arXiv:2601.03934v1 [cs.DS] 7 Jan 2026

快速重路由网络中完美弹性与理想韧性验证的复杂性

Matthias Bentert[1,2],Esra Ceylan-Kettler[1,3],Valentin H¨ubner[3], StefanSchmid[1,4], 和 Jiˇr´ı Srba[5]

[1]柏林工业大学，德国[2]卑尔根大学，挪威[3]奥地利科学技术研究院，奥地利[4]弗劳恩霍夫SIT研究所，德国[5]奥尔堡大学，丹麦

摘要

为了在链路故障后实现快速恢复，大多数现代通信网络都采用完全去中心化的快速重路由机制。这些重新路由机制依赖于预先安装在节点（路由器）上的静态重路由规则，而这些规则仅取决于本地故障信息，即节点 $incident$ 的故障链路。理想情况下，网络具有 $perfectly\ resilient$：只要在故障发生后，源节点和目的地在底层网络中仍保持物理连通，重新路由规则就能确保数据包始终成功路由至目的地。不幸的是，有些情况下无法实现完美弹性。令人惊讶的是，对于何时以及如何实现完美弹性的算法层面问题，目前所知甚少。

我们研究分析此类本地快速重路由机制的计算复杂度。我们的主要结果是否定的：我们证明，即使只是检查一组静态重路由规则是否能确保完美弹性，也是 coNP 完全的。我们还证明，所谓的 $ideal\ resilience$（文献中常考察的一种较弱弹性概念）同样是 coNP 完全的。此外，我们还研究了该问题的其他基本变体。具体而言，我们证明，当重路由规则具有特定模式（文献中称为 $skipping$ ）时，我们的 coNP 完全性证明同样适用。

另一方面，对于节点无法获知数据包从哪个链路到达（即所谓的输入端口）的情形，我们给出了用于验证完美弹性和综合完美弹性的线性时间算法。

1 引言

通信网络是许多分布式系统的关键骨干，无论这些系统运行于企业内部、数据中心，还是地理上分布于互联网各处。为满足严格的可用性要求，现代网络采用本地快速重路由机制：每个路由器（下文称为节点）都有条件数据包转发规则，具体规则取决于其链路状态。这些规则会提前配置，无需了解可能发生的链路故障，并使路由器能够在发生故障时以完全分布式的方式沿备用链路转发数据包。

然而，即使在多链路故障的情况下，要配置此类条件转发规则以提供高弹性，在算法上仍颇具挑战，因为这些规则只能依赖于本地信息：节点无法获知网络其他部分下游可能发生的额外链路故障。因此，如果不加以谨慎处理，不同节点的本地重新路由规则很容易导致转发环路。

理想情况下，网络及其本地快速重路由机制能够提供完美弹性：本地重新路由规则确保，只要源节点在故障发生后仍与底层网络中的目的地保持连接，数据包就能在网络层上成功路由至目的地。早在 ACM PODC 2012，Feigenbaum 等人 [12] 就给出了一个示例，表明某些网络从根本上无法配置为提供完美弹性。

从积极的一面来看，已知在单链路故障的情况下，只要底层网络保持连通，就总能成功地将分组路由到其目的地 [12]。而且，即使发生多链路故障，有时也能实现完美弹性，例如在外平面图中 [14]。

此外，已知可实现的弹性取决于转发规则所能依赖的具体局部信息。虽然通常假设转发规则可以依据关联链路的状态、数据包的目的地，以及数据包到达该节点时所经过的链路（即所谓的输入端口）来制定，但如果规则还能额外获取数据包的源节点信息，可实现的弹性便会提高。具体而言，Dai 等人 [9] 证明，如果能够考虑数据包的源节点，就总能容忍两条链路故障。

然而，时至今日，我们通常仍不清楚在什么情况下能够实现完美弹性。

本文开启了对局部快速重路由算法配置的计算复杂度的研究。具体而言，我们探讨了能否高效验证网络的韧性：

- 给定一个网络及其局部快速重路由机制，它们是否具备完美弹性？

此外，我们还关注网络能够被高效配置的情形 d.

- 在哪些情形下，能够高效配置具备完美弹性的网络？

本文的主要贡献是证明第一个问题为 coNP 完全。积极的一面是，我们还表明，在一个实际相关情形的子类中，第二个问题可以高效求解。

除了完美弹性，我们还研究了文献中常考虑的一种较弱的韧性概念：理想韧性 [6]。这里假设网络初始时（故障发生前）是 k-链路连通的，也就是说，网络无法通过多达 $k-1$ 次链路故障划分为孤立连通分量。在这种高度连通的情形下，如果路由能够容忍多达 $k-1$ 次链路故障（这意味着故障后网络仍保持连通），则称其为 理想 。本文还表明，给定一个网络及其数据包转发规则，判定这些转发规则是否实现了理想韧性是 coNP 完全的。

1.1 主要贡献

我们证明，在节点知道输入端口（数据包到达所经链路）的标准场景下，验证给定网络及其配置是否具备理想韧性是 coNP-完全问题。即使将路由函数限制为优先级列表这种简单形式（“跳过”），结论仍然成立。对于完美弹性，我们证明了同样的结论；而且即使将范围限制在平面图类中，困难性结果仍然成立。

另一方面，在节点转发数据包时不考虑数据包从哪条链路到达的场景中（即路由是“入端口无关”的），验证给定路由是否完全弹性，以及判定给定图是否允许完全弹性路由，都可以在线性时间内完成。入端口无关路由很有意义，因为它们能减少存储路由表所需的内存。

1.2 其他相关工作

文献中早已对本地快速重路由机制进行了深入研究，详细综述请参阅 Chiesa 等人近期发表的调查报告。 [8] 此外，还存在

针对大多数现代网络协议的弹性路由工业标准，涵盖 IP 网络 [27]、MPLS 网络 [26]、最新的段路由 [2] 以及软件定义网络 [31]。然而，这些标准通常只关注单链路故障，无法提供完美或理想的弹性保证。

网络领域在这一主题上有大量应用型文献 [7, 13,15, 19, 21, 24], ，通常侧重于启发式方法。相比之下，我们的论文关注能够提供形式化弹性保证的算法。

对于故障数量 f有上限的故障场景，已有若干有趣的理论研究。在此背景下，能够容忍 f 次链路故障的本地快速重路由方案称为 f-弹性。Chiesa 等人 [4] 证明，如果图是 3-链路连通的，则始终可以实现 2-弹性路由；Dai 等人 [9] 证明，如果重路由规则除了目标节点之外，还可以依赖数据包的源节点，则始终可以实现 2-弹性。在他们发表于 APOCS 2021 的论文中，Foerster 等人 [14] 证明，构成外平面图的网络拓扑始终是完全弹性的，并且支持基于跳过的简单高效重路由算法（每个节点都会为每个输入端口存储一个有序的备用链路优先级列表，遇到故障链路时只需跳过即可）。虽然跳过机制可以生成紧凑的路由表，但一个尚未解决的问题是：仅限于跳过机制的重路由算法是否会以弹性降低为代价 [6, 14]。

在存在完全弹性网络配置的情况下，也可以使用 SyPer [18] 和 SyRep [17], 等近期工具，通过带量化的二叉决策图自动生成这些配置。这也意味着，生成此类规则这一问题的复杂性属于 PSPACE。

理想韧性是否总能实现仍是一个开放的研究问题，但已有研究证明，对于所有 $k \leq 5$ [6]，理想韧性都可以实现。已知对于一般的 k [4]，至少可以容忍 $\lfloor \frac{k}{2} \rfloor - 1$ 次故障。此外，研究还证明，对于团、完全二部图、超立方体或 Clos 网络等特殊图，以及路由规则还可以依赖源节点 [6]的情形，都可以容忍 $k-1$ 次故障。相应算法基于底层图的树枝分解，其中所有树形分支均以目标节点为根。沿着某个树形分支传输的数据包若遇到故障链路，就会按照循环顺序切换到下一个树形分支（这种方法称为循环树枝路由）。该技术基于 Edmond 关于边不相交分支和树枝分解的经典结果 [11], ，并已用于许多论文 [8]。

研究人员也已探讨了局部快速重路由模型的各种变体。具体而言，本文聚焦于确定性算法，但也已有关于随机化算法的研究：在这类模型中，路由器可以生成随机数，或对数据包头进行哈希。Chiesa 等人在其 ICALP 2016 论文中研究了 [3] 链路连通图 k，并证明一种简单的随机化算法不仅能实现较高的鲁棒性，还能产生较短的路径（按跳数计算）。Bankhamer 等人在其 DISC 2021 论文中研究了数据中心网络（基于 Clos 拓扑），并证明只要每个节点上的故障数量不超过某个界限，其算法沿故障切换路径实现的拥塞度在渐近意义上最小，且仅相差多重对数对数因子。

1.3 组织结构

本文余下部分安排如下。我们将在第 2 节更正式地介绍模型和符号。本文的主要贡献，即复杂度结果，将在第 3 节给出。最后，我们将在第 4 节总结全文并提出开放问题。

2 定义

我们将网络定义为无向图 $G = (V, E)$，其中 V是一个有限的节点集合， $E \subseteq \binom{V}{2}$ 是一个链路集合。令 局部(v) 表示与节点 $v \in V$ 相连的链路集合。为了模拟向网络中注入数据包，我们还要求存在

每个节点 $v \in V$ 都有自环 $\{v\} \in E$。由于这些自环始终存在，因此我们不会在图中绘出它们。

A 路径 在 G 中是一个节点序列 $(v_1、\ v_2、\ \ldots\ldots、\ v_n)$，满足 $\{v_i, v_{i+1}\} \in E$ 对所有 $1 \le i < n$ 均成立。若两个节点 $u、\ v \in V$ 之间在 G 中存在路径 $(u = v_1、\ v_2、\ \ldots\ldots、\ v_n = v)$，则称它们连通。在本文余下部分，我们只考虑连通图，也就是说，任意两个节点之间都存在路径。

定义 1. *A* 故障场景 $F \subseteq E$ 是一个不包含任何自环的链路子集，即一个满足 $F \cap \{\{v\} \mid v \in V\} = \emptyset$ 的集合。

我们假设自环仅用于模拟数据包的到达，因此不允许它们发生故障。F 中的链路称为故障链路，$E \setminus F$ 中的链路称为活动链路。根据我们的假设，在任何故障场景中，每个节点至少有一条活动边（自环）。

定义 2。 一个转发模式 ρ 是由一组函数构成的：对于每个故障场景 F 和每个节点 v，都有一个函数 ρ_v^F：$\boldsymbol{local}(v) \to \boldsymbol{local}(v)$ 会针对每条链路 $e \in \boldsymbol{local}(v)$ 返回一条活动链路 $e' \in \boldsymbol{local}(v) \setminus F$。

从现在起，我们假设存在一个固定的目标节点 $t \in V$，使得图中任意节点收到的所有数据包都应转发到节点 t。这相当于在分析数据包头时，仅假设进行目的地匹配（这是可能的最弱假设）。

定义 3。 一个转发模式 ρ、一个起始节点 s 以及一组故障链路 $F \subseteq E$，共同确定一个路由 $\pi = (\pi_1, \pi_2, \ldots)$（一个节点序列），如下所示：$\pi_1 = s$ 对于每个 $i \ge 1$，$\rho_{\pi_i}^F(\{\pi_{i-1}, \pi_i\}) = \{\pi_i, \pi_{i+1}\}$（其中 $\pi_0 = s$）。序列在满足 $\pi_j = t$ 的最小索引 j 处停止（此时数据包到达目的节点 t）。

图 1 展示了转发模式和路由的一个示例。

定义 4. 设 $G = (V, E)$ 是一个无向图，$t \in V$ 是给定的目标节点。如果对于每个故障场景 F 以及每个通过某条路径与 t 连通的节点 $v \in V$，由 ρ、v 和 F 确定的路由都是有限的（包含 t），则转发模式 ρ 是完全弹性的。

换言之，如果对于任意故障场景 F 以及在 F 下与 t 连通的任意节点 v，通过自环向节点 v 注入的数据包最终都会送达目标节点 t，则路由 ρ 是完全弹性的。

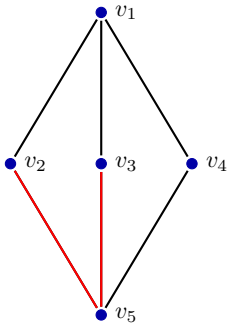
文献中备受关注的完美弹性属性的一个特殊情形是理想韧性 [4, 5]。

定义 5. 设 $G = (V, E)$ 是一个 k-链路连通图（移除任意一组 $k-1$ 条链路都不会使 G 断开）。如果对于大小至多为 $k-1$ 的任意故障场景 F 以及每个节点 $v \in V$，由 ρ、v 和 F 确定的路由都是有限的（包含 t），则转发模式 ρ 是理想韧性的。

显然，如果转发模式 ρ 完全了解全局故障场景，就总能构造出完全/理想弹性的转发模式。由于我们关注的是仅依据局部失效链路信息进行快速重路由的转发模式，因此定义局部转发模式这一概念。

定义 6. 如果对于每个 $v \in V$、每个 $e \in E$ 以及任意两个故障场景 F 和 F'，只要 $F \cap \boldsymbol{local}(v) = F' \cap \boldsymbol{local}(v)$，就有 $\rho_v^F(e) = \rho_v^{F'}(e)$，则称转发模式 ρ 为局部的。

因此，下一跳的选择只能取决于给定节点所关联的活跃链路和故障链路信息。我们也将这种一般的局部转发模式称为组合式，因为节点 v 的局部路由表大小需要 $2^{|local(v)|}$ 条路由表条目，因此会随 v 的度（局部(v) 的大小）呈指数级增长。由于内存需求过高，存储此类转发模式并不实际。为了节省路由表的内存，若干研究考虑了更实用的跳过转发模式概念 [4, 6, 14]。



(a) 含目标节点 $t = v_5$ 的示例图。

node	输入端口	优先级列表
v_1	v_1	v_2, v_3, v_4, v_1
v_1	v_2	v_3, v_4, v_2, v_1
v_1	v_3	v_4, v_2, v_3, v_1
v_1	v_4	v_2, v_3, v_4, v_1
v_2	v_1	v_5, v_1, v_2
v_2	v_2	v_5, v_1, v_2
v_3	v_1	v_5, v_1, v_3
v_3	v_3	v_5, v_1, v_3
v_4	v_1	v_5, v_1, v_4
v_4	v_4	v_5, v_1, v_4

(b) 完全弹性的跳过转发模式；当数据包注入 v_1 时，不存在故障边的路由为 (v_1, v_2, v_5)。

node	输入端口	优先级列表
v_1	v_1	v_2, v_3, v_4, v_1
v_1	v_2	v_3, v_4, v_2, v_1
v_1	v_3	v_4, v_2, v_3, v_1
v_1	v_4	v_2, v_3, v_4, v_1
v_2	v_1	~~v_5~~, v_2
v_2	v_2	~~v_5~~, v_1, v_2
v_3	v_1	~~v_5~~, v_3
v_3	v_3	~~v_5~~, v_1, v_3
v_4	v_1	v_5, v_1, v_4
v_4	v_4	v_5, v_1, v_4

(c) 针对故障场景 $F = \{\{v_2, v_5\}, \{v_3, v_5\}\}$ 的路由为 $(v_1, v_2, v_1, v_3, v_1, v_4, v_5)$。

图 1：具有目标 $t = v_5$ 和完全弹性跳过转发模式的图示例。为便于阅读，我们将与节点 v 相连的链路表示为（此处及后续图中）仅列出该链路的另一个端点。由该转发模式、故障场景 $F = \{\{v_2, v_5\}, \{v_3, v_5\}\}$和起始节点 $s = v_1$确定的路由为 $\pi = (v_1, v_2, v_1, v_3, v_1, v_4, v_5)$。

定义 7。跳过优先级列表是一个函数 $\rho_v : \mathit{local}(v) \to \mathit{local}(v)$，使得对于每条链路 $e \in \mathit{local}(v)$，函数 $\rho_v(e) = (e_1, e_2, \ldots, e_n)$ 返回 $\mathit{local}(v)$ 中所有链路的一个排列。给定的跳过优先级列表 ρ 确定一个跳过转发模式 ρ，使得 $\rho_v^F(e) = e_i$，其中 $\pi(e) = (e_1, e_2, \ldots, e_n)$，且 i 是满足 $e_i \notin F$的最小索引。

观察 1. 显然，任何跳过转发模式也是局部的（组合式），因为转发决策仅基于关联链路的状态。此外，表示跳过优先级列表 $\rho_v(e)$ 时，我们只需存储 $|局部(v)|$ 条链路。

图 1 展示了一个具有完全弹性跳过转发模式的图示例。我们将优先级列表函数表示为一张路由表：第一列是通过给定输入端口（第二列）接收数据包的节点，第三列则给出相应的优先级列表。例如，路由表的第一行表示路由条目 $\rho_{v_1}(\{v_1, v_1\}) = (\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_1\})$。该示例展示了在没有链路故障以及链路 $\{v_2, v_5\}$和 $\{v_3, v_5\}$ 发生故障时，从节点 v_1 注入数据包的路由情况。由于该转发模式具有完全弹性，在这两种情况下，数据包都能送达目标节点 $t = v_5$。

一种限制更多的路由类型是不考虑输入端口信息来做出路由决策。这在以下定义中形式化表述。

定义 8. 如果对于任意节点 $v \in V$、任意故障场景 F，以及任意两条链路 $e, e' \in \mathit{local}(v)$，转发模式 ρ 入端口无关，则 $\rho_v^F(e) = \rho_v^F(e')$。

因此，使用入端口无关转发模式时，无论数据包从哪个链路（端口）到达节点 v，节点 v 始终会选择相同的下一跳。入端口无关转发模式的内存效率更高，因为它们所需的路由表条目更少。

最后，我们将介绍两个与完美弹性和理想韧性相关的判定问题。在验证问题中，给定一个无向图 $G = (V, E)$、一个目标节点 $t \in V$以及一种局部/跳过转发模式 ρ。任务是判断 ρ 是否具有完美/理想韧性。在综合问题中，我们要判断：对于给定的图 G 和目标节点 t，是否存在一种具有完美/理想韧性的局部（或跳过）转发模式。

3 复杂度结果

我们首先陈述完美韧性验证复杂性的一个显而易见的上界，同时指出一个有趣的事实：得益于著名的 Robertson 和 Seymour [30] 定理，综合问题可以在多项式时间内判定。

定理 2. Perfect Resilience 的验证和 Ideal Resilience的验证属于 *coNP*。
Perfect Resilience的综合属于 *P*。

证明。对于验证问题，我们可以猜测一个故障场景，以及一个与目标节点相连、但其转发模式会产生转发环路的源节点；对于给定的故障场景，可以在多项式时间内检查这一点。因此，如果存在会导致转发环路的故障场景，那么给定的验证问题实例就不是完全弹性的，这表明该问题属于 coNP。

综合问题属于 P，这是直接推论，因为完全弹性网络在取子式后仍保持完全弹性；此外，Robertson 和 Seymour 的开创性结果表明，任何此类性质都可以用有限个禁用子式来刻画。结合一个用于检查图是否包含禁用子式的多项式时间算法，我们知道，必然存在一个多项式时间运行的算法，用于判定在所有节点都可能成为目标节点的情况下是否具有完美弹性。利用根子式（目标节点 t 是一个根），很容易证明，对于给定的图和固定的目标节点，判定其是否具有完美弹性也可以在多项式时间内完成 [28, 29]。

□

然而，用于证明完美弹性验证问题可在多项式时间内求解的论证思路，并未给出具体算法，因为它没有告知我们禁用子式集合，只指出该集合存在。若完全弹性转发模式存在，如何有效地（在多项式时间内）设计出这类模式，仍是一个重大开放问题。

接下来，我们将研究入端口无关完美弹性的复杂性。从实际角度看，这是一个值得关注的问题，因为入端口无关转发模式所需的内存占用更少。

3.1 无输入端口匹配的路由

首先，我们将给出一个图允许采用入端口无关转发模式的充要条件。

引理 3。 设 $G = (V, E)$ 是一个连通图。当且仅当 G的所有简单环的长度都不超过 3时，对于任意给定的目标节点 t，都存在完全弹性的入端口无关局部（或跳过式）转发模式。

证明。对于正向，反设存在矛盾：对于给定的目标节点 t，存在一个入端口无关的完全弹性局部转发模式 ρ，且图 G 包含一个长度为 4 或更长的简单环。该环的存在意味着存在三个不同的节点 s，u，$v \in V \setminus \{t\}$，使得

- $\{s, u\} \in E$和 $\{s, v\} \in E$,
- 存在一条从 u 到 t 的路径，该路径不包含节点 v 或 s，并且
- 存在一条从 v 到 t 的路径，该路径不包含节点 u 或 s。

现在，在一个故障场景中，注入到 s 的数据包必须被转发到 u 或 v，其中节点 s 相邻的所有链路均发生故障，但 $\{s, u\}$ 和 $\{s, v\}$ 除外。不妨假设该数据包被转发到 u。我们进一步扩展该故障场景，使节点 u 相邻的所有链路也发生故障，但 u 和 s 之间的链路除外。这意味着数据包会返回 s；由于转发模式是局部的，无法在节点 u 处感知发生故障的链路，并且我们采用入端口无关

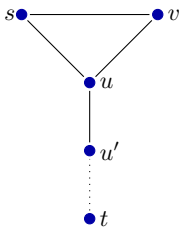
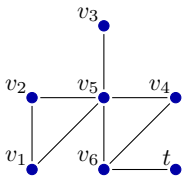


图2：由三个节点 s、v、和 u组成的三角形，其中 u 是所有从 s 或 v 到 t 的路径都经过的唯一节点，而 u' 是从 u 到 t 的最短路径上的第二个节点。



node	优先级列表
v_1	v_5 , v_2 , v_1
v_2	v_5 , v_1 , v_2
v_3	v_5 , v_3
v_4	v_6 , v_5 , v_4
v_5	v_6 , v_4 , v_1, v_2, v_3,v_5
v_6	t, v_4, v_5, v_6

图3：一个图以及针对目标节点 t 的完全弹性入端口无关跳过转发模式。

的转发模式，节点 s 必须将数据包转发回 u。因此形成了一个转发环路；然而，节点 s 仍可通过节点 v 与目标节点 t 连通。这与 ρ 是完全弹性转发模式的假设相矛盾。

在反向方向上，假设图 G 仅包含长度至多为 3 的环。这意味着，从 G 中的任意节点到给定目标节点 t 都存在唯一的最短路径。我们按如下方式构造一个入端口无关的完全弹性跳过（因而也是局部的）转发模式 ρ 。对于不在 G 中任何长度为 3 的简单环上的每个节点 s，我们定义 $\rho(\{*, s\}) = \{s, u\}$，其中 u 是从 s 到 t 的唯一最短路径上的第一个节点，而 $*$ 表示 s 的任意邻居。沿任何其他出链路发送数据包都没有意义，因为数据包必然会返回 s，从而形成转发环路。现在考虑图2所示长度为 3 的环上的三个节点 s、u、$v \in V$。设 u 是唯一一个能够在不经过 s 或 v 的情况下到达目标节点 t 的节点。我们定义 $\rho(\{*, u\}) = \{u, u'\}$，其中 u' 是从 u 到 t 的最短路径上的唯一下一跳节点。与之前一样，如果 ρ 要实现完全弹性，这就是唯一的选择，优先级列表中 $\{u, u'\}$ 之后的链路无关紧要。最后，我们设置 $\rho(\{*, s\}) = \{s, u\}\{s, v\}$ 以及 $\rho(\{*, v\}) = \{v, u\}\{v, s\}$。注意，只要存在从 s 到 t 的路径，每条这样的路径都包含 u。因此，如果链路 $\{s, u\}$ 未发生故障，则向 u 转发始终会朝着到达 t 的方向取得进展。如果该链路发生故障，则从 s 到 t 的所有路径都同时包含 v 和 u，因此向 v 转发也会取得进展。同样的论证适用于从 v 开始的路径，因此所构造的转发模式是完全弹性的，证明完毕。

□

图3展示了一个带有目标节点 t 的图，该图允许采用完全弹性跳过转发模式。只有高亮显示的条目是重要的。由此我们可以得出结论：入端口无感知路由的验证问题和综合问题都可以高效求解。

定理4. 使用入端口无感知的局部路由和跳过路由时，Perfect Resilience 验证问题和综合问题均可在线性时间内判定。

证明。 首先，利用引理 3，我们将检查

算法 1 检验是否存在入端口无感知的完全弹性转发模式。
以图 $G = (V,\ E)$ 和目标节点 t 作为输入；若存在这样的路由，则输出 true；
否则输出 false。

```
过程 SYNTHESIS(V, E, t)
    初始化队列 Q ← [t]
    对所有 v ∈ V 执行
        visited[v] ←假
        parent[v] ←空值
    visited[t] ←真
    当 Q ≠ ∅ 时
        u ← DEQUEUE(Q)
        cycleFound ← 假
        对所有 {u, v} ∈ E 执行
            如果 不是 visited[v]，则
                visited[v] ←为真
                parent[v] ← u
                QUEUE(Q, v)
            else                                  ▷ {u, v} 是一条非树边
                如果 parent[u] ≠ parent[v]，则
                    返回 否
                如果 cycleFound 则                ▷ u 有多条非树边
                    返回 否
                cycleFound ← 为真
    返回 是
```

包含 t 的连通分量中是否存在长度为 4 或更长的环。例如，可以从 t 开始执行略作修改的广度优先搜索，并在线性时间内完成此检查：对于每个节点 v，记录其父节点 p（v）；对于每条不属于广度优先搜索树的链路，检查其两个端点是否具有相同的父节点，并且每个节点是否只与一条这样的链路相连。如果这项线性时间检查未通过，我们就输出不存在完全弹性的局部转发模式。否则，根据引理 3，存在一个入端口无关跳过转发模式 ρ，因此这是一个综合问题的肯定实例。算法 1 是该算法的伪代码实现。

对于验证问题，我们只需检查：对于给定的跳过路由或组合路由，在任意局部故障场景下，重要的优先级列表项是否与 ρ 所做的选择一致。如果这些项与引理 3 所述过程计算出的条目一致，那么给定的转发模式就是完全弹性的。

接下来要证明，只要转发模式的重要条目与定理 3 所述转发模式不一致，它就不是完全弹性的。为此，请注意，节点 v 的转发模式中的每个重要条目都是从 v 到 t 的某条路径上的一个节点。考虑一个故障场景 F_1，在该场景中，假设的转发模式选择了与定理 3 所依据的条目不同的条目。设 u 为定理 3 所选择的节点，u' 为假设解所选择的节点。注意，$u' \neq t$。考虑这样一个故障场景：F_1 中与 v 相连的所有链路均发生故障，而与 u' 相连的链路中，除 $\{v, u'\}$ 外，其余链路均发生故障。此时，转发模式必须将数据包送回初始节点 v，因此会形成转发环路。然而，仍然存在一条从 v 到 t、经过 u 且仅由活动链路组成的路径，这与假设的转发模式能够实现完美弹性相矛盾。 □

3.2 完美韧性验证的困难性

在本节中，我们证明完美弹性的验证问题是 coNP 完全的。这一结果与看似更加困难的综合问题可在多项式时间内判定这一事实形成对比（见定理 2）。

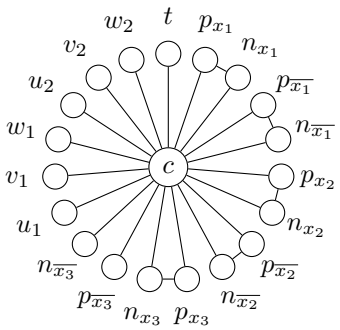
定理 5. 即使输入图是平面的，带有跳过优先级的 PERFECT RESILIENCE 验证问题也是 *coNP* 完全的。

证明。 属于 coNP 已在定理 2 中证明。为证明 coNP 困难性，我们给出从 3-SAT的归约；这是一种 SATISFIABILITY 变体，其中每个子句恰好包含 3 个文字。该问题属于 Karp 最初提出的 21 个 NP 完全问题之一 [22]。我们展示从 3-SAT 到 PERFECT RESILIENCE 验证问题的多项式时间归约，其中，当且仅当原始实例 3-SAT （一个公式 ϕ）不可满足时，所构造的跳过路由才具有完全弹性。

设 $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$和 $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$分别为输入公式 ϕ中的变量集和子句集。我们为带有跳过优先级的 PERFECT RESILIENCE验证问题构造一个实例。首先，我们取两个节点 c 和 t，它们由一条链路相连，其中 t 是目标节点。接下来，对于每个变量 $x_i \in \mathcal{V}$，我们添加四个节点 p_{x_i}、n_{x_i}、p_{x_i}和 n_{x_i}，以及六条链路 $\{c, p_{x_i}\}$、$\{c, n_{x_i}\}$、$\{p_{x_i}, n_{x_i}\}$、$\{c, p_{x_i}\}$、$\{c, n_{x_i}\}$和 $\{p_{x_i}, n_{x_i}\}$。对于每个子句 $C_j \in \mathcal{C}$，我们添加三个节点 u_j、v_j、w_j以及三条链路 $\{c, u_j\}$、$\{c, v_j\}$、$\{c, w_j\}$。构造示例如图 4a 所示。请注意，该图是平面的，且节点和链路的数量与 ϕ中的变量和子句数量成线性关系。接下来，我们给出优先级列表。请注意，如果链路 $\{c, t\}$ 失效，那么只有从 t 出发的数据包才能到达 t。因此，在证明的余下部分中，我们假设这条链路不会失效。于是，我们只需说明 c 从不同输入端口进入时，直到 t的优先级列表。优先级列表见图 4b。在正式陈述证明之前，我们先给出一些总体直观说明。目标是找到一组失效链路，使数据包在 c 与变量节点（p_{x_i}、n_{x_i}、p_{x_i} 和 n_{x_i}）及子句节点（u_j、v_j和 w_j）之间无限循环。首先，如果中心节点 c 曾从节点 p_{x_i}或 p_{x_i} 接收到数据包（其中某个 x_i），就会立即将数据包转发到 t。接下来，我们证明，对于每个 i，在任何解中，链路 $\{c, n_{x_i}\}$ 和 $\{c, n_{x_i}\}$ 恰有一条失效。假设 c 从 $n_{x_{i-1}}$ 或 $n_{x_{i-1}}$ 接收到数据包（或者从 u_m、v_m、w_m 接收数据包，或者数据包从 c 出发，其中 $i = 1$）。那么，链路$\{c, p_{x_i}\}$和$\{c, p_{x_i}\}$不能同时失效，否则数据包会被发送到 t。假设数据包被发送到 p_{x_i} （另一种情况对称）。如果链路 $\{p_{x_i}, n_{x_i}\}$ 或 $\{n_{x_i}, c\}$ 失效，那么数据包就会从 p_{x_i} 发回 c，然后被发送到 t。因此，这两条链路都不会失效，数据包会经由它们转发。现在假设链路 $\{c, n_{\overline{x_i}}\}$ 没有失效。那么，数据包会经由该链路发送。此时，链路 $\{n_{x_i}, p_{x_i}\}$ 和 $\{p_{x_i}, c\}$ 至少有一条必须失效，否则 c 会从 p_{x_i}接收到数据包。因此，数据包会从 n_{x_i} 发回 c。然而，按照构造，数据包现在会被发送到 n_{x_i}，进而到达 p_{x_i}，再返回 c，然后到达 t。因此，在任何解中，链路 $\{c, n_{x_i}\}$和 $\{c, n_{x_i}\}$恰有一条失效，而未失效的那条链路所对应的整个环路也不会失效。我们规定，若链路 $\{c, n_{x_i}\}$ 失效，则将 x_i 设为真；若链路$\{c, n_{x_i}\}$ 失效，则将 x_i 设为假。

现在我们证明该归约是正确的，也就是说，当且仅当 ϕ可满足时，构造的路由不是完全弹性的。正向地，假设存在一个满足赋值 β，使得 ϕ。对于 β赋值为真的每个变量 x_i，我们令链路$\{c, p_{x_i}\}$和$\{c, n_{x_i}\}$发生故障；对于 β赋值为假的每个变量 x_i，我们令链路 $\{c, p_{x_i}\}$和 $\{c, n_{x_i}\}$发生故障。此外，对于每个子句 $C_j = (q_j \vee r_j \vee s_j)$，我们选取一个在 β下满足该子句的变量。如果选中的是 q_j，那么我们令链路 $\{c, v_j\}$和 $\{c, w_j\}$发生故障。如果选中的是 r_j，那么我们令链路 $\{c, u_j\}$和 $\{c, w_j\}$发生故障。如果选中的是 s_j，那么我们令链路 $\{c, u_j\}$和 $\{c, v_j\}$发生故障。现在，对于所有按递增顺序排列的 i，数据包会经过 p_{x_i}和 n_{x_i}，或 p_{x_i} 和 n_{x_i}。然后，对于按递增顺序排列的每个子句 C_j，数据包会前往先前选定的节点（u_j，v_j，或 w_j），并立即返回 c。接着，由于我们有一个满足赋值，因此通往 $n_{q_j}/n_{r_j}/n_{s_j}$ 的链路发生故障，数据包会前往下一个子句。之后，它再次经过所有变量，这个环便会无限循环。因此，构造的路由不是完全弹性的。

反过来，假设构造的路由并非完全弹性，并令 F为一个故障场景，在该场景中，数据包无法从某个源节点到达 t。我们已经证明，对于每个变量 x_i，链路 $\{c, n_{x_i}\}$和 $\{c, n_{x_i}\}$恰有一条发生故障，并说明了这与赋值的对应关系。如果数据包到达某个子句节点 u_j，v_j或 w_j（或者从该处出发，且存在一条通往 c 和 t的路径），那么它就会被发送到 c。接着，c 会尝试将数据包发送到某个节点 $n_{\overline{x_i}}$ 或 n_{x_i}，其中 x_i。如果这条链路没有发生故障（这对应于.....的情况）



（a）为含有三个变量和两个子句的公式构造图 ϕ。

node	输入端口	优先级列表
p_{x_i}	c	n_{x_i}, c
p_{x_i}	n_{x_i}, p_{x_i}	c, n_{x_i}
n_{x_i}	c	p_{x_i}, c
n_{x_i}	p_{x_i}, n_{x_i}	c, p_{x_i}
$p_{\overline{x_i}}$	c	n_{x_i}, c
$p_{\overline{x_i}}$	n_{x_i}, p_{x_i}	c, n_{x_i}
$n_{\overline{x_i}}$	c	p_{x_i}, c
$n_{\overline{x_i}}$	p_{x_i}, n_{x_i}	c, p_{x_i}
u_j	$*$	c
v_j	$*$	c
w_j	$*$	c
c	c	p_{x_1}, p_{x_1}, t
c	p_{x_i}, p_{x_i}	t
c	n_{x_i}	n_{x_i}, $p_{x_{i+1}}$, $p_{x_{i+1}}$, t
c	$n_{\overline{x_i}}$	n_{x_i}, $p_{x_{i+1}}$ $p_{\overline{x_{i+1}}}$ t
c	n_{x_n}	$n_{\overline{x_n}}$ u_1, v_1, w_1, t
c	$n_{\overline{x_n}}$	n_{x_n}, u_1, v_1, w_1, t
c	u_m	n_{q_m}, p_{x_1}, $p_{\overline{x_1}}$, t
c	v_m	n_{r_m}, p_{x_1}, $p_{\overline{x_1}}$, t
c	w_m	n_{s_m}, p_{x_1}, $p_{\overline{x_1}}$, t
c	u_j	n_{q_j}, u_{j+1}, v_{j+1}, w_{j+1}, t
c	v_j	n_{r_j}, u_{j+1}, v_{j+1}, w_{j+1}, t
c	w_j	n_{s_j}, u_{j+1}, v_{j+1}, w_{j+1}, t

(b) 所有节点（目标 t除外）的优先级列表表。符号 $*$表示优先级列表不依赖于输入端口的情况。我们分别用 q_j, r_j, 和 s_j 表示子句 C_j中的三个文字。例如，如果子句为 $C_0 = (x \vee y \vee z)$，那么 $q_0 = x$、$r_0 = y$ $s_i = \bar{z}$ c 和 。节点的优先级列表仅显示到目标 t为止。

图 4：定理 5 证明中实例构造的示例。

子句 C_j 未被相应变量的赋值满足时，数据包会被发送到相应的节点 n_y，随后发送到 p_y，再从 p_y 返回 c；只要链路 $\{c, n_y\}$ 未发生故障，整个环路就不会发生故障（如上所示）。如果通往子句构件的三条链路全部发生故障，那么数据包会直接发送到 t，因此我们可以假设其中至少有一条链路未发生故障。第一条未发生故障的链路将编码该子句对应的满足赋值。随后，数据包会被发送到通往下一个子句的三个节点之一，环路继续运行。只有当所选赋值满足所有子句时，这个环才能无限持续，也就是说，公式 ϕ是可满足的。至此，证明完成。我们注意到，即使已知数据包的源节点，困难性结果仍然成立；例如，将数据包注入 c 就足以证明 coNP 困难性。

□

3.3 理想韧性验证的困难性

接下来我们讨论理想韧性。我们还证明了该问题是 coNP 完全的，但这一次无法证明其在平面图这一特殊情形下的困难性。注意，任何 6-连通图都没有度小于六的节点。因此，这类图都不是平面图，因为平面图是 5-退化的，这意味着每个子图（包括整个图）都至少有一个度不超过

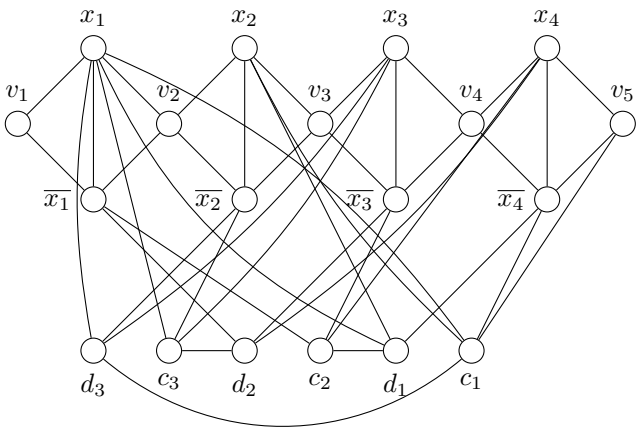


图 5：定理 6 证明中针对该公式构造的图

$$\phi = (x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee x_3)$$

不包括团 K 和目标节点 t。

至多 5。由于对于每个常数 k，验证问题都可以在多项式时间内求解，因此如果输入图是平面图，针对 Ideal Resilience的验证问题实际上可以在多项式时间内求解。

定理 6. 带有跳过优先级的 Ideal Resilience 验证问题是 *coNP* 完全的。

证明。 由于定理 2 已指出该问题属于 coNP，我们重点讨论其困难性部分。我们再次给出一个从 3-Sat出发的归约，其中节点数与输入公式中的变量和子句数量成线性关系；构造出的转发模式理想地具有韧性，当且仅当输入公式不可满足。

设 ϕ 是一个含有 $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$ 个变量和 $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ 个子句的公式。我们首先构造一个由 $2n + 2m + 1$ 个节点 u_0、u_1、.....、u_{2n+2m} 组成的团 K。下文中的所有节点都将与 K 中的所有节点相连。接下来，对于每个变量 $x_i \in \mathcal{V}$，我们添加目标节点 t、三个节点 v_i、x_i 和 $\overline{x_i}$；对于每个子句 $C_j \in \mathcal{C}$，添加两个节点 c_j 和 d_j 。我们还添加一个节点 v_{n+1}。除与 K 的链路外，对于每个变量 x_i，我们添加以下链路：$\{v_i, x_i\}$、$\{v_i, x_i\}$、$\{x_i, x_i\}$、$\{x_i, v_{i+1}\}$、$\{x_i, v_{i+1}\}$。对于每个子句 C_j，设 p_j、q_j、r_j 为出现在 C_j 中的三个文字。我们添加链路 $\{c_j, p_j\}$、$\{c_j, q_j\}$、$\{c_j, r_j\}$、$\{d_j, p_j\}$、$\{d_j, q_j\}$、$\{d_j, r_j\}$。最后，对于每个 $j < m$，我们添加链路 $\{d_j, c_{j+1}\}$ 和 $\{c_{j+1}, d_j\}$，并添加链路 $\{d_m, c_1\}$ 和 $\{c_1, d_m\}$。构造出的网络示例见图 5。接下来，我们给出优先级列表，并将其列于图 6 中。注意， K 确保该图是 $(2n + 2m + 1)$-连通的，而 t 恰好有 $2n + 2m + 1$ 条关联链路。因此，我们假设最多有 $k = 2n + 2m$ 条链路发生故障，并且每个优先级列表中只展示前 $k + 1$ 个条目。在正式陈述证明之前，我们先给出一些高层次的直观解释。稍后我们将证明，如果数据包曾到达 K 中的某个节点，那么它就会到达 t。因此，若要使数据包无法到达 t，它必须从节点 v_1 开始，因为其他所有节点都会立即将其发送到 K 中的某个节点（假设最多有 k 条链路发生故障）。接下来，我们将证明，对于每个 $i \in [n]$，如果数据包到达 v_i，那么唯一不会将其发送到 K 中某个节点的情形，就是链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$，或链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$ 发生故障。在这两种情况下，数据包都会到达 v_{i+1}，而与 v_i 关联的哪条链路发生故障，将编码变量 x_i 在某个解中的赋值。更具体地说，在第一种情况下，我们称 x_i 被设为 true，在第二种情况下称其被设为 false。一旦数据包到达 v_{n+1}，它就会被发送到 c_1 （或发送到 K，随后再发送到 t）。现在，对于每个 c_j，如果分别连接到 p_j、q_j 和 r_j 的三条链路均发生故障，则数据包会被发送到 K。因此，这些链路中至少有一条不会发生故障；当数据包被发送到这三个节点中的某个节点时，系统会检查相应变量是否被正确设置，以满足 C_j。如果不是

node	输入端口	优先级列表
u_i	$*$	$t, u_{i+1}, \ldots u_{2n+2m}, u_0, \ldots, u_{i-1}$
v_1	v_1	$x_1, \overline{x_1}, K$
v_1	$*$	K
v_i	x_{i-1}	$\overline{x_{i-1}}, x_i, \overline{x_i}, K$
v_i	$\overline{x_{i-1}}$	$x_{i-1}, x_i, \overline{x_i}, K$
v_i	$*$	K
v_{n+1}	x_n	$\overline{x_n}, c_1, K$
v_{n+1}	$\overline{x_n}$	x_n, c_1, K
v_{n+1}	$*$	K
x_i	v_i	$\overline{x_i}, K$
x_i	$\overline{x_i}$	v_i, v_{i+1}, K
x_i	c_j	v_i, d_j, K
x_i	$*$	K
$\overline{x_i}$	v_i	x_i, K
$\overline{x_i}$	x_i	$v_i, \quad v_{i+1}, \quad K$
$\overline{x_i}$	c_j	$v_i, \quad d_j, \quad K$
$\overline{x_i}$	$*$	K
c_1	$v_{n+1}, \quad d_m$	$p_1, \quad q_1, \quad r_1, \quad K$
c_1	$*$	K
c_j	d_j	$p_j, \quad q_j, \quad r_j, \quad K$
c_j	$*$	K
d_m	$p_m, \quad q_m, \quad r_m$	$c_1, \quad K$
d_m	$*$	K
d_j	$p_j, \quad q_j, \quad r_j$	$c_{j+1}, \quad K$
d_j	$*$	K

图 6：构造中所有节点（目标节点 t 除外）的优先级列表表格，该构造见定理 6 的证明。符号 $*$ 表示表格中其他位置未列出的所有输入端口。我们分别用 $p_i, q_i,$ 和 r_i 表示子句 C_i中的三个文字。例如，如果子句 $C_1 = (x_1 \vee x_2 \vee x_4)$，那么 $p_1 = x_1$、 $q_1 = x_2$和 $r_1 = x_4$。我们仅给出每个优先级列表中的前 $k + 1 = 2n + 2m + 1$ 个条目，而 K 表示优先级列表 $u_0, u_1, \ldots, u_{2n+2m}$。

这种情况，那么数据包会被发送到某个 v_i，然后发送到 K 中的某个节点，随后到达 t。因此，避免这种情况的唯一方法，就是将数据包发送到某个满足相应子句的变量节点。在这种情况下，数据包会被转发到 d_j，然后到达 c_{j+1}。一旦到达节点 d_m（这对应于所选赋值满足整个公式的情形），路由会继续经过 c_1，随后无限重复这一环。

现在我们将论证形式化。首先，我们证明：如果到达了 K 中的某个节点，且至多有 k 条链路发生故障，那么就能到达目标节点 t。令 i 为满足 $u_i \in K$ 已到达的索引，并反设存在一个大小至多为 $k = 2m + 2n$ 的故障场景 F，使得 t 未被到达。根据构造，如果到达了某个节点 $u_j \in K$，且链路 $\{u_j, t\}$ 未发生故障，那么就能到达 t。因此，$\{u_i, t\} \in F$。我们考虑两个索引 a 和 b，初始时 $a = i$ 且 $b = i + 1$。节点 u_a 是 K 中最后一个到达的节点，而 u_b 是下一个将被访问的节点；也就是说，如果 $b > a$，则对于所有 $j \in \{a + 1, a + 2, \ldots, b - 1\}$，链路 $\{u_a, u_j\}$ 都发生故障；否则，对于所有 $j \in \{a + 1, a + 2, \ldots, 2n + 2m, 0, 1, \ldots, b - 1\}$，链路 $\{u_a, u_j\}$ 都发生故障。如果链路 $\{u_i, u_{i+1}\}$ 和 $\{u_{i+1}, t\}$ 均未发生故障，那么就能到达 t。如果 $\{u_i, u_{i+1}\}$ 未发生故障，我们就更新 $u_a = u_b$；否则，保持 u_a 不变。无论哪种情况，如果 $b \neq 2n + 2m$，我们就更新 $u_b = u_{b+1}$；否则更新 $u_b = u_0$。注意，对于每个满足 $u \neq i$ 的节点 u_b，都存在某条满足 $a \in \{i, i + 1, \ldots, b - 1\}$ 的链路 $\{u_a, u_b\}$，如果 $i < b$ 或 $a \in \{i, i + 1, \ldots, 2n + 2m, 0, 1, \ldots, b - 1\}$，或者链路 $\{u_b, t\}$ 发生故障。由于同一条链路不会被计入两个不同的索引 b（如果 a 位于 i 和 b 之间，那么 b 就不可能位于 i 和 a 之间），这就在 F 中产生了 $2n+2m$ 条链路（此外还有 $\{u_i, b\}$），与 $|F| \leq 2n + 2m = k$ 矛盾。

因此，如果到达了 K 中的某个节点，且至多有 k 条链路发生故障，那么就能到达 t 。

最后，为完成证明，请注意，如果 ϕ 可满足，则令 β 为一个满足赋值。对于每个变量 x_i，如果 β 将其设为真，则令链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$ 发生故障；否则，令链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$ 发生故障。接下来，对于每个子句 C_j，令 c_j 与所有在 β 下无法满足 C_j 的文字之间的链路发生故障。由于每个子句的大小为三，且 β 满足 ϕ，因此至多有 $2m$ 个子句会因此失效。现在，如上所示，如果路由从 v_1 开始，则数据包会经过所有变量并到达 v_{n+1}。从那里，它会被转发到 c_1。接下来，对于每个子句，数据包会被转发到一个文字，该文字在 β 下满足 C_j 。请注意，这对应于该文字（x_i 或 x_i）与 v_i 之间的链路发生故障的情况。因此，数据包接着会被发送到 d_j，然后发送到 c_{j+1}（或从 d_m 发送到 c_1）。由于这个环会无限持续，所构造的转发模式并不具备理想韧性。

反过来，如果构造出的转发模式不是理想的弹性模式，那么路由必须从 v_1 开始，并如上所示避开 K 。因此，对于每个变量 x_i ，链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$，或链路 $\{v_i, x_i\}$ 和 $\{x_i, v_{i+1}\}$ 必然发生故障。我们通过在 $\{v_i, x_i\}$ 发生故障时将 x_i 设为真、在 $\{v_i, x_i\}$ 发生故障时将其设为假，为所有变量定义一个赋值。注意，这两条链路不可能同时发生故障，否则 v_i 会将数据包发送到 K中的某个节点。现在，路由要想无限期地继续下去，唯一的可能是数据包经过由所有节点 c_j 和 d_j组成的一个环。在这两个节点之间，数据包必须经过节点 x_i 或 $\overline{x_i}$。该节点从 c_j接收到数据包后，如果链路 $\{x_i, v_i\}$ 未发生故障，就会将数据包发送到 v_i 。随后，节点 v_i 将其发送到 K中的某个节点。因此，链路 $\{x_i, v_i\}$ 发生故障，这对应于在构造的赋值中 C_j 被 x_i 满足的情况。该赋值满足所有子句，因此输入公式 ϕ 是可满足的，证明得证。 □

我们指出，在理想韧性的情形下，高阶节点是必需的，因为任何常度节点都意味着输入图的连通度（因此 k ）为常数，这会使该问题成为多项式时间可解问题。至于能否修改完美弹性的归约，使其具有有界度，我们将其留作一个开放问题。

请注意，前述证明中构造的网络，其节点数量与输入公式中的变量和子句数量呈线性关系。假设一个称为指数时间假设（ETH）的复杂性假设成立，则 3-SAT 无法在 $2^{o(n+m)}$ 时间内求解，其中 n和 m 分别是输入公式中的变量数和子句数。因此，这两个验证问题都无法在 $2^{o(n)}$ 时间内求解。这与通常的结果形成对比：许多关于平面图的问题都能在亚指数时间内（通常为 $2^{\sqrt{n}}$ 时间）求解 [10, 23, 25]。然而，正如我们刚刚证明的那样，对于理想韧性或完美弹性的验证问题，情况出乎意料地并非如此（假设 ETH 成立）。

4 结论与开放问题

能够对链路故障做出本地且即时响应的快速重路由机制，对于当今的可靠计算机网络至关重要。多种快速重路由方案已在几乎所有类型的计算机网络中实现并部署。然而，对于哪些图能够高效构建完全弹性的保护机制这一问题，目前尚无定论。所谓完全弹性，是指只要源节点和目标节点在物理上仍然连通，就能保证在任何故障场景下传递数据包。通过应用 Robertson-Seymour定理 [30] 以及关于存在面向所有可能目标节点和给定目标节点的完全弹性路由的 minor稳定性结果 [16] （后者借助该结果的根化变体 [28, 29]），我们证明了判定给定网络拓扑及目标节点是否允许完全弹性路由保护的精确计算复杂度为多项式时间。这与我们的验证问题在计算上困难这一结果形成对照——我们通过从 3-SAT 进行一次非平凡归约，证明了判定给定固定目标节点的一组路由表是否完全弹性是 coNP 完全问题。

我们还研究了一种局部快速重路由的变体，即入端口无感知路由（也就是说，数据包的入接口不属于路由表的一部分）。这有望实现更快速的故障切换

保护，同时也能以更节省内存的方式存储转发表。在这一受限情形下，我们提出了一种线性时间算法，用于判断给定的入端口无感知路由是否具有完全弹性；我们还证明，综合问题也可在线性时间内判定，从而能够为确实可行的拓扑高效构造入端口无感知路由表。

一个重要的开放问题是，设计一种具体算法，使其能够针对给定的网络在多项式时间内构造完全弹性的路由表（只要这种构造是可行的）。

致谢

本研究部分得到了欧盟地平线2020研究与创新计划下欧洲研究理事会（ERC）的资助（资助协议编号：819416），以及ERC巩固者资助项目 AdjustNet 的资助（协议编号：864228）。

参考文献

[1] Gregor Bankhamer、Robert Els¨asser 和 Stefan Schmid。《面向数据中心网络的随机化局部快速重路由：实现近乎最优的拥塞》。载于第*35*届分布式计算国际研讨会（*DISC*）论文集，第9:1–9:19页。Schloss Dagstuhl - Leibniz- Zentrum f¨ur Informatik，2021年。doi：10.4230/LIPICS.DISC.2021.9。

[2] Ahmed Bashandy、Clarence Filsfils、B Decraene、S Litkowski、P Francois、D Voyer、F Clad 和 P Camarillo。使用段路由实现拓扑无关快速重路由。工作草案，2018年。

[3] Marco Chiesa、Andrei V. Gurtov、Aleksander Madry、Slobodan Mitrovic、Ilya Nikolaevskiy、Michael Schapira 和 Scott Shenker。《随机化路由抵御多重边故障的韧性》。发表于*ICALP*，第134:1–134:15 页，2016 年。

[4] Marco Chiesa、Ilya Nikolaevskiy、Slobodan Mitrovic、Aurojit Panda、Andrei V. Gurtov、Aleksander Madry、Michael Schapira 和 Scott Shenker。《探寻具有韧性的（静态）转发表》。发表于第*35* 届 *IEEE* 计算机通信国际会议（*INFOCOM*）论文集，第 1–9 页。IEEE，2016 年。doi：10.1109/INFOCOM.2016.7524552。

[5] Marco Chiesa、Ilya Nikolaevskiy、Aurojit Panda、Andrei V. Gurtov、Michael Schapira 和 Scott Shenker。《探索静态故障切换路由的极限（v4）》。abs/1409.0034.v4，2016 年。

[6] Marco Chiesa、Ilya Nikolaevskiy、Slobodan Mitrovic、Andrei V. Gurtov、Aleksander Madry、Michael Schapira 和 Scott Shenker。《静态转发表的韧性》。*IEEE/ACM*网络汇刊，25(2):1133–1146，2017年。doi: 10.1109/TNET.2016. 2619398。

[7] Marco Chiesa、Roshan Sedar、Gianni Antichi、Michael Borokhovich、Andrzej Kamisinski、Georgios Nikolaidis 和 Stefan Schmid。《PURR：一种可重配置快速重路由原语：抱最好的希望，为最坏的情况做好准备》。载于第*15*届新兴网络实验与技术国际会议（*CoNEXT*）论文集，第1–14页。ACM，2019年。doi: 10.1145/3359989.3365410。

[8] Marco Chiesa、Andrzej Kamisinski、Jacek Rak、G´abor R´etv´ari 和 Stefan Schmid。《分组交换网络中的快速恢复机制综述》。*IEEE*通信调查与教程，23(2):1253–1301，2021年。doi: 10.1109/COMST.2021.3063980。

[9] Wenkai Dai、Klaus-Tycho Foerster 和 Stefan Schmid。快速故障切换路由的紧致刻画：对两条链路故障的弹性是可实现的。载于第*35*届*ACM*算法与架构并行性研讨会论文集（*SPAA*），第153–163页。ACM，2023年。doi：10.1145/3558481.3591080。

[10] Erik D. Demaine 和 MohammadTaghi Hajiaghayi。双维性理论及其算法应用。《计算机期刊》，51(3)：292–302，2008年。[11] Jack R. Edmonds。边不相交分支。《组合算法》，第91–96页，1973年。[12] Joan Feigenbaum、Brighten Godfrey、Aurojit Panda、Michael Schapira、Scott Shenker 和 Ankit Singla。简短公告：论路由表的弹性。载于*PODC*，第237–238页。ACM，2012年。[13]Klaus-Tycho Foerster、Mahmoud Parham、Marco Chiesa 和 Stefan Schmid。TI-MFA：保持冷静，快速重路由段。载于第*37*届*IEEE*计算机通信会议研讨会论文集（*INFOCOM*），第415–420页。IEEE，2018年。doi：10.1109/INFCOMW.2018.8406885。[14] Klaus-Tycho Foerster、Juho Hirvonen、Yvonne-Anne Pignolet、Stefan Schmid 和 Gilles Tr´edan。关于通过本地快速故障切换实现完美弹性的可行性。载于第*2*届*SIAM*计算机系统算法原理研讨会论文集（*APOCS*），第55–69页。SIAM，2021年。doi：10.1137/1.9781611976489.5。[15] Klaus-Tycho Foerster、Andrzej Kamisinski、Yvonne-Anne Pignolet、Stefan Schmid 和 Gilles Tr´edan。通过嫁接树形分支增强快速重路由方案的弹性。载于*INFOCOM*，第1–10页。IEEE，2021年。[16] Klaus-Tycho Foerster、Juho Hirvonen、Yvonne-Anne Pignolet 和 Stefan Schmid。静态快速重路由中的局部性代价。载于第*52*届*IEEE/IFIP*可信系统与网络国际会议论文集（*DSN*），第215–226页。IEEE，2022年。doi：10.1109/DSN53405.2022.00032。[17] Csaba Gy¨orgyi、Kim G. Larsen、Stefan Schmid 和 Jir´ı Srba。Syrep：面向弹性网络的快速重路由转发表高效综合与修复。载于第*54*届*IEEE/IFIP*可信系统与网络国际会议论文集（*DSN*），第483–494页。IEEE，2024年。doi：10.1109/DSN58291.2024.00053。[18] Csaba Gy¨orgyi、Kim G. Larsen、Stefan Schmid 和 Ji˘r´ı Srba。Syper：为高可信网络综合完全弹性的本地快速重路由规则。载于*IEEE INFOCOM 2024 - IEEE*计算机通信会议，第2398–2407页，2024年。doi：10.1109/INFOCOM52122.2024.10621323。[19] Thomas Holterbach、Stefano Vissicchio、Alberto Dainotti 和 Laurent Vanbever。SWIFT：预测性快速重路由。载于第*31*届*ACM*数据通信特别兴趣组会议论文集（*SIGCOMM*），第460–473页。ACM，2017年。doi：10.1145/3098822.3098856。[20] Russell Impagliazzo、Ramamohan Paturi 和 Francis Zane。哪些问题具有强指数复杂度？《计算机与系统科学杂志》，63(4)：512–530，2001年。[21]Peter Gjøl Jensen、Dan Kristiansen、Stefan Schmid、Morten Konggaard Schou、Bern- hard Clemens Schrenk 和 Jir´ı Srba。AalWiNes：面向MPLS 网络的快速定量假设分析工具。载于第*16*届国际新兴网络实验与技术会议论文集（*CoNEXT*），第474–481页。ACM，2020年。doi：10.1145/3386367.3431308。[22] Richard M. Karp。组合问题之间的可约性。载于《计算复杂性》，第85–103页。Plenum Press，1972年。[23] Philip N. Klein 和 D´aniel Marx。平面图上子集TSP的亚指数参数化算法。载于第*25*届*ACM-SIAM*离散算法研讨会论文集（*SODA*），第1812–1830页。SIAM，2014年。

[24] Junda Liu、Aurojit Panda、Ankit Singla、Brighten Godfrey、Michael Schapira 和 Scott Shenker。通过数据平面机制确保连通性。载于第*10*届*USENIX*网络系统设计与实现研讨会（*NSDI*）论文集，第113–126页。USENIX协会，2013年。[25] Jesper Nederlof。在次指数参数化时间内检测并计数平面图中的小型模式。载于第*52*届*ACM SIGACT*计算理论研讨会（*STOC*）论文集，第1293–1306页。ACM，2020年。[26] Ping Pan、George Swallow 和 Alia Atlas。面向LSP隧道的RSVP-TE快速重路由扩展。*RFC*，4090：1–38，2005年。[27]Jozef Pap´an、Pavel Sege˘c、Marek Morav˘c´ık、Martin Kont˘sek、L'udov´ıt Miku˘s 和 Jana Uramov´a。IP快速重路由解决方案概述。载于国际新兴电子学习技术与应用会议（*ICETA*）论文集，第417–424页，2018年。[28] Neil Robertson 和 Paul D. Seymour。图子式 IV：树宽与良拟序。组合理论杂志*B*辑，48(2)：227–254，1990年。doi：10.1016/0095-8956(90) 90120-O。[29] Neil Robertson 和 Paul D. Seymour。图子式 XIII：不相交路径问题。组合理论杂志*B*辑，63(1)：65–110，1995年。doi：10.1006/JCTB.1995.1006。[30] Neil Robertson 和 P.D Seymour。图子式 VIII：一般曲面上的Kuratowski定理。组合理论杂志*B*辑，(48(2))：255–288，1990年。[31] 交换机规范1.3.1。OpenFlow。载于 *https://bit.ly/2Vj0077*，2013年。