计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(7):72−83 [doi: 10.15888/j.cnki.csa.009918] [CSTR: 32024.14.csa.009918]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

比特币网络节点探测方法的实验评估与改进①

周子铭, 黎文伟

(湖南大学 信息科学与工程学院, 长沙 410082)
通信作者: 黎文伟, E-mail: liww@hnu.edu.cn

摘　要: 探测网络节点组成并分析其特征是提升比特币网络稳定性和安全性的基础. 现有研究主要侧重于网络节点属性分析, 较少关注节点探测方法本身的优化. 现有比特币网络节点探测方法存在时间长、开销大等不足. 本文将现有方法概括为无去重全遍历 (full traversal without deduplication, FTWD) 测量模型并进行大量测量实验评估, 分析了影响其探测时间、开销及准确度的主要因素. 在此基础上, 提出一种改进的比特币网络节点探测方法 BNP (Bitcoin node probe). 该方法通过增加初始轮次种子节点数量、引入轮次间新增节点比例指标、采用部分遍历策略等措施, 减少了探测时间和探测开销, 提升了探测效率. 实验结果表明, 与现有方法比较, BNP 方法在随机选择比例为 50% 时, 虽探测节点总数略有下降, 但探测时间平均减少 40.4%, 探测数据包开销平均减少 21.4%.

关键词: 区块链; 比特币网; 比特币网络节点探测; 方法评估; 方法改进

引用格式: 周子铭,黎文伟.比特币网络节点探测方法的实验评估与改进.计算机系统应用,2025,34(7):72–83. http://www.c-s-a.org.cn/1003-3254/9918.html

Experimental Evaluation and Improvement of Bitcoin Network Node Detection Method

ZHOU Zi-Ming, LI Wen-Wei

(College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China)

Abstract: Detecting the composition of network nodes and analyzing their characteristics is essential for improving the stability and security of the Bitcoin network. Existing research mainly focuses on the analysis of network node attributes, with less attention given to the optimization of the node detection method itself. Existing Bitcoin network node detection methods have limitations, such as long detection time and high overhead. In this study, the existing methods are generalized into a full traversal without deduplication (FTWD) measurement model, which is evaluated through a large number of experimental measurements. The main factors affecting detection time, overhead, and accuracy are analyzed. Based on this, an improved Bitcoin network node probing method, Bitcoin node probe (BNP), is proposed. This method reduces probing time and overhead and improves probing efficiency by increasing the number of seed nodes in the initial rounds, introducing an indicator for the proportion of new nodes added between rounds, and adopting a partial traversal strategy. Experimental results show that, compared to existing methods, the BNP method reduces probing time by 40.4% on average and probing packet overhead by 21.4% on average when the random selection ratio is 50%, although the total number of probing nodes decreases slightly.

Key words: blockchain; Bitcoin network; Bitcoin network node detection; method evaluation; method improvement

自 2008 年中本聪首次提出比特币系统[1]以来, 其价值不断攀升, 成为迄今为止最成功的数字货币[2]之一. 与其他数字货币不同, 比特币并不通过中央机构向用户分发, 而是依托区块链这一公共账本, 记录并维护

① 基金项目: 岳麓山工业创新中心创新项目 (2024YCII0113)
收稿时间: 2024-12-05; 修改时间: 2025-02-12, 2025-02-14; 采用时间: 2025-02-24; csa 在线出版时间: 2025-05-29
CNKI 网络首发时间: 2025-05-29

所有用户的每笔交易, 实现去中心化的数字货币管理.

比特币系统可以分为事务层和网络层, 以往的研究大多侧重于事务层而较少关注网络层. 比特币网络具有 3 大特点. 第一, 去中心化, 这意味着网络中不存在超级节点或中央组织, 系统中的节点通过消息的交互来达成共识; 第二, 匿名性, 指的是比特币用户的账户和地址被加密, 从而确保隐私和安全; 第三, 规模较大, 这意味着比特币用户众多, 比特币网络中的节点也很多. 比特币的这些特性使得对比特币网络的监控较为困难, 通常需要基于整个网络的拓扑结构来进行监控分析, 而要清晰了解整个比特币网络的拓扑结构, 需要对其组成节点有足够的了解, 因此对比特币网络的节点进行探测具有重要的工程意义.

比特币网络节点探测并非全新课题, 已有研究通过 Bitnode 和 DSN Bitcoin monitoring[3]等平台每日提供节点数量和属性的监控信息. 然而, 现有研究更多关注节点的统计分析, 较少涉及探测方法的优化细节. 随着网络规模扩展, 现有探测方法暴露出时间长、效率低、资源开销大的问题.

为解决这些问题, 本文首先使用 FTWD 模型对现有探测方法进行模拟实验以分析比特币网络节点探测的关键细节. 基于该模型, 对比特币网络中的节点数量、探测时间、新增节点比例以及节点连接情况进行了统计分析.

在此基础上, 本文对现有方法的 3 个阶段进行了优化改进. 首先, 增加了初始轮次的种子节点数量, 使得前两轮能够发现更多的节点, 从而减轻后续轮次的探测负担, 平衡各轮探测时间并缩短整体耗时; 其次, 提出了 DFT (depulication full traversal) 模型, 结合新增节点比例与去重全遍历方法, 当新增节点比例达到预设阈值时即终止探测, 解决了现有方法收敛缓慢的问题, 减少了无效探测所带来的时间消耗; 最后, 通过分析节点地址的连通性和地址列表的重叠情况, 发现后续轮次探测中存在大量重复节点, 且部分待探测节点无法返回有效的地址列表. 为此, 本文结合 DFT 模型与随机选择算法, 提出了一种部分遍历模型, 通过对每轮待探测节点进行随机筛选, 减少重复节点与无效连接的数量, 从而提升探测效率.

最后, 本文将这些改进整合为 BNP 方法并将其与现有方法进行实验对比, 实验结果表明, 当随机选择比例固定为 50% 时, BNP 方法在仅牺牲了 5.2% 的节点覆盖率的条件下减少了 40.45% 的探测时间, 且减少了 21.4% 的数据包处理和 9.22% 的数据字节处理. 这些数据综合表明 BNP 方法在提高探测效率的同时保持了较高的可达性, 具备良好的综合性能与实用性.

1 相关工作

1.1 比特币网络节点探测

已有许多研究针对比特币网络中的节点探测进行了深入分析. 2017 年, Wang 等人[4]设计了 bcclient, 使其能够在无需下载完整区块链数据的情况下作为比特币客户端使用, 并对可达和不可达节点进行了测量和比较. 2019 年, Park 等人[5]对比特币节点进行了测量, 37 天内收集了近 100 万个节点的数据, 并与以往研究进行对比. Zhang 等人[6]通过 Wireshark 抓取比特币流量, 利用 Bitcoin Seeder 工具大规模爬取节点信息, 并对节点变化进行了统计分析, 使用公式定量描述了这种变化. 2021 年, Grundmann 等人[7]提出了一种通过监控网络中的 addr 消息来估计不可达对等体数量的方法, 使用部署的监视器获取地址信息, 估算活跃对等体数量. 2022 年, Li 等人[8]设计了比特币嗅探器, 在 1.5 h 内收集了 410 万个节点数据, 并与 9515 个可达节点建立了 TCP 连接. 2023 年, Essaid 等人[9]设计了 Node-Probe 工具, 集成发现客户端与分析服务器, 可收集节点和区块链数据, 利用基于神经网络的分析引擎进行社区监测和拓扑动态分析, 然而该工具并未公开其获取的节点数据.

综上所述, 本文列举了与我们研究最为相关的先前工作, 主要集中于通过网络监控和 IP 地址收集, 从网络视角研究比特币网络中的节点拓扑.

1.2 比特币网络拓扑

接下来介绍一些与比特币网络拓扑相关的工作, 这与本文内容密切关联. 2014 年, Biryukov 等人[10]通过分析地址传播机制, 提出了一种拓扑推断方法, 他们收集了入口节点的广播地址, 并推断了节点的连接关系. 尽管方法合理, 但噪声过大, 影响了准确性. 2016 年, Neudecker 等人[11]通过观察特定交易到达时间的变化来推断网络拓扑和节点间的连接, 准确率和召回率均约为 40%. 2018 年, Deshpande 等人[12]提出了一种名为 BTCmap 的仿真拓扑推断方法. 该方法首先收集在线节点数据并输入仿真平台, 使用真实算法生成拓扑, 但结果与实际网络差异较大. 2019 年, Grundmann 等人[13]

通过向相邻节点发送“双花交易”提出了两种拓扑推断方法, 准确率为 71%, 召回率为 87%. Delgado-Segura 等人[14]提出了 Txprobe, 通过分析“孤立事件”推断相邻节点的连接, 准确率达 100%, 召回率为 95%. 尽管其准确性高, 但成本高昂. Grundmann 和 Delgado-Segura 的方案均未在真实比特币网络中验证, 且执行成本较高. 同年, Essaid 等人[15]提出了一个使用自定义 Page-Rank 算法的拓扑发现系统, 能够实时收集数据, 不仅分析了比特币 P2P 连接, 还对比特币的社区结构特性及地理分布进行了深入研究.

2 现有方法描述与实验分析

2.1 方法分析

为使网络节点可发现邻居节点以建立通信, 比特币网络通信协议设计了两种场景的“请求地址-响应地址”机制. 第 1 种场景通信过程如图 1 所示, 当节点初加入比特币网络时, 节点会向比特币种子节点发出 nslookup 消息, 请求初始的网络节点 IP 地址列表; 种子节点为固定长期在线的服务节点, 其 IP 地址一般被硬编码在比特币客户端软件中; 种子节点收到请求后会向客户端回应其所记录的当前活跃网络节点 IP 地址列表; 节点收到 IP 地址列表后会在本地维护、更新, 并根据列表中给出的 IP 地址信息与网络其他节点通信.

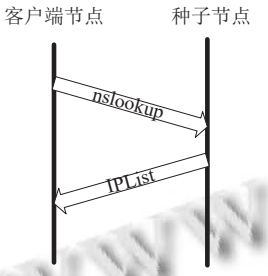

图 1 第 1 种通信场景原理

第 2 种通信场景如图 2 所示, 节点在加入比特币网络后, 可向其他节点发送 getAddr 消息请求对方维护的 IP 地址列表; 收到 getAddr 消息后, 节点会向请求节点发送包含自身地址列表的 addr 消息.

虽然实现细节有所不同, 但现有比特币网络节点探测方法, 如文献[5]和文献[8]使用的方法均以上述比特币网络通信协议的“请求地址-响应地址”机制为基础, 进行迭代式的节点探测, 其基本过程如图 3 所示, 分为 6 个基本步骤.

(1) 探测节点向比特币种子节点发出 nslookup 命令请求初始节点 IP 地址列表;

(2) 种子节点向探测节点响应网络节点 IP 地址列表;

(3) 探测节点收到节点 IP 地址列表后, 将节点 IP 地址信息保存到节点地址集合 A 并遍历该集合, 标记并向每个未被探测过的网络节点发送 getAddr 消息请求;

(4) 收到 getAddr 消息请求的活跃节点向探测节点响应 addr 消息, 此消息包含了自身维护的地址列表;

(5) 探测节点将收到的 addr 消息中所含网络节点 IP 地址去重后存入节点地址集合 A;

(6) 探测节点重复步骤 (3)–(5), 直至节点地址集合 A 中再无未被探测过的网络节点.

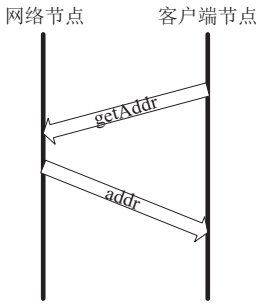

图 2 第 2 种场景原理

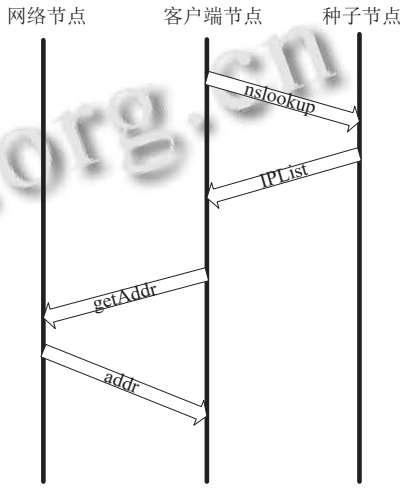

图 3 探测原理

为评估现有方法的探测开销和探测效率, 本文在现有方法的基础上设计了无去重全遍历的节点探测模型 FTWD. FTWD 模型分轮次以上述 6 个步骤进行节点探测, 每一轮的结果保存为一个地址集合. “无去重”指的是新一轮获取的 IP 即使在上一轮的地址集合中已存在, 也会被加入本轮次的地址集合中; “全遍历”指

的是每一轮均对上一轮保存的地址集合中节点全部遍历, 依次对每个节点进行地址探测.

2.2 实验设计

针对 FTWD 模型进行了代码设计, 参考文献[5]提供的代码参数, 本文将超时时间设为 10 s, 这意味着如果成功连接到节点但未收到消息, 会等待一定时间, 若超过 10 s 无响应, 则断开连接并继续尝试其他节点. 另外, 本文将最大线程数设为 256, 每个线程一次处理一个节点的连接尝试, 从而实现最多 256 个并发连接. 最后, 由于 FTWD 是分轮次进行的节点探测, 为了更好地评估现有方法并且不会使单次测量的时间过长, 本文将单次 FTWD 模型的运行轮次设定为 10 轮.

测试平台选择了阿里云, 服务器配置为 8 核 2 GB, 在服务器上运行了多次 FTWD 模型, 并对不同次序相同轮次的结果取平均值作为最后的结果.

2.3 结果分析

本节将展示多次运行 FTWD 模型后不同轮次的结果的平均值.

2.3.1 节点数量与探测时间变化分析

FTWD 模型不同轮次的探测时间变化如图 4. 由于第 1 轮的时间较短, 因此未展示. 从图 4 中可以看出, 前 4 轮的探测时间迅速增加, 而从第 5 轮开始趋于稳定, 表明从第 5 轮起, 需要遍历的节点数量基本一致.

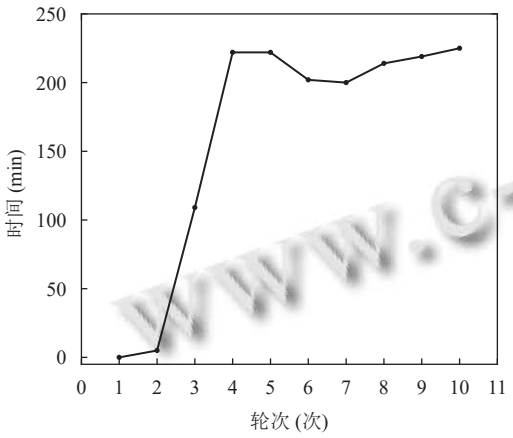

图 4 探测时间变化

现有方法的节点数量及连接情况如图 5 所示, 图中展示了连接成功、连接失败的节点数以及总节点数. 分析图 5 可以发现, 无去重全遍历模型下, 节点总数先是缓慢增长, 随后快速增加, 最终趋于稳定, 总数约为 30 万. 连接失败的节点数量远超连接成功的节点, 比例接近 30:1.

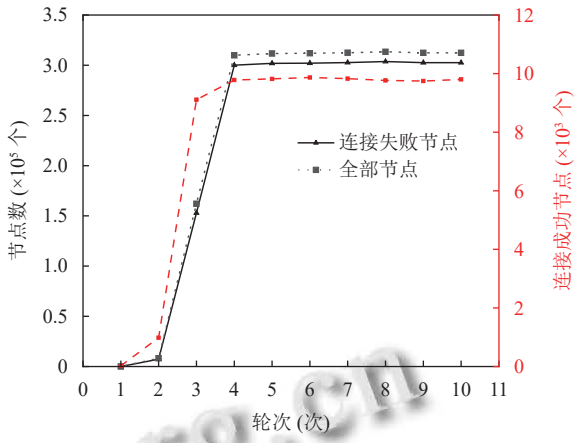

图 5 节点数量变化

2.3.2 新增节点比例分析

通过观察节点总数的变化, 可以发现, 在后续几轮中节点总数趋于稳定. 这意味着每一轮新增节点的数量逐渐减少, 如果花费大量时间却只探测到较少的新节点, 在效率上得不偿失. 因此, 本文提出了一个用于描述探测到的新节点数量的参数, 即新增节点比例, 具体计算方式如式 (1). 其中, Num_{round_i} 代表第 i 轮的节点个数. 当新增节点比例小于某个阈值后, 本文认为在此时是节点探测率和效率的平衡点, 从而在此时停止探测过程比较合理.

$$ratio(i) = \frac{Num_{\text{round}_i} - Num_{\text{round}_i} \cap Num_{\text{round}_{i-1}}}{Num_{\text{round}_i}} \times 100\% \tag{1}$$

式 (1) 计算过程为, 首先本轮节点集合和上一轮节点集合取交集, 然后将本轮节点集合减去该交集并计算其大小, 最后将该值除以本轮节点集合的大小, 即得到新增节点比例.

使用式 (1) 计算了从第 2 轮开始的新增节点比率, 所得结果如图 6 所示.

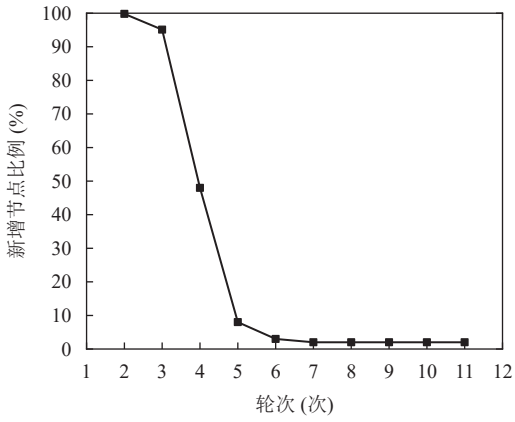

图 6 新增节点比例变化曲线图

从图 6 可以看出, 从第 6 轮开始, 新增节点比率趋于稳定, 维持在 5% 以下. 因此可以认为, 当使用 FTWD 模型且新增节点比率低于 5% 时, 本轮探测过程结束.

2.3.3 节点连接性变化

本文使用集合记录每一轮中每个节点返回的地址列表. 连接成功的节点中能够返回地址列表的节点数, 以及这些节点中返回空地址列表的节点数如图 7 所示.

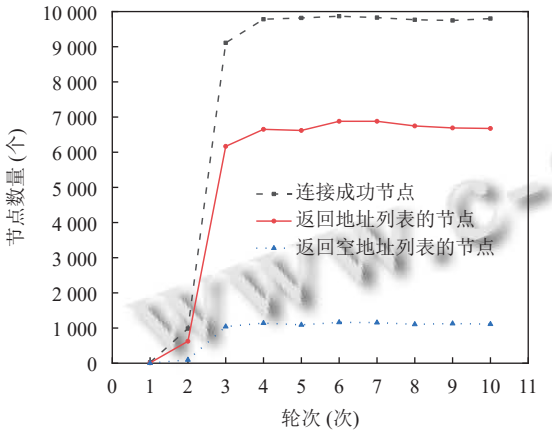

图 7 节点连接性与返回地址列表变化

通过图 7 的数据可以总结出以下两个现象: 首先, 在连接成功的节点中, 约有 2/3 能够返回地址; 其次, 在能够返回地址的节点中, 有 1/6 的节点仅返回一个包含自身地址的 addr 消息, 即为空响应. 对于第 1 个现象, 抓包分析显示, 一些节点在成功连接后迅速断开连接, 这一行为通常由远程节点主动触发, 可能是出于远程节点的防御机制或客户端版本差异导致的不同响应. 此外, 部分节点在发送 version 消息后没有进一步交互, 直接断开连接, 这可能是由于客户端差异, 使得我们的消息被过滤或忽略. 网络波动也可能导致部分连接异常. 对于第 2 个现象, 空响应的产生可能有两个原因: 一是比特币节点根据网络类型对返回的地址列表进行过滤; 二是节点返回的地址列表可能来自缓存数据, 而非实时获取. 这些因素共同导致了大量的空响应现象. 综上所述, 尽管每次全遍历需要处理超过 30 万个节点, 实际能够返回有效地址列表的节点数量相对较少. 这一现象为本文后续对比特币网络节点探测方法的优化提供了理论依据.

2.3.4 交集分析

本文记录了每个节点返回的地址列表, 并对这些列表两两求交, 统计每个 IP 在不同地址列表中的出现次数. 结果显示, 在无去重全遍历模型下的每轮数据中, 某些 IP 在多个节点返回的列表中出现了 600–700 次.

进一步分析交集情况发现, 任意两个返回地址列表的交集节点数通常不超过 10 个 (每个列表平均包含约 1 000 个节点). 这表明任意两个节点返回列表的重复率相对较低. 然而, 某些 IP 的出现次数仍然能够接近返回列表节点数的 1/10. 这一现象的原因在于, 返回列表中的节点数量足够多. 比特币网络的节点总数大约为 30 万, 而能够返回地址列表的节点数在 5 000–7 000 之间波动. 假设每个节点返回 1 000 个互不重复的地址, 整体上将涵盖 50–70 万个节点, 远超比特币网络中实际可探测的节点数. 因此, 在每个节点的返回列表中, 重复节点是不可避免的.

这一点为本文后续对比特币网络节点探测方法的优化提供了重要的理论依据.

3 改进方法

3.1 现有问题

通过对现有方法的分析, 本文发现当前节点探测方法存在以下问题: 首先, 前两轮探测中获取的节点数量较少, 导致初始效率低下. 这是由于现有方法在不同阶段均采用固定线程数处理连接请求, 当连接数量较少时, 固定分配的线程资源未能充分利用, 反而造成浪费, 延长了探测时间. 其次, 后续轮次中重复节点比例显著增加, 不仅降低了新节点的发现率, 还导致额外的计算和网络资源消耗, 进一步拉长了整体探测时间.

此外, 随着探测轮次增加, 新增节点数量逐渐减少, 节点总数趋于稳定, 表明现有方法收敛速度较慢. 后续轮次消耗大量时间却只能获取极少量新节点, 降低了整体效率. 因此, 若能在前两轮获取更多节点, 同时减少后续轮次的重复探测, 平衡各轮次的节点增量分布, 将有助于提升效率, 缩短探测时间并降低资源占用.

3.2 优化方法

针对第 3.1 节提出的问题, 本节提出了 3 种优化策略, 旨在不同网络环境下兼顾效率与资源消耗.

(1) 优化初始轮次的种子节点选择策略

该策略用以提高前两轮的节点获取量. 为避免单纯增加种子节点数量可能引发的网络拥塞, 本文综合考虑了网络带宽和可用连接资源, 动态调整初始种子节点数量, 在提升效率的同时避免资源浪费.

(2) 新增节点比例与有去重全遍历方法结合的模型 DFT

结合新增节点比例的分析, 提出将其与有去重全遍历方法结合的模型 DFT. 其中, “有去重”指不再与已连接节点重复连接, “全遍历”则是对未连接节点进行全面处理. 通过新增节点比例控制 DFT, 当比例达到预设阈值时结束探测. 这一策略可有效解决收敛慢的问题, 减少低效探测的时间开销, 但其复杂度和资源消耗需进一步分析, 尤其是在大规模网络中可能带来较高的计算和存储开销.

(3) 提出了部分遍历模型

针对后续轮次中重复节点多且部分节点无法返回地址列表的问题, 将 DFT 模型与随机选择算法结合, 提出了部分遍历模型. 该模型在每轮待访问集合中进行随机选择, 减少重复节点和无效连接的数量, 从而提高效率. 然而, 随机选择可能引入不确定性, 需在实际应用中权衡随机概率与探测全面性.

3.3 实验设计

为验证优化策略的有效性并减少方法修改的潜在影响, 本文设计了以下实验.

首先, 将初始轮次探测作为独立阶段, 根据网络环境动态调整种子节点数量和线程资源分配, 以提升初始效率并适应不同条件. 为减少网络环境影响, 在相同网络环境和硬件环境下测试了修改初始轮次的方法与未修改方法.

然后, 初始探测结束之后, 后续探测过程尽量与现有方法保持一致, 仅调整探测结束条件. 本文通过多次实验确定了新增节点比例的阈值, 并在相同环境下对比仅修改初始轮次的方法与结合新增节点比例控制的方法.

最后, 本文在相同条件下对比了 DFT 模型与基于随机选择算法的部分遍历模型, 以评估随机选择对探测效率的影响.

3.4 资源与时间消耗评估

优化策略的时间复杂度与现有方法基本保持一致. 由于探测的核心流程未发生本质变化, 整体仍基于对节点的遍历与连接, 因此优化方法的时间复杂度不会显著增加. 然而, 在实践中, 由于优化策略减少了低效探测轮次的执行, 能够在相同或更短的时间内获取更多有效节点, 从而有效缩短整体探测时间. 这一优化在后文的实验部分有具体的数据对比与分析.

在资源消耗方面, 优化策略的影响主要体现在内存使用与计算负载上. 其中, DFT 模型需要维护未连接节点集合, 可能带来一定的存储开销; 但相比于减少的重复探测, 这部分额外消耗在可接受范围内. 此外, 部分遍历模型通过随机选择降低了重复节点的比例, 减少了网络流量与无效连接所导致的计算资源浪费. 因此, 综合来看, 优化策略在时间开销上较现有方法有所减少, 而在资源消耗方面, 尽管 DFT 可能增加一定的存储需求, 部分遍历模型则能够平衡这一问题, 使整体资源消耗得以优化. 第 4 节将通过实验数据具体量化这些优化策略的实际影响.

4 改进方法实验与评估

本节采用与 FTWD 模型相同的实验设计来测试每个改进方法, 对每个方法进行多次实验, 记录实验结果及平均值, 最后基于这些记录结果进行评估.

4.1 不同初始轮次种子节点评估分析

为了测试初始轮次种子节点数量对探测效率的影响, 本文在优化初始轮次节点选择策略下, 调整了 FTWD 模型中用于获取初始网络节点 IP 地址列表的种子节点选择方式. 在实验中, 基于网络环境的不同设置了多种初始轮次种子节点数量, 以评估其对整体探测性能的影响.

实验过程中, 本文分别运行了原版 FTWD 模型 (固定初始种子节点数量) 和优化后的 FTWD 模型 (根据网络环境动态调整种子节点数量). 为确保测量结果的稳定性与普适性, 本文在相同硬件和网络环境下进行了 5 次实验, 并对结果取平均值进行分析. 为减少数据展示的冗余, 本文仅呈现最终轮次的探测结果, 并对探测时间、节点覆盖率、资源消耗等关键指标进行比较分析.

此外, 为了进一步验证不同网络环境下的适应性, 本文分别在高带宽环境和受限带宽环境下测试了优化策略, 以分析其在不同资源条件下的影响. 实验结果记录在表 1 中, 并展开详细讨论.

通过分析表 1 中的数据, 可以发现优化初始轮次种子节点数量后, 探测结果发生了如下变化.

(1) 节点数变化

平均可达节点数由 9 070 下降至 9 050, 平均总节点数由 296 119 下降至 290 775, 差异均较小. 这可能是由于网络环境的波动所致, 表明两种方法在探测覆盖率上的结果基本一致.

(2) 探测时间变化

平均探测时间从 225.4 min 减少至 214.6 min, 缩短了 5.03%. 在探测到的节点数量相近的情况下, 探测时间明显减少, 表明优化策略提高了探测效率.

表 1　不同初始轮次种子节点结果对比

实验次数	原版FTWD模型 (固定初始种子节点数量)					优化后的FTWD模型 (根据网络环境动态调整种子节点数量)				
	可达节点数 (个)	节点总数 (个)	探测时间 (min)	探测开销数据包 (个)	探测开销-数据字节 (byte)	可达节点数 (个)	节点总数 (个)	探测时间 (min)	探测开销数据包 (个)	探测开销-数据字节 (byte)
1	9083	299152	218	478169	141645307	9018	292208	219	422038	136918497
2	9133	296044	229	438570	135123248	9028	290363	225	481346	133749142
3	9083	295293	207	437898	136867136	9060	291868	239	484039	138852443
4	9043	296342	215	481001	139419440	9070	290243	191	469096	141808045
5	9008	293764	258	514595	136176262	9075	289191	199	421384	135549239
平均值	9070	296119	225.4	470190.6	137846278	9050	290775	214.6	455581	137375473

(3) 数据包与流量消耗

接收端的数据包数量和数据字节数变化不大, 这可能是由于两种方法在探测过程中接收到的地址信息数量相近.

发送端的数据包数量和数据字节数减少了约 3%, 表明优化策略在降低网络开销方面具有一定优势.

整体而言, 优化后的方法在平均处理的数据包数量和数据字节数方面减少了约 3%, 进一步降低了探测过程中的计算和网络资源消耗.

4.2　DFT 模型评估分析

由于 DFT 模型与 FTWD 模型的探测方式略有不同, 因此 DFT 模型的新增节点比例阈值需要重新计算, 但计算方式与 FTWD 模型保持一致. 通过实验分析, 本文确定当新增节点比例低于 1% 时, 可认为探测过程已基本收敛, 可以终止探测, 从而减少后续轮次的低效探测开销.

在此基础上, 本文采用该新增节点比例阈值控制 DFT 模型的探测终止条件, 以优化探测效率并减少资源消耗. 为了确保测量结果的普适性, 本文对 DFT 模型进行了 5 次独立运行, 并对结果取平均值进行分析.

为简化统计分析, 实验过程中各轮次的详细数据不再逐一展示, 仅呈现最终轮次的探测结果, 以便直观比较优化前后的探测效果.

运行 DFT 模型探测到的节点数量变化如图 8 所示, 相应花费的探测时间变化如图 9 所示.

通过分析图 8 数据可以发现, 该模型获得的平均可达节点数为 8 907 个, 平均总节点数为 278 641.2 个. 虽然获得的可达节点数略少于运行优化后的 FTWD 模型获得的可达节点数, 但这并非模型性能的问题, 而是比特币主网在不同时间段的节点数量差异所致. 参考 DSN[3]数据, 实验期间比特币主网的可达节点数有所下降, 且本文对新增节点比例模型的测试比现有方法的测试稍晚了几天, 因此这种差异可以忽略不计.

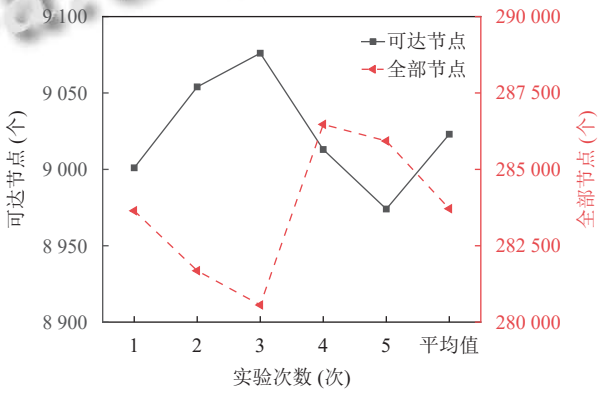

图 8　DFT 探测节点数量变化

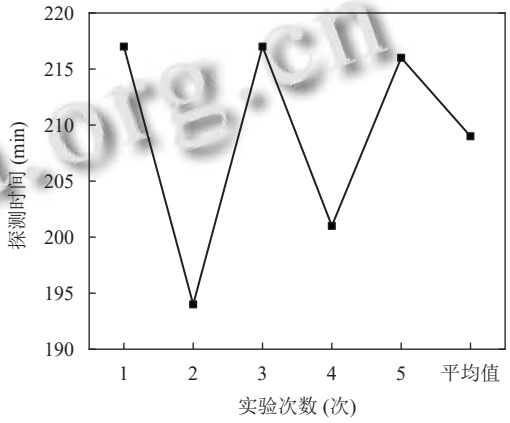

图 9　DFT 节点探测时间变化

通过分析表 1 和图 9 可以发现, DFT 模型的平均节点探测时间为 176.8 min, 相比于仅增加初始轮次种子节点的探测时间减少了 14.3%.

DFT 模型探测到上述节点所需的探测开销变化如图 10 所示.

在探测开销方面, 分析图 10 和表 1 的数据发现, DFT 模型与仅增加初始轮次种子节点的方法相比, 探

测期间处理的数据包总量和数据字节总量整体减少了 1%. 虽然两种方法在探测开销上的差距很小, 但 DFT 模型的探测时间更短, 整体效率更高.

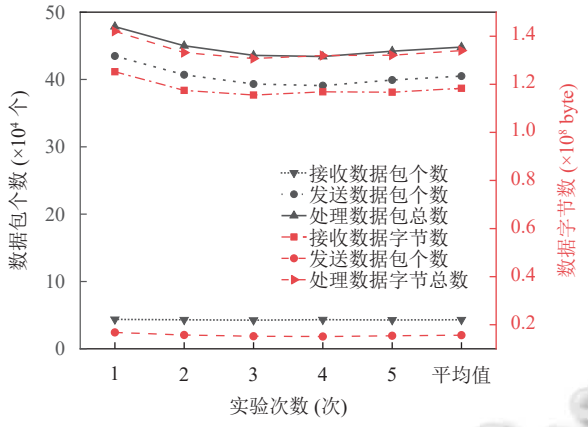

图 10 DFT 探测开销变化

4.3 部分遍历方法评估分析

本节对部分遍历方法进行了实验测试, 该方法基于 DFT 模型, 并在每一轮从待访问节点集合中随机选择一定比例的节点进行探测. 为了确保测量结果的普适性, 本文对 DFT 模型进行了 5 次运行, 并对结果取平均值进行分析. 为简化统计分析, 模型运行过程中各轮次的数据不再展示, 仅呈现最终轮次的运行结果.

部分遍历方法在不同随机选择比例下的新增节点比例阈值如图 11 所示. 通过分析图 11 可以看出, 当随机选择比例大于等于 30% 时, 新增节点比例能够降至 1% 以下, 该阈值与 DFT 模型计算得到的新增节点比例阈值相同.

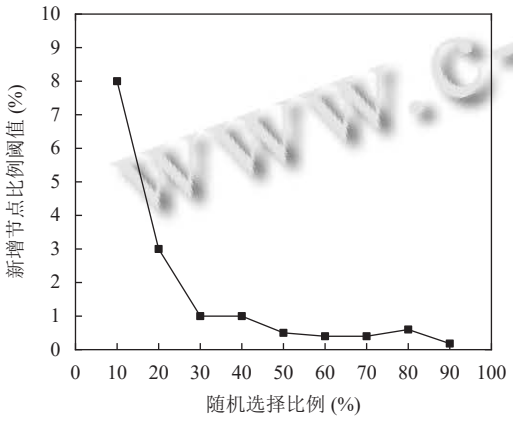

图 11 不同随机选择比例的新增节点比例阈值

部分遍历方法在不同随机选择比例下首次使新增节点比例低于 1% 时探测到的节点数量如图 12 所示, 相应的探测时间如图 13 所示.

根据图 12 的数据, 当随机选择比例低于 30% 时, 新增节点的比例无法降至 1%, 这是由于本文方法基于 Netty 框架处理 TCP 连接, 并分配了 256 个线程. 当随机选择比例过低时, 即便待连接节点集中包含大量节点, 随机选择得到的节点数量仍然不足. 节点数过少而线程数过多, 容易导致网络连接超时、线程间竞争加剧, 甚至可能引发死锁, 最终可能导致程序提前终止, 从而无法使新增节点比例降至 1% 以下.

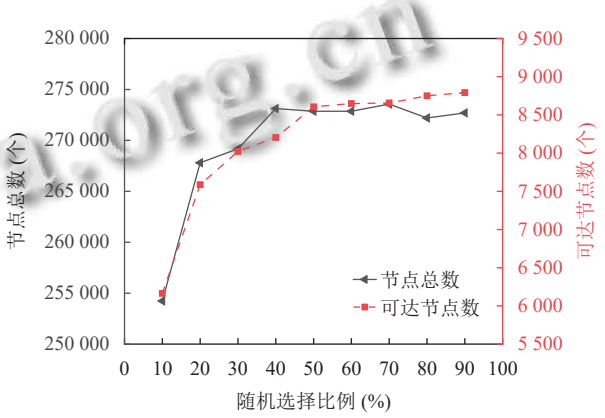

图 12 部分遍历方法探测节点数量

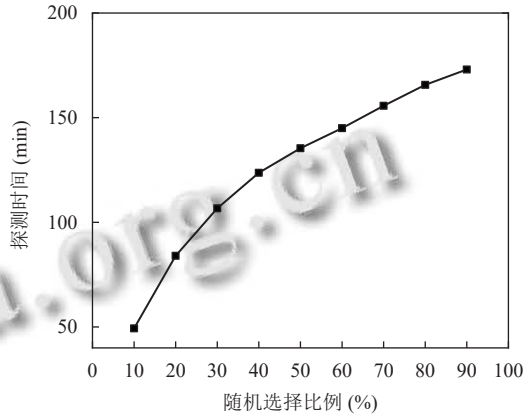

图 13 部分遍历方法探测时间

虽然在随机选择比例为 10%–20% 时, 新增节点比例未能降至 1%, 但本文仍将展示该范围内的数据. 然而, 在对比分析中将主要关注随机选择比例在 30%–90% 之间的情况.

通过分析图 13 中的数据可以发现, 当随机选择比例在 30%–90% 之间时, 节点总数几乎保持不变, 而可达节点数随着随机选择比例的增加逐渐上升, 但增长速度逐渐放缓. 结合图 12 和图 13 的数据, 计算可达节点数与探测时间的比值后发现, 可达节点的发现率随着随机选择比例的上升逐渐下降.

基于这些数据, 本文对比了不同随机选择比例下部分遍历方法与 DFT 模型的运行结果. 图 14 展示了不同随机选择比例下, 部分遍历模型相比于 DFT 模型的探测时间缩减的变化. 这里的缩减比例指的是部分遍历方法在不同随机选择比例下的探测时间相比于 DFT 模型减少的比例.

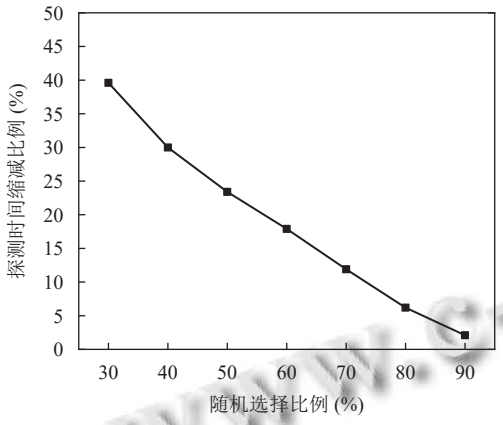

图 14 部分遍历方法的探测时间缩减比例变化

图 15 记录了在不同随机选择比例下, 部分遍历模型相比于 DFT 模型的节点总数覆盖率和可达节点覆盖率的变化. 其中, 可达节点覆盖率是部分遍历方法获得的可达节点数与 FTWD 模型可达节点数的比值, 而节点总数覆盖率是部分遍历方法获得的节点总数与 FTWD 模型节点总数的比值. 这样可以更清晰地反映不同随机选择比例对探测效果的影响.

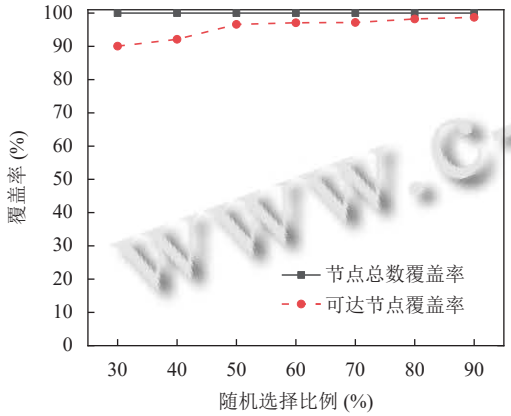

图 15 节点总数及可达节点覆盖率变化

通过分析图 14 和图 15 数据, 可以得到以下结论.

(1) 与 DFT 模型相比, 不同随机选择比例下的节点总数覆盖率几乎相同, 均接近 100%, 这意味着不同随机选择比例的部分遍历方法探测到的节点总数基本一致. 此外, 随着随机选择比例的增加, 可达节点的覆盖率和探测时间的缩减比例也逐步提高.

(2) 当随机选择比例为 50% 时, 部分遍历方法在可达节点覆盖率和探测时间缩减比例之间达到了平衡. 相比于 DFT 模型, 牺牲了 3.4% 的节点覆盖率, 换来了 23.4% 的探测时间缩减.

部分遍历方法的探测开销如图 16 所示. 通过分析图 16 的数据, 在探测开销方面可得以下结论.

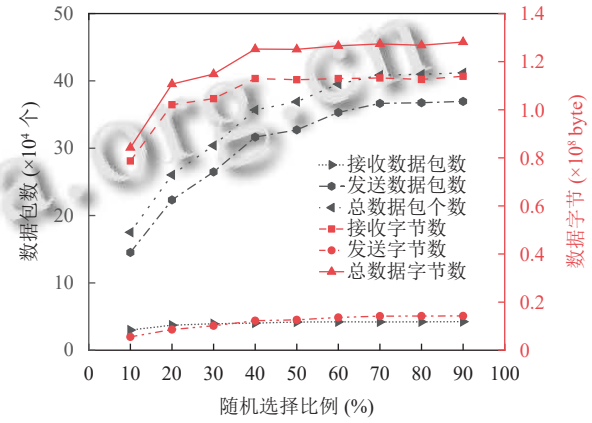

图 16 不同随机选择比例的探测开销

(1) 随着随机选择比例的增加, 探测总量先迅速上升, 随后逐渐趋于稳定. 这是因为随机选择比例提升后, 节点集合中的可达节点逐渐饱和, 导致有效数据包的交换和数据字节的处理趋于稳定. 然而, 较高的随机选择比例也会带来额外的无效开销, 从而降低整体效率.

(2) 接收的数据字节数远超发送的数据字节数. 原因在于发送的数据主要为命令消息, 而接收端可能会收到大量的地址消息, 因此接收的数据量明显大于发送的数据量.

(3) 发送的数据包数量显著多于接收的数据包数量. 由于在建立 TCP 连接过程中, 需要发送大量数据包, 而比特币网络中存在许多不可达节点, 这些节点无法响应发送的数据包, 导致大量单向发送的情况, 从而使发送的数据包数量远超过接收的数据包数量.

基于上述分析和测量数据, 本文对比了不同随机选择比例下部分遍历模型与 DFT 模型的探测开销, 其中, 不同随机选择比例下的探测量差异结果如图 17 所示.

结合图 17 数据与 DFT 模型的探测开销分析可知, 部分遍历方法在随机选择比例为 30%–70% 时数据包总数迅速减少, 80%–90% 时趋于平稳. 这种现象的出现是由于发送的数据包明显多于接收的数据包, 总体数据包数量的变化趋势与发送数据包的变化趋势一致.

随着随机选择比例的增加, 发送的数据包数量增加, 进而导致总数据包数量呈现出如图 17 所示变化趋势.

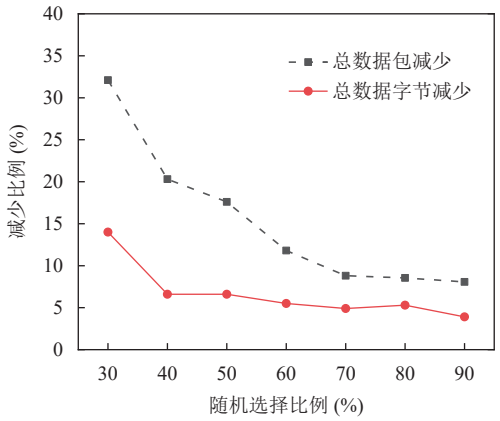

图 17 不同随机选择比例下的探测量差异

从图 17 中还可以看出, 当随机选择比例达到 40% 及以上时, 总数据字节数的减少趋于平稳. 结合前文分析可知, 接收的数据字节数远多于发送的数据字节数. 随机选择比例超过 40% 后, 总数据字节数的减少幅度变小, 表明此时接收的数据字节数变化不显著. 换言之, 大部分地址信息在随机选择比例达到 40% 时便基本被接收和处理完毕.

4.4 改进方法综合评估分析

第 4.1–4.3 节分析表明, 3 种改进方法在确保节点覆盖率的前提下, 均能显著缩短探测时间. 然而, 单独使用某一种方法时, 仍可能存在资源浪费、重复探测等问题. 为进一步优化探测效率, 本文提出了综合这 3 种方法的 BNP (balanced node probing) 方法, 该方法在略微牺牲节点覆盖率的情况下, 有效减少了探测时间. 实验结果表明, 当随机选择比例为 50% 时, BNP 方法的探测时间较 DFT 模型减少 39.9%, 而节点覆盖率仅降低 5.2%.

为了进一步验证 BNP 方法的改进效果, 本文复现了文献[8]的方法并进行了对比实验. 为确保实验的稳定性, 两种方法均运行 5 次, 并取其平均值进行分析. 表 2 展示了 BNP 方法与文献[8]在探测到的节点数量上的对比, 表 3 统计了探测时间、资源开销及两种方法的差异, 并以百分比形式计算差异 (计算方式: 以文献[8]的结果减去 BNP 方法的结果, 再除以文献[8]的结果). 对比实验中, BNP 方法的随机选择比例固定为 50%.

表 2 BNP 方法和文献[8]方法探测节点数量结果比较

实验次数	可达节点			节点总数		
	BNP (个)	文献[8] (个)	差异百分比 (%)	BNP (个)	文献[8] (个)	差异百分比 (%)
1	8625	9001	4.1	272891	283643	3.7
2	8600	9033	4.7	273100	281682	3.0
3	8575	9043	5.1	272255	280561	2.9
4	8596	9013	4.6	273020	281471	3.0
5	8630	8975	3.8	273028	282929	3.4
平均值	8605.2	9013	4.5	272856.8	282057	3.2

表 3 BNP 方法和文献[8]方法探测时间与开销结果比较

实验次数	探测时间			探测开销-数据包			探测开销-数据字节		
	BNP (min)	文献[8] (min)	差异百分比 (%)	BNP (个)	文献[8] (个)	差异百分比 (%)	BNP (byte)	文献[8] (byte)	差异百分比 (%)
1	136	218	37.6	369640	478169	22.6	123670776	141645307	12.6
2	135	229	41.0	372242	438570	15.1	127379393	135123248	5.7
3	135	217	37.7	369631	437898	15.5	125091875	136867136	8.6
4	135	215	37.2	367029	481001	23.6	124748917	139419440	10.5
5	136	258	47.2	367079	514595	28.6	124749207	136176262	8.3
平均值	135.4	227.4	40.4	369124.2	470190.6	21.4	125128033.6	137846278	9.2

实验结果表明, BNP 方法有效减少了探测时间, 同时保持了较高的节点覆盖率. 这种优化效果源于 3 种改进方法的协同作用, 通过增加种子节点、引入随机选择策略以及优化去重机制, 使得探测过程更加高效. BNP 方法的优化机制不仅在实验中得到了验证, 同时也具备理论上的支持. 为深入分析 BNP 方法的优化原理, 本节从数学角度探讨其提高探测效率的内在逻辑.

(1) BNP 方法优化效果的理论支持

在分析无去重全遍历模型 (FTWD) 时, 发现该模型的探测效率低下, 尽管每轮探测需要遍历数十万个节点, 但实际返回有效地址节点比例极低 (不足 0.1%), 导致严重的资源浪费. 基于此, 本文提出核心假设: 通过智能缩减待访问集合的规模, 可以在保证探测结果完整性的前提下优化探测效率. 为验证该假设, 本文基于历史探测日志构建了理论模拟框架, 重点考察随机选择策略对探测效果的影响. 具体而言, 我们对 FTWD 模型的运行轨迹进行了重构, 从第 3 轮探测开始 (前两轮因数据量过小不具备统计意义), 提取原始待访问集合 C 的日志数据. 以第 3 轮探测为例, 原始待访问节点数为 162 403, 通过 50% 随机采样选取 81 021 个节点

构建实验组集合 TEMP-A, 随后关联该轮次的响应集合 E 日志, 建立地址映射关系库 TEMP-B, 并采用哈希索引技术实现跨集合匹配, 自动完成地址去重处理. 此外, 我们保留了原有集合的生成机制, 以确保实验条件的可比性. 在 3 次独立模拟实验后, 结果表明, 随机选择策略可达到 91.6% 的有效节点覆盖率 (相较于全遍历方法), 且 99.8% 的地址完整性得以保留, 说明采样策略不会显著影响探测结果的准确性. 进一步的比例敏感性分析显示, 当随机采样率从 10% 增加到 90% 时, 有效节点数与地址总数呈近似线性增长. 这一定量分析表明, 适度缩减待访问集合不会显著影响探测质量, 同时能有效减少资源消耗, 从而提高探测效率.

(2) BNP 方法优化原因探究

BNP 方法的优化效果源于 3 种方法的协同作用, 可通过数学建模进行解释. 首先, 种子节点扩展与随机选择相结合能够提高探测增益. 假设某轮探测的待访问集合为 C, 其中实际可达节点集合为 V, 在传统方法下, 探测效率可按式 (2) 计算:

$$E_{\mathrm{FTWD}} = \frac{|V|}{|C|} \tag{2}$$

由于$|V| \ll |C|$, 导致探测效率极低. 而通过增加种子节点并采用随机选择策略, 可构造更具代表性的采样集合 C', 其中 $C'=\alpha C$, $0<\alpha<1$. 实验数据显示, 当 α=50% 时, $|V'|\approx$91.6%, 处理后的探测效率可按式 (3) 进行计算:

$$E_{\mathrm{BNP}} = \frac{|V'|}{|C'|} \tag{3}$$

由于$|C'| \ll |C|$, 且$|V'| \approx |V|$, 因此可以得出结论: BNP 方法在探测效率上显著优于 FTWD 模型. 其次, DFT 模型通过去重机制减少重复探测, 从而提高有效探测比例; 同时, 利用新增节点的比例来控制探测的结束时间, 从而进一步优化探测过程. 令原始集合 C 包含重复节点集合 D, 则新的集合 C 可按式 (4) 进行计算:

$$|C_{\mathrm{eff}}| = |C| - |D| \tag{4}$$

BNP 方法结合 DFT 进行去重, 使得待访问集合 C 进一步缩减, 集合 C 具体大小可按式 (5) 进行计算:

$$|C_{\mathrm{BNP}}| = \alpha(|C| - |D|) \tag{5}$$

从而有效减少探测时间和资源消耗, 同时保持高覆盖率. 最终, BNP 方法综合上述 3 种优化策略, 最终的探测效率可按式 (6) 方式进行计算:

$$E_{\mathrm{BNP}} = \frac{|V'|}{\alpha(|C| - |D|)} \tag{6}$$

由于 α<1 以及$|D|$>0, 意味着 BNP 方法在缩小探测规模的同时保持了较高的节点发现率, 从而有效提升探测性能.

通过实验验证和数学分析, 可以得出以下结论: BNP 方法能够在减少待访问集合的同时保持高覆盖率和较好的地址完整性, 这一结论由模拟实验和数学建模共同验证. 实验数据显示, BNP 方法的探测时间减少 39.9%, 节点覆盖率仅降低 5.2%, 在探测效率与覆盖率之间取得了良好平衡. 进一步分析表明, 两种方法探测到的节点总数基本一致, 差异不足 3%, 且由于比特币网络节点基数较大, 这种差异可忽略不计. 然而, BNP 方法探测到的可达节点数比文献[8]方法减少 3.8%–5.1%, 平均减少 4.5%, 主要是部分遍历策略导致部分节点未测试连接性, 但 BNP 方法参数可调, 可通过调整探测参数接近文献[8]的可达节点数. 此外, BNP 方法的探测时间比文献[8]减少 37.2%–47.2%, 平均减少 40.4%, 探测数据包量减少 21.4%, 数据字节量减少 9.22%, 在显著降低探测时间和开销的同时, 仅以微小的节点覆盖率下降为代价, 展现出优异的探测效率和综合性能. 数学分析进一步解释了 3 种优化方法的协同作用, 证明了 BNP 方法的理论可行性, 而非仅依赖实验结果. 综上所述, BNP 方法通过理论分析与实验验证, 充分证明了 3 种优化策略的综合优势, 为比特币网络探测提供了一种高效、可行的改进方案.

5 总结

本文首先简要介绍了现有的比特币网络节点探测方法, 并通过实验验证得出了初步结论. 接着, 使用 FTWD 模型模拟现有方法, 深入研究了比特币网络节点探测的具体细节. 基于该模型, 本文对比特币网络中的节点数量、探测时间、新增节点比例以及节点连接情况进行了统计分析.

在此基础上, 本文对现有方法的 3 个阶段进行了改进. 首先, 增加了初始轮次的种子节点数量, 使得前两轮能够发现更多节点, 减轻后续轮次的探测压力, 平衡探测时间并缩短整体时长; 其次, 提出了 DFT 模型, 将新增节点比例与有去重全遍历方法结合, 当新增节点比例达到预设阈值时, 探测过程即可终止, 解决了现有方法收敛慢的问题, 减少了低效探测的时间开销; 最

后, 通过分析节点地址的连接性和地址列表交集, 发现后续探测轮次中存在大量重复节点, 且许多待探测节点无法返回地址列表. 为此, 本文结合 DFT 模型与随机选择算法, 提出了部分遍历模型, 通过对每轮获取的待访问节点集合进行随机选择, 减少重复节点和无效连接的数量, 从而提升探测效率. 这 3 项改进均在保证可达节点覆盖率的同时, 显著缩短了探测时间.

本文整合这些改进策略, 提出了 BNP 方法, 并通过数学分析与实验验证相结合的方式证明了其高效性与可行性. 实验结果表明, 当随机选择比例为 50% 时, BNP 方法在仅牺牲 5.2% 节点覆盖率的情况下, 将探测时间减少 39.9%, 同时降低 21.4% 的数据包处理量和 9.22% 的数据字节处理量. 这一结果充分表明, BNP 方法在显著提升探测效率的同时, 保持了较高的节点可达性, 具备优异的综合性能与实用性.

尽管如此, 本文的方法仍有改进空间. 例如, 目前仅对待访问节点集合进行简单随机选择, 未来可以基于地理位置、网络端口等因素进行更精细地筛选. 此外, 本文提供的代码仅能探测可达节点, 对于不可达节点的探测仍有待深入研究, 未来将在这一方向进行进一步优化.

参考文献

1 Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. www.bitcoin.org, 2008. [2024-12-01].

2 Tschorsch F, Scheuermann B. Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Communications Surveys & Tutorials, 2016, 18(3): 2084–2123.

3 Grundmann M, Amberg H, Baumstark M, *et al*. Short paper: What peer announcements tell us about the size of the Bitcoin P2P network. Proceedings of the 26th International Conference on Financial Cryptography and Data Security. Grenada: Springer, 2022. 694–704.

4 Wang L, Pustogarov I. Towards better understanding of Bitcoin unreachable peers. arXiv:1709.06837, 2017.

5 Park S, Im S, Seol Y, *et al*. Nodes in the Bitcoin network: Comparative measurement study and survey. IEEE Access, 2019, 7: 57009–57022. [doi: 10.1109/ACCESS.2019.2914098]

6 Zhang Y, Tan RN, Kong XY, *et al*. Bitcoin node discovery: Large-scale empirical evaluation of network churn. Proceedings of the 5th International Conference on Artificial Intelligence and Security. New York: Springer, 2019. 385–395.

7 Grundmann M, Amberg H, Hartenstein H. On the estimation of the number of unreachable peers in the Bitcoin P2P network by observation of peer announcements. arXiv: 2102.12774, 2021.

8 Li RG, Zhu JW, Xu DW, *et al*. Bitcoin network measurement and a new approach to infer the topology. China Communications, 2022, 19(10): 169–179. [doi: 10.23919/JCC.2022.00.030]

9 Essaid M, Lee C, Ju H. Characterizing the Bitcoin network topology with Node-probe. International Journal of Network Management, 2023, 33(6): e2230.

10 Biryukov A, Khovratovich D, Pustogarov I. Deanonymisation of clients in Bitcoin P2P network. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. Scottsdale: ACM, 2014. 15–29.

11 Neudecker T, Andelfinger P, Hartenstein H. Timing analysis for inferring the topology of the Bitcoin peer-to-peer network. Proceedings of the 2016 International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). Toulouse: IEEE, 2016. 358–367.

12 Deshpande V, Badis H, George L. Btcmap: Mapping Bitcoin peer-to-peer network topology. Proceedings of the 2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN). Toulouse: IEEE, 2018. 1–6.

13 Grundmann M, Neudecker T, Hartenstein H. Exploiting transaction accumulation and double spends for topology inference in Bitcoin. Proceedings of the 2018 International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2018. 113–126.

14 Delgado-Segura S, Bakshi S, Pérez-Solà C, *et al*. TxProbe: Discovering Bitcoin's network topology using orphan transactions. Proceedings of the 23rd International Conference on Financial Cryptography and Data Security. Frigate Bay: Springer, 2019. 550–566.

15 Essaid M, Park S, Ju H. Visualising Bitcoin's dynamic P2P network topoogy and performance. Proceedings of the 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Seoul: IEEE, 2019. 141–145.

(校对责编: 王欣欣)