$,且 $\forall x\in X(G')$,$x.e\leq Y[k]$. 根据 α-MX 紧致点的定义可推知,k 是一个 α-MX 紧致点,且 $Y[k]\geq\hat{x}.e$. 因此,若不存在 α-MX 紧致点 i 且 $Y[i]\geq\hat{x}.e$,则不存在 $\hat{x}$ 的 MX 紧致子图,此时,根据引理 1,则 $\hat{x}$ 可增广 MX.

若存在 α-MX 紧致点 i 且 $Y[i]\geq\hat{x}.e$,令 k 为 x 的 α-MX 最小紧致点. 根据定义和引理 2 不难证明,$G[MX[1,k]\cup Y[1,k]]$ 是 x 的 MX 最小紧致子图,其中 MX 按 end-id 序列排序. 因此,根据推论 1,可得 $R(\hat{x},MX)=MX[1,k]$.

再考察 x' 可被 $\hat{x}$ 出发的 MX 交错路可达,且 $x'.s\leq Y.\min$,即在图 $G+\hat{x}$ 中 $x'\in R(\hat{x},MX)$. 若存在 $\hat{x}$ 的 MX 紧致子图,令 G' 是 $\hat{x}$ 的 MX 最小紧致子图,则根据推论 1,有 $x'\in R(\hat{x},MX)=X(G')$,$Y[1]\subseteq Y(G')$ 且 $Y[j]\subseteq Y(G')$. 根据命题 6,$Y(G')$ 是 Y 上的连续区间,则 $Y(G')=Y[1,k]$,其中 $Y[k]=Y(G').\max$. 后面的证明与 $\hat{x}.s\leq Y.\min$ 情况下相类似. 证毕.

对称地,对应于 β-MX 最大紧致点情形有如下引理,证明与引理 3 类似.

引理 4. 设 MX 为凸二分图 $G=(X,Y)$ 中 X

匹配集，对于顶点 $\hat{x}\notin X$，当 $\hat{x}.e\geq Y.\max$，或存在 $x'\in MX$，使得从 $\hat{x}$ 出发的 MX 交错路可达 x' 且 $x'.e\geq Y.\max$ 时，在图 $G+\hat{x}$ 中，若不存在 β-MX 紧致点 i 且 $Y[i]\leq\hat{x}.s$，则 $\hat{x}$ 可增广 MX；若存在这类紧致点，$\hat{x}$ 的 MX 可替换集 $R(\hat{x},MX)=MX[|MX|-|Y|+k,|MX|]$，其中 k 为 $\beta_{\mathrm{Pre}}(\hat{x}.s,MX)$ 紧致点，MX 按 start-id 序列排序.

证明.　略.

本节研究了在求解左侧带权凸二分图动态匹配问题中所涉及到的一些关键的概念和定理. 下文将研究具体的动态匹配算法，包括动态算法和数据结构的设计与分析.

4　改进的二叉计算树

二叉计算树数据结构最早由文献[18]提出以解决凸二分图上的并行匹配问题. 随后文献[17]基于该结构结合隐性表征，解决不带权凸二分图上的动态匹配问题. 本节改进二叉计算树隐性表征结构，使其可支持带权问题的求解.

4.1　改进的二叉计算树数据结构

二叉计算树是增广的平衡二叉查找树(如红黑树，参阅文献[22]). 给定左侧带权凸二分图 $G=(X,Y)$，将集合 $S=\{x.s:x\in X\}$ 按照 $x.s$ 增序排列，进而 S 可表示为 $\{s_1,\cdots,s_k\}$，令 $s_{k+1}=Y.\max+1$. 进而 Y 可划分为 k 个连续区间：$[s_i,s_{i+1}-1],1\leq i\leq k$. 在二叉计算树中，从左往右第 i 个叶子节点与子图 $G[X_i\cup Y_i]$ 相关联，其中 $X_i=\{x\in X:x.s=s_i\}$，$Y_i=[s_i,s_{i+1}-1]$. 令 P 为某中间节点，L 为其左孩子节点，R 为其右孩子节点. 节点 P 关联子图 $G[X_P\cup Y_P]$，其中 $X_P=X_L\cup X_R$，$Y_P=Y_L\cup Y_R$. 故而，根节点关联图 G.

每个节点 P 中增广数个辅助集合，如表 1 所示.

表 1　二叉计算树节点中的辅助集合

X_P	与 P 关联的 x 顶点集合.
Y_P	与 P 关联的 y 顶点集合. Y_P 划分为 Y_P^l 和 Y_P^r，$Y_P^l=\{y\in Y_P:y<Y_P.\mathrm{mid}\}$；若 P 为中间节点，$Y_P.\mathrm{mid}=Y_R.\min$，$Y_P^l=Y_L$，$Y_P^r=Y_R$；若 P 为叶子节点，$Y_P.\mathrm{mid}=Y_P.\min$，$Y_P^l=\varnothing$，$Y_P^r=Y_P$.
FX_P	游离集，$FX_P=\{x\in X_P:x.e>Y.\max\}$.
MX_P	$G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 中的 X 匹配集；MX_P 划分为 MX_P^l 和 MX_P^r.
UX_P	未匹配集，$UX_P=\{x\in X_P-FX_P-MX_P\}$.

对于节点 P 中的辅助集合，算法维护一些循环不变式，其中关键的两个为：

(1) ϕ_1：MX_P 是子图 $G_{\hat{P}}$ 中的最优 X 匹配集.

(2) ϕ_2：不存在 $x\in MX_P^r$ 使得 MX_P^l+x-x' 在子图 $G[MX_P\cup Y_P^l]$ 中是一个 X 匹配集，其中 $x'\in MX_P^l$，且在 end-id 排序下 $x<x'$.

由 ϕ_1 可推知当 P 为树中根节点时，MX_P 即为图 G 的最优 X 匹配集. ϕ_2 决定 MX_P 的划分，在动态更新操作中保证了算法效率(下文 5.1 节论及).

4.1.1　节点内部表征

二叉计算树中每个节点的辅助集合同样基于增广的平衡二叉树来表征，因此整体结构是二维树结构. 节点 P 中的数据结构表征包括：

(1) 对于辅助集合 MX_P^l，MX_P^r，MX_P，FX_P，UX_P，每个集合有一颗对应的树结构表征；

(2) 一颗 α 树表征 $G[MX_P^r\cup Y_P^r]$；

(3) 一颗 β 树表征 $G[MX_P^l\cup Y_P^l]$.

辅助集合表征：辅助集合树结构的表征方法以集合 MX_P^r 对应的 MX_P^r 树为例. 在该树中，每个叶子节点对应一个 $x\in MX_P^r$. 叶子节点按照 end-id 序列排序，即叶子节点所关联的关键字为 $(x.e,x.s,x.id)$. 对于树中每个节点 u，增广数个辅助域，包括 $u.size=|leaves(u)|$，其中 $leaves(u)$ 表示以 u 为根节点的子树中所包含的叶子节点集合；$u.\max e$，$u.\min s$，$u.\min w$ 分别表示集合 $leaves(u)$ 中叶子节点所关联的 x 集合中，end-id 排序下最大的 x，start-id 排序下最小的 x，weight-id 排序下最小的 x. 其他辅助集合的树结构可同理表征.

α 树表征：在 α 树中，每个叶子节点对应一个 $y\in Y_P^r$. 叶子节点的关键字为所关联的 y 顶点，按照 y 的增序排列. 从左往右第 i 个叶子命名为 y_i，叶子 y_i 对应 $Y_P^r[i]$. 每个节点 u 增广一个整型的辅助域 $u.d$，对于中间节点，$u.d$ 的初值为 0，对于叶子节点 y_i，$u.d$ 的初值为 i(下一小节将会讨论：每次基本操作至多影响 3 条从根节点到叶子节点的路径，使得其中的节点所对应的 d 的值发生改变). 定义 $d(u,v)=\sum_{n\in\boldsymbol{P}[u,v]}n.d$，其中 v 是 u 的祖先节点，$\boldsymbol{P}[u,v]$ 代表从 u 到 v 的不包含 v 的路径. 对于节点 u，定义 $d(u)=d(u,root)$，$root$ 代表 α 树中根节点. 这样，对于叶子 y_i，$d(y_i)=d_\alpha(i,MX_P^r)$($d_\alpha$ 定义请参考 3.3 节). 每个节点 u 增广辅助域 $d_{\min}$，其值为 $\min\{d(k,u):k\in leaves(v)\}$. 因此，对于节点 u，等式 $d(u)+u.d_{\min}=\min\{d(v):v\in leaves(u)\}$ 成立. 每个节点 u 维护一个整型域 σ，初值为 0，随后在每次使用前更新为 $d(u)$. 每个节点 u 增广辅助域 $\max y$，表示 $leaves(u)$ 所关联

的 y 集合中最大的 $y.\beta$ 树结构可同理表征.

4.2 基本操作

在上述表征的基础上,二叉计算树数据结构支持如下操作:

(1)在辅助集合树结构中查询某子集中的最大或最小元素以及更新集合的操作;

(2)在 α 树查询 $\alpha_{\text{Pre}}(y,MX_P^r)$ 或 $\alpha_{\text{Post}}(y,MX_P^r)$ 紧致点,以及更新集合的操作;

(3)在 β 树查询 $\beta_{\text{Pre}}(y,MX_P^l)$ 或 $\beta_{\text{Post}}(y,MX_P^l)$ 紧致点以及更新集合的操作.

该小节分别以一种情况为例,描述和分析这几类基本操作的算法及其时间复杂度.

辅助集合查询和更新:考虑在 MX_P^r 集合中寻找满足 $x.e\leq y'$ 的条件时 weight-id 排序下最小元素的操作,即寻找 $\hat{x}=\min w\{x\in MX_P^r: x.e\leq y'\}$. 在 MX_P^r 树中,根据域 max$e$,自顶向下遍历路径 $\boldsymbol{P}(root,l)$,其中 l 代表满足 $x.e\leq y'$ 中最大的叶子节点;对于路径中每个节点 u,若存在 u 的左兄弟节点 v 且在 weight-id 排序下 $v.\min w<\hat{x}$,置 $\hat{x}=v.\min w$,其中 $\hat{x}.w$ 的初值置为 $+\infty$. 令叶子 l 所关联的 x 顶点为 x'. 若在 weight-id 排序下 $\hat{x}>x'$,则返回 x'. 否则,自下而上遍历路径 $\boldsymbol{P}[l,root]$,直到遇到第 1 个满足下述条件的节点 u:u 的左兄弟节点 v 中 $v.\min w=\hat{x}$. 以 v 为根节点的子树包含对应 $\hat{x}$ 的叶子节点,根据域 min$w$ 可定位该叶子. 故而在 MX_P^r 集合中寻找 weight-id 排序下最小元素的操作,至多遍历从根节点到某叶子节点的路径 3 次. 在 MX_P^r 集合中加入一个新 x 顶点时,同样可根据域 max$e$ 定位加入的叶子节点,进而完成添加操作;若其中涉及到树的平衡操作(如旋转),每个节点中所对应的辅助域可在常数时间内进行调整.

α 树查询:基于 α 树可实现在子图 $G_{\hat{P}}^r$ 中查找 y 顶点 y' 的 $\alpha_{\text{Post}}(y',MX_P^r)$ 紧致点 k 的操作,即 $k=\alpha_{\text{Post}}(y',MX_P^r)$. 若 $y'>Y_P.\max$,则当前 α 树中不存在 x 的紧致点,故假设 $y'\leq Y_P.\max$. 在 α 树中,根据域 max$y$,自顶向下遍历路径 $\boldsymbol{P}(root,y')$,对于路径中每个节点 u,更新 $u.\sigma$. 定位到 y' 之后,自下而上遍历路径 $\boldsymbol{P}[y',root]$,直到遇到第 1 个满足下述条件的节点 u:u 的右兄弟节点 v 中 $v.d_{\min}+v.d+u.parent.\sigma=0$. 若不存在这样的节点,则当前 α 树中不存在 x 的紧致点;否则,以 v 为根节点的子树包含对应 x 的 α-M 最小紧致点 k 的叶子节点 y_k,即 $leaves(v)$ 中满足 $d(y_i)=0$ 的最小叶子节点 y_i. 以如下方法确定从 v 到 y_k 路径,对于路径中每个节点 u',更新 $u'.\sigma$,并从 u' 的两个孩子节点中选择满足 $u'.\sigma+v'd+v'.d_{\min}=0$ 的孩子节点 v'(优先选择左孩子节点). 当定位到 y_k 之后,即可通过相似的方法求解 y_k 在 Y_P^r 中的序列位置,即为 k 的值. 查找 $\hat{x}$ 的 α-M 最小紧致点 k 的操作遍历从根节点到叶子节点的路径至多 3 次. $\alpha_{\text{Pre}}(y',MX_P^r)$ 紧致点可类似求解. 同样,在 β 树中,$\beta_{\text{Pre}}(y',MX_P^l)$ 和 $\beta_{\text{Post}}(y',MX_P^l)$ 紧致点也可以类似求解.

α 树更新:MX_P^r 中加入 $\hat{x}$ 之后更新 α 树的操作与上述讨论类似. 通过 $\hat{x}.e$ 即可找到对应的叶子节点 y',将 $y'.d$ 减 1,然后自下而上遍历路径 $\boldsymbol{P}[y',root]$,对于路径中的每个节点 u,若 u 有右兄弟节点 v,则 $v.d$ 减 1,同时对于路径中的每个节点更新其 $d_{\min}$ 域的值. 例如图 5 所示,每个叶子节点下面的数值代表 d 域的初值,设 $\hat{x}.e$ 对应第 2 个叶子节点,树中的深色节点表示 d 域更新的节点. 更新 α 树的操作遍历从根节点到叶子节点的路径至多 2 次. 若 MX_P^r 中发生替换操作,即 $\hat{x}$ 替换了其中某个 x',则涉及从 $\hat{x}.e$ 和 $x'.e$ 两个叶子节点到根节点的更新值操作,方法类似. 在 β 树中的更新和替换操作同理可实现.

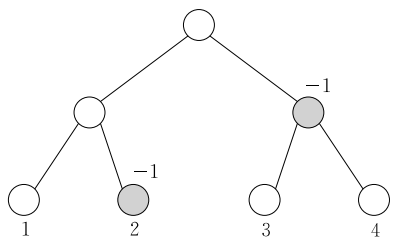

图 5 α 树更新

引理 5. 改进的二叉计算树中的基本操作的时间复杂度为 $O(\log|V|)$.

证明. 在辅助集合对应的树结构中,查询和更新操作至多遍历从根节点到某叶子节点的路径 3 次,在每个节点中操作为常数时间复杂度,故而可在对数时间复杂度下完成. 在 α 树以及 β 树中的查询紧致点和更新树的操作同样至多遍历从根节点到叶子节点的路径 3 次,对于树中每个节点中的操作为常数时间,故而可在对数时间复杂度下实现. 证毕.

本节我们基于二叉计算树,设计了支持左侧带权凸二分图动态匹配的数据结构,给出了其隐性表征结构,以及该数据结构支持的基本操作. 下一节将研究动态匹配算法.

5　动态匹配算法

动态匹配算法的设计基于改进的二叉计算树数据结构.本节详细描述并证明图中添加新的 x 顶点的动态更新操作,其他更新操作可类似求解;动态查询操作可采用我们已有方法[23]来实现.

5.1　*insert*(x)更新操作

动态更新操作保证二叉计算树的每个节点 P 中的辅助集合满足相应的循环不变式,即维护子图 $G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 的最优 X 匹配集.我们首先定义相对于节点 P 的可替换集.

定义 7.　设 MX_P 为二叉计算树中节点 P 所对应子图 $G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 的最优 X 匹配集,给定顶点 $x\notin X_P$ 且 $x.e\leq Y_P.\max$,若 x 不能增广 MX_P,即 MX_P+x 不是 $G'_{\hat{P}}=G[(X_P-FX_P+x)\cup Y_P]$ 的 X 匹配集,则称图 $G'_{\hat{P}}$ 中 x 的 MX_P 可替换集为 x 在节点 P 中的可替换集,记为 $RS(x,P)$.当不致混淆时,简记 $RS(x,P)$ 为 $RS(x)$.

易知 $\forall x'\in RS(x,P)$,$MX'=MX_P+x-x'$ 为子图 $G'_{\hat{P}}$ 的一个 X 匹配集.以下推论由给出了在 $G'_{\hat{P}}$ 中维护最优 X 匹配集的方法.

推论 2.　设 MX_P 为二叉计算树中节点 P 所对应子图 $G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 的最优 X 匹配集,给定顶点 $x\notin X_P$ 且 $x.e\leq Y_P.\max$,若 $MX'_P=MX_P+x$ 为 $G'_{\hat{P}}=G[(X_P-FX_P+x)\cup Y_P]$ 的 X 匹配集,则 MX'_P 为 $G'_{\hat{P}}$ 的最优 X 匹配集;否则,令 rx 为 $RS(x,P)\cup x$ 中按 weight-id 序列排序下的最小元素,$MX+x-rx$ 为 $G'_{\hat{P}}$ 的最优 X 匹配集.

证明.　由定理 1 直接推论可证.　　证毕.

更新操作的计算遍历二叉计算树中从某个叶子节点到根节点的路径,对于该路径中的每个节点,依次进行更新操作,且路径中每个父节点中的操作依赖于其子节点的操作计算结果,该结果用消息机制进行传递.在 *insert*(x)的操作中,定义消息 msg 为一个五元组 $(X^+,MX^+,MX^-,FX^+,UX^+)$,由节点 P 传递给其父节点,其中 X^+ 表示当前加入的 x 顶点,MX^+ 和 MX^- 分别表示加入和移除出 MX_P 集合的 x 顶点,FX^+ 和 UX^+ 类似定义.五元组中元素的值可能为空,用符号 Λ 表示.

5.1.1　叶子节点中加入 x

算法 1 是在一个叶子节点中 *insert*(x)的更新操作.令加入的 x 顶点为 $\hat{x}$,根据 $\hat{x}.s$ 来定位到相应的叶子节点 P(因为 $\forall x\in X_P$,$x.s=\hat{x}.s=Y_P.\min$).若 $\hat{x}.e>Y_P.\max$,$\hat{x}$ 属于 FX_P 集合.否则,在 α 树中查找 $\alpha_{\text{Post}}(\hat{x}.e,MX^r_P)$ 紧致点.若不存在此类紧致点则 MX^r_P 可被 $\hat{x}$ 增广(叶子节点中,$MX_P=MX^r_P$);若存在,则可根据其值来确定可替换集 $RS(\hat{x},P)$.进而在该叶子节点中维护循环不变式,即维护相对应于该节点子图 $G_{\hat{P}}+\hat{x}$ 的最优 X 匹配集.上述情况都涉及相关的辅助集合更新以及相应的 msg 消息的赋值;若 P 有父节点,则 msg 作为参数向上传递.

算法 1.　*InsertXinLeaf*($\hat{x}$).

输入:新加入的 x 顶点

输出:msg 消息

1.　$X_P\leftarrow X_P+\hat{x}$, $msg.X^+\leftarrow\hat{x}$
2.　IF ($\hat{x}.e>Y_P.\max$) THEN
3.　　$FX_P\leftarrow FX_P+\hat{x}$, $msg.FX^+\leftarrow\hat{x}$
4.　ELSE IF ($\exists\,\alpha_{\text{Post}}(\hat{x}.e,MX^r_P)$) THEN
5.　　$MX^r_P\leftarrow MX^r_P+\hat{x}$, $msg.MX^+\leftarrow\hat{x}$
6.　ELSE
7.　　$\alpha'\leftarrow MX^r_P[\alpha_{\text{Post}}(\hat{x}.e,MX^r_P)].e$
8.　　$RS(\hat{x},P)\leftarrow\{x\in MX^r_P:x.e\leq\alpha'\}$
9.　　$rx\leftarrow\min w\{x\in RS(\hat{x},P)\cup\hat{x}\}$
10.　　$MX^r_P\leftarrow MX^r_P+\hat{x}-rx$, $UX_P\leftarrow UX_P+rx$
11.　　$msg.MX^+\leftarrow\hat{x}$, $msg.MX^-\leftarrow msg.UX^+\leftarrow rx$
12.　RETURN msg

以下定理保证了算法 1 的正确性.

定理 2.　算法 *InsertXinLeaf*($\hat{x}$)在二叉计算树中的叶子节点中维护了循环不变式,时间复杂度为 $O(\log|V|)$ 平摊时间.

证明.　叶子节点 P 中,$Y^l_P=\varnothing$,故 $MX_P=MX^r_P$.已知加入 $\hat{x}$ 之前 MX_P 是子图 $G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 中的最优 X 匹配集,现证明加入 $\hat{x}$ 之后 MX_P 是子图 $G'_{\hat{P}}=G[(X_P-FX_P+x)\cup Y_P]$ 中的最优 X 匹配集.根据引理 3,通过 $\alpha_{\text{Post}}(\hat{x}.e,MX^r_P)$ 的计算结果,即可判断 $\hat{x}$ 是否可增广 MX_P.若不能增广,则可通过 $MX^r_P[\alpha_{\text{Post}}(\hat{x}.e,MX^r_P)]$ 来确定可替换集 $RS(\hat{x},P)$(MX^r_P 按照 end-id 序列排序).再根据推论 2,从中替换 weight-id 序列排序下的最小元素即可求解最优 X 匹配集.

根据引理 5,在算法 1 中,每一步的操作都可基于辅助集合对应的树结构在对数时间复杂度下实现,故而在叶子节点中加入 x 的操作可在对数时间复杂度实现.若加入的 $\hat{x}$ 的 $\hat{x}.s\notin\{x.s:X-\hat{x}\}$,则二叉计算树的结构会发生变化,产生新的叶子节点,而

调整平衡操作的开销使得算法效率成为平摊时间复杂度. 证毕.

5.1.2 中间结点中加入 x

在二叉计算树中,添加 x 顶点的动态更新算法是以自下而上的方式进行计算,即沿着某个叶子节点到根节点的路径进行计算.考虑从右孩子 R 到父节点 P 的路径,算法 2 描述了在这种情况下在 P 中加入 x 节点 $\hat{x}$ 的更新操作.

算法 2. $InsertXinR2P(msg)$.

输入:msg 消息

输出:msg 消息

1. $\hat{x}\leftarrow msg.X^+$
2. IF ($\hat{x}.e>Y_P.\max$) THEN
3. $FX_P\leftarrow FX_P+\hat{x}$
4. ELSE IF($msg.UX^+=\hat{x}\,\|\,msg.UX^+\in MX_P^r$) THEN
5. P 节点中集合更新以及 msg 设置均与 R 节点相同
6. ELSE IF($\exists\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$) THEN
7. $MX_P^r\leftarrow MX_P^r+\hat{x}$
8. ELSE
9. $\alpha'\leftarrow MX_P^r[\alpha_{\text{Post}}(\hat{x}.e,\ MX_P^r)].e$
10. $x_1\leftarrow\min s\{x\in MX_P^r:x.e\leq\alpha'\}$
11. $\beta'\leftarrow MX_P^l[\beta_{\text{Pre}}(x_1.s,\ MX_P^l)].s$
12. $RS(\hat{x},P)\leftarrow\{x\in MX_P^l:x.s\geq\beta'\}\cup\{x\in MX_P^r:x.e\leq\alpha'\}$
13. $rx\leftarrow\min w\{x\in RS(\hat{x},P)\cup\hat{x}\}$
14. 更新相关辅助集合与 msg
15. RETURN msg

以下定理保证了算法 2 的正确性.

定理 3. 对于从右孩子 R 到父节点 P 的情况,算法 $InsertXinR2P(msg)$ 在节点 P 中维护了循环不变式,时间复杂度为 $O(\log|V|)$ 最坏时间.

证明. 中间节点 P 中,$MX_P=MX_P^l\cup MX_P^r$.已知加入 $\hat{x}$ 之前 MX_P 是子图 $G_{\hat{P}}=G[(X_P-FX_P)\cup Y_P]$ 中的最优 X 匹配集,现证明加入 $\hat{x}$ 之后 MX_P 是子图 $G'_{\hat{P}}=G[(X_P-FX_P+x)\cup Y_P]$ 中的最优 X 匹配集.与叶子节点类似,若 $\hat{x}.e>Y_P.\max$,$\hat{x}$ 属于 FX_P 集合.若 $\hat{x}.e\leq Y_P.\max$,有两种情况下 P 节点中的操作与 R 节点相同,故可直接根据消息中的信息来计算.第 1 种情况是 $msg.UX^+=\hat{x}$,表明 $\hat{x}$ 不能增广 MX_R,且在 weight-id 序列下,$\hat{x}$ 小于 R 中 $\hat{x}$ 的可替换集中的最小元素.可证明若 $\hat{x}$ 在孩子节点中不能增广匹配,则在父节点中也不能增广匹配,且 $\hat{x}$ 的可替换集的最小元素单调递增.故而在 P 节点 $\hat{x}$ 仍然替换 $\hat{x}$ 自身,与 R 节点相同.第 2 种情况是 $msg.UX^+\in MX_P^r$(约定 MX_P^r 不包含 Λ),表明在 R 节点中 $\hat{x}$ 替换 $x'=msg.UX^+$,且可推知 $\hat{x}$ 的可替换集在 P 中与 R 中相同,故而 P 节点中 $\hat{x}$ 仍然替换 x',与 R 节点相同.

若上述两种情况都不满足,即 $msg.UX^+=\Lambda$ 或者 $msg.UX^+\neq\hat{x}\,\&\&\,msg.UX^+\notin MX_P^r$.前一种情况下,若 P 中存在 $\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$,可推出必然存在 $x'\in FX_L$ 加入到图 $G[MX_P^r\cup Y_P^r]$ 中,且在 x' 和 $\hat{x}$ 之间存在一条交错路.故而根据引理 3,$\hat{x}$ 不可增广 MX_P^r,且通过 $\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$ 可确定可替换集.考虑若 P 中不存在 $\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$ 的情况.假设 $\hat{x}$ 在 P 中不可增广匹配,因为 $msg.UX^+=\Lambda$,即 $\hat{x}$ 在 R 中可增广匹配,则在 P 中必存在 $x'\in FX_L$ 加入到图 $G[MX_P^r\cup Y_P^r]$ 中,且在 x' 和 $\hat{x}$ 之间存在一条交错路,根据引理 3,存在 α-MX 紧致点 i 且 $Y[i]\geq\hat{x}.e$,与前提矛盾,故而若 P 中不存在 $\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$,$\hat{x}$ 在 P 中可增广匹配.后一种情况的分析类似.

因此,通过 $\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)$ 是否存在可判断 $\hat{x}$ 在 P 中是否可增广匹配;若不可增广,根据引理 3,通过 $MX_P^r[\alpha_{\text{Post}}(\hat{x}.e,MX_P^r)]$ 可求解 $\hat{x}$ 在 $G[MX_P^r\cup Y_P^r]$ 中的可替换集,并在其中取 $x.s$ 最小的元素 x_1,且可推知 $x_1.s<Y_P.\text{mid}$.由引理 4,根据 $\beta_{\text{Pre}}(x_1.s,MX_P^l)$ 即可求解 x_1 在 $G[MX_P^l\cup Y_P^l]$ 中的可替换集.由 ϕ_2 可推知,在 MX_P^l 中不存在 x' 使得 $x'.s\geq\beta'$ 且 $x.e>\alpha'$,意味着 x_1 在 $G[MX_P^l\cup Y_P^l]$ 中的可替换集与 $\hat{x}$ 的相同,于是 $\hat{x}$ 在 P 中的可替换集 $RS(\hat{x})$ 可确定.根据推论 2,从 $RS(\hat{x})\cup\hat{x}$ 中替换 weight-id 序列排序下的最小元素即可求解对应于子图 $G'_{\hat{P}}$ 的最优 X 匹配集.需注意的是,在更新相关辅助集合时,需维护不变式 ϕ_2.

根据引理 5,在算法 2 中,每一步的操作都可基于辅助集合对应的树结构在对数时间复杂度下实现,故而在从右孩子到父节点的情况加入 x 顶点的动态操作可在 $O(\log|V|)$ 最坏时间下完成. 证毕.

考虑从左孩子 L 到父节点 P 的路径,算法 3 描述了此时在 P 中加入 x 顶点 $\hat{x}$ 的动态更新操作.

算法 3. $InsertXinL2P(msg)$.

输入:msg 消息

输出:msg 消息

1. $\hat{x}\leftarrow msg.X^+$
2. IF ($\hat{x}.e>Y_P.\max$) THEN
3. $FX_P\leftarrow FX_P+\hat{x}$

4. ELSE IF ($msg.UX^{+}=\hat{x} \| msg.UX^{+} \in MX_P^r$) THEN
5. P 节点中集合更新以及 msg 设置均与 L 节点相同
6. ELSE IF ($\exists \beta_{\text{Pre}}(\hat{x}.s, MX_P^l)$) THEN
7. $MX_P^l \leftarrow MX_P^l + \hat{x}$
8. ELSE
9. $\beta' \leftarrow MX_P^l[\beta_{\text{Pre}}(\hat{x}.s, MX_P^l)].s$ //在图 G_P^l 中
10. $x_1 \leftarrow \max e\{x \in MX_P^l : x.s \geq \beta'\}$
11. $\alpha' \leftarrow MX_P^r[(x_1.e, MX_P^r)].e$ //在图 G_P^r 中
12. $x_2 \leftarrow \min s\{x \in MX_P^r : x.e \leq \alpha'\}$
13. IF ($x_2.s < \beta'$) THEN
14. $\beta' \leftarrow MX_P^l[\beta_{\text{Pre}}(x_2.s, MX_P^l)].s$ //图 G_P^l
15. $RS(\hat{x}, P) \leftarrow \{x \in MX_P^l : x.s \geq \beta'\} \cup \{x \in MX_P^r : x.e \leq \alpha'\}$
16. $rx \leftarrow \min w\{x \in RS(\hat{x}, P) \cup \hat{x}\}$
17. 更新相关辅助集合与 msg
18. RETURN msg

以下定理保证了算法 3 的正确性.

定理 4. 对于从左孩子 L 到父节点 P 的情况，算法 $InsertXinL2P(msg)$ 在节点 P 中维护了循环不变式，时间复杂度为 $O(\log|V|)$ 最坏时间.

证明. 除了对于可替换集的求解，算法 3 与从右孩子到父节点的算法 2 情形类似. 下面仅证明算法 3 中求解可替换集的过程. 若在子图 $G[MX_P^l \cup Y_P^l]$ 中存在 $\beta_{\text{Pre}}(\hat{x}.s, MX_P^l)$（第 10 行），根据引理 4，集合 $\{x \in MX_P^l : x.s \geq \beta'\}$ 为 $\hat{x}$ 在子图 $G[MX_P^l \cup Y_P^l]$ 中的可替换集，选取其中 end-id 排序下最大的元素 x_1. 可证明，$x_1.e \geq Y_P.\text{mid}$，且子图 $G[MX_P^r \cup Y_P^r]$ 中必然存在 x_1 的 α-MX 最小紧致点. 再根据引理 4，通过 $\alpha' = MX_P^r[\alpha_{\text{Post}}(x_1.e, MX_P^r)].e$ 即可确定 x_1 在子图 $G[MX_P^r \cup Y_P^r]$ 中的可替换集 $\{x \in MX_P^r : x.e \leq \alpha'\}$，该集合也即 $\hat{x}$ 在子图 $G[MX_P^r \cup Y_P^r]$ 中的可替换集. 若该集合中按 start-id 排序下最小的元素 x_2 的 $x_2.s < \beta'$，即存在一条从 $\hat{x}$ 出发的交错路可达小于 β' 的 y 顶点，需根据 $x_2.s$ 更新 β'，进而求解出 $\hat{x}$ 在 P 中的可替换集 $RS(\hat{x})$. 与从 R 到 P 的计算类似，更新辅助集合时需维护不变式 ϕ_2. 根据引理 5，在算法 3 中，每一步的操作都可基于辅助集合对应的树结构在对数时间复杂度下实现，故而在从左孩子到父节点的情况下，在节点 P 中，$insert(\hat{x})$ 操作的时间复杂度为 $O(\log|V|)$ 最坏时间. 证毕.

5.1.3 算法实例

图 6 为一个 $insert(x)$ 算法的实例，左上角列出顺次加入的 x 顶点的四元组信息，中间的 3 个方框代表二叉计算树中的树节点，节点旁边列出了相应的辅助集合. 设初始状态存在 4 个 y 顶点，对应计算树中唯一的节点 P. 当加入第一个 x 顶点 x_1 时，因为 $x_1.s=3 \neq Y_P.\min=1$，P 节点分裂并生成新的叶子节点 L 和 R. 算法 1 先尝试在 R 节点中加入 x_1，根据 α 树计算得出不存在 $x_1.e$ 的 α_{Post} 紧致点，故而 x_1 在 R 中成功增广匹配，加入 X 匹配集 MX_R^r. 下一步，因为 R 是 P 的右孩子，调用算法 2，尝试在 P 节点加入 x_1；由于 P 节点中同样不存在 $x_1.e$ 的 α_{Post} 紧致点，故而 x_1 增广 X 匹配集 MX_P^r.

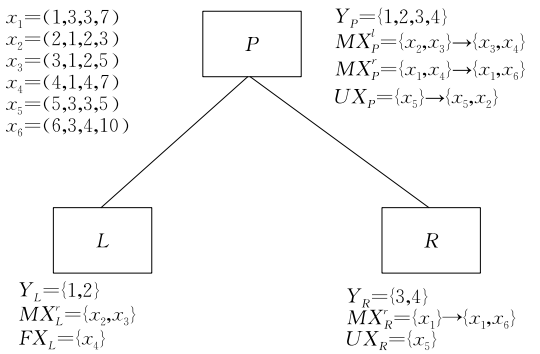

图 6　$insert(x)$ 算法实例

类似的，x_2，x_3，x_4 都顺次成功加入 P 的 X 匹配集中；其中 x_4 在节点 L 中进入 FX_L 集合，因为 $x_4.e=4>Y_L.\max=2$，再根据算法 3，在 P 节点 x_4 增广匹配成功. 对于 x_5，算法 1 在 R 中尝试添加 x_5；根据 α 树计算 $\alpha_{\text{Post}}(x_5.s, MX_R^r)=3$，故而可替换集 $RS(x_5, R)=\{x_1\}$；由于 $x_1.w=7>x_5.w=5$，故而 $rx=x_5$，R 中匹配未增广成功，x_5 被加入 UX_R 集合. 继而算法 2 在 P 节点尝试加入 x_5，由于 $msg.UX^{+}=\hat{x}=x_5$，在 P 节点中仍然不能增广匹配，x_5 加入 UX_P 集合.

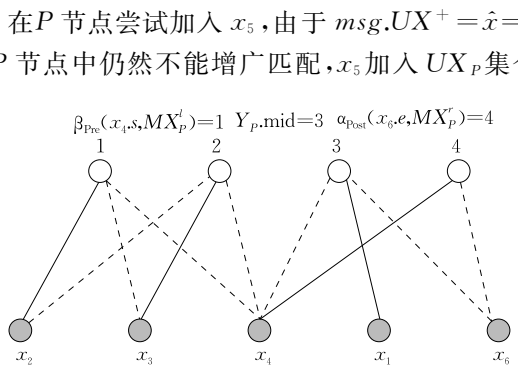

图 7　节点 P 中添加 x_6 的计算

下面描述加入 x_6 的过程，图 6 中节点 P 和 R 的辅助集合中的箭头分别指出在此过程中集合的变化情况. 首先根据 $x_6.s$ 定位到 R 节点，根据算法 1，在 R 节点加入 x_6 成功. 继而调用算法 2，在 P 节点中加入 x_6. 根据 α 树计算可得 $\alpha_{\text{Post}}(x_6.s, MX_P^r)=4$，在右侧的可替换集中找到 $x.s$ 最小的 x 顶点 x_4（参考算法 2 第 10 行和图 7），计算 $\beta_{\text{Pre}}(x_4.s, MX_P^l)=1$，得

出可替换集 $RS(x_6,P)=\{x_1,x_2,x_3,x_4\}$，其中按 weight-id 排序最小的 x 顶点为 x_2，故而 x_6 在 P 的 X 匹配集中替换 x_2.

6 算法复杂度分析与实验仿真

在维护最优 X 匹配集的基础上，动态查询操作可采用文献[17]或[23]中的方法求解. 我们基于文献[23]中的分离匹配区间(disjoint matching interval)方法，在最坏 $O(k)$时间复杂度下求解动态查询操作，其中 $k\leq\min\{|X|,|Y|\}$. 综合更新和查询操作算法，我们给出以下定理.

定理 5. 左侧带权凸二分图的最大权值动态匹配问题可在 $O(\log^2|V|)$平摊时间复杂度下维护更新操作，在常数时间复杂度下查询是否匹配，在最坏 $O(k)$时间复杂度下查询某已匹配节点的相匹配节点，其中 $k\leq\min\{|X|,|Y|\}$.

证明. 根据定理 2～4，算法中每一步操作都可基于辅助集合对应的树结构在对数时间复杂度下实现，故而在中间节点中加入 x 的操作在对数时间复杂度实现. 加入 x 顶点的计算包含至多 $\log|V|$ 次加入二叉计算树中某节点的操作，因此可在 $O(\log^2|V|)$平摊时间复杂度下求解. 其他动态操作同理可在相同时间复杂度下进行维护. 证毕.

针对不带权凸二分图的最大基数动态匹配问题，文献[17]设计了一种算法在 $O(\log^2|V|)$平摊时间复杂度下维护更新操作，在最坏常数时间复杂度下查询某顶点是否已匹配，查询某已匹配顶点的相匹配顶点操作的最坏时间复杂度为 $O(\min\{k\log^2|X|+\log|X|,|X|\log|X|\})$，其中 $k<\min\{|X|,|Y|\}$，或者为 $O(\sqrt{|X|}\log^2|X|)$平摊时间复杂度. 左侧带权凸二分图的最大权值动态匹配问题之前一直未被解决，直到我们在文献[23]中提出一种方法在 $O(\log^3|V|)$平摊时间复杂度下维护更新操作，在最坏 $O(k)$时间复杂度下维护查询操作，其中 $k\leq\min\{|X|,|Y|\}$.

下面的表 2 中，呈现了文献[17,23]和本文的工作结果的时间复杂度比较.

表 2 时间复杂度比较

	文献[17]	文献[23]	本文工作
研究问题	凸二分图动态最大基数匹配	左侧带权凸二分图动态最大权值匹配	左侧带权凸二分图动态最大权值匹配
动态更新	$O(\log^2\|V\|)$平摊时间	$O(\log^3\|V\|)$平摊时间	$O(\log^2\|V\|)$平摊时间
动态查询	最坏接近线性时间	最坏线性时间	最坏线性时间

本文基于 C++实现了左侧带权凸二分图的最大权值动态匹配算法. 实验评估过程运行在一台 Lenovo-X1 笔记本上，其 CPU 型号为 Intel Core i5-5200，主频 2.2 GHz，内存 8 GB，操作系统是 64 位 Windows 8，程序运行环境是 Visual Studio 2013. 其中数据集是随机生成.

图 8 显示了运行的结果，其中横坐标表示当前图的规模，如 1000 表示图中含有 1000 个 x 节点和 1000 个 y 顶点，纵坐标表示在当前规模的图上，新加入一个 x 节点所花的时间，单位是 ms.

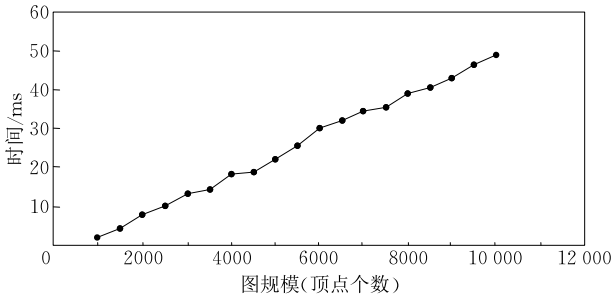

图 8 实验仿真结果

7 结 论

本文研究左侧带权凸二分图的最大权值动态匹配问题. 发现该问题的求解的关键在于维护最优 X 匹配集以及求解可替换集合，提出紧致子图的概念，证明可替换集合的求解等价于紧致子图的求解，将求解匹配问题转换成为求解子图问题. 同时研究凸二分图结构中的凸性质，找到一种快速求解该子图的方法. 在此基础之上，基于改进的二叉计算树数据结构和隐性表征技术，设计算法在 $O(\log^2|V|)$平摊时间下维护图的更新操作，在最坏线性时间复杂度下支持顶点匹配信息的查询操作. 进而在与文献[17]解决不带权问题的相同时间复杂度下，解决了难度更高的左侧带权问题.

参 考 文 献

[1] Hopcroft J E, Karp R M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, 1973, 2(4): 225-231

[2] Plaxton C G. Fast scheduling of weighted unit jobs with release times and deadlines//Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'08). Reykjavik, Iceland, 2008: 222-233

[3] Chen H-T, Lin H-H, Liu T-L. Multi-object tracking using dynamical graph matching//Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01). Hawaii, USA, 2001: II-210-217

[4] Deng Shui-Guang, Yin Jian-Wei, Li Ying, et al. A method of semantic Web service discovery based on bipartite graph matching. Chinese Journal of Computers, 2008, 31(8): 1364-1375(in Chinese)
(邓水光, 尹建伟, 李莹等. 基于二分图匹配的语义 Web 服务发现方法. 计算机学报,2008,31(8):1364-1375)

[5] Wang Y, Makedon F, Ford J, Huang H. A bipartite graph matching framework for finding correspondences between structural elements in two proteins//Proceedings of the 26th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (IEMBS'04). San Francisco, USA, 2004: 2972-2975

[6] Zhang Wei-Feng, Zhou Yu-Ming, Xu Lei, Xu Bao-Wen. A method of detecting phishing web pages based on hungarian matching algorithm. Chinese Journal of Computers, 2010, 33(10): 1963-1975(in Chinese)
(张卫丰, 周毓明, 许蕾, 徐宝文. 基于匈牙利匹配算法的钓鱼网页检测方法. 计算机学报, 2010, 33(10): 1963-1975)

[7] Mahmood K, Webb G I, Song J, et al. Efficient large-scale protein sequence comparison and gene matching to identify orthologs and co-orthologs. Nucleic Acids Research, 2012, 40(6): e44

[8] Plaxton C G. Vertex-weighted matching in two-directional orthogonal ray graphs//Proceedings of the Algorithms and Computation (ISAAC'13). Hong Kong, China, 2013: 524-534

[9] Zhang N, Cheng N, Liang H, et al. Efficient channel assignment for cooperative sensing based on convex bipartite matching//Proceedings of the 2014 IEEE International Conference on Communications (ICC'14). Sydney, Australia, 2014: 1403-1408

[10] Glover F. Maximum matching in a convex bipartite graph. Naval Research Logistics Quarterly, 1967, 14(3): 313-316

[11] van Emde Boas P. Preserving order in a forest in less than logarithmic time and linear space. Information Processing Letters, 1977, 6(3): 80-82

[12] Lipski Jr W, Preparata F P. Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. Acta Informatica, 1981, 15(4): 329-346

[13] Aho A V, Hopcroft J E, Ullman J D. The Design and Analysis of Computer Algorithms. Addison-Wesley, 1974

[14] Tarjan R E. Efficiency of a good but not linear set union algorithm. Journal of the ACM, 1975, 22(2): 215-225

[15] Steiner G, Yeomans J S. A linear time algorithm for maximum matchings in convex, bipartite graphs. Computers and Mathematics with Applications, 1996, 31(12): 91-96

[16] Katriel I. Matchings in node-weighted convex bipartite graphs. INFORMS Journal on Computing, 2008, 20(2): 205-211

[17] Brodal G, Georgiadis L, Hansen K, Katriel I. Dynamic matchings in convex bipartite graphs//Proceedings of the Mathematical Foundations of Computer Science (MFCS'07). Ceský Krumlov, Czech Republic, 2007: 406-417

[18] Dekel E, Sahni S. A parallel matching algorithm for convex bipartite graphs and applications to scheduling. Journal of Parallel and Distributed Computing, 1984, 1(2): 185-205

[19] Berge C. Two theorems in graph theory. Proceedings of the National Academy of Sciences of the United States of America, 1957, 43(9): 842-844

[20] Spencer T, Mayr E. Node weighted matching//Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'84). Antwerp, Belgium, 1984: 454-464

[21] Balakrishnan R, Ranganathan K. A TextBook of Graph Theory. New York, USA: Springer, 2000

[22] Cormen T H, Leiserson C E, Rivest R L, Stein C. Introduction to Algorithms. 3rd Edition, Massachusetts, USA: The MIT Press, 2009

[23] Zu Q, Zhang M, Yu B. Dynamic matchings in left weighted convex bipartite graphs//Proceedings of the Frontiers of Algorithmics Workshop (FAW'14). Zhangjiajie, China, 2014: 330-342

ZU Quan, born in 1982, Ph. D. candidate. His research interests include combinatorial optimization, model checking.

ZHANG Miao-Miao, born in 1971, Ph. D., professor, Ph. D. supervisor. Her research interests include modeling and verification of timed and hybrid systems.

LIU Jing, born in 1965, Ph. D., professor, Ph. D. supervisor. Her research interests include model driven software development and software engineering.

Background

Matching in bipartite graphs forms a classical area of combinatorial optimization and of graph theory. In convex bipartite graphs (CBGs), matching problems are equivalent to several classic scheduling problems. Dynamic matching in CBGs also belongs to the field of dynamic graph problems, which study the properties of graphs in the context of vertices or edges being changing.

To the best of our knowledge, in the area of dynamic matching in CBGs, only maximum cardinality matching problem was studied previously. Our project hence focuses on the problems in weighted situations, i. e., dynamic maximum weight matching (DMWM) in weighted CBGs. Matching problems in weighted situations are different from and more difficult than the problem in the corresponding unweighted situation. In our previous work, the authors gave a solution for the DMWM problem in the left weighted CBGs. They designed an algorithm that maintains the update operations of changing vertices or edges in $O(\log^3|V|)$ amortized time.

In this work, the authors improve the solution for DMWM in the left weighted CBGs. They propose an approach that reduces traditionally computing matchings via alternating paths to computing matchings via finding a certain subgraph structure. Then they extend a two-dimensional balanced binary search tree data structure and use it to compute the subgraph structure efficiently, based on which an algorithm is designed to maintain the update operations in $O(\log^2|V|)$ amortized time. Our future works include DMWM problems in general weighted CBGs and to find more efficient solution for maintaining the query operations.

This work is supported by the National Natural Science Foundation of China (Nos. 61472279, 61332008, 91318301).