文章编号:1007-5321(2020)04-0113-07 DOI:10.13190/j.jbupt.2019-198

基于度约束最小生成树的域间路由恢复算法

王 禹[1,2,3], 张连成[2,3], 张宏涛[2,3,4], 郭 毅[2,3]

(1. 河南工程学院 计算机学院, 郑州 451191; 2. 数学工程与先进计算国家重点实验室 网络密码研究室, 郑州 450002;
3. 解放军信息工程大学 网络空间安全学院, 郑州 450002; 4. 郑州大学 网络管理中心, 郑州 450001)

摘要: 低速拒绝服务攻击对于域间路由系统造成威胁,已有失效恢复算法未能有效解决恢复拓扑计算的时间复杂度高和节点聚合控制等问题,为此,提出一种基于度约束最小生成树的失效恢复算法. 通过设计基础迁移子算法和复杂迁移子算法,在满足度约束的条件下根据遭袭路由系统生存拓扑构建新的恢复拓扑,并针对上述两类迁移子算法,分别提出关键点选择子算法,用于判定和计算迁移过程所需的关键节点. 理论分析和仿真实验结果证明,该算法生成的恢复拓扑在有效控制节点度的同时,具有较优的性能.

关 键 词: 域间路由系统; 失效恢复; 度约束最小生成树; 时间复杂度; 节点聚合控制

中图分类号: TP393 **文献标志码**: A

A Failure Recovery Algorithm for Inter-Domain Routing System Based on Degree-Constrained Minimum Spanning Tree

WANG Yu[1,2,3], ZHANG Lian-cheng[2,3], ZHANG Hong-tao[2,3,4], GUO Yi[2,3]

(1. School of Computer Science, Henan University of Engineering, Zhengzhou 451191, China; 2. Network Cryptography Laboratory, State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450002, China;
3. School of Cyberspace Security, PLA Information Engineering University, Zhengzhou 450002, China;
4. Network Management Center, Zhengzhou University, Zhengzhou 450001, China)

Abstract: In view of the threat of low-rate denial of service attacks on inter-domain routing system, the existing failure recovery methods fail to solve problems including high time complexity and node aggregation control. A failure-recovery algorithm based on degree-constrained minimum spanning tree named degree-constrained minimum spanning tree based failure recovery (DR) is proposed. By designing the Fundamental Transfer sub-algorithm and the complex transfer sub-algorithm, a new recovery topology can be constructed according to the survival topology of the attacked routing system under the condition of given degree constraint. For the above two types of transfer sub-algorithms, two selection sub-algorithms are respectively advanced for the determination and calculation of key nodes. Theoretical analysis and simulation experiments verify that the recovery topology generated by DR has better performance while effectively controlling the node degree.

Key words: inter-domain routing system; failure recovery; degree-constrained minimum spanning tree; time complexity; control of node aggregation

收稿日期: 2019-08-28

基金项目: 国家自然科学基金项目(61802115); 河南省高等学校重点科研项目(18A520004, 9A520008); 河南省科技攻关计划项目(182102310925, 192102310445, 192102210283)

作者简介: 王 禹(1984—), 男, 讲师, E-mail:stonchor@163.com.

近年来,域间路由系统面临的安全防护压力逐步加大[1-3],边界网关协议低速拒绝服务(BGP-LDoS,border gateway protocol-low-rate denial of service)攻击的攻击威胁与复杂程度不断攀升[4]. Schuchar 和 Kan 的实验结果均证明,BGP-LDoS 攻击能够导致域间路由系统局部乃至整体失效. 伴随物联网的发展,以大规模僵尸网络为基础攻击条件的 BGP-LDoS 将给域间路由系统带来更为深远的影响.

域间路由系统失效恢复,旨在利用多种路由恢复技术保障网络遭袭后生存节点的通信. 按照失效自治域和域间链路的数量,可将已有路由失效恢复算法划分为基于单链/单域失效的路由恢复和基于多链/多域失效的路由恢复. 前者以路径拼接[5]和混合抗干扰方法[6]为代表,难以应对 BGP-LDoS 攻击造成的多点无关联区域失效场景. 后者指无邻接关系的多个自治域区域近乎同时并发呈现通信中断态势下,针对尚存活自治域实施报文转发路径的动态切换,以 IP 快速重路由[7]和多级可信传输机制(CETML, credible transmission with multiple levels)[8]为代表. 此类算法在预计算备份子图生成方面存在时间复杂度较高的问题,且 BGP-LDoS 攻击以高度聚合的中心节点为攻击目标,已有 CETML 等算法未考虑对生成子图中节点度的控制,因此在面对 BGP-LDoS 攻击时其生成的备份子图大都具有高中心度节点特性,导致其二次抗毁性较差.

综上,借鉴中心化网络控制的设计理念,通过定义 BGP-LDoS 袭击域间路由系统后生存路由的规划,提出一种基于度约束最小生成树的路由失效恢复算法(DR, degree-constrained minimum spanning tree based failure recovery). 笔者的主要贡献为:①基于 DR 算法实现的域间路由失效恢复方案,在产生最小生成树,即路由系统恢复拓扑的过程中能够有效控制节点的聚合,避免生成具有过高度数节点;②针对已有多链/多域恢复算法在备份子图生成方面存在时间耗费较高的问题,DR 算法能够有效降低恢复路由的生成时长.

1 域间路由系统路由失效恢复

针对自治域(AS,autonomous system)和域间路由系统拓扑开展研究. 应用互联网数据分析中心(CAIDA,center for applied internet data analysis)[9]针对来自多个数据源的 AS 级拓扑特性的统计表明,平均度值的范围基本落于[4,6],数据源中的最大度值范围为[2 607,4 386]. 由 CAIDA 基于 ARK(Archipelago)和基于 traceroute 的计算节点度值的统计可知,度值大于 10 的节点概率均为 0.09 左右,大于 1 000 的节点概率约为 0.005,表明大多数节点的度值小于 10.

1.1 度约束最小生成树

度约束最小生成树(DCMST,degree-constrained minimum spanning tree),即如果针对最小生成树中各顶点的度数加上一定的数目限制,则满足条件的最小生成树记为度约束最小生成树. 对于连通图 $G=(V,E)$,其节点集合 $V=\{v_1,v_2,\cdots,v_n\}$,节点个数为 $|V|$;边的集合 $E=\{e_1,e_2,\cdots,e_k\}$,边的个数为 k;每条边对应的权值为 w_{ij};假定 T 表示 G 中所有满足度约束的生成树集合.

1.2 生成树编码

针对域间路由系统中面对的 BGP-LDoS 攻击,提出一种节点表示法(NR,node representation),便于图中节点编码. 对于一个简单无向图 $G=(V,E)$,存在一个生成森林 $F=(V,E_F)$ 是 G 的无环子图,F 中的一棵树 $T=(V_T,E_T)$ 是 F 中的一个连接部分,有 $V_T\subseteq V$. 对于一个图 G 及其中一棵树 T,$G^T=(V_G^T,E_G^T)$ 是由节点集合 V_T 生成的一个子图,即 E_G^T 包含的所有边 $(x,y)\in E$,均满足 $x\in V_T$ 和 $y\in V_T$.

对于一棵树 T,根为 Root,定义节点 n 的进深为从 n 到 Root 的路径长度,记为 $p(n)$. $N_G(n)$ 表示 G 中毗邻节点 n 的节点集合. G 中一个节点的度表示为 $d_G(x)$,T 中一个节点的度表示为 $d_T(x)$. G 中一棵树 T 的度记为 $d_G(T)$,是指对于 G 的边集 E,射入属于 V_T 集合节点的边的总数. 与之类似,T 中边的个数为 $d_T(T)=|V_T|-1$.

一棵树 T 的 NR 为一个有序列表,其中每个 NR 涵盖 4 个元素 (n,I,p,d):分别表示节点、索引、节点进深和节点度值. 对于一个节点 n,$I(n)$ 表示其索引值,$p(n)$ 和 $d_G(n)$ 分别表示其进深和度值. 对于 T,考虑到深度优先搜索对于节点的顺序特性,基于深度优先法则实施 NR 编码时,搜寻到一个未访问过的节点,则可将该节点以及对应的进深、度值加入 NR 编码序列.

一棵生成树 F 的 NR 编码,记为森林表示(FoR, forest representation),是 F 中每棵树 T 的 NR 合并. 由于一片森林包含多棵树,则可用一个对偶 (h_T,d_T) 表示一棵树,前者表示指向树 T 的指针,后

者表示该树的度值.

1.3 DR 算法建立

1)对于代表多个自治域组成的域间路由系统,其对应原始图 G. 在应对路由失效而生成 DCMST 时,由于域间路由节点在不断迁移和变化,因此通过创建并逐步完善编码矩阵的方式记录该过程.

子算法 1　节点迁移编码 TransferNodesEncode( )

输入:图 G(代表多个自治域组成的域间路由系统). **输出**:节点的迁移编码矩阵 M_N.

① 对于 G 中一棵树 F_i 中的某节点 n,创建其对应的组数 $A_n=[i,H_i,I(n)_i]$,i 表示该节点所属树的序号,H_i 为指向树 F_i 对应 NR 编码的指针,最后的 $I(n)_i$ 为NR 中 n 对应的索引值;② 假设 F_1 是来自 G 的首个森林,在 F_1 的基础上,随着节点或子树的裁剪和迁移,不断生成的森林序列记为 $L=[F_1,F_2,\cdots,F_i]$;③ 若节点 n 所在的一棵子树 SubTr,从树 F_j 迁移至树 F_k,则节点 n 将被重新定位,该节点的迁移编码矩阵记为 $\boldsymbol{M}_n=[\boldsymbol{A}_1^{\mathrm{T}},\cdots,\boldsymbol{A}_j^{\mathrm{T}},\boldsymbol{A}_k^{\mathrm{T}}]$.

2) DR 算法通过定义 FundamentalTransfer(FT)和 ComplexTransfer(CT)以产生新的生成森林,即从已有生成森林 F 对应的森林表示 FoR,产出新生成森林 F'的 FoR. FT 和 CT 操作的目标是从森林中的 T_{origin}(一组树)里的子树迁移至另一棵树 T_{dst}.

子算法 2　基础迁移 FundamentalTransfer( ),FT

FT 针对 BGP-LDoS 攻击后,导致域间路由系统中某一关键节点失效的典型场景,针对该问题构建新的森林,并有效满足度约束值.

输入:森林 F 的 FoR 序列中树 T_{origin} 和 T_{dst} 的索引;被迁移子树 T_{origin} 中的根 r,此时也称之为修剪节点;新树 T_{dst} 中的节点 t,t 满足在图 G 中与 r 相邻;度约束值 c. **输出**:新生成森林 F'的 FoR.

① 寻找 T_{origin} 中修剪节点 r 所属子树,其对应的索引范围是$[I(r),I(m)]$,该范围涵盖了节点 r 及其后继节点 m 等,且满足 $p(r)$ 值小于后继节点. 考虑到 $I(r)$ 值已知,通过顺序搜索可发现后继节点的 $I(m)$;② 临时创建一个中间树 T_{mid},用于存放迁移过程中的节点及子树;③ 将 T_{origin} 中索引范围$[I(r),I(m)]$内的 NR 编码放入临时 T_{mid},按照 $p(n)-p(r)+p(t)+1$ 计算 T_{mid} 每个 n 的 $I(n)$;④ 创建一个序列 T'_{dst},包含 T_{dst} 和 T_{mid} 所有节点. 通过生成一棵新树,将以 r 为根的子树迁移至 T'_{dst}. 仅当满足 $d(t)<c+1$ 时,将 T_{mid} 插入 T'_{dst} 的$[I(t)+1]$索引处,之后修正 $p(t)=p(t)+1$;若不满足则跳转到步骤④,对 t 之后的下一节点进行判断;如果 t 已是 T_{dst} 中最后节点,则表明 c 过小,FT 结束;⑤ 创建序列 T'_{origin},包含排除 T_{mid} 所含节点以外 T_{origin} 中的所有节点,之后修正 T_{origin} 中 $d(r)=d(r)-1$;⑥ 复制 F 对应的 NR,生成一个新的 F'. 将指向 F'中新旧两棵树的指针分别记为 $h\_T'_{\mathrm{origin}}$ 和 $h\_T'_{\mathrm{dst}}$;⑦ 计算 $d_G(T'_{\mathrm{origin}})$和$d_G(T'_{\mathrm{dst}})$,并存储于 F';⑧ 对矩阵 M_N 中 T'_{origin} 和 T'_{dst} 的每个节点 n 修正.

子算法 3　复杂迁移 ComplexTransfer( ),CT

针对域间路由系统在遭受 BGP-LDoS 攻击后,导致毗邻的多个自治域节点失效场景,设计 CT 子算法生成度约束最小生成树,予以实施路由恢复.

输入:森林 F 的 FoR 序列中树 T_{origin} 和 T_{dst} 的索引;被迁移树 T_{origin} 中的根 r;修剪节点 f;新树 T_{dst} 中的节点 t,t 满足在图 G 中与 r 相邻;度约束值 c. **输出**:新生成森林 F'的 FoR.

① 寻找 T_{origin} 中根节点 r 所属子树,其对应索引范围$[I(r),I(m)]$,该范围涵盖节点 r 及其后继节点 m 等,且满足 $p(r)$ 值小于后继节点;② 临时创建 2 个中间树 T_{mid1} 和 T_{mid2},用于存放迁移过程中的节点及子树;③ 将 T_{origin} 中索引范围$[I(r),I(m)]$内的 NR 编码,放入临时 T_{mid1},按照 $p(x)-p(r)+p(t)+1$ 计算 T_{mid1} 每个节点 n 的 $I(n)$;④ T_{origin} 中从 r 到 f 的路径中节点,标记为$[r_1,r_2,\cdots,r_n]$,对应于 $r=r_1$,以及 $f=r_n$;⑤ 寻找从 r 到 f 的路径. 初始化 T_{mid2} 为空 NR,对于每个 $i(1<i\leqslant n)$,仅当满足 $d(i)<c+1$ 时,复制以 r_i 为根的子树(但不包括以 r_{i-1} 为根的子树)到 T_{mid2} 的末端,同时,按照 $p(n)-p(r)+i+p(t)+1$ 方式修正每个加入 T_{mid2} 的节点 n 的进深值;之外,执行 $p(r)=p(r)+1$ 和 $p(f)=p(f)-1$ 以修正节点 r 和 f;若不满足 $d(i)<c+1$,在跳转至步骤⑤,对下一个节点 r_{i+1} 判断;如果 r_i 已是 T_{dst} 中最后节点,则表明 c 过小,CT 结束;⑥ 将被裁剪子树和 T_{dst} 连接组合成新树 T'_{dst},即包含 T_{dst},T_{mid1} 和 T_{mid2},以$[T_{\mathrm{mid1}}\ T_{\mathrm{mid2}}]$序列方式置于 T'_{dst} 的$(t+1)$处,执行 $d(t)=d(t)+1$ 修正 t 的度值;⑦ 利用 T_{origin} 中$[T_{\mathrm{tmp1}}\ T_{\mathrm{tmp2}}]$之外的节点创建 T'_{origin},将原 T_{origin} 中 f 的前趋节点度值减 1;⑧ 复制 F 对应的 NR 生成一个新的森林 F',将指向 F'中新旧两棵树的指针分别记为 $h_T'_{\mathrm{origin}}$和 $h\_T'_{\mathrm{dst}}$;计算 $d_G(T'_{\mathrm{origin}})$ 和 $d_G(T'_{\mathrm{dst}})$并存于 F';⑨修正矩阵 $\boldsymbol{M}_N$ 中 T'_{origin} 和 T'_{dst} 中的每个节点 n.

子算法 4　单树森林修正算法 UpdateForSingle-

Tree( )

上述算法在应用 CT 与 FT 时,均假设一个森林中至少存在两棵树. 为了扩展算法的适用性,可将仅有一棵树的森林予以修正,依照特殊方式实施操作.

输入:森林 F 的唯一树 T_{origin}. **输出**:新生成森林 F'的 FoR.

① 建立一棵树 T_{sup}进行辅助,假设该树仅包含一个节点 n,此时该节点 $n \notin G$,且 n 与 T_{origin}中每个节点均保持连接;② 令 T_{dst}为 T_{sup},执行 FT,由此可将 T_{origin}中一棵子树变为 T_{sup}中 n 的后继;③ 利用 FT 或 CT 操作,将以 n 为根子树变为原始树,从而生成一个新的森林 F'.

子算法 5 FT 关键节点选择 KeyNodeSelectionForFT( )

输入: 森林 F 的 FoR 序列. **输出**:FT 中新树 T_{dst}中节点 t,被迁移子树 T_{origin}中的根 r.

① 随机寻找 F 中一棵树 T,需满足 $d_G(T)$大于 $d_T(T)$,令 T_{origin}为 T. 若无法找到,则说明图 G 本身即单个森林,停止该过程;② 从 T_{origin}中随机确定一个非根节点 r,$d_G(r)$大于 $d_T(r)$;③ 从 $N_G(r)$随机选择一个节点 n,若边(n,r)不属于 E_T,则令 t 为 n,否则跳过本步骤;之后通过搜寻矩阵以确定节点 t 在森林 F 中的索引 $I(t)$;④ 若 t 属于 T_{origin}的节点集合,则执行单树森林操作;否则执行 FT.

子算法 6 CT 关键节点选择 KeyNodeSelectionForCT( )

输入:森林 F 的 FoR 序列. **输出**:被迁移树 T_{origin}中的根 r;修剪节点 f;新树 T_{dst}中的节点 t.

① 随机寻找 F 中一棵树 T,需满足 $d_G(T)$大于 $d_T(T)$,令 T_{origin}为 T. 若无法找到,则说明图 G 本身即单个森林,停止该过程;② 从 T_{origin}中随机确定一个非根节点 r,$d_G(r)$大于 $d_T(r)$;③ 寻找 r 至 T_{origin}根的路径,令 i 为 1 且 r_i为 r,以向根的方向从 T_{origin}中 $I(r)$开始遍历,步长为 1,同时判别遍历过程中的每个节点 n,若 $p(n)$小于 $p(r_i)$,则令 r_{i+1}为 n. 若 $i>2$,则在$[2,\cdots,i-1]$范围内随机选择 j,令 f 为 r_j;若整个路径仅包含节点 r 和根,则令 f 为 r;④ 从 $N_G(r)$中随机选择一个节点 n,若边(n,f)不属于 E_T,则令 t 为 n,否则重复本步骤;之后,通过搜寻矩阵以确定节点 t 在森林 F 中的索引 $I(t)$;⑤ 若 t 属于 T_{origin}的节点集合,则执行单树森林操作;否则执行 FT.

1.4 基于 DR 算法的恢复路由生成流程

首先,基于节点表示法 NR,对由多个 AS 组成的路由系统编码, 将系统表示为含有多棵树的森林,记为 FoR;其次,如果目前 FoR 代表仅含单树的森林,由于后续的基础迁移 FT 和复杂迁移 CT 算法均假设森林中至少存在两棵树,因此为拓展算法的适用性,利用单树森林修正算法 UpdateForSingleTree( )对该森林修正. 反之,则直接进行失效 AS 节点迁移操作;再次,路由系统遭袭后通常会造成单个失效自治域节点或多个毗邻失效自治域节点,根据 2 种不同的情况,分别采用 FT 关键节点选择算法和 CT 关键节点选择算法,率先计算出 2 种情况下的关键节点;之后,分别利用 FT 算法和 CT 算法,实施失效节点的迁移操作,完成域间路由系统恢复.

2 时间复杂度分析

分析度约束最小生成树的时间复杂度. 对于多域社团抽象出的连通图 G,包含的自治域数目为 n.

定义 1 $s=|T_{\text{origin}}|+|T_{\text{dst}}|$,其中$|T_{\text{origin}}|$和$|T_{\text{dst}}|$分别表示参与节点交换的两棵树的容量大小,$s$ 即代表交换操作的时间复杂度,其平均值为 $\bar{s}$.

定义 2 $t=|T|$表示 G 中生成森林中树的个数.

定义 3 $s_{\text{op}}=s_{\text{FT}}+s_{\text{CT}}$,表示 FT 和 CT 操作的时间复杂度,其平均值为 $\bar{s}_{\text{op}}$.

定义 4 s_{init}为初始化子树指针 h、新树根 r_{new}、备选节点序列等参数的复杂度,其平均值为 $\bar{s}_{\text{init}}$.

度约束最小生成树的时间复杂度 $C=s_{\text{op}}+s_{\text{init}}$.

1) 计算 $\bar{s}$. 如果连通图 G 中的 n 个节点均匀分布于 t 棵树(假定 $t=\lceil\sqrt{n}\rceil$),令 K 为森林中的不同树对组成的集合,则 s 的平均大小为

$$\bar{s}=\sum_{\{i,j\}\in K}\frac{|T_i|+|T_j|}{|K|}$$

其中,若固定 i 不变,则对于$|T_i|+|T_j|$有 $t-1$ 组,此时$|T_i|+|T_j|$之和为

$$(t-1)|T_i|+\sum_j|T_j|-|T_i|$$

之后对每种 T_i 情形累加,求和可得

$$\bar{s}=\sum_{\{i,j\}\in K}\frac{|T_i|+|T_j|}{|K|}=\frac{2n}{t}=O\left(\frac{n}{t}\right)=O(\sqrt{n})$$

2) 计算 $\bar{s}_{\text{op}}$. 根据 FT 和 CT 操作步骤,可知

$$s_{\text{op}}=O(|T_{\text{origin}}|+|T_{\text{dst}}|+t)=O(s+t)$$

由上述 1)结论,则有

$$\bar{s}_{op} = O(\bar{s} + \bar{t}) = O(\sqrt{n})$$

3）计算 $\bar{s}_{init}$. 根据初始化过程可知

$$\bar{s}_{init} = O(t) + O(3|T_{init}|) + O\left(\bar{s} + \frac{n}{s}\right)$$

$|T_{init}|$ 的上限为 $|T_{origin}| + |T_{dst}|$，则由 1）、2）可知

$$O(3|T_{init}|) = O(s + t)$$

$$\bar{s}_{init} = O\left(\bar{s} + t + \frac{n}{s}\right) = O(\sqrt{n})$$

4）度约束最小生成树的时间复杂度 $\bar{C}$

$$\bar{C} = \bar{s}_{op} + \bar{s}_{init} = O(\sqrt{n})$$

分析可知，度约束最小生成树的平均时间复杂度为 $O(\sqrt{n})$，根据 CNM（Clauset Newman Moore）算法对 AS 级域间路由系统的分解，约 70% 的社团 AS 数目小于 100，因此基于度约束最小生成树的恢复算法在时间耗费方面完全能够满足多域环境下恢复路由的快速生成.

3　验证和比较

利用锦标赛选择法挑选子代，操作数选择为 2，分别对应 FT 操作和 CT 操作. 基于精英选择策略，通过对收敛速度的判别，确保仅有最佳解决方案被用于构建下一代生成树.

3.1　基于随机图的仿真

利用拓扑产生器，基于广义线性优先（GLP，generalized linear preference）模型生成随机图，代表路由系统 AS 级拓扑. 设置随机图节点个数分别为 10、50、100、200、400、600、800、1 000. 模拟 BGP-LDoS 对上述随机图实施袭击，针对存活节点及链路，分别利用 DR 算法同边集表示（ESR，edge set representation）[10]、基于蚁群的度约束生成树（AB-DCST，ant-based algorithm for degree constrained spanning trees）[11] 和混合稳定态遗传算法（HSSGA，hybrid approach combining a steady-state genetic algorithm）[12] 3 种算法生成 DCMST. 对生成树节点度值范围限定在[3，6]；GLP 生成的随机图平均节点度为 4.1.

1）DR 与 ESR 算法对比

ESR 利用特定交换和变异，由边集操作产生新树. 仿真实验同样采用文献[10]中对 ESR 的设置，交换概率和变异概率均为 0.8.

图 1 和图 2 分别给出了在限制度值分别为 3 和 6 的情况下，包含不同节点个数的随机图，利用 ESR 算法和 DR 算法分别生成 DCMST 所花费时间的对比. 由图可知，2 种算法下，随着度值约束的放松，即在约束值越高的情况下，随机图生成时间相对越短. 整体上，同样拓扑规模对应的 DCMST 生成时间差距较小.

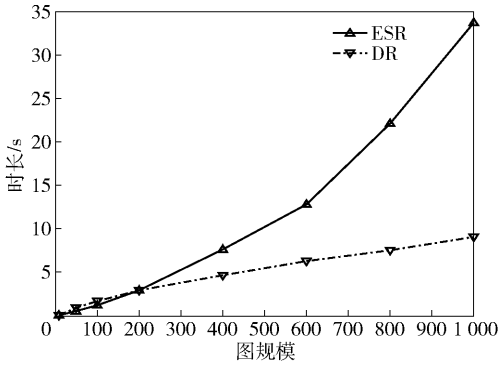

图 1　限定度约束为 3 的不同图规模下 DCMST 生成时间

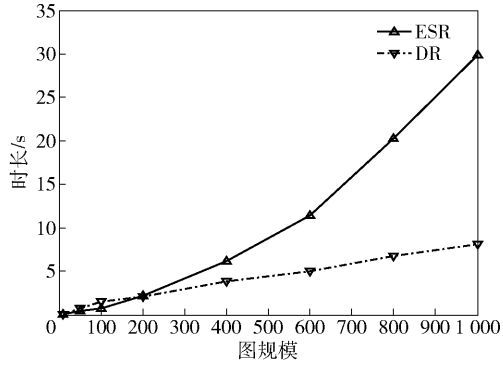

图 2　限定度约束为 6 的不同图规模下 DCMST 生成时间

DR 算法相对于 ESR 算法具有明显的性能优势，伴随着拓扑规模增大，性能优势越发突出. 与时间复杂度理论分析相比，DR 算法在随机图仿真条件下性能表现更好，优于平均时间复杂度.

2）DR 与 AB-DCST 算法对比

AB-DCST 算法模拟蚁群在图中移动，识别候选边集，从而建立度约束最小生成树. 表 1 中 $|V|$ 代表节点个数，生成所用时长均以 s 为单位，可知随着度约束值的逐步增大，DR 与 AB-DCST 算法生成度约束最小生成树的时间均在减少. 在具备同样度约束值的情况下，对于相同节点个数的随机图，DR 算法的生成速度要快于 AB-DCST 算法.

值得注意的是，随着图中节点个数的增加，DR 算法相比于 AB-DCST 算法的性能优势相对在扩大；其他度约束值情况下，也具有相似特征，表明 DR 算法在面对大规模随机图时，具有更好的生成效率.

表1 DR与AB-DCST算法性能比较

$\|V\|$	时长/s($c=3$)		时长/s($c=4$)		时长/s($c=5$)	
	DR	AB-DCST	DR	AB-DCST	DR	AB-DCST
100	1.69	3.62	1.75	3.92	1.53	3.21
200	2.34	5.17	2.27	4.95	2.34	4.08
600	6.15	12.98	5.19	10.81	5.37	10.32
1 000	9.06	20.44	8.73	19.35	8.16	17.50

3）DR与HSSGA算法对比

HSSGA为混合式的构建算法. 表2所示为DR与HSSGA在度约束为3、4、5情况下的消耗时间.

由表2可知,伴随度约束的宽松,2种算法下生成DCMST的时长都在逐步减少. 对于相同节点个数和相同度约束,比较二者消耗时长,发现DR算法优于HSSGA算法. 此外,需要注意的是,在相同度约束下,随着节点数的增大,DR算法相比于HSSGA算法的性能优势在逐步降低.

表2 DR与HSSGA算法性能比较

$\|V\|$	时长/s($c=3$)		时长/s($c=4$)		时长/s($c=5$)	
	DR	HSSGA	DR	HSSGA	DR	HSSGA
100	1.69	3.12	1.75	3.40	1.52	2.98
200	2.35	4.60	2.28	4.34	2.34	3.97
600	6.14	10.55	5.19	9.02	5.37	8.12
1 000	9.06	14.78	8.73	13.51	8.16	11.86

3.2 基于CAIDA的仿真

利用来自CAIDA的真实互联网拓扑对DR算法进行仿真,分别从名为BGP_tables和WHOIS的数据集获得样本拓扑,从2个原始数据集中抽取涵盖前3 000个AS节点的拓扑进行仿真. 将取自BGP_tables的局部拓扑命名为B_tpl,平均度值为2.51;将取自WHOIS的局部拓扑命名为W_tpl,平均度值为6.53.

模拟BGP-LDoS对B_tpl和W_tpl拓扑实施攻击,针对存活节点及链路,分别利用DR算法同ESR、AB-DCST和HSSGA算法生成DCMST,作为域间路由系统遭袭后的恢复拓扑,对生成树节点度值范围也限定为[3,6].

图3给出了B_tpl样本下,度约束范围在[3,6]内使用DR算法和其他3种算法生成DCMST的时间对比. 可知,4种算法下,随着度值约束的放松,DCMST的生成时间整体上均逐步缩减;虽然DR在度约束5和度约束4情形下时间损耗基本持平,但整体趋势仍为缩减. 相比于ESR、AB-DCST和HSSGA算法,DR算法具有相对一定的优势,有效降低了DCMST生成时间.

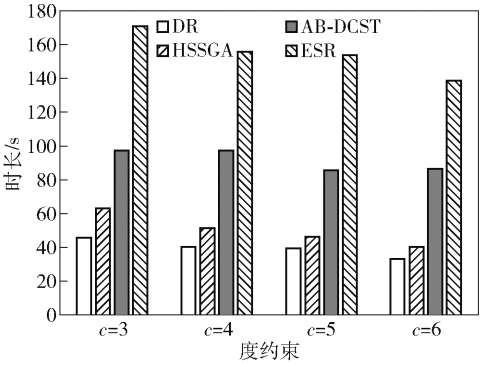

图3 B_tpl样本不同度约束条件下DCMST生成时间对比

图4所示为W_tpl样本下,度约束范围在[3,6]内使用4种算法生成DCMST的时间比较. 同B_tpl样本仿真结果类似,一方面,随着度值约束条件的放松,DCMST的生成时间也在逐步缩减;另一方面,DR算法对比ESR和AB-DCST算法整体具有显著的性能优势;此外,DR相对于HSSGA算法,在度约束值为[3,5]区间内也有较优表现,在度约束为6时,二者差异较小.

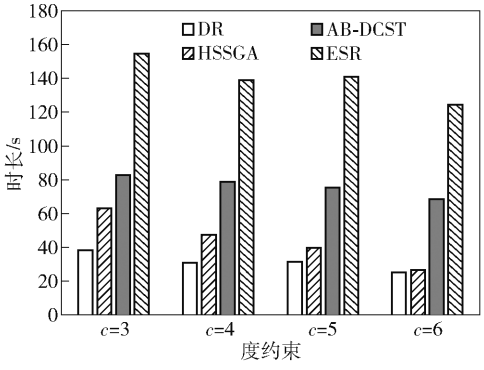

图4 W_tpl样本不同度约束条件下DCMST生成时间对比

4 结束语

基于DCMST的域间路由恢复算法DR,在降低生成恢复拓扑时间复杂度的同时,有效控制节点聚合问题. 为记录拓扑生成时的节点迁移信息,设计了子算法TransferNodesEncode();为从已有生成森林产出新的生成森林,设计了子算法Fundamental-Transfer()/ComplexTransfer();为了判定和计算迁移过程中的关键节点,分别提出KeyNodeSelectionF-

orFT( )/KeyNodeSelectionForCT( ). 理论分析结果表明, DR 计算 DCMST 的平均时间复杂度为 $O(\sqrt{n})$,仿真实验验证了该算法在满足较小度约束条件的同时具有性能上的优越性.

参考文献:

[1] Sermpezis P, Kotronis V, Dainotti A, et al. A survey among network operators on BGP prefix hijacking[J]. ACM SIGCOMM Computer Communication Review, 2018, 48(1): 64-69.

[2] Florian S, Franziska L, Robert B, et al. BGP communities: even more worms in the routing can[C]// Internet Measurement Conference. Boston: ACM Press, 2018: 279-292.

[3] Brivaldo J, Ronaldo A F, Italo C, et al. High-fidelity interdomain routing experiments[C] // ACM SIGCOMM Conference on Posters and Demos. Budapest: ACM Press, 2018: 36-38.

[4] Li Heshuai, Zhu Junhu, Wang Qingxian, et al. LAAEM: a method to enhance LDoS attack[J]. IEEE Communications Letters, 2016, 20(4): 708-711.

[5] Motivala M, Elmore M, Feamster N, et al. Path splicing [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(4): 27-38.

[6] 孙子文, 张炎棋, 徐宜敏. 一种采用混合抗干扰方法的路由恢复机制[J]. 系统仿真学报, 2019, 32(5): 874-884.
Sun Ziwen, Zhang Yanqi, Xu Yimin. A route recovery mechanism using hybrid anti-interference method[J]. Journal of System Simulation, 2019, 32(5): 874-884.

[7] Gopalan A, Ramasubramanian S. IP fast rerouting and disjoint multipath routing with three edge-independent spanning trees[J]. IEEE/ACM Transactions on Networking, 2016, 24(3): 1336-1349.

[8] 陈文龙, 赵一荣, 肖融, 等. 基于虚拟拓扑的多级可信传输体系及路由计算[J]. 计算机研究与发展, 2018, 55(4): 729-737.
Chen Wenlong, Zhao Yirong, Xiao Rong, et al. Packets transmission with multiple levels of credibility and routing calculation based on virtual topologies[J]. Journal of Computer Research and Development, 2018, 55(4): 729-737.

[9] CAIDA. CAIDA's internet topology data comparison[EB/OL]. 2012(2012-05-15)[2019-08-26]. http://www.caida.org/research/topology/topo_comparison.

[10] Raidl G R. Edge sets: an effective evolutionary coding of spanning trees[J]. IEEE Transactions on Evolutionary Computation, 2003, 7(3): 225-239.

[11] Thang N B, Xianghua D, Catherine M Z. An improved ant-based algorithm for the degree-constrained minimum spanning tree problem[J]. IEEE Transactions on Evolutionary Computation, 2012, 16(2): 266-278.

[12] Singh K, Sundar S. A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem[J]. Soft Computing, 2019, 24, 2169-2186.