Advances in Applied Mathematics 应用数学进展, 2023, 12(2), 718-727
Published Online February 2023 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2023.122073

Hans汉斯

度限制条件下最大概率恢复树的路由算法设计

刘银惠，张淑蓉*

太原理工大学数学学院，山西 太原

收稿日期：2023年1月26日；录用日期：2023年2月21日；发布日期：2023年2月28日

摘　要

随着通信网络规模的日益扩大，对通信网络的可靠性提出了更高要求，因此针对提高通信效率的路径设计及其优化问题的研究具有广阔的应用前景。通信网络中存在着各种干扰因素，并且由于通信网络中节点的数据处理能力以及信息交互能力有限，所以与该节点同时进行通信的节点个数也是有限的。基于上述分析，该文章主要考虑节点容量和节点间高效通信的概率，构建点、边赋权的网络模型，并提出度限制条件下的最大概率红蓝恢复树问题，旨在最大化源点与任意汇点间的两条点不交容错路径的传输总概率。在路径优化过程中，利用耳分解和点插入法设计有效多项式算法构造出包含所需路径的最大概率红蓝恢复树，得到满足度限制的高效传输路径。最后，通过实例仿真验证了该算法能够提供通信网络中稳定且具有容错性的有效路径设计方案。

关键词

路径优化算法，点不交路径，红蓝恢复树，耳分解

The Design of the Routing Algorithm for the Degree-Constrained Recovery Tree with the Maximum Probability

Yinhui Liu, Shurong Zhang*

School of Mathematics, Taiyuan University of Technology, Taiyuan Shanxi

Received: Jan. 26th, 2023; accepted: Feb. 21st, 2023; published: Feb. 28th, 2023

Abstract

With the increasing scale of communication network, higher requirements are put forward for the

*通讯作者。

文章引用: 刘银惠, 张淑蓉. 度限制条件下最大概率恢复树的路由算法设计[J]. 应用数学进展, 2023, 12(2): 718-727.
DOI: 10.12677/aam.2023.122073

reliability of communication network. Therefore, the research on path designs and related optimization problems for improving communication efficiency has broad application prospects. There are various interference factors and due to the limited data processing ability and information interaction ability of nodes in the communication network, the number of nodes communicating with one node at the same time is also limited. Based on the above analysis, this paper mainly considers the capacity of nodes and the probability of efficient communication between nodes, and constructs a network model with node and edge weights; then, proposes the maximum probability red/blue recovery trees problem under degree constraints, which is aiming to maximize the total transmission probability of two node disjoint fault tolerant paths between source node and any sink node. In the process of path optimization, ear decomposition and node insertion are used to design an effective polynomial algorithm to construct a maximum probability red/blue recovery trees containing the required path, and an efficient transmission path satisfying the degree constraints is obtained. Finally, the simulation results show that the algorithm can provide a stable and fault-tolerant path design scheme in the communication network.

Keywords

Path Optimization Algorithms, Node Disjoint Paths, Red/Blue Recovery Trees, Ear Decomposition

1. 引言

在通信网络被广泛应用且规模不断增大的背景下，日益扩充的网络规模，使得原本有限的网络容量以及频率资源更加紧张，进而使两者出现难以调和的矛盾。因此，各种频率紧密复用技术应运而生，使得网络容量以及频率利用效率提升的同时，也增大了网络干扰发生的概率。随之带来的网络稳定问题也日益引起重视。在多方面不确定因素的干扰下，特别是网络中节点间距离和网络环境的变化使得在外部干扰下提高成功通信的概率受到关注[1] [2]。文献[2]建立了随机网络最大概率路径问题的数学模型，然后结合该问题的特点，设计了一种改进的蚁群算法，为求解满足一定时间限制下随机网络最大概率路径问题提供了一种快速、可行的方法。故而，本文在通信网络的背景下，主要考虑网络中节点间成功实现通信的概率，并将其以边赋权函数的形式运用于网络模型中。另一方面，由于网络的承载力有限并且其节点的数据处理能力以及容量也是有限的，通过与图中顶点度的限制条件[3] [4]相结合，本文针对网络模型中的节点运用度限制条件，得到了点赋权函数。因此将点、边赋权的无向图作为网络模型。

该模型不仅适用于通信网络，也可应用于量子网络。因为量子网络的通信是基于量子纠缠[5]的成功建立，并且成功率是基于与保真度相关的概率密度函数积分的形式得出[6] [7]，从而得到网络中值为概率的边赋权函数。同时，量子网络中每个节点所含量子位个数也是有限的[8] [9]，为避免资源争用，可将值为量子位个数的度限制函数作用于节点，这样量子网络便可建模为一个点、边赋权的无向图，建立优化问题模型。

大规模网络中，节点故障的发生是不可避免的，故障的存在必然会对网络连通能力造成损害，甚至导致整个网络系统的瘫痪，而被动应对故障干扰是不够的，所以设计高容错性传输路径势在必行[10] [11]，而设计不交路径是提高容错性的重要方法。文献[11] [12] [13] [14]分别研究了不同网络模型以及不同性能约束条件下的不交路设计和优化问题。其中，文献[11]是在图中找两条从不同源点到不同汇点的不交路。

文献[12]则提出了计算 k 条不相交限制最短路径的高效近似算法。

在上述研究的基础上，为同时得到源点和任意节点间的点不交路径，文献[15]在文献[16]和[17]的基础上对构建红蓝恢复树(一对独立生成树)进行了研究，进一步提出了提高网络保护质量(QoP)和网络服务质量(QoS)的快速算法并且具有较低的时间复杂度。近年来，文献[18]在高度可扩展的数据中心网络中并行构造多个独立生成树并应用于数据传输中，文献[19]通过冗余树建立了高效的多路由路径。

基于上述研究，本文在通信网络的背景下，将红蓝恢复树构建、传输概率、节点容量相结合，进一步设计了满足实际需求的有效容错路径。首先，构建点、边赋权的网络模型，当对红蓝树的节点提出度限制条件时，定义最大概率红蓝树的优化问题。其次，先利用点插入法得到源点到其余所有节点的两条最大概率节点不交路径，并在此基础上，再利用耳分解的方法[20]将上述不交路经进行组合优化，设计了在通信网络中构建大概率红蓝树的多项式时间算法。最后，通过实例仿真说明算法得到的满足度限制的大概率红蓝树具有很好的优化效果，能够充分满足实际应用的需求。

本文结构如下：第二节对基本概念进行了阐述并建立了问题模型；第三节提出算法并计算了算法的时间复杂度，同时在第四节中对算法进行实例说明并给出了算法的主要结果；第五节总结全文。

2. 问题提出及模型建立

依据问题研究背景，网络模型用一个点、边赋权的无向图依据问题研究背景，网络模型用一个点、边赋权的无向图 $G=\left(V,E,Q(v),w(e)\right)$ 来表示，其中 $V=\left\{v_i:i=1,2,\cdots,n=|V|\right\}$ 是节点集，图的边集合为 $E=\left\{e_{v_iv_j}:v_i,v_j\in V\right\}$，$Q(v)$ 和 $w(e)$ 分别为非负点、边赋权函数，用以表示每个节点的容量值以及边的高效传输概率。设 $W(G)$ 为 n 阶方阵，其中任意元素 $W[i][j]$ 表示经过 G 中路径从节点 v_i 到节点 v_j 能够成功传输的最大概率值。图 G 中节点 v 的邻点数记为 $d_G(v)$ [21]。

在文献[22]中，Medard 等人提出了称为 MFBG 的方案，即：在网络冗余的前提下，设计根节点为 r 的一对有向生成树 T_r^R 和 T_r^B，满足：对于根节点 r 与任意节点 v，若将两棵树上两点之间的连接路径分别记为 $path\left(r,v,T_r^R\right)$ 和 $path\left(r,v,T_r^B\right)$，则这两条路径是内部节点不相交的。$T_r^R$ 和 T_r^B 分别被称为红树和蓝树。本文根据节点容量有限性，进一步对红蓝树提出“度限制条件”，要求：$d_{T_r^R\cup T_r^B}(v)\le Q(v)$。

下面对于路径的概率计算则利用函数 $w(e)$ 得到计算公式：$P_{r,v}\left(T_r^R\right)=\prod_{e\in E\left(path\left(r,v,T_r^R\right)\right)}w(e)$。同理，路径 $path\left(r,v,T_r^B\right)$ 的概率为：$P_{r,v}\left(T_r^B\right)=\prod_{e\in E\left(path\left(r,v,T_r^B\right)\right)}w(e)$。因此 $P_{r,v}(T_r)=P_{r,v}\left(T_r^R\right)+P_{r,v}\left(T_r^B\right)$ 为红蓝树中从 r 到 v 的两条点不交路径的概率之和，我们希望找到能够使得红蓝树概率和 $P_r(T_r)=\sum_{v\ne r}P_{r,v}(T_r)$ 达到最大值的满足度限制条件的红蓝树设计方案。

接下来，在一个给定的点、边赋权的无向图 $G=\left(V,E,Q(v),w(e)\right)$ 中，本文研究的路径优化问题具体定义如下：

定义 2.1 (MPNDP 问题)给定一个无向图 $G=\left(V,E,Q(v),w(e)\right)$，令 $r\in V$ 为源点，寻找从 r 到其余所有顶点的满足度限制条件的两条概率和最大的点不交路，称为最大概率点不交路(Maximum probability node-disjoint paths (MPNDP))问题。

基于上述问题可进一步研究度限制条件下最大概率红蓝树问题。

定义 2.2 (DMPRBT 问题)给定一个无向图 $G=\left(V,E,Q(v),w(e)\right)$ 和一个根节点 r，在 G 中构建满足度限制条件的红蓝树 T_r^R 和 T_r^B 使得 $P_r(T_r)$ 达到最大的问题，称为度限制条件下最大概率红蓝树(Degree-constrained maximum probability red/blue trees (DMPRBT))问题。

3. 算法设计

在本节中，我们针对上述提出的两个问题，设计了相应算法，其中算法 1 得到根节点 r 到其他点 v 的两条最大概率点不交路，算法 2 得到图 G 的一对根节点为 r 的最大概率红蓝树。

3.1. MPNDP 算法

算法 1 MPNDP 算法

输入 图 $G=(V,E,Q(v),w(e))$ 概率矩阵 W 根节点 r

输出 根节点到其他点的两条最大概率不交路

阶段一：先找到图中根节点到其他点的最大概率路，记为 $path_1(r,v,G)$。

1. 初始化 $d_G(v)=0$，对于每个 $t_i\in V\setminus\{r\}$ 且 $Q(t_i)\geq 1$ 的节点，考虑其余顶点 $k\in V\setminus\{r,t_i\}$ 且 $d_G(k)\leq Q(k)$。

2. 令 $f(i,j)=W[i][j]$。

3. $f(r,t_i)=\max\{f(r,k)\cdot f(k,t_i),f(r,t_i)\}$。

4. $d_G(k)=d_G(k)+2$，$d_G(t_i)=d_G(t_i)+1$，$d_G(r)=d_G(r)+1$。

5. 循环执行以上步骤直到找到所有 r 到 t_i 的最大概率路经，并且记该路径为 $path_1(r,t_i,G)$，同时更新 $Q(v)=Q(v)-d_G(v)$。

阶段二：找图 G 中根节点到其他点除 $path_1(r,t_i,G)$ 外的最大概率路，记为 $path_2(r,t_i,G)$。

6. 令 $G_i=G\setminus\{e_{v_iv_j}:e_{v_iv_j}\in path_1(r,t_i,G)\}$。

7. 在 G_i 中进行阶段一的步骤，找到路径 $path_2(r,t_i,G)$。

MPNDP 算法分为两个阶段。在阶段一中我们先利用类似于 Floyd 算法[22]中插点法和[23]中 Dijkstra 算法的思想找到图 G 中根节点 r 到其余所有点的最大概率路。对于每个非根节点的 $t_i\in V\setminus\{r\}$，我们首先判断该点的度函数限制函数 $Q(t_i)$ 是否大于等于 1，若不是则说明该点没有度去和其它节点建立通信，否则我们进行以下步骤：考虑顶点 $k\in V\setminus\{r\}$，若该点的度 $d_G(k)\leq Q(k)$，我们定义函数 $f(i,j)=W[i][j]$ 表示点 i 到点 j 的路径概率。在步骤 3 中比较 $f(r,k)\cdot f(k,t_i)$ 与 $f(r,t_i)$ 的大小，若前者大于后者，我们将 $f(r,t_i)$ 更新为 $f(r,k)\cdot f(k,t_i)$，否则不更新，并且将该路径上节点的度函数更新。最后，记录找到的 r 到 t_i 的路径为 $path_1(r,t_i,G)$，当找完所有除 r 外的点的 $path_1$ 后阶段一结束。

在阶段二中我们将每个 $t_i\in V\setminus\{r\}$ 对应的路径 $path_1(r,t_i,G)$ 中包含的边在 G 中删去得到一个新图 G_i，注意这里每个 t_i 对应一个 G_i，并且每次都是从 G 中进行删边而非 G_{i-1}。然后在 G_i 中进行阶段一对 t_i 的步骤，从而找到除 $path_1(r,t_i,G)$ 外的“最大概率”路径 $path_2(r,t_i,G)$。

定理 3.1.1 算法 1 的时间复杂度为 $O(n^2)$。

证明假设图 G 有 n 个顶点，即：$|V|=n$。在阶段一中，对于每个顶点 v 作为目标节点 t_i 时，我们考虑其余顶点 $k\in V\setminus\{r,t_i\}$，并比较 $f(r,t_i)$ 与 $f(r,k)\cdot f(k,t_i)$ 的大小，这需要 $O(n^2)$ 的复杂度。在阶段二中，我们在 G_i 中进行与找路过程，同理需 $O(n^2)$ 的复杂度。因此，算法 1 共需 $O(n^2+n^2)$ 的复杂度，即 $O(n^2)$。

3.2. DMPRBT 算法

耳分解是一种对图的分解方法，即将图分解为一个环和若干个耳。图论[24]中耳的定义就是一条极大的路径。在 2-连通图 $G=(V,E,Q(v),w(e))$ 中通过耳分解我们将其可以分解为一系列 2-连通图，即 $G_0\subset G_1\subset\cdots\subset G_k$，其中 $G_0=(V_0,E_0)$，$V_0=\{r\}$，$G_k=(V_k,E_k)$，$V_k=V$。对于每个 $i=1,2,\cdots,k$：

$G_i = G_{i-1} \bigcup ear_i$，其中 ear_i 是一条 x_i 到 y_i 的路径，即 $ear_i = \{x_i - \cdots - v - \cdots - y_i\}$，并且 $x_i, y_i \in V_{i-1}$，$ear_i \bigcap V_{i-1} = \{x_i, y_i\}$，满足这样条件的 ear_i 叫做耳。耳在图中有两种不同形式，当 $x_i = y_i$ 时，耳为一个环；当 $x_i \neq y_i$ 时，耳是一条 x_i 到 y_i 的简单路。例如在第三节中图 1(a)所示的实例中，当 $x_i = y_i$ 时，$ear = \{1-2-4-1\}$ 为一个环；当 $x_i \neq y_i$ 时，例如 $ear = \{2-3-4\}$ 是一条 $x_i = v_2$ 到 $y_i = v_3$ 的简单路。

DMPRBT 算法是在 MPNDP 算法的基础上利用耳分解来构建最大概率红蓝树的过程。首先将 T_r^R 和 T_r^B 设为空集，$V_0 = \{r\}$ 为耳分解第一个子图 G_0 的顶点集。然后运用 MPNDP 算法找到 r 到其余各点的两条节点不交路，分别将两条路按以下规则加入两个红蓝树集合 T_r^R 和 T_r^B。算法步骤 3~7 是将路作为耳添加到红蓝树中的过程，将 i 初始化为 1，当 $V \setminus V_{i-1}$ 不是空集时，我们将 r 到 V 的两条节点不交路分成两个片段，其中我们只考虑未被添加到 T_r^R 和 T_r^B 的部分。如步骤 4 所示，假设 $path_1(r, v, G) = \{v - a_1 - \cdots - a_k - x_i - \cdots - r\}$ (注意，该片段无方向)，我们考虑片段 $\{v - a_1 - \cdots - a_k - x_i\} \subseteq path_1(r, v, G)$，其中 $x_i \in V_{i-1}$，$a_1, \cdots, a_k \notin V_{i-1}$，这意味着片段 $\{x_i - \cdots - r\}$ 已经被添加到了 T_r^R 和 T_r^B。同理，对于 $path_2(r, v, G)$，我们只考虑片段 $\{v - b_1 - \cdots b_l - y_i\}$，其中 $y_i \in V_{i-1}$，$b_1, \cdots, b_l \notin V_{i-1}$。因此，如果片段 $\{x_i - a_k - \cdots - a_1 - v - b_1 - \cdots - b_l - y_i\}$ 满足构成耳的条件，在步骤 6 和 7 中我们将这一对节点不交路添加到 T_r^R 和 T_r^B 中，此时计算分别将两条节点不交路放入不同红蓝树集合中的概率值之和，即：$P_{r,v}^{path_1}(T_r^B) + P_{r,v}^{path_2}(T_r^R)$ 与 $P_{r,v}^{path_1}(T_r^R) + P_{r,v}^{path_2}(T_r^B)$，若前者大于后者，则将片段 $\{x_i \to a_k \to \cdots \to b_l\}$ 添加到 T_r^B 中，而片段 $\{a_k \leftarrow b_l \leftarrow \cdots \leftarrow y_i\}$ 添加到 T_r^R 中；否则我们交换添加顺序如步骤 7 所示。添加结束后我们更新 $G_i = G_{i-1} \bigcup ear_i$ 并且更新 $i = i + 1$ 继续添加下一个耳，直到 $V \setminus V_{i-1}$ 为空集时算法结束。

算法 2 DMPRBT 算法

输入 算法 1 得到的最大概率路集合

输出 一对根节点为 r 的最大概率红蓝树

1. 初始化 T_r^R 和 T_r^B 为空集，$G_0 = (V_0, E_0, Q(v), w(e))$ 其中 $V_0 = \{r\}$，$N(u)$ 为顶点 u 的邻点集。

2. 当 $i = 1$ 时，将 $path_1(r, v, G)$ 和 $path_2(r, v, G)$ 加入 T_r^R 和 T_r^B。

3. 当 $V \setminus V_{i-1}$ 非空时，考虑每个满足 $v \in N(u) \bigcap V \setminus V_{i-1}$ 的点 v，其中 $u \in V_{i-1}$。

4. 令 $v_i = \arg\max\{W[u][v], u \in V \setminus V_{i-1}\}$，片段 $\{v_i - a_1 - \cdots - a_k - x_i\} \subseteq path_1(r, v_i, G)$，其中 $a_1, \cdots, a_k \notin V_{i-1}, x_i \in V_{i-1}$。

片段 $\{v_i - b_1 - \cdots - b_l - y_i\} \subseteq path_2(r, v_i, G)$，其中 $b_1, \cdots, b_l \notin V_{i-1}, y_i \in V_{i-1}$。

5. 如果 $ear_i = \{x_i - a_k - \cdots - a_1 - v_i - b_1 - \cdots - b_l - y_i\}$，我们考虑 $P_{r,v_i}^{path_1}(T_r^B) + P_{r,v_i}^{path_2}(T_r^R)$ 与 $P_{r,v_i}^{path_1}(T_r^R) + P_{r,v_i}^{path_2}(T_r^B)$ 的大小。

6. 若 $P_{r,v_i}^{path_1}(T_r^B) + P_{r,v_i}^{path_2}(T_r^R) > P_{r,v_i}^{path_1}(T_r^R) + P_{r,v_i}^{path_2}(T_r^B)$，我们将片段加入 $\{x_i \to a_k \to \cdots \to b_l\}$ 加入 T_r^B，片段 $\{a_k \leftarrow b_l \leftarrow \cdots \leftarrow y_i\}$ 加入 T_r^R。

7. 若 $P_{r,v_i}^{path_1}(T_r^B) + P_{r,v_i}^{path_2}(T_r^R) < P_{r,v_i}^{path_1}(T_r^R) + P_{r,v_i}^{path_2}(T_r^B)$，我们将片段加入 $\{x_i \to a_k \to \cdots \to b_l\}$ 加入 T_r^R，片段 $\{a_k \leftarrow b_l \leftarrow \cdots \leftarrow y_i\}$ 加入 T_r^B。

8. 更新 $G_i = G_{i-1} \bigcup ear_i$，$i = i + 1$，返回步骤 2 直至 $V \setminus V_{i-1}$ 为空。

定理 3.2.1 算法 2 的时间复杂度为 $O(n)$。

证明假设图 G 有 n 个顶点，即：$|V| = n$。算法 2 需要将算法 1 的结果输入。因此只需计算在添加耳过程中的需要的复杂度。在最坏的情况下，除根节点所在的耳中的节点外，其余每个顶点都属于不同的耳，故至多需要添加 $n-2$ 个耳，需 $O(n)$ 的复杂度。所以，算法 2 共需 $O(n)$ 的复杂度。

4. 算法实例

接下来，如图 1 所示我们给出一个实例，图 1(a)给出 $G=(V,E,Q(v),w(e))$ 的拓扑结构，其中：$Q(v)=\{10,6,8,9,7,4,6,8,5,4\}$ 表示顶点 v 的度函数，概率矩阵 $W(G)$ 如下所示。下面给出算法 1 的运行过程：

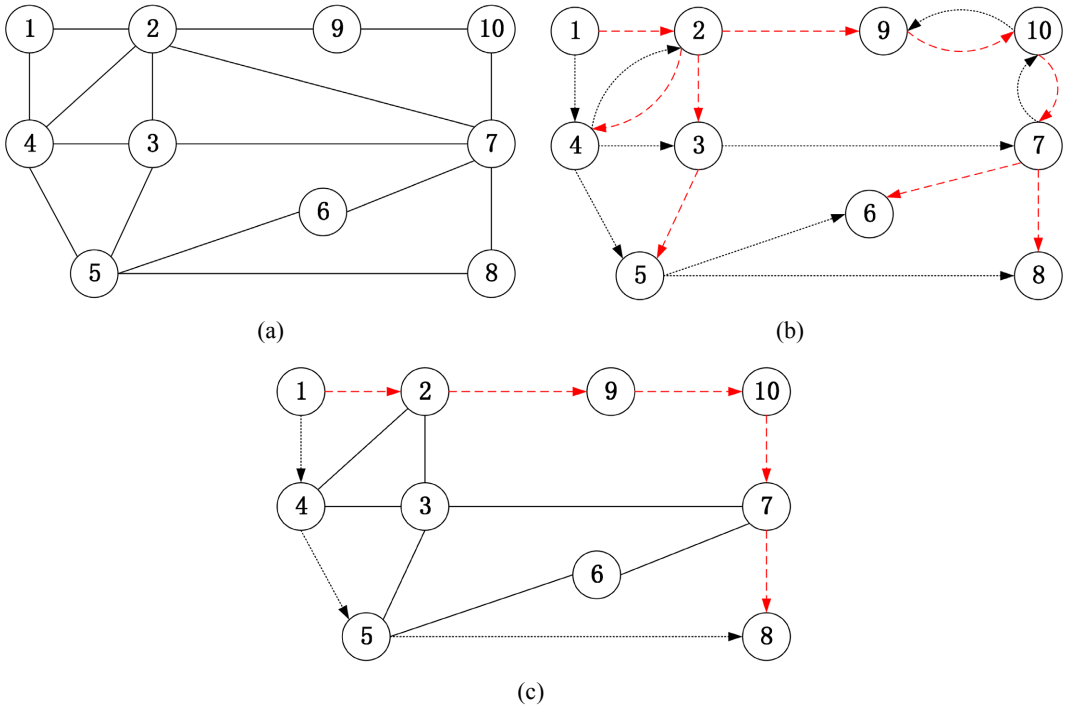

Figure 1. An illustrative example, the root node is v_1, and its probability matrix is $W(G)$. (a) Topology of a graph G; (b) A pair of maximum probability red/blue trees with root node v_1; (c) Two maximum probability disjoint paths from v_1 to v_8 in the red/blue trees

图 1. 说明性示例，根节点为 v_1，其概率矩阵为 $W(G)$。(a) 图 G 的拓扑结构；(b) 根节点为 v_1 的一对最大概率红蓝树；(c) 红蓝树中 v_1 到 v_8 的两条最大概率不交路

$$W(G)=\begin{pmatrix}1&0.6&0&0.7&0&0&0&0&0&0\\0.6&1&0.7&0.6&0&0&0.6&0&0.8&0\\0&0.7&1&0.8&0.7&0&0.8&0&0&0\\0.7&0.6&0.8&1&0.5&0&0&0&0&0\\0&0&0.7&0.5&1&0.9&0&0.9&0&0\\0&0&0&0&0.9&1&0.7&0&0&0\\0&0.6&0.8&0&0&0.7&1&0.6&0&0.9\\0&0&0&0&0.9&0&0.6&1&0&0\\0&0.8&0&0&0&0&0&0&1&0.8\\0&0&0&0&0&0&0.9&0&0.8&1\end{pmatrix}$$

阶段一：对于每个 $t_i\neq r$，我们找 r 到 t_i 的最大概率路。当 $t_i=v_2$ 时，$Q(v_2)=6>1$，因此我们定义 $f(1,2)=W[1][2]=0.6$，随后在根节点 r 到顶点 $t_i=v_2$ 之间插入其它点来更新 $f(1,2)$，更新后我们最终得到 r 到顶点 v_2 的一条最大概率路，即 $path_1(r,v_2,G)=\{1-2\}$，最后更新路上顶点的 $Q(v)$，$Q(v_1)=9$ 且 $Q(v_2)=5$；当 $t_i=v_3$ 时，$Q(v_3)=8>1$，因此定义 $f(1,3)=W[1][3]=0$，同样的我们在根节点 r 到顶点 $t_i=v_3$

之间插入其它点来更新 $f(1,3)$，当 $k=v_2$ 时，进行步骤 3 来更新 $f(1,3)=f(1,2)\cdot f(2,3)=0.6\cdot 0.7=0.42$，当 $k=v_4$ 时，更新 $f(1,3)=f(1,4)\cdot f(4,3)=0.7\cdot 0.8=0.56$，更新后我们最终得到 r 到顶点 v_3 的一条最大概率路，即 $path_1(r,v_3,G)=\{1-4-3\}$，其余点也是同样的运行过程……

下面我们给出 r 到各点的最大概率路经及其概率如表 1 所示：

Table 1. $path_1(r,t_i,G)$ and its probability

表 1. $path_1(r,t_i,G)$ 及其概率

$path_1(r,t_i,G)$	$f(r,t_i)$
$path_1(r,v_2,G)=\{1-2\}$	$f(r,v_2)=0.6$
$path_1(r,v_3,G)=\{1-4-3\}$	$f(r,v_3)=0.56$
$path_1(r,v_4,G)=\{1-4\}$	$f(r,v_4)=0.7$
$path_1(r,v_5,G)=\{1-4-3-5\}$	$f(r,v_5)=0.392$
$path_1(r,v_6,G)=\{1-4-3-5-6\}$	$f(r,v_6)=0.353$
$path_1(r,v_7,G)=\{1-4-3-7\}$	$f(r,v_7)=0.448$
$path_1(r,v_8,G)=\{1-4-3-5-8\}$	$f(r,v_8)=0.353$
$path_1(r,v_9,G)=\{1-2-9\}$	$f(r,v_9)=0.48$
$path_1(r,v_{10},G)=\{1-4-3-7-10\}$	$f(r,v_{10})=0.403$

阶段二：在该阶段中，寻找路的过程与阶段一类似，不同的是阶段二要在新图 G_i 中找。当目标节点为 t_i 时，由于阶段一已经找到 r 到 t_i 的最大概率路经 $path_1(r,t_i,G)$，为寻找与阶段一不相交的路径，因此我们将 $path_1(r,t_i,G)$ 中包含的边在图 G 中删去，从而得到新图 G_i。最后，在新图 G_i 中我们重复阶段一的找路过程得到次最大概率路 $path_2(r,t_i,G)$。下面我们给出 r 到各点的次最大概率路经及其概率如表 2 所示。

Table 2. $path_2(r,t_i,G)$ and its probability

表 2. $path_2(r,t_i,G)$ 及其概率

$path_2(r,t_i,G)$	$f'(r,t_i)$
$path_2(r,v_2,G)=\{1-4-2\}$	$f'(r,v_2)=0.42$
$path_2(r,v_3,G)=\{1-2-3\}$	$f'(r,v_3)=0.42$
$path_2(r,v_4,G)=\{1-2-4\}$	$f'(r,v_4)=0.36$
$path_2(r,v_5,G)=\{1-2-4-5\}$	$f'(r,v_5)=0.18$
$path_2(r,v_6,G)=\{1-2-7-6\}$	$f'(r,v_6)=0.252$
$path_2(r,v_7,G)=\{1-2-7\}$	$f'(r,v_7)=0.36$
$path_2(r,v_8,G)=\{1-2-7-8\}$	$f'(r,v_8)=0.216$
$path_2(r,v_9,G)=\{1-4-3-7-10-9\}$	$f'(r,v_9)=0.323$
$path_2(r,v_{10},G)=\{1-2-9-10\}$	$f'(r,v_{10})=0.384$

在算法 1 结束后，我们便找到了 r 到其余目标节点的两条最大概率不交路，随后，我们进行算法 2 添加耳朵的过程。首先初始化 $V_0=\{r\}$，当 $i=1$ 时，进行下面的步骤 2~7 的循环过程，此时 $V_{i-1}=V_0$，$u=\{1\}$，$N(u)=\{2,4\}$，$W[1][2]=0.6$，$W[1][4]=0.7$，所以 $v_i=v_4$ 且片段 $\{1-4\}\subseteq path_1(r,v_4,G)$，片段 $\{1-2-4\}\subseteq path_2(r,v_4,G)$。这时，$ear_1=\{1-2-4-1\}$ 构成耳，故将片段 $\{1-4-2\}$ 加入 T_r^B，片段 $\{1-2-4\}$ 加入 T_r^R，然后将 ear_1 加入 G_0 构成 G_1；当 $i=2$ 时，$V\setminus V_1=\{3,5,6,7,8,9,10\}$，$v_i=v_3$ 且片段 $\{4-3\}\subseteq path_1(r,v_3,G)$，片段 $\{2-3\}\subseteq path_2(r,v_3,G)$。这时，$ear_2=\{4-3-2\}$ 构成耳，并且 $P_{r,v_3}^{path_1}(T_r^B)+P_{r,v_3}^{path_2}(T_r^R)=P(1-4-3)+P(1-2-3)=0.56+0.42=0.98$，而且我们得到：$P_{r,v_3}^{path_1}(T_r^R)+P_{r,v_3}^{path_2}(T_r^B)=P(1-2-4-3)+P(1-4-2-3)=0.288+0.294=0.582$，故将片段 $\{4-3\}$ 加入 T_r^B，片段 $\{2-3\}$ 加入 T_r^R，然后将 ear_2 加入 G_1 构成 G_2；当 $i=3$ 时，$V\setminus V_2=\{5,6,7,8,9,10\}$，$v_i=v_9$ 且片段 $\{2-9\}\subseteq path_1(r,v_9,G)$，片段 $\{3-7-10-9\}\subseteq path_2(r,v_9,G)$。这时，$ear_3=\{2-9-10-7-3\}$ 构成耳并且 $P_{r,v_9}^{path_1}(T_r^B)+P_{r,v_9}^{path_2}(T_r^R)=P(1-4-2-9)+P(1-2-3-7-10-9)=0.336+0.242=0.578$，$P_{r,v_9}^{path_1}(T_r^R)+P_{r,v_9}^{path_2}(T_r^B)=P(1-2-9)+P(1-4-3-7-10-9)=0.48+0.323=0.803$。故将片段 $\{3-7-10-9\}$ 加入 T_r^B，片段 $\{2-9-10-7\}$ 加入 T_r^R，然后将 ear_3 加入 G_2 构成 G_3……继续添加耳直至 $V\setminus V_{i-1}$ 为空集算法停止，最终得到如图 1(b)的红蓝树。红蓝树的构成过程如表 3 所示，并且我们给出在红蓝树中根节点 r 到其余目标节点的概率矩阵 $W(T_r^R)$ 和 $W(T_r^B)$。

$$W(T_r^R)=\begin{pmatrix}1&0.6&0.42&0.36&0.294&0.252&0.36&0.216&0.48&0.384\\0.6&1&0.7&0.6&0.49&0.42&0.6&0.36&0.8&0.64\\0.42&0.7&1&0.42&0.7&0.294&0.42&0.252&0.56&0.448\\0.36&0.6&0.42&1&0.294&0.252&0.36&0.216&0.48&0.384\\0.294&0.49&0.7&0.294&1&0.206&0.294&0.176&0.392&0.314\\0.252&0.42&0.294&0.252&0.206&1&0.7&0.42&0.504&0.63\\0.36&0.6&0.42&0.36&0.294&0.7&1&0.6&0.72&0.9\\0.216&0.36&0.252&0.216&0.176&0.42&0.6&1&0.432&0.63\\0.48&0.8&0.56&0.48&0.392&0.504&0.72&0.432&1&0.8\\0.384&0.64&0.448&0.384&0.314&0.63&0.9&0.63&0.8&1\end{pmatrix}$$

$$W(T_r^B)=\begin{pmatrix}1&0.42&0.56&0.7&0.35&0.245&0.448&0.315&0.323&0.403\\0.6&1&0.48&0.6&0.3&0.27&0.384&0.27&0.276&0.346\\0.42&0.48&1&0.8&0.4&0.36&0.8&0.36&0.576&0.72\\0.36&0.6&0.8&1&0.5&0.45&0.64&0.45&0.46&0.576\\0.294&0.3&0.4&0.5&1&0.9&0.32&0.9&0.23&0.288\\0.252&0.27&0.36&0.45&0.9&1&0.288&0.81&0.207&0.26\\0.36&0.384&0.8&0.64&0.32&0.288&1&0.288&0.72&0.9\\0.216&0.27&0.36&0.45&0.9&0.81&0.288&1&0.207&0.26\\0.48&0.276&0.576&0.46&0.23&0.207&0.72&0.207&1&0.8\\0.403&0.346&0.72&0.576&0.288&0.26&0.9&0.26&0.8&1\end{pmatrix}$$

以第三节的例子为例分析我们的计算结果，由于我们算法的目的是最大化 $P_r(T_r)=\sum_{v\neq r}P_{r,v}(T_r)$，当 $r=1$ 时 $P_1(T_1)=\sum_{v\neq r}P_{1,v}(T_1)=1.02+0.98+1.06+0.644+0.497+0.808+0.531+0.803+0.787=7.13$；而在例子中，显然可以得到以下结果：

$P_1'(T_1)=\sum_{v\neq r}P_{r,v}'(T_1)=1.2+1.12+1.4+0.7+0.504+0.896+0.62+0.96+0.806=8.206>7.13$。

因此，$P_r'(T_r)>P_r(T_r)$，而且两者差距较小，所以算法输出结果接近问题最优解，同时低复杂性使其具有更好的实际应用价值。

Table 3. The composition of red/blue trees
表 3. 红蓝树的构成

ear_i	T_r^B	T_r^R
$ear_1=\{1-2-4-1\}$	$\{1-4-2\}$	$\{1-2-4\}$
$ear_2=\{4-3-2\}$	$\{4-3\}$	$\{2-3\}$
$ear_3=\{2-9-10-7-3\}$	$\{3-7-10-9\}$	$\{2-9-10-7\}$
$ear_4=\{3-5-4\}$	$\{4-5\}$	$\{3-5\}$
$ear_5=\{5-6-7\}$	$\{5-6\}$	$\{7-6\}$
$ear_6=\{5-8-7\}$	$\{5-8\}$	$\{7-8\}$

5. 结语

本文首先研究源点与所有点之间的两条满足度限制(节点容量限制)下最大概率点不交路；其次，进一步研究构建度限制下最大概率红蓝树问题，以此来提高网络通信的传输可靠性。度限制条件的运用显著提升了高概率路径设计问题的复杂性，本文研究发现，通过应用插点法和耳分解的技术可以给出高效的问题求解算法，复杂度分别为$O(n^2)$和$O(n)$。通过算法实例对算法的具体思路与步骤进行详解，经实际分析和计算进一步验证了所得路径设计方案接近最优解，体现了算法的有效性。

基金项目

山西省自然科学基金“时变网络信道容错路径优化算法研究”(202103021224058)。

参考文献

[1] Nie, Y. and Wu, X. (2009) Shortest Path Problem Considering on-Time Arrival Probability. *Transportation Research Part B*: *Methodological*, **43**, 597-613. https://doi.org/10.1016/j.trb.2009.01.008

[2] 周光发, 陈亮. 随机网络最大概率路径问题的模型与算法[J]. 解放军理工大学学报(自然科学版), 2016, 17(4): 391-395.

[3] Ribeiro, C.C. and Souza, M.C. (2002) Variable Neighborhood Search for the Degree-Constrained Minimum Spanning Tree Problem. *Discrete Applied Mathematics*, **118**, 43-54. https://doi.org/10.1016/S0166-218X(01)00255-4

[4] Chan, T.M. (2003) Euclidean Bounded-Degree Spanning Tree Ratios. *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, San Diego, 8-10 June 2003, 11-19. https://doi.org/10.1145/777792.777795

[5] Clegg, B. (2006) The God Effect: Quantum Entanglement, Science's Strangest Phenomenon. Macmillan Publishers, New York.

[6] Gyongyosi, L. and Imre, S. (2019) Entanglement Access Control for the Quantum Internet. *Quantum Information Processing*, **18**, Article No. 107. https://doi.org/10.1007/s11128-019-2226-5

[7] Victora, M., Krastanov, S., de la Cerda, A.S., Willis, S. and Narang, P. (2020) Purification and Entanglement Routing on Quantum Networks. ArXiv Preprint arXiv: 2011.11644.

[8] Shi, S. and Qian, C. (2020) Concurrent Entanglement Routing for Quantum Networks: Model and Designs. *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications*, *Technologies*, *Architectures*, *and Protocols for Computer Communication*, Virtual Event, 10-14 August 2020, 62-75. https://doi.org/10.1145/3387514.3405853

[9] Zhang, S., Shi, S., Qian, C. and Yeung, K.L. (2021) Fragmentation-Aware Entanglement Routing for Quantum Networks. *Journal of Lightwave Technology*, **39**, 4584-4591. https://doi.org/10.1109/JLT.2021.3070859

[10] Suurballe, J.W. and Tarjan, R.E. (1984) A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Networks*, **14**, 325-336. https://doi.org/10.1002/net.3230140209

[11] Shiloach, Y. and Perl, Y. (1978) Finding Two Disjoint Paths between Two Pairs of Vertices in a Graph. *Journal of the ACM*, **25**, 1-9. https://doi.org/10.1145/322047.322048

[12] Guo, L., Liao, K., Shen, H. and Li, P. (2015) Efficient Approximation Algorithms for Computing k Disjoint Restricted Shortest Paths. *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, Portland, 13-15 June 2015, 62-64. https://doi.org/10.1145/2755573.2755608

[13] Enyedi, G. and Rétvári, G. (2010) Finding Multiple Maximally Redundant Trees in Linear Time. *Periodica Polytechnica Electrical Engineering* (*Archives*), **54**, 29-40. https://doi.org/10.3311/pp.ee.2010-1-2.04

[14] Narvaez, P., Siu, K.Y. and Tzeng, H.Y. (2000) New Dynamic Algorithms for Shortest Path Tree Computation. *IEEE/ACM Transactions on Networking*, **8**, 734-746. https://doi.org/10.1109/90.893870

[15] Zhang, W., Xue, G., Tang, J. and Thulasiraman, K. (2008) Faster Algorithms for Construction of Recovery Trees Enhancing QoP and QoS. *IEEE/ACM Transactions on Networking*, **16**, 642-655. https://doi.org/10.1109/TNET.2007.900705

[16] Xue, G., Chen, L. and Thulasiraman, K. (2003) Quality-of-Service and Quality-of-Protection Issues in Preplanned Recovery Schemes Using Redundant Trees. *IEEE Journal on Selected Areas in Communications*, **21**, 1332-1345. https://doi.org/10.1109/JSAC.2003.816597

[17] Xue, G., Chen, L. and Thulasiraman, K. (2002) QoS Issues in Redundant Trees for Protection in Vertex-Redundant or Edge-Redundant Graphs. 2002 *IEEE International Conference on Communications*. *Conference Proceedings*. *ICC* 2002, New York, 28 April-2 May 2002, 2766-2770.

[18] Yang, J.-S., Li, X.-Y., Peng, S.-L. and Chang, J.-M. (2022) Parallel Construction of Multiple Independent Spanning Trees on Highly Scalable Datacenter Networks. *Applied Mathematics and Computation*, **413**, Article ID: 126617. https://doi.org/10.1016/j.amc.2021.126617

[19] Tapolcai, J., Rétvári, G., Babarczi, P. and Bérczi-Kovács, E.R. (2019) Scalable and Efficient Multipath Routing via Redundant Trees. *IEEE Journal on Selected Areas in Communications*, **37**, 982-996. https://doi.org/10.1109/JSAC.2019.2906742

[20] Korte, B.H. and Vygen, J. (2011) Combinatorial Optimization: Theory and Algorithms. Springer, Heidelberg.

[21] Bondy, J.A. and Murty, U.S.R. (1976) Graph Theory with Applications. Macmillan, New York.

[22] Medard, M., Finn, S.G., Barry, R.A. and Gallager, R.G. (1999) Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs. *IEEE/ACM Transactions on Networking*, **7**, 641-652. https://doi.org/10.1109/90.803380

[23] Dijkstra, E.W. (1959) A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, **1**, 269-271. https://doi.org/10.1007/BF01386390

[24] Warshall, S. (1962) A Theorem on Boolean Matrices. *Journal of the ACM*, **9**, 11-12. https://doi.org/10.1145/321105.321107