

分类号：

密级：

UDC：

学校代码：

东 岛 大 学

硕 士 学 位 论 文

机会网暨蜂窝网算法与协议
—路由，数据分发与能耗优化

由 磊

指 导 教 师 魏长江教授，李建波副教授

学 科 专 业 名 称 软件工程

论 文 提 交 日 期 2015 年 6 月 16 日

论 文 答 辩 日 期 2015 年 6 月 16 日

答 辩 委 员 会 主 席 _____

摘 要

关键词：机会网络 蜂窝网络 路由协议 数据分发 优化算法

Abstract

Keywords: opportunistic network; cellular network; routing protocol; data dissemination; optimization algorithm

目 录

引言	1
I. 无线网络分类概述	1
II. 移动无线网研究历程——从 MANET 到 PCN	1
III. 相辅相成——机会网与蜂窝网技术融合	1
IV. 核心技术简介：机会路由，数据分发与能耗优化	1
IV-A. 机会路由	1
IV-B. 数据分发	1
IV-C. 能耗优化	1
第一章 研究现状	3
1.1 机会路由协议	3
1.2 数据分发协议	3
1.3 蜂窝网能耗优化	3
第二章 论文工作总览	5
第三章 基于移动模式最优节点群组选取的路由算法	7
3.1 系统模型及基本定义	8
3.1.1 网络模型	8
3.1.2 基本定义	8
3.2 路由问题概览	10
3.2.1 移动模式	10
3.2.2 路由相关的两个关键属性	12
3.2.3 路由问题形式化定义	15
3.3 N_{opt} 搜索问题分析	16
3.3.1 计算复杂性证明	16
3.3.2 局部陷阱	19
3.3.3 基于禁忌搜索的求解方法	19
3.4 最优化路由算法	24
3.4.1 Local-MPAR: 基于局部搜索的路由算法	27
3.4.2 Tabu-MPAR: 基于禁忌搜索的路由算法	29
3.5 仿真实验	33

3.5.1 改变消息生存时间	34
3.5.2 改变节点缓存大小	34
3.6 本章小结	39
第四章 基于消息传输收益的最优队列调度算法	41
4.1 系统模型	41
4.2 问题形式化	42
4.2.1 问题目标	42
4.2.2 吞吐量预测	43
4.2.3 问题定义	45
4.3 基于数据项选择的改进路由	46
4.3.1 动态规划求解	46
4.3.2 路由协议描述	46
4.4 仿真实验	49
4.4.1 Cambridge-iMote 场景仿真	49
4.4.2 Helsinki City Scenario 仿真	53
4.5 本章小结	55
第五章 基于跳数的启发式距离向量算法	57
5.1 系统模型	57
5.2 消息投递跳数预测	58
5.2.1 消息收集	58
5.2.2 启发函数	59
5.3 路由协议	62
5.4 仿真实验	64
5.4.1 Helsinki City 场景	65
5.4.2 Cambridge-iMote 场景	65
5.5 本章小结	65
全文总结	67
参考文献	69
攻读学位期间发表的学术论文目录	77
致谢	79
学位论文独创性声明、学位论文知识产权权属声明	81

引　　言

首段先从无线网络分类讲起，然后撇开较成熟的网络，而转到移动无线网上，此外再介绍 4G,5G 技术及亟待攻克的问题，然后，介绍一下引言的结构安排。

I. 无线网络分类概述

主要参考来源： Wikipedia，介绍无线网络分类，及其区别

II. 移动无线网研究历程——从 MANET 到 PCN

主要参考来源： Mobile Ad Hoc Networking [1]

III. 相辅相成——机会网与蜂窝网技术融合

其中一项，可以介绍一下 opportunistic offloading

IV. 核心技术简介：机会路由，数据分发与能耗优化

IV-A. 机会路由

IV-B. 数据分发

IV-C. 能耗优化

第一章 研究现状

1.1 机会路由协议

1.2 数据分发协议

1.3 蜂窝网能耗优化

第二章 论文工作总览

第三章 基于移动模式最优节点群组选取的路由算法

在许多节点具有一定社会属性的机会网络中，具有共同兴趣的移动用户往往访问一些与其兴趣相关的地点。研究表明，50% 的移动用户会在某一个特定的接入点 (access point, AP) 上花费约 74% 的时间 [2]。换言之，节点往往具有频繁访问某一或某一部分地点（简称为常访地点）的特点。这些常访地点可被看做“连接”这些节点的枢纽。可以通过在常访地点部署缓存设备，用以辅助消息传递，例如投掷盒 (throw-box) [3] 等设备。缓存设备具有普通移动节点不具备的优势。首先，由于部署的缓存设备位置固定在常访地点，且节点往往在常访地点停留一段时间，所以节点与缓存设备之间具有比移动节点之间更加稳定的连接。其次，这些固定的缓存设备，没有如同移动节点那样便携性的限制和要求，故其存储容量往往比移动节点要大许多。由此，可以利用在常访地点部署缓存设备作为中继枢纽，从而提升机会网络中路由算法的性能表现。

本章研究多副本单播机会路由算法，基本假设如下。每条消息具有一对“源—目的”节点对 (source-destination node pair)，并且每条消息可在网络中具有多份拷贝。不同于以往基于社会网络分析求解社会关系的方法，或自定义社会属性的方法，本章利用节点收集的移动记录信息，从中提取出节点（群组）移动模式，并以此为依据选出最优中继节点群组。基本思想为：将持有某一条特定消息（或其拷贝）的任意一组节点看成一个整体，从记录信息中提取出移动模式，从而确定该节点群组的常访地点集合 A_1 。目的节点的常访地点 A_2 ，也以相同方法获取。进而，取交集求得 $A = A_1 \cap A_2$ ，作为该节点群组与目的节点的共同常访地点集合。利用集合 A ，可以求出消息的预测投递率。路由目标即为求得能使预测投递概率最大的节点群组。此外，超出有效期 (deadline) 的消息不再具有价值，在求解移动模式的过程中，消息的时效性也被考虑在内。本章基于节点的移动模式，提出构造最优节点群组作为中继节点群的移动模式相关最优路由算法 (Movement Pattern-Aware optimal Routing, MPAR)。就目前掌握的资料来看，这是首次利用节点群组移动模式进行社会相关的路由算法研究。

本章组织如下：第3.1节引入系统模型及基本定义；第3.2节分析路由问题背后蕴含的两个关键属性，并给出最优路由问题的形式化定义；第3.3节证明了该最优问题的计算复杂性，并提出了启发式算法用以求解近似最优解；第3.4节给出路由算法的详细过程及描述；第3.5节对仿真实验结果进行分析；第3.6节总结本章内容。

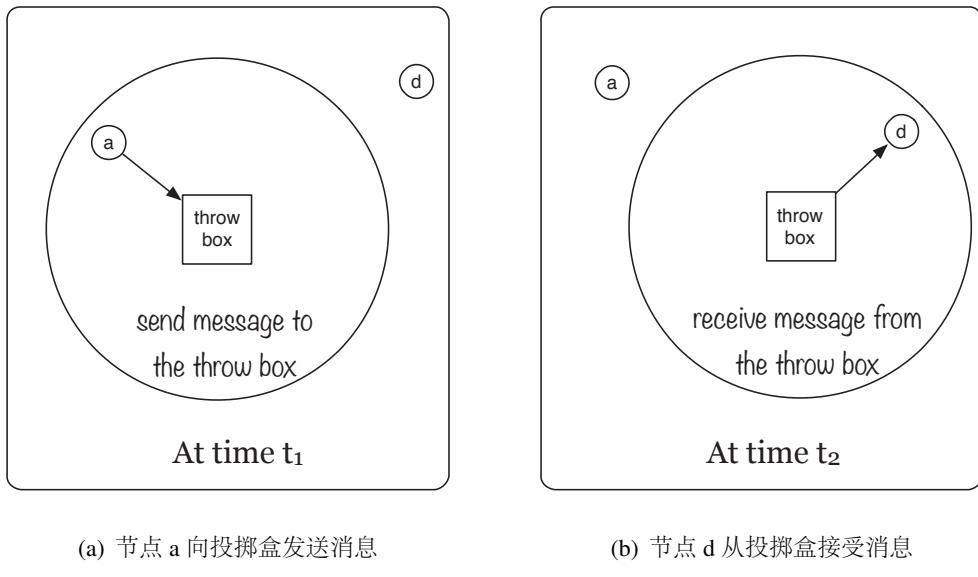


图 3-1 利用投掷盒完成消息从节点 a 到节点 d 的传递操作.

3.1 系统模型及基本定义

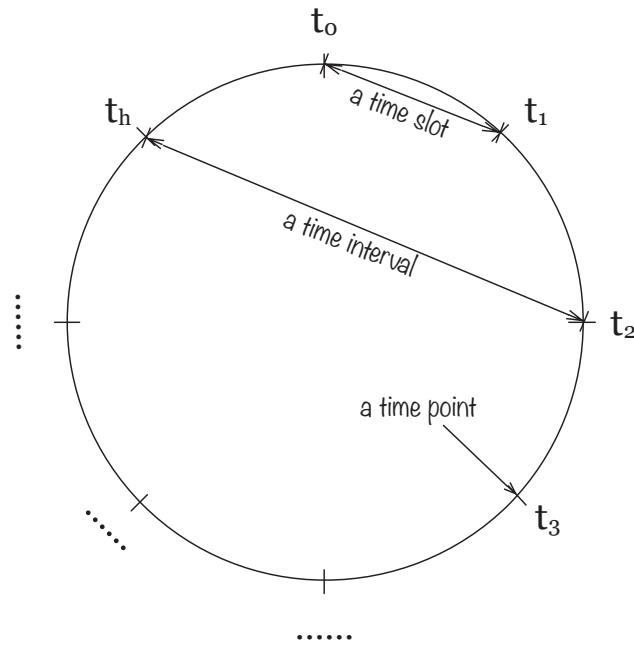
3.1.1 网络模型

网络节点集合记为 $\bar{N} \triangleq \{n_i | 1 \leq i \leq n\}$. 节点在给定的地点集合间移动，地点集合记为 $\bar{A} \triangleq \{a_j | 1 \leq j \leq m\}$. 任意一个节点 n_i 的常访地点集合 $A(n_i) \subseteq \bar{A}$. 该模型源于真实的移动网络，一个典型的实例是 Dartmouth College 的 Wi-Fi Campus Network [4]。另一符合该模型的一类网络是 VANET。在 VANET 中，大量的公交车，电车等在车站等固定地点间移动。我们假设节点 n_i 访问任意一个地点 a_j 的时间间隔服从指数分布。此外，在任意地点内都部署有一个投掷盒装置，用于接受，缓存及传递消息。在此模型中，投掷盒的缓存容量假设为足够大，即在投掷盒中不会因缓存满而产生消息丢弃。如图 3-1 所示，节点 **a** 在 t_1 时刻将消息托管给投掷盒，在 t_2 时刻，该消息的目的节点 **d** 进入投掷盒的传输范围内，并从投掷盒接受该消息，至此，消息的投递操作完成。

我们假设每条消息 l 都具有一个值 τ_l ，代表消息 l 在网络的剩余生存时间。随着时间消逝， τ_l 的值逐渐减少，当 $\tau_l = 0$ 时，该消息将从节点缓存中删除，不再存在于网络中。消息的生存时间属性，一定程度上避免了消息在网络中停留的时间过长。此外，应用程序所产生的消息因具有时效性，往往也会对消息设定一个生存时间。例如新闻发布或者广告发布服务，消息只在一定时间内有效。当消息超出某个时间即不再具有投递价值。

3.1.2 基本定义

由于作为手持设备的节点附属于具有社交属性的人群，节点的移动模式具有一定的周期性。举例而言，Smith 先生在周一到周五上班，故其常访地点可能包含家，办


 图 3-2 周期 T 划分为 h 个时间槽

公室，某些公交站点及一些快餐店等。周末，Smith 先生通常去健身俱乐部或咖啡馆，或者一些不同于工作时间去的地方。然而，他的行为通常以一星期为一个周期，在一定程度上重复，即其移动记录具有一定的周期性。假设周期为 T ，若将 T 划分为 h 个时间槽，则每个时间槽的长度为 $\frac{T}{h}$ 。将这些划分好的时间槽的边界时间点，用序列 $< t_0, t_1, t_2, \dots, t_h >$ 表示，则任意两个区间点 t_s 与 t_e ($t_s < t_e$) 形成时间区间 $[t_s, t_e]$ ，如图图 3-2 所示。针对时间区间 $[t_s, t_e]$ ，可以从节点的移动记录中，提取出对应的移动模式。节点的移动记录见定义 1。

定义 1. 移动记录

节点 n_i 的移动记录定义为一个 $h \times m$ 矩阵，记为 $\mathbb{R}(i)$

$$\mathbb{R}(i) = \left[\underbrace{\begin{matrix} 1/r_{1,1}^i & 1/r_{1,2}^i & \dots & 1/r_{1,m}^i \\ 1/r_{2,1}^i & 1/r_{2,2}^i & \dots & 1/r_{2,m}^i \\ \vdots & \vdots & \ddots & \vdots \\ 1/r_{h,1}^i & 1/r_{h,2}^i & \dots & 1/r_{h,m}^i \end{matrix}}_{m\text{个地点}} \right] \left. \right\} h\text{个时间槽}$$

其中第 k 行的向量 $\mathbb{R}(i, k)$ 代表时间槽 $[t_{k-1}, t_k]$ 内的移动记录，且有

$$\mathbb{R}(i, k) = [1/r_{k,1}^i, 1/r_{k,2}^i, \dots, 1/r_{k,m}^i]$$

其中 $r_{k,j}^i$ 代表节点 n_i 对地点 a_j 在时间槽 $[t_{k-1}, t_k]$ 内的平均访问时间间隔。

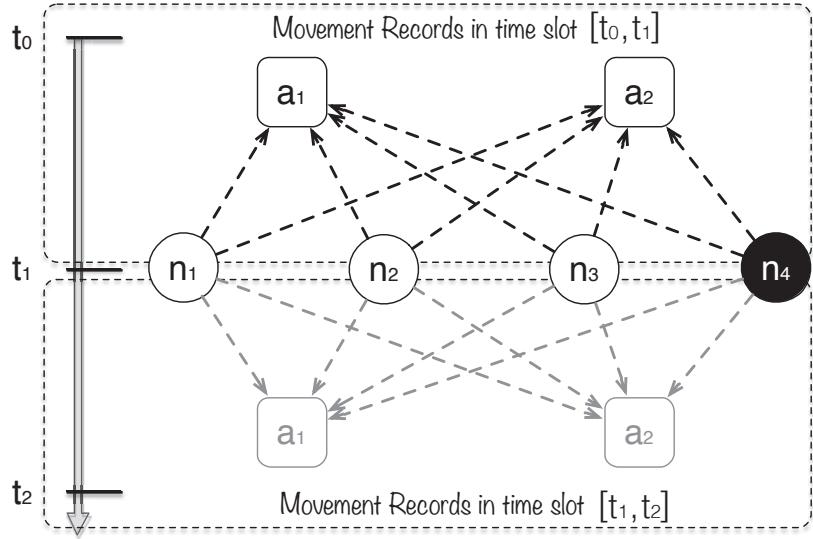


图 3-3 网络场景举例

从上述定义可知, $1/r_{k,j}^i$ 代表节点 n_i 对地点 a_j 的平均访问频率。特别的, 若 n_i 在时间槽 $[t_{k-1}, t_k]$ 内从未到达过 a_j , 则设定 $r_{k,j}^i = \infty$, 于是有 $1/r_{k,j}^i = 0$. 对于所有的时间槽, 可以求出节点 n_i 对于地点 a_j 的平均访问时间间隔 $M_{i,j}$, 记为

$$M_{i,j} = \sum_{k=1}^{h} r_{k,j}^i / h \quad (3-1)$$

3.2 路由问题概览

在讨论路由相关细节之前, 先对路由问题做一个总览。在3.2.1小节中, 讨论如何从一组节点中提取对应的节点群组移动模式。随后在3.2.2小节中, 分析路由相关的两个关键属性。最后, 在3.2.3小节中, 给出最优路由问题的形式化定义。

3.2.1 移动模式

移动模式从节点的移动记录中提取。首先定义函数 \mathbb{E} , 用以将移动记录向量转化为对应的移动模式向量。

定义 2. 函数 \mathbb{E}

$$\mathbb{E}([x_1, x_2, \dots, x_m]) = [\varkappa_1, \varkappa_2, \dots, \varkappa_m]$$

其中

$$\varkappa_j = \begin{cases} 1 & x_j \geq \frac{\delta}{m} \sum_{i=1}^m x_i \\ 0 & \text{otherwise} \end{cases} \quad (3-2)$$

$0 < \delta < 1$ 是预设的系统参数。

函数 \mathbb{E} 用于过滤节点—地点间的访问记录，不常访问的地点将被过滤掉。基于函数 \mathbb{E} 可以直接定义移动模式，如定义3。

定义 3. 移动模式. 对于任意节点群组 N , 其在时间区间 $[t_p, t_q]$ 内的移动模式, 记为 $\mathcal{P}(V, [t_s, t_e])$, 定义为

$$\mathcal{P}(N, [t_s, t_e]) = \mathbb{E}(\sum_{n_x \in N} \sum_{i=s}^{i=e} \mathbb{R}(x, t_i)) \quad (3-3)$$

在定义3中, 移动模式 \mathcal{P} 从某节点(群组) 在某个特定时间区间 $[t_s, t_e]$ 上的所有移动记录向量通过累加后, 以函数 \mathbb{E} 处理后得出。举例而言, 假设如图 3-3所示的网络场景, 网络中总共有四个节点 n_1, n_2, n_3, n_4 以及两个地点 a_1, a_2 。其中 n_4 是目的节点, 周期 T 被划分为两个时间槽, 简记为 $[t_0, t_1]$ 与 $[t_1, t_2]$ 。

图 3-3中每条边的权值如表 3-1所示。权值代表“节点—地点”对访问平均间隔时间。针对每个节点的记录矩阵, 表示为 $\mathbb{R}(1), \mathbb{R}(2), \mathbb{R}(3), \mathbb{R}(4)$ 。

$$\begin{aligned} \mathbb{R}(1) &= \begin{bmatrix} 1/2.08 & 1/4.65 \\ 1/6.02 & 1/2.95 \end{bmatrix} & \mathbb{R}(2) &= \begin{bmatrix} 1/4.4 & 1/4.3 \\ 1/4.0 & 1/4.5 \end{bmatrix} \\ \mathbb{R}(3) &= \begin{bmatrix} 1/8.05 & 1/1.11 \\ 1/6.05 & 1/15.49 \end{bmatrix} & \mathbb{R}(4) &= \begin{bmatrix} 1/2.6 & 1/3.3 \\ 1/3.5 & 1/3.5 \end{bmatrix} \end{aligned}$$

表 3-1 网络场景图 3-3 边权值

		a_1	a_2
$[t_0, t_1]$	n_1	2.08	4.65
	n_2	4.4	4.3
	n_3	8.05	1.11
	n_4	2.6	3.3
$[t_1, t_2]$	n_1	6.02	2.95
	n_2	4.0	4.5
	n_3	6.05	15.49
	n_4	3.5	3.5

节点集合 $\{n_1, n_2, n_3\}$ 的移动模式, 可以利用公式3-3从对应的移动记录中提取出来。对应的移动记录累加计算如下:

$$\begin{aligned} \sum_{i=1}^{i=3} \sum_{j=1}^{j=2} \mathbb{R}(i, j) &= \left[\frac{1}{2.08} + \frac{1}{4.4} + \frac{1}{8.05} + \frac{1}{6.02} + \frac{1}{4.0} + \frac{1}{6.05}, \right. \\ &\quad \left. \frac{1}{4.65} + \frac{1}{4.3} + \frac{1}{1.11} + \frac{1}{2.95} + \frac{1}{4.5} + \frac{1}{15.49} \right] = [1.414, 1.974] \end{aligned}$$

针对公式3-2，设定参数 $\delta = 0.95$ ，从定义2得出

$$\frac{\delta}{m} \sum_{i=1}^{i=m} v_i = \frac{0.95}{2} \times (1.414 + 1.974) \approx 1.609$$

所得值 1.609 大于向量第一个元素值 1.414，且小于向量第二个元素值 1.974，故对应移动模式提取如下

$$\mathcal{P}(\{n_1, n_2, n_3\}, [t_0, t_2]) = [0, 1]$$

移动模式 \mathcal{P} 指明了节点群组 $\{n_1, n_2, n_3\}$ 在时间区间 $[t_0, t_2]$ 内的常访区域集合，即 $A(\{n_1, n_2, n_3\}, [t_0, t_2]) = \{a_2\}$ 。对于集合 \bar{N}

$\{n_d\}$ ，共有 $2^{n-1} - 1$ 个非空子集，每一个子集可被看做用于向目的节点 n_d 合作投递消息的一个整体。在本例中，基于移动记录，提取出非空节点集 $\{n_1, n_2, n_3\}$ 以及目的节点 n_4 的移动模式，结果如下

$$\begin{aligned} \mathcal{P}(\{n_1\}, [t_0, t_2]) &= [1, 0] & \mathcal{P}(\{n_1, n_2\}, [t_0, t_2]) &= [1, 0] \\ \mathcal{P}(\{n_2\}, [t_0, t_2]) &= [1, 1] & \mathcal{P}(\{n_1, n_3\}, [t_0, t_2]) &= [0, 1] \\ \mathcal{P}(\{n_3\}, [t_0, t_2]) &= [0, 1] & \mathcal{P}(\{n_2, n_3\}, [t_0, t_2]) &= [0, 1] \\ \mathcal{P}(\{n_4\}, [t_0, t_2]) &= [1, 1] & \mathcal{P}(\{n_1, n_2, n_3\}, [t_0, t_2]) &= [0, 1] \end{aligned}$$

对应的常访地点集合，如下

$$\begin{aligned} A(\{n_1\}, [t_0, t_2]) &= \{a_1\} & A(\{n_1, n_2\}, [t_0, t_2]) &= \{a_1\} \\ A(\{n_2\}, [t_0, t_2]) &= \{a_1, a_2\} & A(\{n_1, n_3\}, [t_0, t_2]) &= \{a_2\} \\ A(\{n_3\}, [t_0, t_2]) &= \{a_2\} & A(\{n_2, n_3\}, [t_0, t_2]) &= \{a_2\} \\ A(\{n_4\}, [t_0, t_2]) &= \{a_2\} & A(\{n_1, n_2, n_3\}, [t_0, t_2]) &= \{a_1, a_2\} \end{aligned}$$

3.2.2 路由相关的两个关键属性

3.2.2.1 投递概率

设节点集合 N 在时间区间 $[t_0, t_0 + t_l]$ 内的移动模式为 $P(N, [t_0, t_0 + t_l])$ ，目的节点 n_d 的移动模式为 $\mathcal{P}(n_d, [t_0, t_0 + \tau_l])$ ，共同常访地点集合记为 A ，则有 $A = A(N, [t_0, t_0 + \tau_l]) \cap A(n_d, [t_0, t_0 + \tau_l])$ 。记任意地点 a_j 及任意节点 $n_i \in N$ 满足节点 n_i 和 n_d 都在某一时间访问 a_j 。设随机变量 $T_{i,j}$ 及 $T_{d,j}$ 分别代表 n_i 和 n_d 到达 a_j 的时间，若要使消息能够成功传递，则须满足 $T_{i,j} < T_{d,j} < \tau_l$ ，换言之， n_i 须先于 n_d 到达 a_j

托管消息，且 n_d 需在消息到期之前到达 a_j 取出消息。为了便于讨论，设起始时间点 $t_0 = 0$ 。此外，如3.1节所述，任意节点访问任意地点的时间间隔服从指数分布。从公式3-1可以计算出节点 n_i 访问地点 a_j 的指数分布参数 $\lambda_{i,j}$ ，即

$$\lambda_{i,j} = 1/M_{i,j}$$

对于 A 为空集的情形，默认其预测投递概率为 0，到达其它地点的期望时延为 ∞ 。对于 A 非空的情形，可依公式 (4-4) 预测节点 n_i 对 n_d 的投递率。进一步的，节点集 N 对 n_d 的投递率可依公式 (3-5) 预测。

$$\begin{aligned} P_{i,d} &= 1 - \prod_{a_j \in A} (1 - P(T_{i,j} < T_{d,j} < \tau_l)) \\ &= 1 - \prod_{a_j \in A} \left(1 - \int_0^{\tau_l} \int_{t_{i,j}}^{\tau_l} f_{T_{i,j}, T_{d,j}}(t_{i,j}, t_{d,j}) dt_{d,j} dt_{i,j} \right) \\ &= 1 - \prod_{a_j \in A} \left(1 - \int_0^{\tau_l} \int_{t_{i,j}}^{\tau_l} f_{T_{i,j}}(t_{i,j}) f_{T_{d,j}}(t_{d,j}) dt_{d,j} dt_{i,j} \right) \\ &= 1 - \prod_{a_j \in A} \left(1 - \int_0^{\tau_l} \int_{t_{i,j}}^{\tau_l} \lambda_{d,j} \lambda_{i,j} e^{-(\lambda_{d,j} t_{d,j} + \lambda_{i,j} t_{i,j})} dt_{d,j} dt_{i,j} \right) \\ &= 1 - \prod_{a_j \in A} \left(1 - \frac{\lambda_{i,j} (1 - e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})})}{\lambda_{d,j} + \lambda_{i,j}} - (e^{\tau_l \lambda_{i,j}} - 1) e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})} \right) \quad (3-4) \end{aligned}$$

$$\begin{aligned} P_{N,d} &= 1 - \prod_{n_i \in N} (1 - P_{i,d}) \\ &= 1 - \prod_{n_i \in N} \prod_{a_j \in A} \left(1 - \frac{\lambda_{i,j} (1 - e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})})}{\lambda_{d,j} + \lambda_{i,j}} - (e^{\tau_l \lambda_{i,j}} - 1) e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})} \right) \quad (3-5) \end{aligned}$$

特别而言，当消息生存时间设为无穷时，即 $\tau_l = \infty$ ，有

$$\begin{aligned} P_{i,d} &= 1 - \prod_{a_j \in A} \left(1 - \int_0^{\infty} \int_{t_{i,j}}^{\infty} \lambda_{d,j} \lambda_{i,j} e^{-(\lambda_{d,j} t_{d,j} + \lambda_{i,j} t_{i,j})} dt_{d,j} dt_{i,j} \right) \\ &= 1 - \prod_{a_j \in A} \left(1 - \frac{\lambda_{i,j}}{\lambda_{d,j} + \lambda_{i,j}} \right) \quad (3-6) \end{aligned}$$

对于 $P_{N,d}$ ，有

$$P_{N,d} = 1 - \prod_{n_i \in N} \prod_{a_j \in A} \left(1 - \frac{\lambda_{i,j}}{\lambda_{d,j} + \lambda_{i,j}} \right) = 1 - \prod_{n_i \in N} \prod_{a_j \in A} \frac{\lambda_{d,j}}{\lambda_{d,j} + \lambda_{i,j}} \quad (3-7)$$

对于图 3-3 中的例子，为简便起见，讨论 $\tau_l = \infty$ 的情况（形式化定义时，会给出一般情况下的定义）。对于节点集合 $\{n_1, n_2, n_3\}$ 而言，常访地点集合 $A(N, [t_0, t_3]) = \{a_2\}$ ；对于目的节点 n_d 而言，其常访地点集合 $A(n_4, [t_0, t_3]) = \{a_1, a_2\}$ ；由此可得出共同常访地点 $A = \{a_2\}$ 。每一个“节点—地点”对的平均访问时间间隔，可从公式 (3-1) 获得，即

$$\begin{aligned} M_{1,1} &= 4.05 & M_{2,1} &= 4.2 & M_{3,1} &= 7.05 & M_{4,1} &= 3.05 \\ M_{1,2} &= 3.8 & M_{2,2} &= 4.4 & M_{3,2} &= 8.3 & M_{4,2} &= 3.4 \end{aligned}$$

由以上计算结果，可预测节点集合 N 的投递概率，如下

$$\begin{aligned} P_{\{n_1, n_2, n_3\}, 4} &= 1 - \left(\frac{\lambda_{4,2}}{\lambda_{1,2} + \lambda_{4,2}} \right) \left(\frac{\lambda_{4,2}}{\lambda_{2,2} + \lambda_{4,2}} \right) \left(\frac{\lambda_{4,2}}{\lambda_{3,2} + \lambda_{4,2}} \right) \\ &= 1 - \left(\frac{M_{1,2}}{M_{1,2} + M_{4,2}} \right) \left(\frac{M_{2,2}}{M_{2,2} + M_{4,2}} \right) \left(\frac{M_{3,2}}{M_{3,2} + M_{4,2}} \right) = 0.789 \end{aligned}$$

3.2.2.2 期望时延

除了投递概率之外，另一个关键的属性即节点移动到其他地点的期望时延。设节点 n_i 的常访地点集合为 A_i ，用随机变量 D_i 代表 n_i 移动到下一个地点 $a_j \in A_i$ 的期望时延。形式上，有 $D_i = \min\{T_{i,1}, T_{i,2}, \dots, T_{i,|A_i|}\}$ ，由此可求得 D_i 的概率密度函数为

$$f_{D_i}(t) = \sum_{a_k \in A_i} \lambda_{i,k} \prod_{a_h \in A_i} e^{-\lambda_{i,h} t}$$

于是有

$$\begin{aligned} E[D_i] &= \int_{-\infty}^{\infty} t f_{D_i}(t) dt = \sum_{a_k \in A_i} \int_0^{\infty} t \lambda_{i,k} \prod_{a_h \in A_i} e^{-\lambda_{i,h} t} dt \\ &= \sum_{a_k \in A_i} \frac{\lambda_{i,k}}{\left(\sum_{a_h \in A_i} \lambda_{i,h} \right)^2} = \frac{1}{\sum_{a_k \in A_i} \lambda_{i,k}} \quad (3-8) \end{aligned}$$

对于图 3-3 所示例子，可以分别对节点 n_1, n_2 及 n_3 根据公式 (3-8) 计算期望时延 $E[D_i]$ 。

$$\begin{aligned} E[D_1] &= 1/\lambda_{1,1} = M_{1,1} = 4.05 \\ E[D_2] &= \frac{1}{\lambda_{2,1} + \lambda_{2,2}} = \frac{M_{2,1} M_{2,2}}{M_{2,1} + M_{2,2}} = 2.15 \\ E[D_3] &= 1/\lambda_{3,2} = M_{3,2} = 8.3 \end{aligned}$$

对于任意一对节点 n_i 和 n_j ，若 $E[D_i] < E[D_j]$ ，则从期望意义上讲， n_i 到达下一个新地点的时间要早于 n_j ，这意味着 n_i 能够更早的将消息托管到新的投掷盒。在机会网络中多使用多拷贝策略进行路由，快速的向网络中传播足够数目的消息副本，对提升路由性能至关重要。期望时延 $E[D]$ 是一个很好的衡量指标，应尽可能的利用 $E[D]$

较小的节点分发消息。在本章提出的基于局部搜索的路由算法 Local-MPAR 中, $E[D]$ 被用于衡量节点作为网络中唯一的活跃节点是否合适。在基于禁忌搜索的路由算法 Tabu-MPAR 中, $E[D]$ 被用作向网络中非对称喷洒消息副本的衡量指标。在表 3-1 所示例子中, 节点 n_2 最适合作为活跃节点, 因为其期望时延 $E[D]$ 最低, 因此在期望意义上, n_2 可以较早的将消息拷贝托管给其它地点的投掷盒。

3.2.3 路由问题形式化定义

机会网络中的路由问题, 其优化目标可有多种。衡量路由算法性能的最主要的两个指标, 即是消息投递率和平均端到端时延。尽管端到端时延在路由优化中被看做一项非常重要的指标, 但路由的性能表现仍需要以可被接受的投递概率为基础。换言之, 较高的投递概率是保证网络正常工作的基础。本章意图最大化消息的投递概率, 进而提高路由性能表现。即寻找子集 $N \subseteq \bar{N}$, 从而使得 $P_{N,d} \geq P_{N',d}$ 对于 $\forall N' \subseteq \bar{N}$ 成立, 其中 n_d 是消息的目的节点。

在图 3-3 所示例子中, 可依 3.2.2 小节中公式 (3-5), 预测集合 $\{n_1, n_2, n_3\}$ 所有非空子集的投递率, 如下

$$\begin{array}{ll} P_{\{n_1\},4} = 0.430 & P_{\{n_1,n_2\},4} = 0.670 \\ P_{\{n_2\},4} = 0.673 & P_{\{n_2,n_3\},4} = 0.600 \\ P_{\{n_3\},4} = 0.291 & P_{\{n_1,n_3\},4} = 0.626 \\ P_{\{n_1,n_2,n_3\}} = 0.789 & \end{array}$$

在本例中, 最优集合即 $\{n_1, n_2, n_3\}$ 。注意计算结果为估计投递率, 并不直接与持有该消息的节点数目成正比。投递率 $P_{N,d}$ 基于节点集合 N , N 在整个路由过程中被看做一个整体。对于某个 $N' \subset N$ 而言, 概率 $P_{N,d} < P_{N',d}$ 是可能发生的。事实上, 如图 3-3 所示, 对于集合 $\{n_2, n_3\}$ 而言, 其预测投递率低于集合 $\{n_2\}$ 的预测投递率。如果忽略缓存限制和能耗资源, 向整个网络中广播每条消息会获得最好的性能表现, 因为在这种情况下增加中继节点的数目, 可以提高消息的投递率 (或者说至少不可能降低)。然而这种理想的情况在机会网络中往往并不存在。真实情况下, 往往倾向于选择能够符合一定移动模式的最小节点群组, 对消息进行投递。从例中可以看出, 如果把持有该消息的所有节点视为一个整体, 则其预测投递率被两项指标所影响: (1) 节点集对应的移动模式, (2) 节点集的大小。在正式定义路由问题之前, 首先给出最优节点集合 N_{opt} 的定义。

定义 4. The optimal set N_{opt}

最优集合 N_{opt} 包含网络中的一组节点, 且有

$$N_{opt} = \min \left\{ \arg \max_{N \subseteq \bar{N}} P_{N,d} \right\}$$

最优集合是能够取得最高预测投递率的最小集合，集合中的节点以合作的方式将消息传递给目的节点。基于此，可定义最优路由问题。

定义 5. N_{opt} 搜索问题。

令 $f(\mathbf{x})$ 代表目标函数， $g(\mathbf{x})$ 代表限制函数，其中决策变量表示为 $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ ，

$$\text{有 } f(\mathbf{x}) = \sum_{i=1}^{i=n} x_i \log \left(\prod_{a_j \in A} \left(1 - \frac{\lambda_{i,j} (1 - e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})})}{\lambda_{d,j} + \lambda_{i,j}} - (e^{\tau_l \lambda_{i,j}} - 1) e^{-\tau_l (\lambda_{d,j} + \lambda_{i,j})} \right) \right)$$

$$g(\mathbf{x}) = \left| \sum_{i=1}^{i=n} x_i \right|$$

于是问题可以被形式化为如下

$$\begin{aligned} & \min f(\mathbf{x}) \\ s.t. \quad & \forall \mathbf{x}', g(\mathbf{x}') - g(\mathbf{x}) \geq 0, \\ & \forall i, x_i \in \{0, 1\} \end{aligned}$$

当求出最优节点集合 \mathbf{x}_{opt} 时，有

$$N_{opt} = \{n_i | x_i > 0, 1 \leq i \leq n\}$$

在上述定义中，存在一个 \mathbf{x} 与 N 之间的一对一映射。事实上，有 $f(\mathbf{x}) = \log(1 - P_{N,d})$ ，其中 $f(\mathbf{x})$ 与 $P_{N,d}$ 成负相关。由此， N_{opt} 搜索问题被形式化为最优化问题。

3.3 N_{opt} 搜索问题分析

本节讨论 N_{opt} 问题的计算复杂性，并提出了一种启发式方法，以近似计算最优解。首先证明对于在线算法 (online algorithm)，对于算法输入序列中新加入的元素，能够很大程度改变目标函数的值。对于离线算法 (offline algorithm)，给出了 \mathcal{NP} -hard 性质证明。最后，本章提出基于禁忌搜索的启发式算法。

3.3.1 计算复杂性证明

在线算法可以分片处理输入序列。换言之，算法的输入是以一个序列方式随时间交付由算法本身处理的，而不同于离线算法，需要一次接受整个算法输入。对于 N_{opt} ，其在线算法的输入是网络中节点的移动记录，简单表示为 $\mathbb{R}_1, \mathbb{R}_2, \dots, \mathbb{R}_q$ (其中序列中

的每个元素都是一个向量，例如对于在时间槽 t_i 的节点 n_x 可以有 $\mathbb{R}_j = \mathbb{R}(x, t_i)$ 。假设当前只有序列的前 $1 - k$ 个元素被输入算法。剩下的问题即探究当下一个新元素 \mathbb{R}_{k+1} 被加入网络时，会在多大程度上影响目标函数的值，见定理 1

定理 1. 对于 $\forall k \in [1, z]$ ，即使序列的前 $1 - k$ 部分已知，新加入的元素 \mathbb{R}_{k+1} 仍可能将移动模式改变为至少有一个元素不为 0 的任意向量。

证明. 将任意向量 R_i 表示为如下

$$R_i = [r_1^i, r_2^i, \dots, r_m^i] \quad \forall j \in [1, m], r_j^i \geq 0$$

向量 $\sum_{i=1}^{i=k} \mathbb{R}_i$ 表示为

$$\sum_{i=1}^{i=k} \mathbb{R}_i = [S_1^k, S_2^k, \dots, S_m^k]$$

于是对于 $\forall i \in [1, m]$ 及 $k \in [1, z]$ 有

$$S_i^{k+1} = S_i^k + r_i^{k+1}$$

相应地，用 $\mathcal{P}' = \mathbb{E}(\sum_{i=1}^{i=k+1} \mathbb{R}_i)$ 和 $\mathcal{P} = \mathbb{E}(\sum_{i=1}^{i=k+1} \mathbb{R}_i)$ 分别表示在加入新元素 \mathbb{R}_{k+1} 之前和之后的移动模式向量。现证明对于任意的 \mathcal{P} ，都存在一个可能的新元素 \mathbb{R}_{k+1} ，能使向量 \mathcal{P}' 变为 \mathcal{P} 。当已知 $\mathcal{P} = [\varkappa_1, \varkappa_2, \dots, \varkappa_m]$ 时，显然 \mathbb{R}_{k+1} 存在，当且仅当如下线性不等式组成立

$$\forall i \in [1, m] \quad \begin{cases} S_i^{k+1} \geq \delta \sum_{j=1}^{j=m} (S_j^{k+1}/m) & \varkappa_i = 1 \\ S_i^{k+1} < \delta \sum_{j=1}^{j=m} (S_j^{k+1}/m) & \varkappa_i = 0 \end{cases}$$

即

$$\begin{cases} \sum_{j=1}^{j=m} r_j^{k+1} - \frac{m}{\delta} r_i^{k+1} \leq \frac{m}{\delta} S_i^k - \sum_{j=1}^{j=m} S_j^k & \varkappa_i = 1 \\ \sum_{j=1}^{j=m} r_j^{k+1} - \frac{m}{\delta} r_i^{k+1} > \frac{m}{\delta} S_i^k - \sum_{j=1}^{j=m} S_j^k & \varkappa_i = 0 \end{cases}$$

对于 $\forall i$ ，将 $\frac{m}{\delta} S_i^k - \sum_{j=1}^{j=m} S_j^k$ 表示为 w_i ，并将 $\varkappa_i = 1$ 和 $\varkappa_i = 0$ 时的 w_i 分别表示为 $w_{i|\varkappa_i=1}$ 和 $w_{i|\varkappa_i=0}$ 。同理，用 r_i^{k+1} 表示 $\varkappa_i = 1$ 和 $\varkappa_i = 0$ 时的对应状态，有

$$\sum_{j=1}^{j=m} r_j^{k+1} - \frac{m}{\delta} r_{i|\varkappa_i=0}^{k+1} > w_{i|\varkappa_i=0} \tag{3-9}$$

且有

$$\sum_{j=1}^{j=m} r_j^{k+1} - \frac{m}{\delta} r_{i|\varkappa_i=1}^{k+1} \leq w_{i|\varkappa_i=1} \tag{3-10}$$

令 $r_{i|\varkappa_i=0}^{k+1} = 0$ 及 $\sum_{j=1}^{j=m} r_j^{k+1} = \max\{w_i\} + \epsilon$ ($\epsilon > 0$)，则等式 (3-9) 对于 $\forall \varkappa_i = 0$ 都成立。

接下来证明可以选择足够大的 ϵ 值，使得等式 3-10 成立，即只需让 $\sum_{j=1}^{j=m} r_j^{k+1} - \frac{m}{\delta} r_{i|\varkappa_i=1}^{k+1} \leq \min\{w_i\}$ 对于 $\forall \varkappa_i = 1$ 都成立，即

$$\max\{w_i\} + \epsilon - \frac{m}{\delta} r_{i|\varkappa_i=1}^{k+1} \leq \min\{w_i\}$$

从上述讨论, 只需让下述不等式同时成立.

$$r_{i|\varkappa_i=1}^{k+1} \geq \frac{\delta}{m}(\max\{w_i\} - \min\{w_i\} + 1) \quad (0 < \delta < 1) \quad (3-11)$$

$$\sum_{j=1}^{j=m} r_{j|\varkappa_j=1}^{k+1} = \max\{w_i\} + \epsilon \quad (3-12)$$

假设 \mathcal{P} 中 $\varkappa_i = 1$ 的数量是 z , 由于 $0 < z \leq m$, 有

$$z \cdot \frac{\delta}{m}(\max\{w_i\} - \min\{w_i\} + \epsilon) \leq \delta(\max\{w_i\} - \min\{w_i\} + \epsilon)$$

为使等式 (3-11) 和 (3-12) 同时成立, 只须有

$$\delta(\max\{w_i\} - \min\{w_i\} + \epsilon) < \max\{w_i\} + \epsilon = \sum_{j=1}^{j=m} r_{j|\varkappa_j=1}^{k+1}$$

即

$$\epsilon > \frac{\delta}{\delta - 1} \min\{w_i\} - \max\{w_i\}$$

证毕 □

定理 2 给出 N_{opt} 计算复杂性的证明。

定理 2. N_{opt} 问题属于 \mathcal{NP} -hard 类.

证明. 将子集和问题 (Subset Sum Problem, SSP) 归约为 N_{opt} 问题。假设已知 SSP 问题的一个解实例, 用 $R = r_1, r_2, \dots, r_n$, 目标值用 v 表示。为清楚起见, 将 R 变为如下形式

$$R = \{\log 2^{r_1}, \log 2^{r_2}, \dots, \log 2^{r_n}\} = \{\log g_1, \log g_2, \dots, \log g_n\}$$

其中有 $g_i = 2^{r_i}$ ($i \in [1, n]$). 将其所有非空子集表示为 $R_1, R_2, \dots, R_{2^{|N|-1}}$ 。接下来, 构造对应的 N_{opt} 实例。目标值为 $P = v$, 节点集合为 $N = n_1, n_2, \dots, n_n$, 对应上述提到的集合 R . 对于每一个 N 的非空子集 N_i , 都存在一个 R 的子集 R_i 与其一一对应(所有的角标都是一一对应的), 将任意子集 N_j 表示为如下

$$N_j = \{n_{j_1}, n_{j_2}, \dots, n_{j_k}\}$$

对于 $\forall j$, 令 $\lambda_{i,j} = 1$, $\tau_l = \sqrt{2}$, 对于 $\forall n_i \in N - d$, 令 $\lambda_{i,j} = \Lambda_i$. 此外, 令所有的节点都具有相同的移动记录 $\mathbb{R} = [0, 0, \dots, 0]$. 于是, 对于网络中所有的节点, 都有

$$\mathcal{P} = [1, 1, \dots, 1]$$

于是有

$$P_{N_j, d} = 1 - \prod_{n_i \in N} (1 - \Lambda_i e^{-\Lambda_i - 1})^n$$

令 $\kappa_i = (1 - \Lambda_i e^{-\Lambda_i - 1})^n$, 则有

$$P_{N_j,d} = 1 - \kappa_{j_1} \kappa_{j_2} \cdots \kappa_{j_k}$$

由此, 对于问题 SSP 的任意一个解实例, 一定有一个问题 N_{opt} 的解实例与之对应。换言之

$$v = \log(g_{j_1} g_{j_2} \cdots g_{j_k}) \iff P_{N_j,d} = 1 - \kappa_{j_1} \kappa_{j_2} \cdots \kappa_{j_n}$$

反之, 若 SSP 问题无解, 则 N_{opt} 问题一定也无解. 因为若 N_{opt} 有解, 则存在一个集合 $R_j = r_{j_1}, r_{j_2}, \dots, r_{j_n}$ 使下式成立

$$P = P_{N_j}$$

与题设矛盾。

证毕 □

3.3.2 局部陷阱

对于 N_{opt} , 一种直接的解法是局部搜索算法。然而, 这有可能会陷入局部最优陷阱。对图 3-3 中例子而言, 假设节点 n_2 产生一条目的节点为 n_4 的消息, 则其局部搜索过程如表 3-2 所示。

表 3-2 图 3-3 中的局部搜索过程示例

	current solution			next options		
	\mathbf{x}^{now}	\mathbf{N}	$P_{N,4}$	\mathbf{x}^{next}	\mathbf{N}'	$P_{N',4}$
Step 1	[0, 1, 0]	{ n_2 }	0.673	[0, 1, <u>1</u>]	{ n_2, n_3 }	0.600
				[0, <u>0</u> , 0]	ϕ	0
				[<u>1</u> , 1, 0]	{ n_1, n_2 }	0.670
Stop	[0, 1, 0]	{ n_2 }	0.673	—	—	—

如图 3-2, 在开始时只有节点 n_2 持有该消息, 对于 $N = \{n_2\}$, 其预测投递率为 0.673, 高于其周围 i 任何一个可行解所对应的目标值。算法会误认为当前解 \mathbf{x}^{now} 即是最优解, 然而真正的最优解是 [1, 1, 1], 对应预测投递率 0.789。在本例中, 算法掉入了局部最优陷阱, 而局部搜索算法无法跳出该陷阱。

3.3.3 基于禁忌搜索的求解方法

本小节基于禁忌搜索, 提出解决 N_{opt} 的启发式算法, 框架见算法 1 所示。

禁忌搜索过程可被分为两步, 即 Step 1 和 Step 2, 其基本规则和数据结构见算法 1 内的表格。其中数学符号的正式定义将在下文给出。整个搜索过程即是在解空间

Algorithm 1 Tabu Search Framework.**Require:**

notation	meaning
p	Evaluation function
\mathcal{N}	Neighborhood of a solution
\mathcal{S}	Solution space
\mathcal{E}	Stop rules
\mathcal{T}	tabu table $\left\{ \begin{array}{c c} \varphi & \text{Tabu element} \\ \hline \mathcal{L} & \text{Tabu length} \end{array} \right\}$
\mathcal{A}	Aspiration criteria

Step 1:

- 1: choose an initial feasible solution \mathbf{x}^{now} in \mathcal{S}
- 2: set the current best solution $\mathbf{x}^{best} \leftarrow \mathbf{x}^{now}$
- 3: set the tabu table $\mathcal{T} \leftarrow \emptyset$

Step 2:

- 1: **if** $\mathcal{E}(\mathbf{x}^{best}) = \text{true}$ **then**
 - 2: **return** \mathbf{x}^{best}
 - 3: **end if**
 - 4: get C by filtering $\mathcal{N}(\mathbf{x}^{now})$ according to \mathcal{T} and \mathcal{A}
 - 5: $\mathbf{x}^{now} \leftarrow \arg \max_{\mathbf{x} \in C} p(\mathbf{x})$
 - 6: **if** $p(\mathbf{x}^{now}) < p(\mathbf{x}^{best})$ **then**
 - 7: $\mathbf{x}^{best} \leftarrow \mathbf{x}^{now}$
 - 8: **end if**
 - 9: update \mathcal{T}
 - 10: **goto** Step 2
-

中一步一步更改当前可行解，同时评估该可行解对应的目标函数值，从而尝试获得最优解。Step 1 是禁忌搜索的初始化步骤，其中算法将在解空间 S 中选择一个可行解作为当前解，记为 \mathbf{x}^{now} 。接着，最优解也被初始化为 \mathbf{x}^{now} ，在此时，禁忌表 \mathcal{T} 中并无记录，如 Step 1 中行 3 所示。

在 Step 2 中，若停止规则 \mathcal{E} 被满足，则搜索过程终止并返回当前最优解 \mathbf{x}^{best} 作为最终结果。若不满足停止规则，则从 \mathbf{x}^{now} 的所有邻居 $\mathcal{N}(\mathbf{x}^{now})$ 中选择一个可行解更新 \mathbf{x}^{now} ，该选择过程参照禁忌表 \mathcal{T} 的状态，以及预先定义的特赦准则 \mathcal{A} 。接着，算法将搜索子集 C ，并选择能够使函数 p 取得最大值的 \mathbf{x} 解向量。在 Step 2 的每一轮迭代中，都会尝试更新最优解 \mathbf{x}^{best} 。最终，禁忌表 \mathcal{T} 中对应的禁忌元素 ϕ 将会被更新为预先定义的禁忌步长 $\mathcal{L}(\phi)$ 。

下面给出算法1中所有数学符号的定义。

定义 6. 评估函数.

禁忌搜索中的评估函数表示为 $p(\mathbf{x})$, 有

$$p(\mathbf{x}) = -f(\mathbf{x})$$

在定义 6 中，评估函数设为对定义 5 中函数 $f(\mathbf{x})$ 的值取反。于是，搜索算法的目标即最大化评估函数的值。对于任一可行解向量 \mathbf{x} ，规定其邻域如定义 7

定义 7. 邻域.

令 $\mathbf{x} = [x_1, x_2, \dots, x_n] (\forall i, x_i \in \{0, 1\})$ 为任一可行解向量，则在算法中 \mathbf{x} 的邻域 $\mathcal{N}(\mathbf{x})$ 定义如下

$$\mathcal{N}(\mathbf{x}) = \left\{ \mathbf{x}' \mid \sum_{i=1}^n |x_i - x'_i| \leq 1 \right\} \quad (3-13)$$

如定义 7，某可行解的领域中的任一元素，与该可行解在向量上只有一个元素的差距。例如，如果当前可行解 \mathbf{x} 为 $[0, 1, 0, 1]$ ，则其领域

$$\mathcal{N}(\mathbf{x}) = \{[\underline{1}, 1, 0, 1], [0, \underline{0}, 0, 1], [0, 1, \underline{1}, 1], [0, 1, 0, \underline{0}]\}$$

其中每一个向量都是 \mathbf{x} 的相邻解向量，下划线位置即是对原向量对应位置“取反”得到。事实上，整个解空间可被看做一个超立方体，其中每个顶点对应一个可行解向量，且解的领域对应其在超立方体中的所有邻居节点，如图 3-4 所示。

停止规则是禁忌搜索中最重要的准则之一，其决定搜索过程停止的条件。在本算法中，停止规则如定义 8 所示。

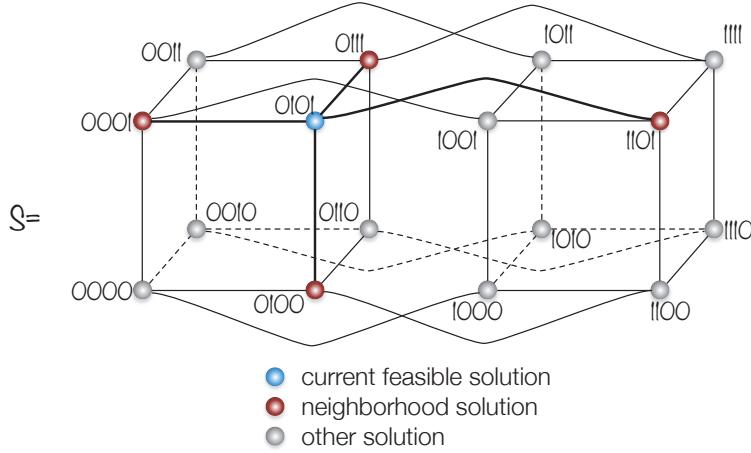


图 3-4 超立方体解空间

定义 8. 停止规则

停止规则 \mathcal{E} 如下定义

$$\mathcal{E}(\mathbf{x}^{best}) = \begin{cases} \text{true} & \text{if } p(\mathbf{x}^{best}) \text{ not improved after } \theta \text{ steps} \\ \text{false} & \text{otherwise} \end{cases}$$

如之前所述，评估函数 p 的值在搜索过程的每一次迭代中都被更新。如果当前解向量 \mathbf{x}^{now} 对应更好的评估函数值，即若有 $p(\mathbf{x}^{now}) > p(\mathbf{x}^{best})$ ，则用 \mathbf{x}^{now} 的值更新 \mathbf{x}^{best} 。然而，若当前记录的最优解对应的评估函数值 $p(\mathbf{x}^{best})$ 在一定步数之内都未被更新，则意味着算法可能无法找到更好的解（在算法本身不被修改的前提下）。在这种情况下，搜索过程停止，当前最优解向量将作为返回值从算法返回。

禁忌表 \mathcal{T} 由两个主要元素组成，禁忌对象 ϕ 和禁忌长度 $\mathcal{L}(\phi)$ 。禁忌对象是禁忌搜索算法中实施禁忌的目标元素，可以定义为解向量的变化，或评估函数的变化。在本算法中，禁忌对象见定义 9。

定义 9. 禁忌对象

禁忌对象为解向量的变化，形式化定义如下

$$\varphi : \mathbf{x} \rightarrow \mathbf{x}'$$

有

$$\mathbf{x} = (x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \quad (i \in [0, n])$$

且

$$\mathbf{x}' = (x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n) \quad (i \in [0, n], x_i \neq y_i)$$

事实上，禁忌对象是当前解向量在超立方体中对应的顶点向邻接顶点的迁移。正如图 3-4 所示，从蓝色顶点（当前解）向红色顶点的迁移，即为禁忌对象。禁忌表中的禁忌长度如定义 10。

定义 10. 禁忌长度

禁忌长度由 $\mathcal{L}(\varphi)$ 表示，有

$$\mathcal{L}(\varphi) = \lfloor T \rfloor$$

T 是服从正态分布的随机变量，有 $T \sim N(\mu, \sigma^2)$ ，且令

$$\mu = \begin{cases} \sqrt{n}[1 + p(\mathbf{x}') - p(\mathbf{x})] & p(\mathbf{x}') > p(\mathbf{x}) \\ \sqrt{n} & \text{otherwise} \end{cases}$$

在定义 9 中，禁忌长度设为对服从正态分布的随机变量 T 做下取整，其中正态分布的参数 μ 保证了对于能让目标函数值变的更好的解向量，在统计意义上具有一个较长的禁忌长度。将禁忌长度设为一个随机数，而非常量，其意义在于向搜索算法中加上一定的随机因素，如同舍伍德算法一样。随机性能够帮助算法跳出一些无休止的循环，且能在算法输入被蓄意设定的情况下表现的更好。对于禁忌长度，值 T 与评估函数的变化程度 $|p(\mathbf{x}) - p(\mathbf{x}')|$ 具有一定关系。评估函数值发生变化具有两种情况；第一种情况下 $p(\mathbf{x}') < p(\mathbf{x})$ ，意味着评估函数值到达了一个新的“山谷”；第二种情况下 $p(\mathbf{x}') > p(\mathbf{x})$ ，意味着评估函数值爬上了一个比以往更高的“峰顶”。在第一种情况下，应将禁忌长度设的更大，从而让算法能够跳出当前的局部陷阱；对于第二种情况，应将禁忌长度设的更小，以免解向量移动的太远，再次掉入另一个“山谷”。定义 9 中的设定正好符合以上几点，此外对于正态分布参数 μ ，有 $\mu \in [\sqrt{n}, 2\sqrt{n}]$ 。

从定义 9 和定义 10 出发，算法禁忌表见定义 11。

定义 11. 禁忌表

禁忌表的结构如下

$\mathcal{T} =$	1	2	\dots	\dots	$n - 1$	n
	t_1	t_2	\dots	\dots	t_{n-1}	t_n

假设禁忌对象为 $\varphi : \mathbf{x} -> \mathbf{x}'$ ，其更新规则如下

$$\forall i, t_i = \begin{cases} \mathcal{L}(\varphi) & \text{if } |x_i - x'_i| = 1 \\ 0 & \text{if } t_i = 0 \\ t_i - 1 & \text{otherwise} \end{cases}$$

禁忌表 \mathcal{T} 设为一个 $2 \times n$ 的表，其中第一行代表禁忌对象中解向量的变化位置，第二行代表对于每个位置当前的禁忌步长。例如，设当前解向量发生变化，其对应的禁忌对象为 $\phi : [0, 1, 0] \rightarrow [1, 1, 0]$ ，则应当按如下更新当前的禁忌表（其中 $t_1, t_2, t_3 > 0$ ）

1	2	3
t_1	t_2	t_3
$\mathcal{L}(\varphi)$	$t_2 - 1$	$t_3 - 1$

对于算法的每一次更新操作，向量变化所对应的位置，在禁忌表中都会以 $\mathcal{L}(\phi)$ 值更新，其它所有非零的表格元素都会做减 1 操作。任意位置，若其对应的禁忌步长不为 0，则仅当禁忌步长降为 0 时，当前解向量才可从该位置向对应的相邻解向量迁移。

禁忌搜索中另一个重要的法则，即为特赦准则。特赦准则可以对禁忌表中特定的禁忌对象进行赦免，从而使其能被算法重新选取。特赦准则如定义 12

定义 12. 特赦准则

特赦准则如下定义

对于 $\forall \mathbf{x}$, 若有 $p(\mathbf{x}) > p(\mathbf{x}^{best})$
则解向量 \mathbf{x} 可以作为下一步选择，即使其在禁忌表 \mathcal{T} 中被禁

当所有的位置都在表 \mathcal{T} 中被禁时，当前解向量无法迁移到任何一个邻居向量。然而当某一个邻居向量所对应的评估函数值，优于当前所记录的最优值，则该向量会以特赦准则被解禁，即可以被算法选取。

图 3-3 所示例子所对应的禁忌搜索过程，如图 3-3。如同表 3-2 所示的局部搜索一样，消息源节点依然被选为 n_2 ，且初始解向量为 $[0, 1, 0]$ 。为了简化表述，在此，禁忌表长 \mathcal{L} 设定为常数 3，而非定义中所述的随机量。此外，定义 8 中的 θ 值设为 3。

在 Step 1 中，禁忌表 \mathcal{T} 初始化为空，对于当前解向量，其所有的邻居解向量都可选。依算法 1 第 5 行，选择能使评估函数取最大值的解向量 $[1, 1, 0]$ 。在 Step 2 中，解向量第一个位置在表中被禁，故下一个解向量只能在 $[0, 0, 0]$ 及 $[1, 1, 1]$ 中选择。搜索过程一直进行到 Step 5，其中最优解在 $\theta = 3$ 步之后还未更新，触发停止规则，算法终止。纵览整个过程，与表 3-2 所述局部搜索过程相比，算法可以成功跳出局部最优陷阱。

3.4 最优化路由算法

本节给出有关路由过程的细节。在本节中，提出两个路由算法，基于局部搜索的 Local-MPAR 和基于禁忌搜索的 Tabu-MPAR。在 Local-MPAR 中，对任一节点而言，其它节点的 λ 值仅当该节点与其接触时获取；在 Tabu-MPAR 中，假设每个节点都知道其它所有节点对应的 λ 值。

表 3-3 图 3-3例子对应的禁忌搜索过程

	current solution			current best solution			tabu table \mathcal{T}	next options									
	\mathbf{x}^{now}	N	$P_{N,4}$	\mathbf{x}^{best}	N^{best}	$P_{N^{best},4}$		\mathbf{x}^{next}	N'	$P_{N',4}$	status						
Step 1	[0, 1, 0]	{ n_2 }	0.673	[0, 1, 0]	{ n_2 }	0.673	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> </table>	1	2	3	0	0	0	[1, 1, 0] [0, 0, 0] [0, 1, 1]	{ n_1, n_2 } ϕ { n_2, n_3 }	0.670 0 0.600	choosable choosable choosable
1	2	3															
0	0	0															
Step 2	[1, 1, 0]	{ n_1, n_2 }	0.670	[0, 1, 0]	{ n_2 }	0.673	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>3</td><td>0</td><td>0</td></tr> </table>	1	2	3	3	0	0	[0, 1, 0] [0, 0, 0] [1, 1, 1]	{ n_2 } ϕ { n_1, n_2, n_3 }	0.673 0 0.789	tabu choosable choosable
1	2	3															
3	0	0															
Step 3	[1, 1, 1]	{ n_1, n_2, n_3 }	0.789	[1, 1, 1]	{ n_1, n_2, n_3 }	0.789	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>2</td><td>0</td><td>3</td></tr> </table>	1	2	3	2	0	3	[0, 1, 1] [1, 0, 1] [1, 1, 0]	{ n_2, n_3 } { n_1, n_3 } { n_1, n_2 }	0.600 0.626 0.670	tabu choosable tabu
1	2	3															
2	0	3															
Step 4	[1, 0, 1]	{ n_1, n_3 }	0.626	[1, 1, 1]	{ n_1, n_2, n_3 }	0.789	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>1</td><td>3</td><td>2</td></tr> </table>	1	2	3	1	3	2	[0, 0, 1] [1, 1, 1] [1, 0, 0]	{ n_3 } { n_1, n_2, n_3 } { n_1 }	0.291 0.789 0.430	tabu tabu tabu
1	2	3															
1	3	2															
Step 5	[1, 0, 1]	{ n_1, n_3 }	0.626	[1, 1, 1]	{ n_1, n_2, n_3 }	0.789	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>0</td><td>2</td><td>1</td></tr> </table>	1	2	3	0	2	1	[0, 0, 1] [1, 1, 1] [1, 0, 0]	{ n_3 } { n_1, n_2, n_3 } { n_1 }	0.291 0.789 0.430	choosable tabu tabu
1	2	3															
0	2	1															
Stop	[1, 1, 1]	{ n_1, n_2, n_3 }	0.789	—	—	—	—	—	—	—	—						

在本章中，算法过程的解释是针对单一消息而言的（该消息在网络中可以具有一份或者多份拷贝）。为了进一步解释路由算法，本章参照文献 [5] 所提出的传染病路由算法的概念——将消息看做传播的病毒。由此，节点间一次成功的消息传输（复制）可被看做一次感染。上述讨论中，不包括消息的投递过程，即假设当 n_a 感染 n_b 时， n_b 不为消息的目的节点。以此出发，可以将网络中的节点状态划分为三类（皆只针对某一消息而言）。

- 被感染节点

被感染节点持有一条或者几条该消息的拷贝，然而被感染节点不具备传染性，即无法传染其他纯净节点。

- 纯净节点

纯净节点不持有消息的任意拷贝，可以被任意传染节点感染。

- 传染节点

传染节点是一类特殊的被感染节点，其可以传染其它纯净节点。

从传染状态可以迁移到被感染状态；换言之，节点将仍然持有该条消息，然而无法将该消息复制给其它节点。纯净状态也可以迁移到被感染状态或传染状态；当此类状态迁移发生，则说明该节点接受了消息。为了进一步解释这类状态迁移，下面将参照 Epidemic 算法 [5] 及 SprayAndWait 算法 [6] 进行说明。

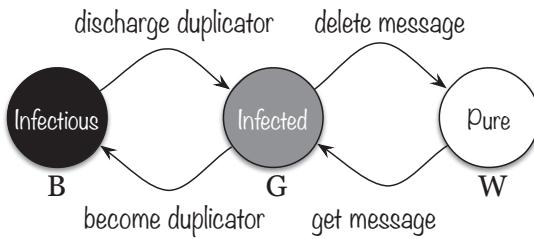


图 3-5 Local-MPAR 节点状态迁移图.

在 Epidemic 路由算法中，有两类状态节点，纯净节点和传染节点。消息副本的分发过程在所有纯净节点都被感染，都迁移至传染状态时结束。SprayAndWait 路由算法中，节点可以处于以上三种状态中的任意一种。在 Source SprayAndWait 中，网络中只有一个传染节点，在 Binary SprayAndWait 中，传染节点允许有多个，但是其最大数量被限制为一个固定数目。当网络中不在有传染节点时，消息分发过程结束。

事实上，这几类节点几乎存在于所有的机会网络路由算法中。特别地，Epidemic 及 SprayAndWait 是两类具有代表性的零知识依赖路由算法。对于其它路由算法，其消息副本分发过程亦可按照此类节点状态划分方法进行分类，从而可视作 Epidemic 及 SprayAndWait 算法的改进。在介绍 Local-MPAR 及 Tabu-MPAR 之前，先提出本章关于路由的三个公设。

公设 1. 对任意节点 n_a 而言， n_a 从 n_b 接受某条消息，当且仅当其不持有该消息的任意拷贝。

公设 2. 当消息的 *time-to-live* 到期时，从缓存中清除该消息的行为叫做丢弃；因节点状态迁移而清除缓存消息的行为，叫做删除。

公设 3. 持有某条消息或该条消息任意拷贝的某一组节点，记为 N ，由所有传染节点及被感染节点组成。

Local-MPAR 及 Tabu-MPAR 中的状态迁移图，如图 3-5 及图 3-6 所示，有如下定理。

定理 3. 对应表 3-5 中的状态迁移，复制操作对应于 $W \rightarrow G$ ，删除操作对应于 $G \rightarrow W$ 。

证明. 从三种状态的定义可知，持有消息的节点，位于状态 B 或 G ，未持有消息的节点，对应于状态 W 。由于 B 与 W 之间无直接迁移路径，状态 B 只能经过状态 G 迁移至 W ，反之亦然。从公设 1 可知，复制操作对应于状态迁移 $W \rightarrow B$ ，其中包含迁移 $W \rightarrow G$ 。从公设 2 可知，删除操作对应于 $B \rightarrow W$ ，即有 $G \rightarrow W$ 。

证毕 □

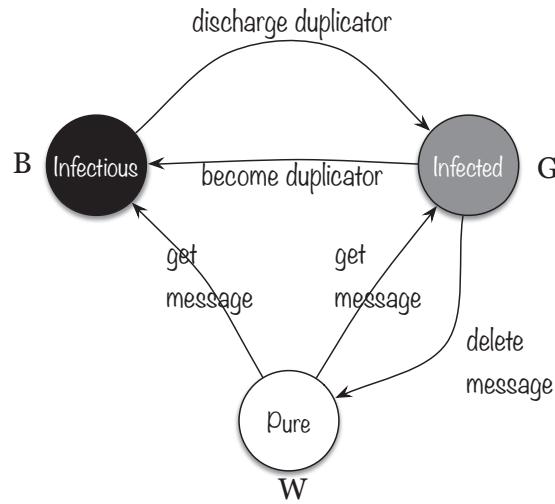


图 3-6 Tabu-MPAR 节点状态迁移图。

表 3-4 Local-MPAR 中的节点状态迁移

	state of n_b		
	B	G	W
B	—	$B \rightarrow \begin{cases} G & \text{if } G \rightarrow B \text{ happens in } n_b \\ B & \text{else} \end{cases}$	B
state of n_a	G	$G \rightarrow \begin{cases} W & \text{if } P_{N-\{n_a\},d} > P_{N,d} \\ B & \text{if } P_{N-\{n_a\}} \leq P_{N,d} \text{ and } E[D_a] < E[D_b] \\ G & \text{else} \end{cases}$	G
	W	$W \rightarrow \begin{cases} G & \text{if } P_{N \cup \{n_b\},d} > P_{N,d} \\ W & \text{else} \end{cases}$	W

3.4.1 Local-MPAR: 基于局部搜索的路由算法

在 Local-MPAR 算法中，节点可以处于三种状态中的任一状态。然而，对于每一条从源节点产生的消息而言，只允许网络中存在不超过一个传染节点。Local-MPAR 的基本思想，即是动态调整集合 N ，从而最大化预测投递概率 $P_{N,d}$ 。

算法 2 阐述了 Local-MPAR 算法的过程。在 Local-MPAR 中共有三种状态，在初始状态时，源节点 n_s 产生了消息 M ，于是其状态被设为传染状态。相应地，集合 N 被初始化为只包含源节点 n_s 。对于集合 N ，有如下公设。

公设 4. 在 Local-MPAR 中，节点集合 N 总是被传染节点维护。

算法 2 中整个路由过程可以被视作动态调整集合 N 的过程，如 Routing Stage 所示。该 Stage 的关键操作，即是让每一个节点完成表 3-4 中的状态转换。为简单起见，称表中行 B 列 G 的位置为 BG。从表 3-4 可以看出，对于任意两个相遇节点 n_a 及 n_b ，状态迁移仅发生在 n_a 和 n_b 至少有一个处于 B 状态时。BB 中无行为发生，其原因是在 Local-MPAR 中，对于某一条消息而言，不可能有两个节点同时处于 B 状态。

表 3-5 Tabu-MPAR 中的节点状态迁移

	state of n_b		
	B	G	W
state of n_a	B	$B \rightarrow B G$ (depends on the tickets left in n_a after allocating to n_b)	$B \rightarrow B G$ (depends on the tickets left in n_a after allocating to n_b)
G	$G \rightarrow B G$ (depends on the tickets left in n_a after allocating to n_b)	G	$G \rightarrow \begin{cases} W & \text{if } n_a \notin N_{opt} \text{ and } n_b \in N_{opt} \\ G & \text{else} \end{cases}$
W	$W \rightarrow B G$ (depends on the tickets left in n_a after allocating to n_b)	$W \rightarrow \begin{cases} G & \text{if } n_a \in N_{opt} \text{ and } n_b \notin N_{opt} \\ W & \text{else} \end{cases}$	W

Algorithm 2 Local-MPAR Algorithm.**Initial Stage:**

- 1: n_s generates message M
- 2: $n_s.state \leftarrow B$
- 3: $N \leftarrow \{n_s\}$

Routing Stage:

- 1: **for** any pair of n_a and n_b **do**
- 2: **if** n_a and n_b encounter **then**
- 3: finish the state transition according to 表 3-4
- 4: update N
- 5: **end if**
- 6: **end for**

End Stage:

- 1: M is delivered

引理 1. 消息的复制过程或删除过程，仅当相遇的两个节点中其中一个为传染节点时，才会发生。

证明. 直接由 表 3-4 得出。

证毕 □

定理 4. Local-MPAR 的路由阶段，即在解空间 \mathcal{S} 中做局部搜索。

证明. 从公设 4 可知，节点集合 N 只会在传染节点中被更新。引理 1 直接可得，任何复制或删除操作可以立即被传染节点所知，即集合 N 可被及时更新。复制操作或删除操作，其只会引起网络中一份拷贝的数量差异，因此仅仅会从集合 N 中删除或增加一个

元素，其本质即是将当前解向量迁移至其某一邻居解向量。从表 3-4 可知， N 当且仅当 $\forall N' \in \mathcal{N}(N), P_{N',d} \leq P_{N,d}$ 才不会再改变；即当且仅当解向量到达局部最优时，算法停止。

证毕 □

另一个需要解释的问题是，既然 Local-MPAR 中只存在一个传染节点，那么该节点应当如何选取？其选取的基本准则，即是基于公式 (3-8) 中对节点移动到下一个位置的延迟时间的预测。在表 3-4 中 GB 位置，当无需将 n_a 从状态 G 迁移至状态 W 时，应当考虑是否改变 n_a 为传染节点。若 $E[D_a]$ 的值小于 $E[D_b]$ ，则认为 n_a 更适合担当消息复制者，因为它具有更低的移动到下一位置的期望时延，意味着更多的传输机会。反之，若将消息复制权交给 n_b ， n_b 将发生 $G \rightarrow B$ 迁移，对称地， n_a 将发生 $B \rightarrow G$ 迁移。这样网络中仍然只保持有仅一个传染节点。

图 3-3 中例子所示的 Local-MPAR 路由过程，如图 3-7 所示。在时间 t_1 ，节点 n_2 产生目的节点为 n_4 的消息。从时间 t_2 到 t_3 ，由于集合 $\{n_1, n_2\}$ 及集合 $\{n_2, n_3\}$ 所对应的投递率都要小于集合 $\{n_2\}$ ， n_2 不会向 n_1 及 n_3 复制消息。最优集合 $N_{opt} = \{n_1, n_2, n_3\}$ 所对应的投递概率，要比 $\{n_2\}$ 所对应的投递概率大，然而，由于 Local-MPAR 算法无法跳出局部最优，故 N 无法变为 N_{opt} 。

3.4.2 Tabu-MPAR: 基于禁忌搜索的路由算法

在 Tabu-MPAR 算法中，与 Local-MPAR 相同，每个节点都有三个状态。如同 SprayAndWait 算法，允许网络中存在多于一个但有最大数目限制的传染节点。

Tabu-MPAR 路由算法如算法 3 所示。如同 Local-MPAR，Tabu-MPAR 算法亦分为三个阶段。在初始节点，当源节点 n_s 产生消息 M 时，节点 n_s 的状态被设为 B，如行 1-2 所示。节点集合 N_{opt} 在源节点中计算，此外，对于每一条产生的消息，都对应有一个网络中该消息的最大副本数量，该值设为 $|N_{opt}|$ 。每个节点对于每条消息，维护有一个计数变量，用以指明当前该节点持有该消息的拷贝数目。节点处于状态 B，当且仅当其消息计数变量大于 1¹。持有消息的计数变量为 1 的节点，处于状态 G，不持有该消息的节点，皆处于状态 P。

算法 3 的第二阶段即是 Tabu-MPAR 的路由过程。与 Local-MPAR 相比，有两个不同点。首先，无需在路由过程中更新 N_{opt} ，因为其已在源节点 n_s 中被计算出来，并写入消息的头部。其次，当任意两个节点 n_a 及 n_b 相遇时，将在状态转换之前重新分配对应的消息计数变量。分配策略基于 $E[D_a]$ 及 $E[D_b]$ 的值，如行 3 及行 4 所示。具有更小 $E[D]$ 值的节点，将被分配更多数目的消息拷贝。行 5-10 保证了变量 a 和 b 都是位于 1 和 $L - 1$ 之间的整数。由此可定义 Tabu-MPAR 的状态转换规则，如表 3-5。除了表中对角线位置以外，其它表位置中都可能发生对应的状态迁移。事实上，当节点

¹严格来讲，该节点是针对该条消息处于状态 B，对于其它消息而言，该节点的状态也可能为 W 或者 G。

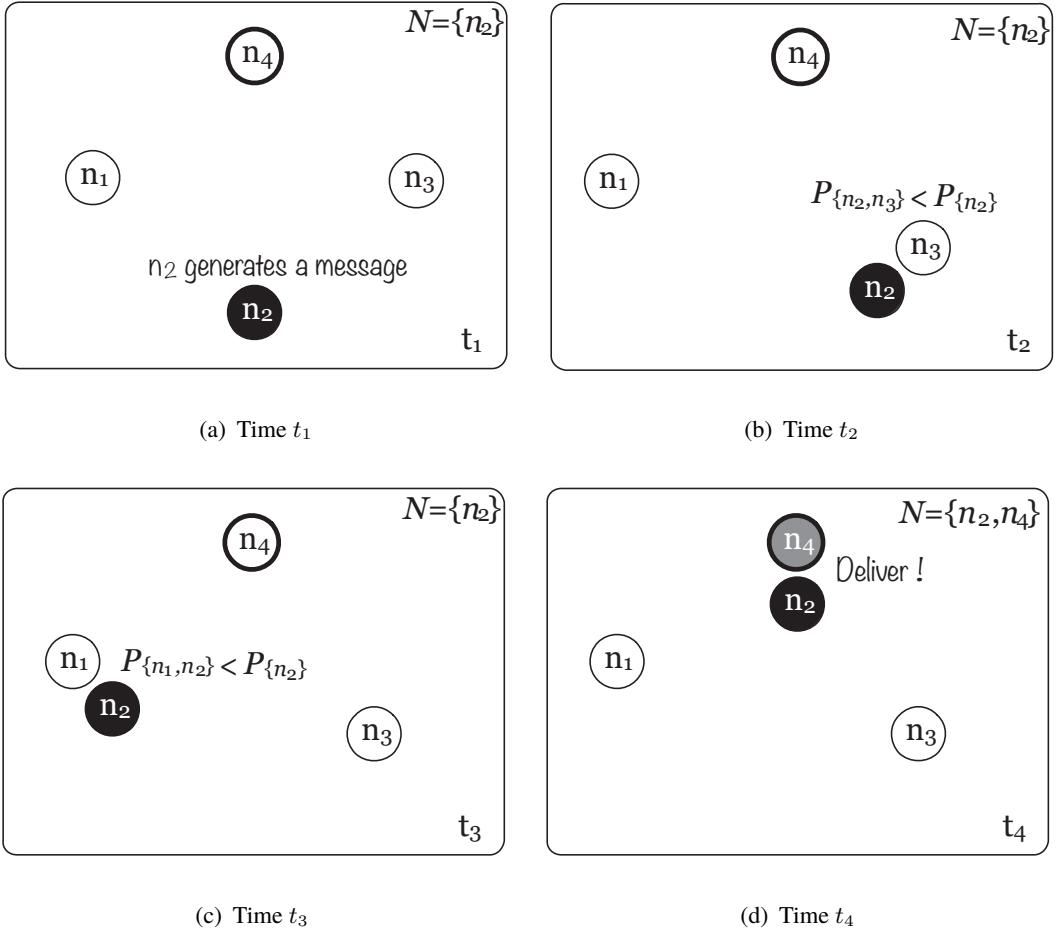


图 3-7 Local-MPAR 路由过程.

n_a 发生 GB 位置的状态迁移时，节点 n_b 也同时发生 BG 位置的状态迁移，反之亦然。该规则也适用于 WB，BW 及 WG，GW。故表表 3-5 可以看做一个 3×3 的对称矩阵。Tabu-MPAR 路由过程如图 3-8 所示。

下面证明 Tabu-MPAR 的路由过程即对应 N 变为最优集合 N_{opt} 的过程。

引理 2. $N = N_{opt}$ 当且仅当网络中不存在传染节点时发生，即所有属于集合 N 中的节点，都处于状态 G。

证明. 由于网络中某条消息最大副本数量为 $|N_{opt}|$ ，若存在传染节点 n_a ，则其对应的拷贝数量至少为 2，故其它节点一共所具有的拷贝数量不会超过 $|N_{opt}| - 2$ 。因此，持有该消息的所有节点的数目，至多为 $|N_{opt}| - 1$ 个，故其必要性成立。若持有该消息的节点集合为 $|N_{opt}|$ ，则该条消息在网络中已达到其最大可能副本数量 $|N_{opt}|$ 。在此情形下，每个节点所持有该消息的拷贝数目为 1 份。根据 Tabu-MPAR 节点状态的定义，集合 $|N_{opt}|$ 中所有节点此时都处于被感染状态，即状态 G；其它所有节点都为纯净节点，即处于状态 W。此时，网络中无传染节点，故充分性成立。

证毕 □

Algorithm 3 Tabu-MPAR Algorithm.

Initial Stage:

- 1: n_s generates message M
- 2: $n_s.state \leftarrow \mathbf{B}$
- 3: n_s computes N_{opt} by tabu search and saves it in M
- 4: $n_s.tickets \leftarrow |N_{opt}|$

Routing Stage:

- 1: **for** any pair of n_a and n_b **do**
- 2: **if** n_a and n_b encounter **then**
- 3: $a = \frac{E[D_b]}{E[D_a] + E[D_b]} \cdot |N_{opt}|$
- 4: $b = \frac{E[D_a]}{E[D_a] + E[D_b]} \cdot |N_{opt}|$
- 5: **if** $a < 1$ **then**
- 6: $n_a.tickets = \lceil a \rceil$
- 7: $n_b.tickets = \lfloor b \rfloor$
- 8: **else**
- 9: $n_a.tickets = \lfloor a \rfloor$
- 10: $n_b.tickets = \lceil b \rceil$
- 11: **end if**
- 12: finish the state transition according to 表 3-5
- 13: **end if**
- 14: **end for**

End Stage:

- 1: M is delivered
-

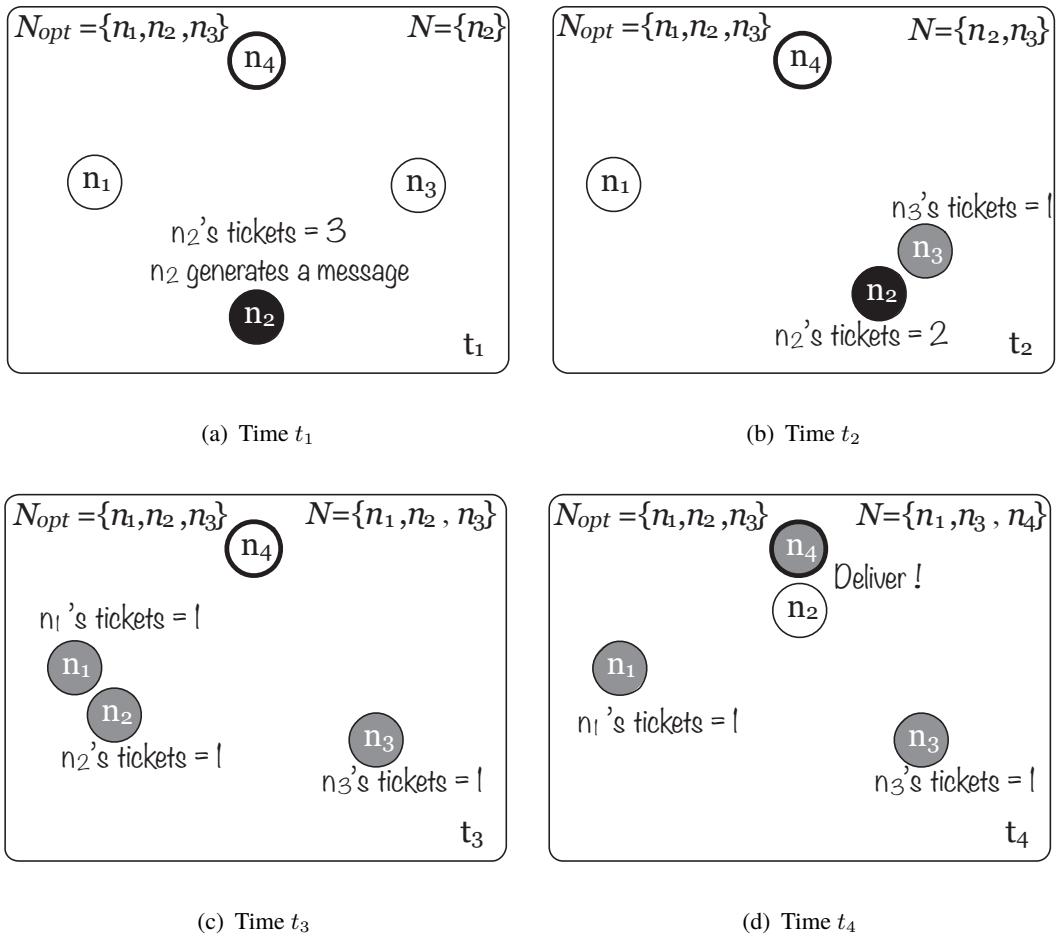


图 3-8 Tabu-MPAR 路由过程.

引理 3. 若集合 N 变为 N_{opt} , 则不会再改变。

证明. 从引理 2 可知, 若 N 变为 N_{opt} , 则所有 N 中的节点都处于状态 G, 且网络中所有的纯净节点都不在 N 中。由于网络中没有位于 B 状态的节点, 状态迁移 $B \rightarrow G$ 不会发生。对于状态迁移 $G \rightarrow B$, 若其发生, 则一定伴随状态迁移 $G \rightarrow W$ 发生; 然而其只可能在处于 G 状态的节点不属于 $|N_{opt}|$ 时发生, 与题设矛盾。故集合 N 中所有节点的状态不会发生改变。

证毕 □

定理 5. 依表 3-5 中状态转移规则能够使集合 N 变为 N_{opt} .

证明. 状态迁移 $B \rightarrow G$ 不会使集合 N 发生改变。从算法 34-10 行可知, 所有节点重新分配后的副本数目至少为 1, 即若有足够多的节点接触机会, 则网络中所有处于 B 状态的节点最终都会变为状态 G。考虑状态迁移 $W \rightarrow G$ 及 $G \rightarrow W$, 其转换规则即符合 N_{opt} 。结合引理 2 及引理 3 即可证得本定理。

证毕 □

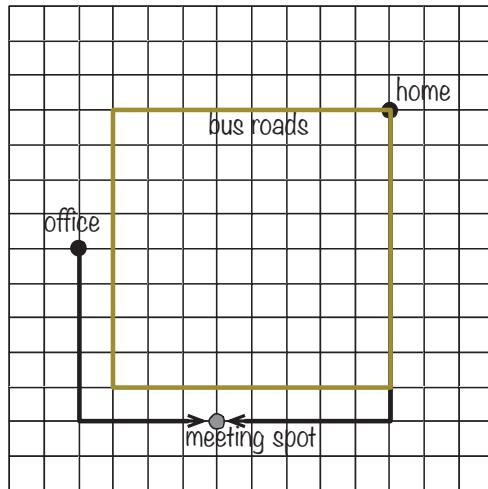


图 3-9 Working day movement 模型，结合 Manhattan blocks.

3.5 仿真实验

在本节中，基于机会网络仿真器（Opportunistic Network Environment, ONE）[7]，对 MPAR 算法进行仿真，并进行性能评估比较。节点移动模型采用 [8] 提出的 Working Day Movement (WDM) 模型，地图设为 Manhattan Blocks。WDM 模型分层结合了许多子节点移动模型。此外，一些常见的移动模型相关元素，如家，办公室，夜晚活动及不同的交通工具（如电车，私家车等）也在 WDM 中设定，用以捕捉人类社会相关属性。办公室模型（office sub-model）产生一种围绕办公桌的星形移动轨迹，相关的办公室人员按此轨迹进行移动。家模型（home sub-model）即是使人员在某个定点固定下自身位置，不做移动。夜晚活动模型（evening activity sub-model）反映了具有朋友关系的人群在工作后的活动，其使节点在街道上服从随机走方式移动。Manhattan Community 模型将节点路径限制在街道上。Manhattan Community 模型包含一组呈矩阵状的小格，如图 3-9。

网络场景设定如下。行人数量分别设为 200, 400, 600, 800。公交车，办公室及活动场所的数量分别设为总节点数目的 2%, 20%, 2.5%。在模拟器时间内，每人每天工作时间设为 8 小时。每人晚上参加夜晚活动的概率为 0.5。此外，每个人有车的概率为 0.2，若有车，则晚上参加活动将会开车，否则将乘巴士。行人的行走速度设为 0.8–1.4m/s。如图 3-9 所示，网络中设有三类地点，家（home），办公室（office）及聚会地点（meeting spot）。在街区中心设有一个环绕中心的公交路线，可以用于搭载行人。在每一个家，办公室及聚会地点，都设有一个 throw-box，用以接管和传递消息。所有节点都以蓝牙接口通信，传输半径设为 10 m，数据率设为 250 KB/s。由于每一类地点（家，办公室或聚会地点）都设为地图上的一个点，仿真中确保每个节点到达这些地点后一定与 throwbox 发生接触。在仿真中，共产生 30,000 条消息，对于每条消

息，其源节点和目的节点从所有行人节点中随机选择。每条消息的大小设为 500 K 至 1 M 之间。仿真区域的大小根据节点数目的多少按同比例设定。对于整个仿真过程，其仿真器时间设为 12 天。

本节将 MPAR 算法与两个经典算法相比较：Delegation Forwarding (DF) [9] 与 Simbet [10]。采用四项评估指标：投递率 (delivery ratio)，平均时延 (average latency)，网络开销 (overhead ratio) 及平均每跳数 (average hop count)。

3.5.1 改变消息生存时间

节点缓存大小固定在 200M。消息的生存时间在 10 小时到 30 小时变化，仿真结果如图 3–10，图 3–11，图 3–12，图 3–13。

仿真结果表明，两种 MPAR 算法，Local-MPAR 及 Tabu-MPAR，其性能表现都要优于 DF 及 SimBet。与 DF 算法相比，Tabu-MPAR 和 Local-MPAR 分别可以增加约 71.1% 及 37.8% 的消息投递率，且能降低约 83.8% 及 57.8% 的平均时延。与 SimBet 算法相比，Tabu-MPAR 和 Local-MPAR 分别可以增加 95.2% 和 55.3% 的投递率，且能降低大概 79.2% 及 60.1% 的平均时延。从图 3–12可以看出，Tabu-MPAR 和 Local-MPAR 具有比 DF 及 SimBet 更低的网络开销。对于三中社会属性相关的路由算法，Tabu-MPAR，Local-MPAR 及 SimBet 而言，网络开销要比 DF 低很多，特别是在消息生存时间设为比较长时。当网络中节点数目更小时，这种提升就越明显。由于 MPAR 路由算法限制了消息的最大副本数量，且利用 throwbox 辅助投递消息，中继传输操作非常少。此外，随着节点数目的增加，对于每个节点而言，其开销也会相应减小，导致整个网络的开销减小。图 3–13显示了消息平均跳数的性能表现。可以看出，两种 MPAR 算法具有比 DF 和 SimBet 更低的平均跳数。当节点数目增加时，平均跳数也增加；其原因是仿真区域按同比例增加后，需要节点间更多的合作，用以投递消息。

3.5.2 改变节点缓存大小

消息生存时间固定为仿真器中的 16 小时。节点缓存大小在 50 MB 到 300 MB 之间变换。仿真结果如图 3–14，图 3–15，图 3–16，图 3–17。

仿真结果表明，两种 MPAR 算法，Local-MPAR 及 Tabu-MPAR，其性能表现皆优于 DF 及 SimBet。与 DF 算法相比，Tabu-MPAR 和 Local-MPAR 分别可以增加约 83.8% 及 40.3% 的消息投递率，且能降低约 83.8% 及 57.8% 的平均时延。与 SimBet 算法相比，Tabu-MPAR 和 Local-MPAR 分别可以增加 3 倍和 2.5 的投递率，且能降低大概 80% 及 60% 的平均时延。从图 3–16可以看出，Tabu-MPAR 和 Local-MPAR 具有几乎相同的网络开销；当缓存大小设为 120 MB 时，MPAR 算法的网络开销比 SimBet 略低一点，比 DF 低很多。当网络中节点数目更小时，性能提升越明显，如图 3–16所示。与消息生存时间作为变量时相同，从图 3–17可以看出，两种 MPAR 算法具有比 DF 和 SimBet 更低的消息平均跳数，且当节点数目增加时，平均跳数也随之增加。

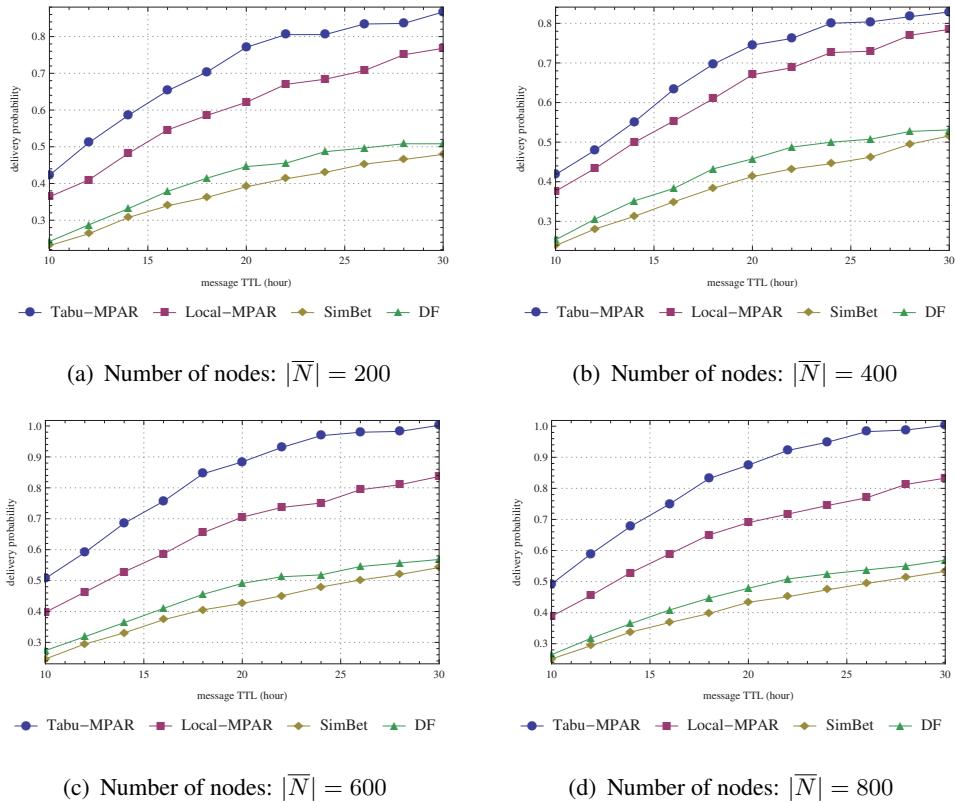


图 3-10 消息投递率 vs. 消息生存时间

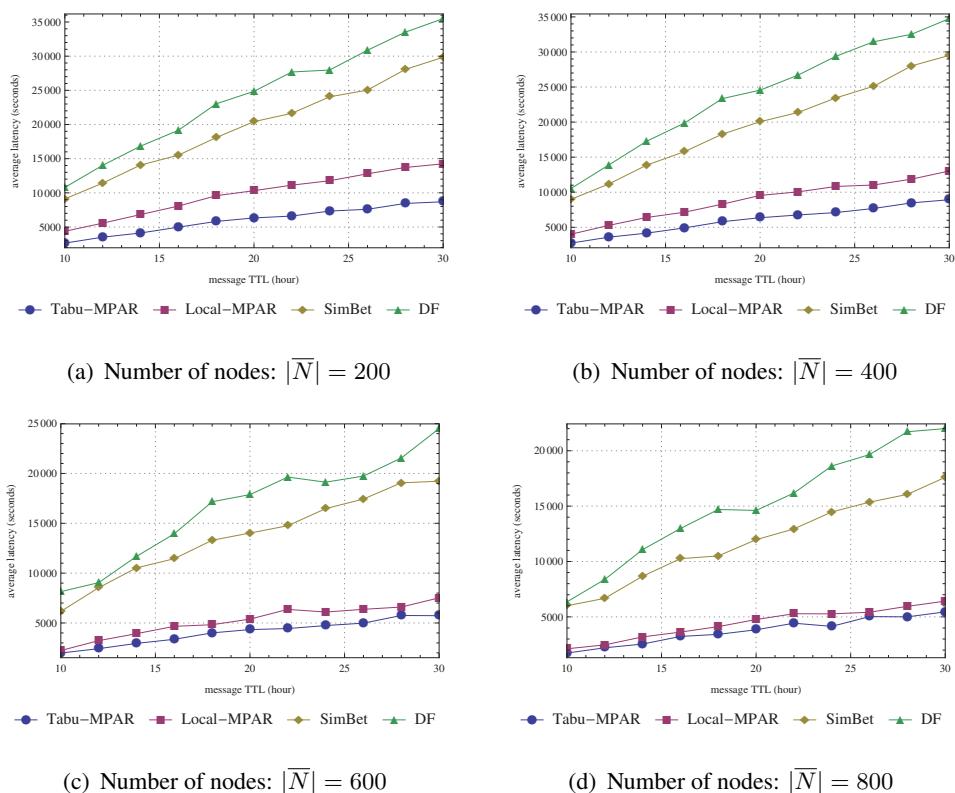


图 3-11 端到端平均时延 latency vs. 消息生存时间

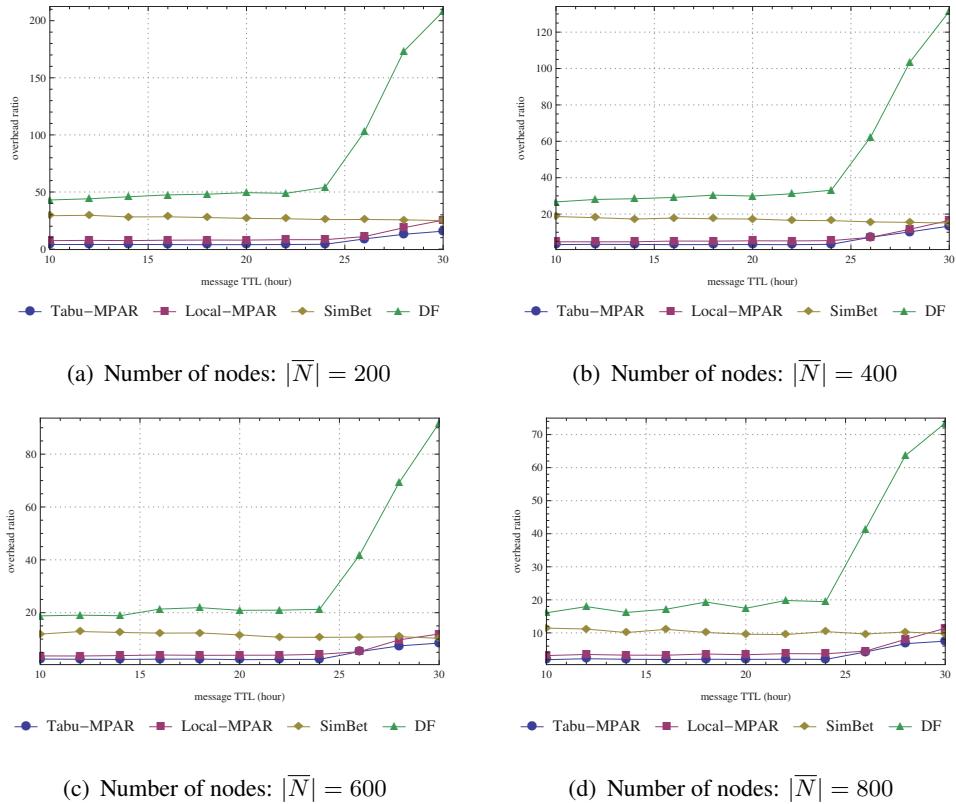


图 3-12 网络开销 vs. 消息生存时间

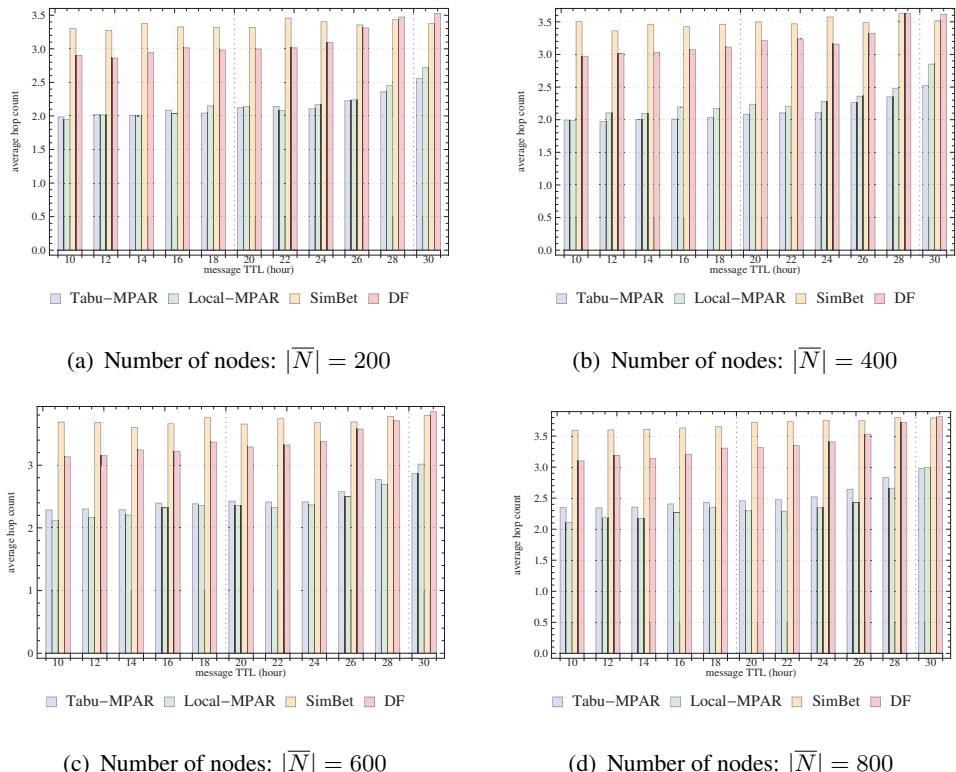


图 3-13 平均跳数 vs. 消息生存时间

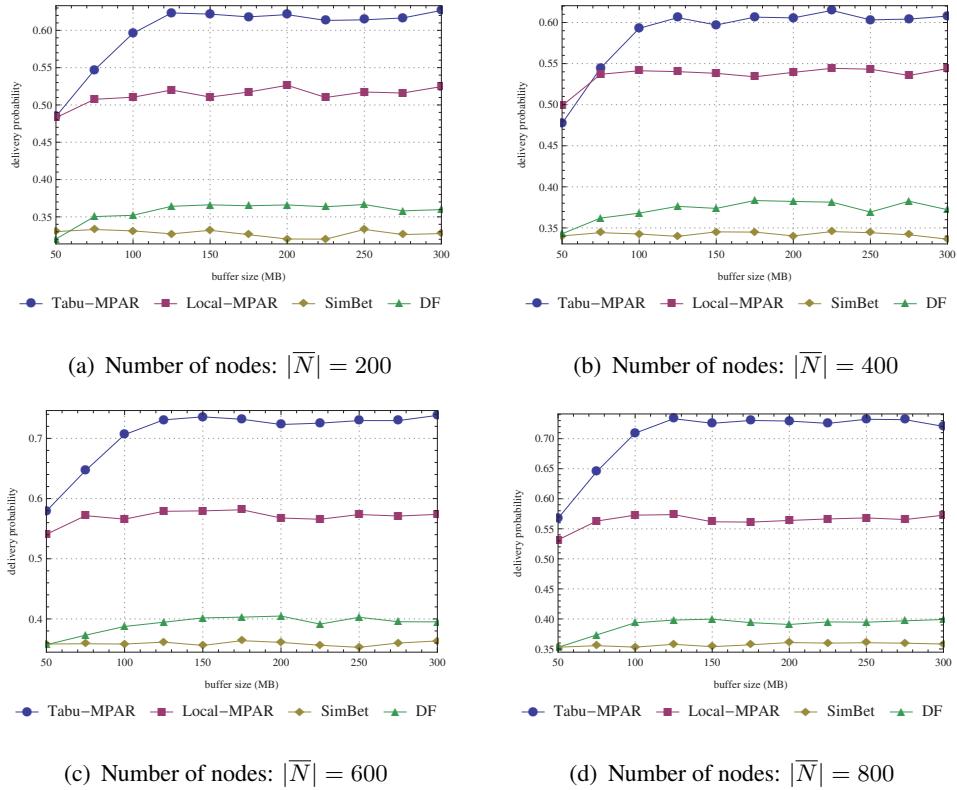


图 3-14 消息投递率 vs. 节点缓存

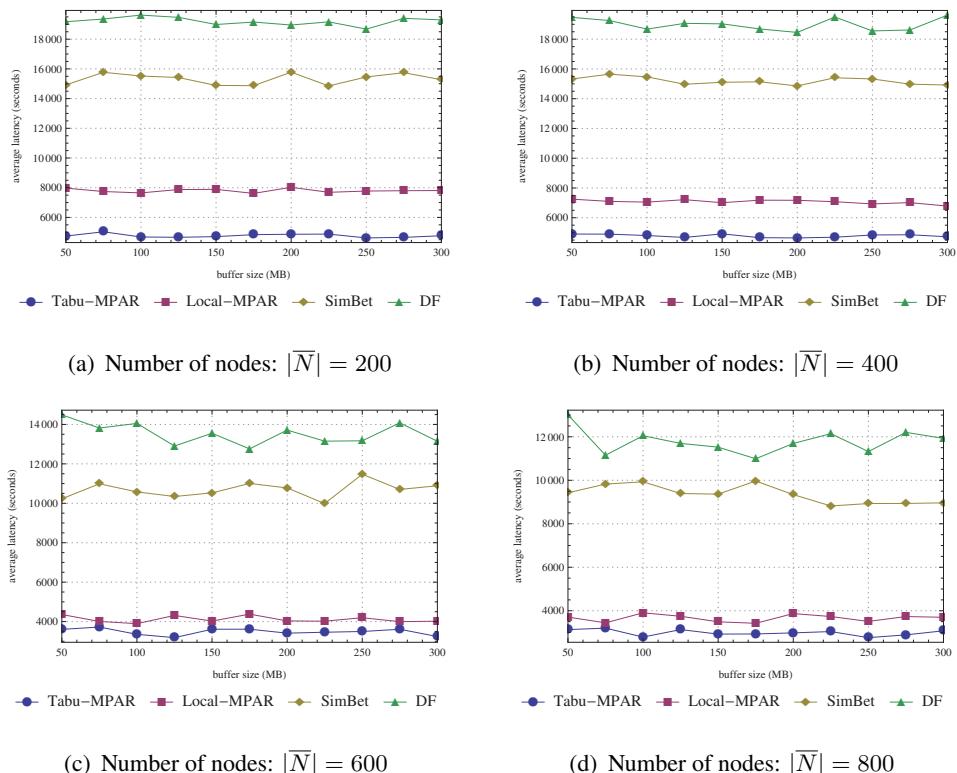


图 3-15 端到端平均时延 vs. 节点缓存

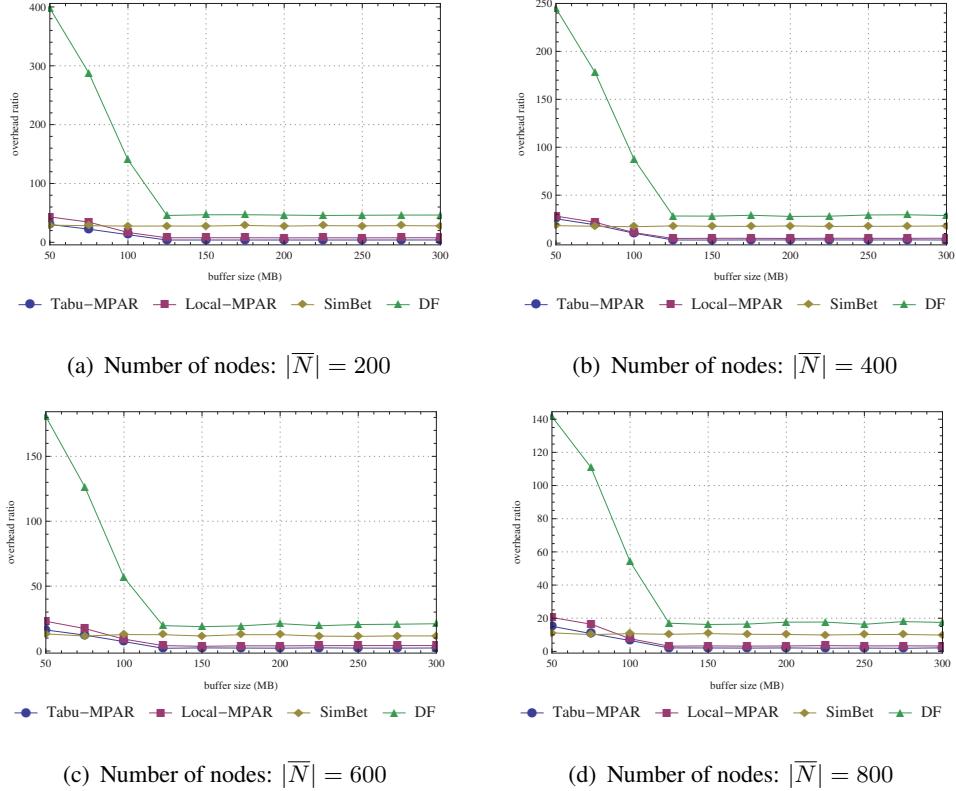


图 3-16 网络开销 vs. 节点缓存

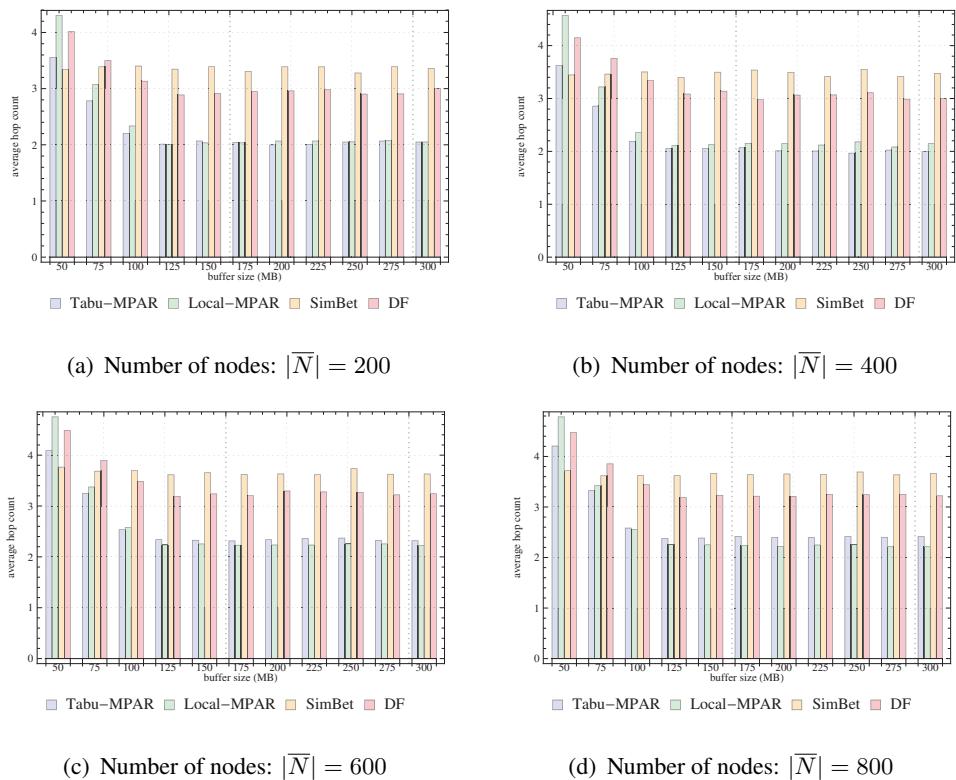


图 3-17 平均跳数 vs. 节点缓存

3.6 本章小结

在本章中，建立了周期相关的移动记录模型，并从移动记录中提取出节点（群组）移动模式。对于成组的节点，其被看做一个整体，并评估整体的预测投递率。本章研究了两个相关路由的关键属性，并将路由问题建模为组合最优化问题，且证明了该问题的 \mathcal{NP} 难解性。为求解该路由问题，基于局部搜索及禁忌搜索，分别提出了两种路由算法 Local-MPAR 及 Tabu-MPAR。此外，本章证明了 Tabu-MPAR 过程可以使得持有消息的节点集合最终达到预测投递概率最优的集合。仿真实验表明，两种 MPAR 算法，在机会网络中的综合移动模型 WDM 上优于 DF 算法及 SimBet 算法。

第四章 基于消息传输收益的最优队列调度算法

本章尝试改进经典的基于概率预测的路由算法 PRoPHET [11]。定义了“传输收益”概念，基于此，提出了传输收益最优化，吞吐量相关的概率路由问题（Optimal Throughput-Aware Probabilistic Routing）。该问题被建模为最优决策问题，并在本章中采用动态规划方法解决。在该模型下，提出了一种数据项选择机制，其具有如下特点：

1. 每一对节点所对应的潜在连接，其平均数据吞吐量可能比缓存中的消息大小要小。
2. 每次传输的能量消耗不可忽略，换言之，消息传输不成功会带来无收益的能量损耗。

例如，针对某条消息的一次传输操作而言，若当前连接所允许的总数据量比消息的大小小很多，则该次传输终止时，消息并未被成功传输。针对该问题的一个解决办法，即是总是传输不超过该连接平均吞吐量大小的消息，从而避免浪费传输机会。退一步而言，即使连接的吞吐量完全足够传输每一条消息，传输不同消息所带来的收益也并不相同（比如投递率收益，及投递时延收益等），然而传输的机会是有限的，故针对每一次传输机会而言，传输不同的消息对路由的整体性能表现具有较大的影响。从这个角度而言，在机会网络中，为了提高路由性能表现，应当设计有效的数据项选择机制。

本章组织如下：4.1节中介绍了系统模型及路由模型；4.2节中形式化定义了关键问题；4.3节中利用设计的数据项选择机制对 PRoPHET 进行改进；4.4节为仿真实验；4.5节概括了本章内容。

4.1 系统模型

数学符号如表 4-1 所示。本模型的时间线设为离散时间，时间轴被划分为多个小的时间槽，每个时间槽的长度定义为一个单位时间。整个节点集合用符号 $\mathcal{N} = \{n_i | n_i \in \mathcal{N}, 1 \leq i < |\mathcal{N}|\}$ 。对于任意一条消息 m_k ，其消息生存时间表示为 $TTL(k)$ 。当消息 m_k 被某节点产生时， $TTL(k)$ 的值即被指定。若某条消息的 TTL 到期，则消息会被当前节点丢弃。消息 m_k 的大小用符号 $S(k)$ 表示。模型假设任一节点 n_i 知道其自身接口的带宽 $B(i)$ 。该假设的合理性建立在网络中的每个节点往往在通信时都采用统一标准的某种接口，如蓝牙，wifi 等。退一步讲，即使接口状态不稳定，也可以利用滑动窗口法对其平均值进行预测。对于任意一对相遇节点 n_i 及 n_j ，令节点 n_i 记录两个值 $t_{s_{(i,j)}}$ 及 $t_{e_{(i,j)}}$ ，分别用于记录两节点间最近一次连接的开始时间及

表 4-1 数学符号

Notation	Meaning
\mathcal{N}	网络节点集合
n_i	第 i 号节点
m_i	第 i 号消息
$TTL(i)$	消息 m_i 的生存时间
$S(i)$	消息 m_i 的大小
$\zeta(i)$	消息 m_i 的传输收益
$B_{(a,b)}$	节点 n_a 与 n_b 之间的带宽
$t_{s_{(a,b)}}/t_{e_{(a,b)}}$	节点 n_a and n_b 最近一次连接的开始/结束时间
$\tau_{a,b}$	节点 n_a 与 n_b 预测接触时间
$P_{(a,b)}$	n_a 's 节点 n_a 与 n_b 的预测接触概率
$ETH_{(a,b)}$	节点 n_a 与 n_b 连接的预测吞吐量
$P^{\{a,b\}}$	节点 n_a 与 n_b 的共同投递率

结束时间。于是 $t_{e_{(i,j)}} - t_{s_{(i,j)}}$ 即代表该次连接的持续时间；为简单起见，符号 $t_{s_{(i,j)}}$ 及 $t_{e_{(i,j)}}$ 在无歧义的上下文中，简记为 t_s 和 t_e 。

PRoPHET 路由算法 [11] 记录了节点的相遇历史，用于预测节点间的相遇概率，并且考虑到了相遇概率具有传递性的特点。在 PRoPHET 中， $P_{(a,b)}$ 即用于衡量节点 n_a 及 n_b 之间投递概率的效用函数，存于节点 n_a 内。 $P_{(a,b)}$ 的计算及更新方法如下式所示。

$$P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old}) \times P_{init} \quad (4-1)$$

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^k \quad (4-2)$$

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) \times P_{(a,b)} \times P_{(b,c)} \times \beta \quad (4-3)$$

其中， P_{init} , β 及 γ 是位于 $[0, 1]$ 范围内的常数。每个节点维护一个 $1 \times |\mathcal{N}|$ 的向量，该向量中第 i 个元素记录了节点 n_a 对节点 n_i 的投递率。

4.2 问题形式化

本节给出问题的形式化定义。

4.2.1 问题目标

问题目标即是最大化每条消息的投递概率。在机会网络中，每个节点以“存储-携带-转发”的方式对消息进行路由。在机会路由中，一种策略是总是让节点选择对目的节点投递率高的消息进行投递。吞吐量与节点间的接触时间及通信接口带宽具有很强的相关性，因此对消息是否能投递成功具有较大的影响。然而在该种方法中，节点

之间连接的吞吐量因素并未考虑进去。由于节点的带宽，接触时间及接触机会非常有限，缓存消息队列的调度策略对路由性能表现具有很大的影响。假设节点 n_a 持有三条消息 m_i, m_j 及 m_k ，其目的节点都为 n_b ，大小分别为 150 K, 200 K 与 100 K。然而当前连接最多只能传输 120 K 的数据量。在此情况下，由于连接吞吐量的限制，消息 m_i 和 m_j 都无法被成功传递。一种做法是让 n_a 按消息大小从小到大传递，然而这种方法虽然能保证部分消息成功传输，但并未设定任何优化目标，由此可能使得某些虽然大小稍大，但也能成功传递，且对投递率具有较大改进的消息无法充分利用该次机会传输。为解决此问题，需要对消息的调度设定一个可行的评估指标。在本章中，主要集中研究如何有效的数据项选择机制提高消息的投递率。下面将给出最优化消息传输投递率收益的分析过程。

若消息被节点 n_a 转发给 n_b ，则该消息宣告传输失败，仅当 n_a 及 n_b 都未成功传输该消息，其投递率可以用公式(4-4)计算。

$$P^{\{a,b\}} = 1 - (1 - P_{(a,d_i)})(1 - P_{(b,d_i)}) \quad (4-4)$$

于是对于传输收益有如下定义。

定义 13. 传输收益.

传输收益函数 $\zeta(i)$ 用于衡量消息 m_i 在本次传输过程中能够改善的投递率，有

$$\zeta(i) = P^{\{a,b\}} - P^{\{a\}} = 1 - (1 - P_{(a,d_i)})(1 - P_{(b,d_i)}) - P_{(a,d_i)} \quad (4-5)$$

函数 $\zeta(i)$ 的值代表将消息 m_i 从节点 n_a 传输给节点 n_b 所带来的投递率收益。从这点出发，可以如下定义“吞吐量相关最优机会路由”概念。

定义 14. 吞吐量相关最优机会路由.

最优机会路由总是尝试最大化消息投递概率，且考虑带宽及接触时间两个因素。节点 n_i 将依照 ζ 函数值选择消息传输给节点 n_j ，利用预测的吞吐量最大化 ζ 值总和。

例如，在表 4-2 所示例子中，节点 n_a 缓存中总共有 5 条消息，记为 m_1, m_2, m_3, m_4, m_5 。消息 m_i 的目的节点记为 d_i 。对于其中任意一条消息 m_i ($1 \leq i \leq 5$)，有 $P_{(b,d_i)} > P_{(a,d_i)}$ 。在这种情形下，当节点 n_a 与 n_b 相遇时，所有这 5 条消息都应由 n_a 转发给 n_b 。由公式(4-5)可以得到每条消息对应的 ζ 值。在下一小节中，将给出吞吐量的预测方法，然后给出问题的形式化定义。

4.2.2 吞吐量预测

定义 15. 连接吞吐量.

给定任意两个节点的接触时间 t ，以及带宽 B (KB/unit)，该次连接的吞吐量，记为 TH ，有

$$TH = B \cdot t$$

表 4-2 节点 n_a 缓存中的 5 条消息

data item	m_1	m_2	m_3	m_4	m_5
$P_{(a,d_i)}$	0	0.2	0.1	0.12	0.2
$P_{(b,d_i)}$	0.1	0.75	0.2	0.25	0.35
value of ζ	0.1	0.6	0.18	0.22	0.28
message size	1K	2K	5K	6K	7K

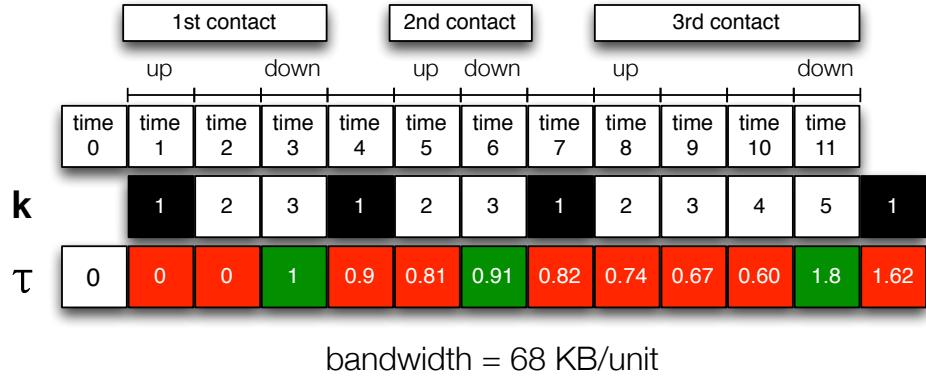


图 4-1 带宽预测过程

在此假设带宽 B 为已知，要预测吞吐量，则只需预测出连接的持续时间 t 。采用公式 (4-6) 预测下一次 n_a 与 n_b 之间连接的持续时间。

$$\tau_{(a,b)_{new}} = (1 - \alpha)\tau_{(a,b)_{old}} + \alpha(t_e - t_s) \quad (4-6)$$

其中 $\alpha \in [0, 1]$ 是常数参数， $t_e - t_s$ 是节点 n_a 与 n_b 之间最近一次连接的持续时间，该部分加权为 α 。当 α 值设的较大时，公式 (4-6) 后半部分的影响较大，反之亦然。

当节点 n_a 一段时间内没有再与节点 n_b 接触时，则用公式 (4-7) 更新 $\tau_{(a,b)}$ 。

$$\tau_{(a,b)_{new}} = \tau_{(a,b)_{old}}\gamma^k \quad (4-7)$$

其中 $\gamma \in [0, 1]$ 是衰减常数，如公式 (4-2) 一样； k 是离上一次更新公式流逝的单位时间数目（即经过的时间槽数目）。

在每个时间槽内，对于任意一对节点 n_a 和 n_b ，在每一个时间槽，节点都会检查 n_a 及 n_b 的连接状态。对 τ 的更新过程如算法 4 所示。根据定义 15 所示关于 n_a 与 n_b 之间吞吐量的定义，吞吐量可以由公式 (4-8) 预测。

$$ETH_{(a,b)} = B_{(a,b)} \cdot \tau_{(a,b)} \quad (4-8)$$

Algorithm 4 Updating the τ value

For the current time unit

```

1: if connection is up then
2:    $t_s \leftarrow current\_time$ 
3:    $\tau_{(a,b)_{new}} = \tau_{(a,b)_{old}} \gamma^k$ 
4:    $k \leftarrow k + 1$ 
5: else if connection is down then
6:    $t_e \leftarrow current\_time$ 
7:    $\tau_{(a,b)_{new}} = (1 - \alpha)\tau_{(a,b)_{old}} + \alpha(t_e - t_s)$ 
8:    $k \leftarrow 1$ 
9: else
10:   $\tau_{(a,b)_{new}} = \tau_{(a,b)_{old}} \gamma^k$ 
11:   $k \leftarrow k + 1$ 
12: end if
```

4.2.3 问题定义

首先证明数据项选择问题可以建模为 0-1 背包问题，然后给出问题的形式化定义。

定理 6. 定义 14 中路由所对应的数据项选择问题，即为 0-1 背包问题。

证明. 将预测吞吐量 $ETH_{(a,b)}$ 看做背包的最大承重，收益函数值 $\zeta(i)$ 看做第 i 项物品的价值，消息的大小 $S(i)$ 看做第 i 项物品的重量，则数据项选择过程等同于 0-1 背包问题填装物品的过程，其中决策目标即是让每个物品所对应的收益的总和，即总收益最大。 \square

由定理 6 可知，数据项选择问题是一个 \mathcal{NP} 完全的最优决策问题。该问题形式化定义如下。

定义 16. 数据项选择问题。

吞吐量相关最优机会路由中的数据项选择问题是一个最优决策问题，即

$$\begin{aligned}
 & Max \quad \sum_{i=1}^n \zeta(i)x_i \\
 s.t. \quad & \sum_{i=1}^n S(i)x_i \leq ETH_{(a,b)} \\
 & x_i \in \{0, 1\}, \quad i = 1, \dots, n
 \end{aligned}$$

Algorithm 5 Get the solution by dynamic programming**Input:** $MessageList = [m_1, m_2, \dots, m_n]$,

$$\zeta = [\zeta(1), \zeta(2), \dots, \zeta(n)], S = [S(1), S(2), \dots, S(n)]$$

Output: $ForwardList$

```

1: for  $i \leftarrow 1$  to  $n$  do
2:   for  $j \leftarrow 0$  to  $ETH$  do
3:      $\mathcal{V}[i, j] = \max\{\mathcal{V}[i - 1, j], \mathcal{V}[i - 1, j - S(i)] + \zeta(i)\}$ 
4:   end for
5: end for
6:  $c \leftarrow ETH$ 
7: for  $i \leftarrow 1$  to  $n - 1$  do
8:   if  $\mathcal{V}[i, c] \neq \mathcal{V}[i + 1, c]$  then
9:      $ForwardList.add(m_i)$ 
10:     $c \leftarrow c - S(i)$ 
11:   end if
12: end for
13: if  $\mathcal{V}[n][c] > 0$  then
14:    $ForwardList.add(m_n)$ 
15: end if
16: return  $ForwardList$ 

```

4.3 基于数据项选择的改进路由

在本节中，给出关于改进 PRoPHET 路由的关键细节。其核心问题已在4.2节中定义。首先，将利用动态规划方法解决该问题。然后将阐明如何在整个路由过程中维护所需要的信息，最后给出路由协议的整个步骤。

4.3.1 动态规划求解

解决该最优决策问题的方法，如算法 5 所示。整个动态规划求解过程如 1–5 行所示，解的构造过程如 6–19 行所示。

这里给出一个贯穿本文的实例，如表 4–2 所示，其中给出了每条消息对应的大小及其传输收益值 ζ 。在4.2.2小节中，已讨论如何对于任意一对节点 n_a 及 n_b 预测吞吐量的值 $ETH_{(a,b)}$ 。基于此，表 4–3 给出本例的计算过程。

4.3.2 路由协议描述

本小节给出路由协议描述。如 PRoPHET 协议一样，首先需要每个节点维护一张表用以记录节点间预测出的接触概率。此外，由于链接吞吐量的预测是基于连接的持

表 4-3 动态规划计算过程

i	1	2	3	4	5
$S(i)$	1K	2K	5K	6K	7K
$\zeta(i)$	0.10	0.60	0.18	0.22	0.28
0	0	0	0	0	0
1	1	1	1	1	1
2	1	6	6	6	6
3	1	7	7	7	7
4	1	7	7	7	7
5	1	7	18	18	18
6	1	7	19	22	22
7	1	7	24	24	28
8	1	7	25	28	29
9	1	7	25	29	34
10	1	7	25	29	35
11	1	7	25	40	40

表 4-4 路由信息表

1	2	3	\cdots	$ \mathcal{N} $
$P_{(a,n_1)}$	$P_{(a,n_2)}$	$P_{(a,n_3)}$	\cdots	$P_{(a,n_{ \mathcal{N} })}$
k_1	k_2	k_3	\cdots	$k_{ \mathcal{N} }$
$\tau_{(a,n_1)}$	$\tau_{(a,n_2)}$	$\tau_{(a,n_3)}$	\cdots	$\tau_{(a,n_{ \mathcal{N} })}$
$t_{s_{(a,n_1)}}$	$t_{s_{(a,n_2)}}$	$t_{s_{(a,n_3)}}$	\cdots	$t_{s_{(a,n_{ \mathcal{N} })}}$
$t_{e_{(a,n_1)}}$	$t_{e_{(a,n_2)}}$	$t_{e_{(a,n_3)}}$	\cdots	$t_{e_{(a,n_{ \mathcal{N} })}}$

续时间的，因此需要让每个节点维护两个变量 t_s 和 t_e ，以及预测值 τ ，该三个变量都用于描述最近一次发生的连接。最后，需要维护一个变量 k ，用以记录流逝的单位时间数目。路由信息表用符号 $TABLE[a]$ 表示，该表结构如表 4-4 所示，其空间复杂度为 $O(|\mathcal{N}|)$ 。

整个路由协议分为两部分，即算法 6 所示的信息交换协议及算法 7 所示的数据传输协议。

在算法 6 中，主要解决的问题即是更新路由协议需要的信息，并且保证信息在节点之间交换。更新过程如 1-4 行所示，其中用到 PRoPHET 中的公式对接触概率进行预测，此外也用到本章提出用于更新 τ 值的算法。在第 5 行，将信息表 $TABLE$ 向网络中 n_a 的所有邻居节点进行广播。在 6-8 行，若当前节点 n_a 收到其它节点的请求， $TABLE[a]$ 将会被传输给发出该请求的节点。如 9-11 行所示，当节点 n_a 从任意其它

Algorithm 6 Information exchange protocol

Triggering Condition:

In each time unit

n_a Executes:

- 1: **for** each column record i of n_a in 表 4-4 **do**
 - 2: update $P(a, b_i)$ by equation (4-1)–(4-3).
 - 3: call **Algorithm 4** to update $\tau_{(a,b_i)}$.
 - 4: **end for**
 - 5: broadcast the request for $TABLE$ to $n_a.neighbors$
 - 6: **if** received request for $TABLE$ from any node n_b **then**
 - 7: forward $TABLE[a]$ to n_b
 - 8: **end if**
 - 9: **if** received $TABLE[b]$ from any node n_b **then**
 - 10: call **Algorithm 7**
 - 11: **end if**
-

Algorithm 7 Data transmission protocol

Triggering Condition:

n_a and n_b are in contact

n_a Executes:

- 1: **for** $\forall M_i$ in the buffer of n_a **do**
 - 2: $d_i \leftarrow$ the destination node of m_i .
 - 3: **if** $TABLE[a].P_{(a,d_i)} > TABLE[b].P_{(b,d_i)}$ **then**
 - 4: $MessageList(n_a).add(M_i)$
 - 5: **end if**
 - 6: **end for**
 - 7: estimate $ETH_{(a,b)}$ by **equation (4-8)**.
 - 8: **for** $\forall M_i \in MessageList(n_a)$ **do**
 - 9: calculate $\zeta(i)$ by **equation 4-5**
 - 10: get $ForwardList(n_a)$ by **Algorithm 5**
 - 11: **end for**
 - 12: $Sort(ForwardList, ascend, TTL)$
 - 13: $ForwardList(n_a).transmitTo(n_b)$
-

节点 n_b 收到 $TABLE[b]$ 时，将会调用数据传输协议。换言之，此为数据传输协议的触发条件。

在算法 7 中，当前节点 n_a 首先扫描其缓存，将满足条件 $P(b, d_i) > P(a, d_i)$ 的任意消息加入消息传输列表。消息传输策略与 PRoPHET 算法相同，即仅当节点 n_b 对该消息的投递率高于 n_a 时， n_a 才将该消息转复制给 n_b 。此外，每条连接的吞吐率也被纳入考虑，如算法 7 余下部分所示。在第 7 行中，预测了节点 n_a 及 n_b 之间连接的吞吐率。随之，对每条消息 m_i ，依照公式 (4-5) 计算传输效用函数值 $\zeta(i)$ 。最终，利用算法 5，从消息列表 *MessageList* 中获取整个转发列表 *ForwardList*。在 *ForwardList* 中，消息按照 *TTL* 的值增序排序，此为确保即将到期的消息具有较高的传输优先级，从而一定程度上降低丢包（如前所述，到期消息将被节点从缓存中删除）。在算法第 13 行，所有位于 *ForwardList* 中的消息将以队列顺序依次由节点 n_a 转发给 n_b 。

4.4 仿真实验

仿真实验结果从仿真器 ONE [7] 中获得。首先，采用被广泛运用的真实数据集 Cambridge-iMote 对协议进行仿真；该数据集记录了几组持有手持蓝牙设备的小组成员两个月的接触记录信息。在本实验中，移动的节点主要为 Cambridge University 的学生组成；在整个实验过程中，这些学生一直携带有 iMote 移动端。另一个实验场景是基于 Helsinki City Scenario 的，该仿真基于综合移动模型。仿真分为如下几组：(1) Cambridge-iMote 场景改变缓存大小；(2) Cambridge-iMote 场景改变消息 TTL；(3) Helsinki City Model 场景改变改变缓存大小；(4) Helsinki City Model 场景改变消息 TTL。本节将基于消息投递率，网络开销以及平均时延三个指标，对五种路由协议进行评估比较。

4.4.1 Cambridge-iMote 场景仿真

在该仿真中，随机产生 3300 条消息，每条消息对应唯一的源节点和目的节点，源节点与目的节点从所有节点中随机选取。对应投递率，网络开销以及投递时延三个指标，仿真时间分别设为整个数据集对应时间的 25%，50%，75% 以及 100%。图 4-2 为改变节点缓存大小的仿真结果，消息的 TTL 固定在 300 分钟。在图 4-3 改变消息 TTL 的仿真中，缓存大小固定在 50M。其它的仿真设定如表 4-5 所示。

如图 4-2 所示，结果表明本章中改进 PRoPHET 后的协议 Throughput，在投递率及投递时延方面的性能表现强于 PRoPHET 以及 Epidemic 协议，且能够一定程度上降低网络开销。尽管与 MaxProp 算法相比，本章算法 Throughput 在投递率方面优势不大，然而 Throughput 协议在其余两方面性能表现，即投递时延和网络开销，要强于 MaxProp 协议。从图 4-2(a), 4-2(d), 4-2(g) 和 4-2(j) 可以看出，随着仿真时间的增加，Throughput 算法改善投递率的效果亦增加。如图 4-2 第二列所示，Throughput 协议具有比 PRoPHET 更低的平均时延。参照图 4-2 第三列的结果，随着仿真时间加长，Throughput 协议能够很大程度上改善网络开销的性能表现。

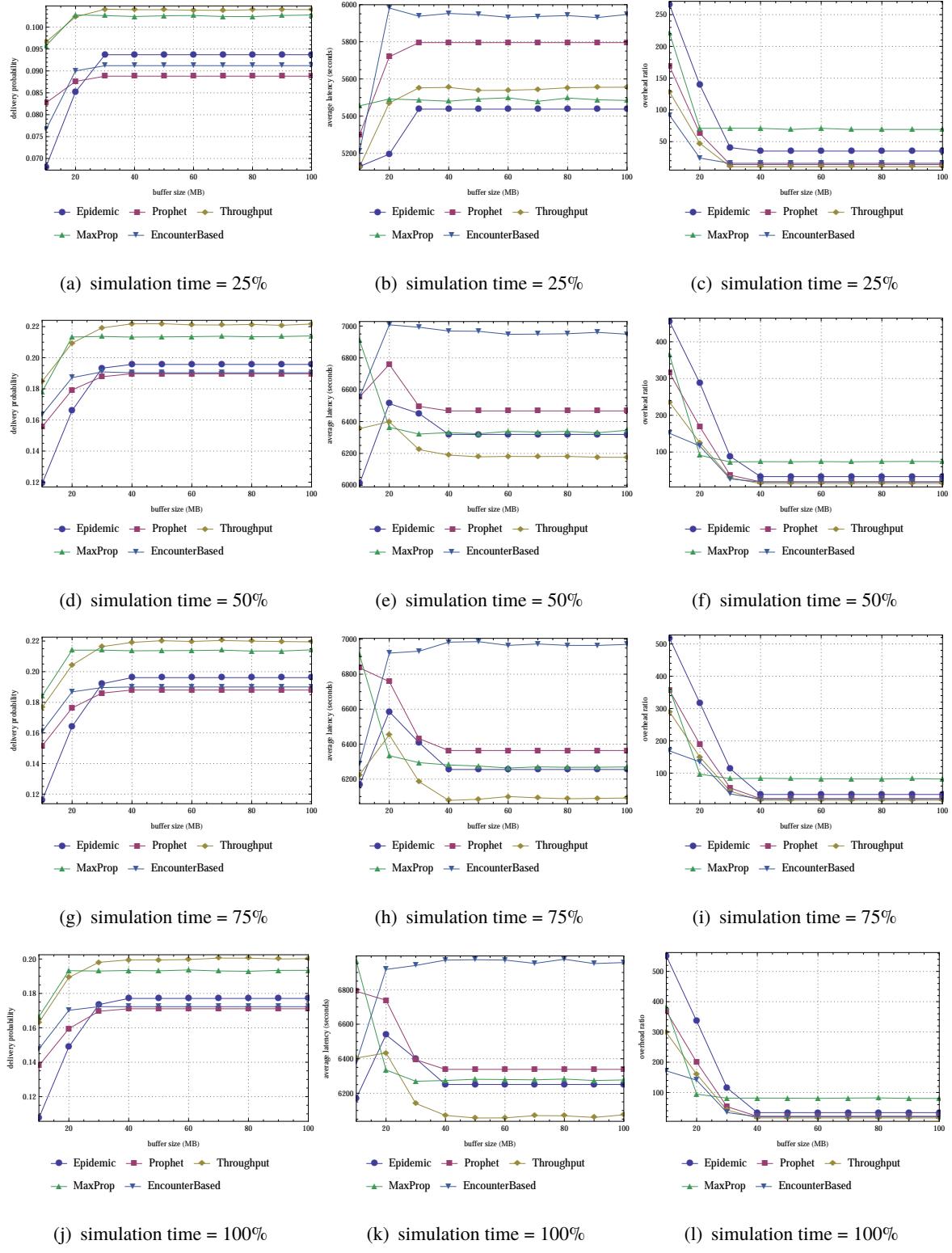


图 4-2 [Cambridge-iMote] 不同仿真时间，改变节点缓存大小

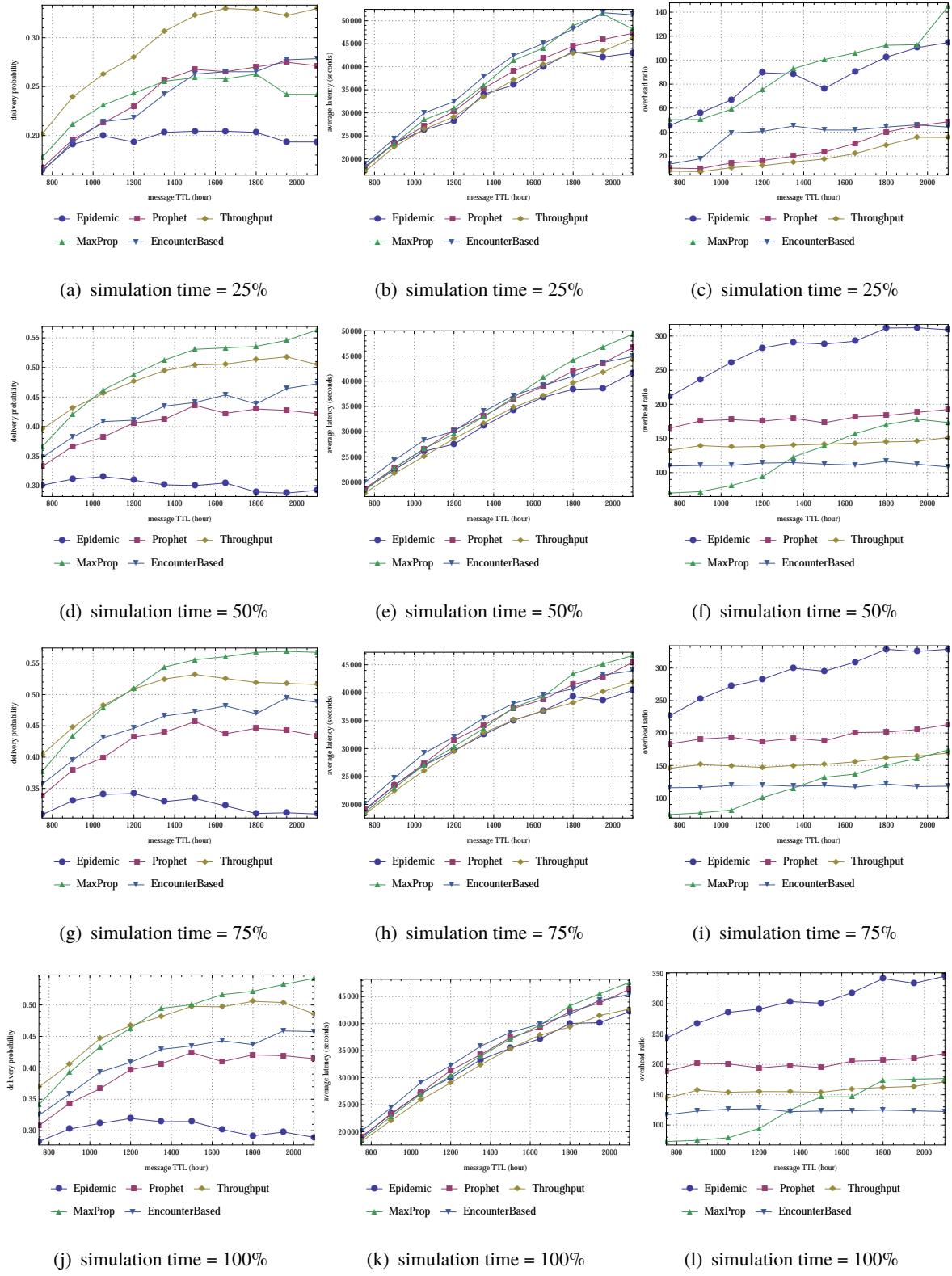


图 4-3 [Cambridge-iMote] 不同仿真时间，改变消息 TTL

表 4-5 Simulation settings of Cambridge-iMote trace

parameter name	range
number of nodes	36
entire sim_time(days)	11.5
message size(KB)	500–1024
bandwidth(KBps)	250
P_{init}	0.75
α	0.5
β	0.25
γ	0.98

表 4-6 Simulation settings of Helsinki City Scenario

parameter name	range(default value)
pedestrians/cars	30-90
world size($m \times m$)	4500×3000
initial tickets number	6
message TTL(min)	180–425 (300)
simulation time(hours)	12
message size(KB)	500–1024
pedestrian buffer(MB)	5–55 (25)
tram buffer(MB)	500
bluetooth range(m)	10
highspeed range(m)	1000
bluetooth (KBps)	250
highspeed (MBps)	10
pedestrian speed(m/s)	0.5–1.5
message interval(s)	35–40

图 4-3 的结果表明，Throughput 协议在投递率以及网络开销方面的表现优于其余四种算法；在平均时延方面有微小改进。与 Epidemic, PRoPHET 以及 EncounterBased 协议相比，Throughput 具有更优的性能表现。与 MaxProp 相比，Throughput 算法在仿真时间较短时，性能表现明显好于 MaxProp。该结果表明 Throughput 能够更快的达到算法最佳表现。

4.4.2 Helsinki City Scenario 仿真

在 Helsinki City Scenario[12] 中，节点持有移动设备，并利用蓝牙接口进行通信，数据率设为 250 KBps，传输范围直径设为 10 m。在该场景中，每个节点的初始缓存大小设为较小的值，在 5 M 到 55 M 之间变动。地图中设有六辆电车沿着指定路径移动，并且配有额外的高速通信接口，数据率为 10 MBps，传输直径为 1000 m 电车之间可以使用该接口进行通信。其余仿真设置如表 4-5

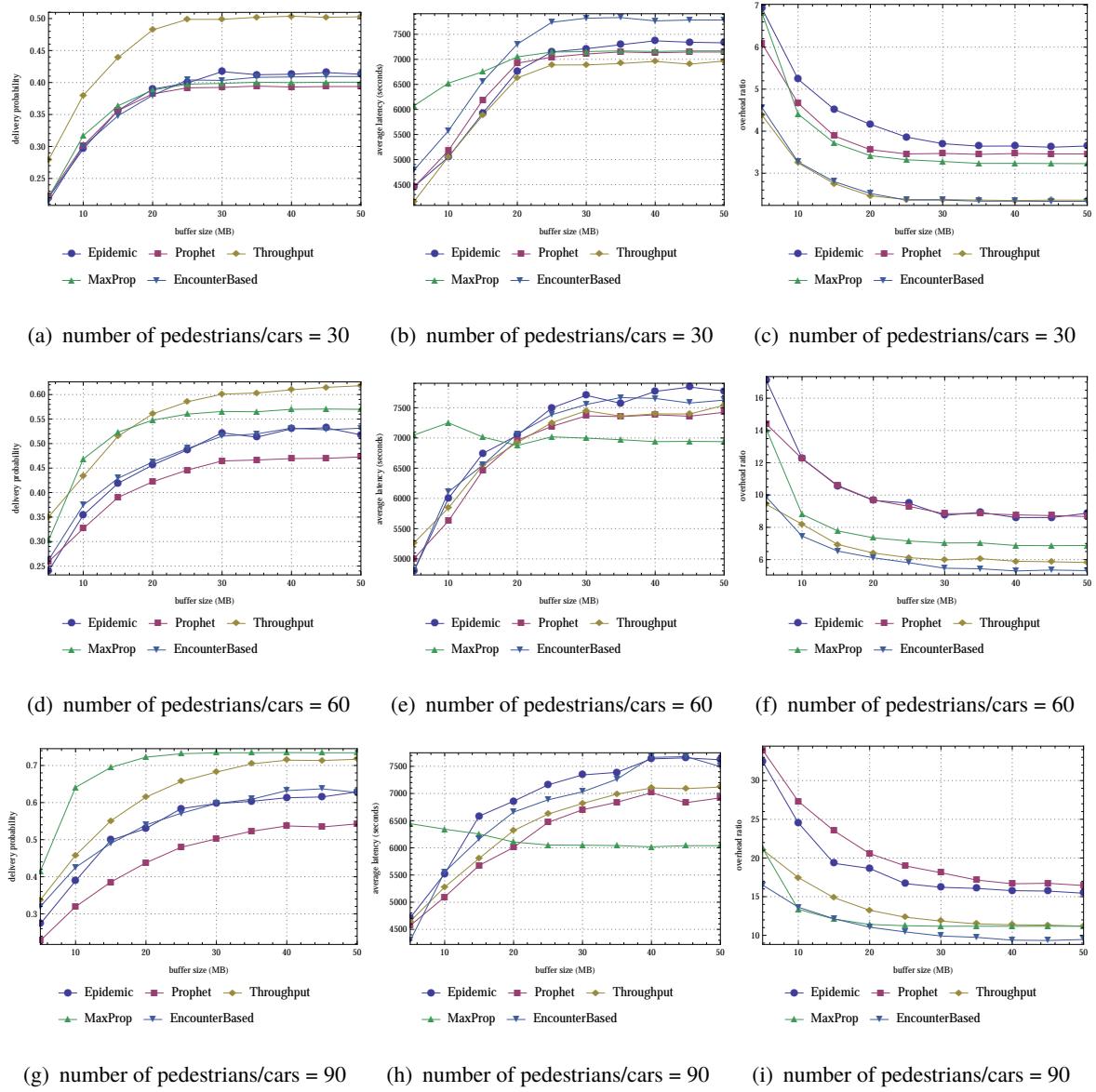


图 4-4 [Helsinki City Scenario] 不同节点数目，改变节点缓存大小

如图 4-4 结果所示，Throughput 路由算法在投递率上较大程度优于 Encounter-Based, PRoPHET 以及 Epidemic 三种路由。随着仿真场景中行人数目以及汽车数目增加，MaxProp 路由协议的性能表现逐渐好转，最终在三个指标上优于其余算法。然而

当网络中不具备足够多的行人及车辆节点时, MaxProp 的性能表现几乎与 PRoPHET 一样。在这种情况下, Throughput 在三项指标上的表现全面优于 MaxProp。从图 4-4(a), 图 4-4(d) 以及图 4-4(g) 中, 可以看出投递率的性能表现随着行人/汽车节点数目的增加而上升。如图 4-4 第二列所示, Throughput 算法与 PRoPHET 有几乎相同的投递时延。从图 4-4 第三列可以看出, Throughput 在网络开销上的改善效果较为明显。综合图 4-4 所有结果来看, 当节点数目较少时, Throughput 具有综合最优的性能表现。

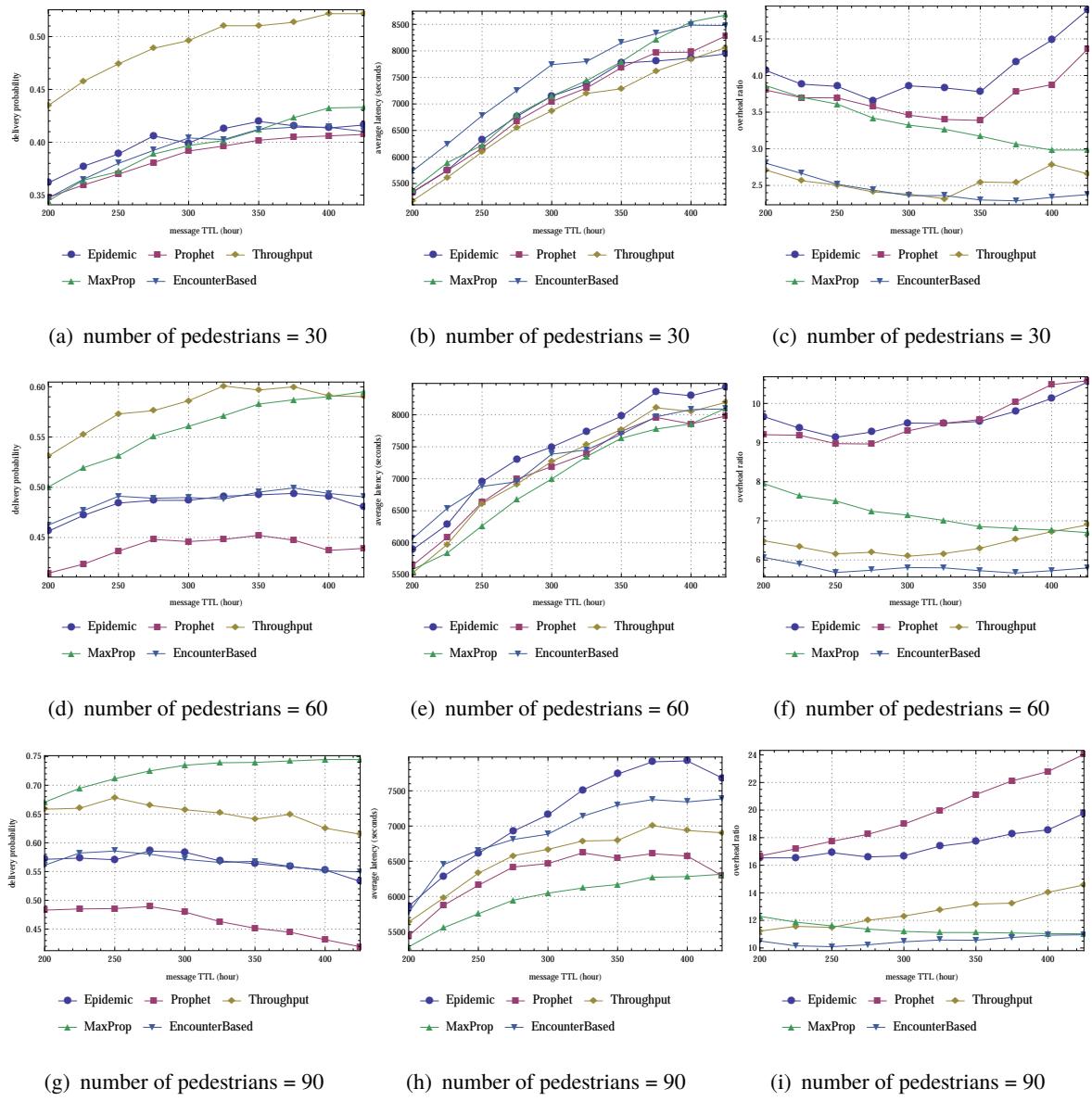


图 4-5 [Helsinki City Scenario] 不同节点数目, 改变消息 TTL

图 4-5 展示的结果与图 4-4 类似, 即当节点数目较小时, Throughput 算法在投递率和网络开销两个指标上优于其它四种路由算法; 在投递时延上有微小优势。总而言之, 若是在节点数目相对丰富, 网络中的节点密度较大时, 应采用 MaxProp 算法。当网络

中节点较为稀疏时，采用 Throughput 算法能获得更好的性能表现。

4.5 本章小结

本章通过设计适用于机会路由的数据项选择机制，改进了机会路由的性能表现。其出发点在于，由于节点间带宽和接触时间受限所导致的连接有限的吞吐量会造成消息的传输失败，从而浪费了节点间宝贵的接触机会。通过定义概念“消息传输效用”，在 PRoPHET 的基础上加上数据项选择机制，得到改进后的路由算法 Throughput。仿真结果表明，该数据项选择机制能够在消息投递率，消息投递时延，网络开销三方面较大程度的改进路由性能表现。进一步地，本章定义的“消息传输效用”概念亦可用于其它路由协议上以设计相应的数据项选择机制，该课题将作为以后的研究工作。

第五章 基于跳数的启发式距离向量算法

本章提出一种用于机会网络的基于跳数的启发式距离向量算法 HCH，主要贡献如下：

1. 设计了启发函数用以预测消息投递所需跳数。启发策略所需的信息由节点间传递的消息数据包携带。
2. 形式化定义了矩阵运算，从而将跳数预测计算转化为矩阵运算。
3. 利用 ONE 模拟器对 HCH 算法进行了仿真，结果表明 HCH 具有较高的投递率以及较低的投递时延，且网络开销保持在可接受水平。

本章组织如下：5.1 节中介绍了系统模型及路由模型；5.2 节中提出了跳数预测算法；5.3 节中提出了基于跳数的启发式距离向量路由算法；5.4 节为仿真实验；5.5 节概括了本章内容。

5.1 系统模型

在该模型中，网络包含一组移动节点，节点之间对等通信。基本假设如下：

- 所有节点以对等方式通信，即网络中不存在任何辅助消息进行传递的基础设备。换言之，不存在路由器类似的设备用于转发消息，所有的节点合作以多跳的方式进行消息投递，节点自身将做出对消息的转发决策。
- 节点的移动方式多变难以预测，即难以对某个节点预测其下一时间或下一时间段的路径及地点。

数学符号如表 5-1 所示。节点集合以 $V = \{v | 1 \leq v \leq n\}$ 表示。为便于分析，描述路由的过程只针对某一条消息而言，并以 s 代表其源节点， d 代表其目的节点，该消息用符号 $M_k(s, d)$ 表示，其索引号为 k 。符号 $hop(k)$ 为一个整数，表述消息 M_k 所经过的跳数值。符号 $\overline{hop}(i, j)$ 表示消息从节点 i 到达节点 j 之间的平均跳数。

在此模型下，本章解决如下问题：

- 如何设计效用指标衡量网络当前状态
- 如何从网络中收集信息，从而动态计算效用指标
- 如何根据效用值选择节点的转发策略

表 5-1 数学符号定义

notation	meaning
n	节点总数
V	节点集合 ($ V =n$)
$M_k(s, d)$	索引号为 k 的消息，其中源节点为 s ，目的节点为 d
$\overline{hop}(i, j)$	节点 i 和节点 j 之间的平均跳数
$hop(k)$	消息 M_k 当前经过的跳数
$h(i, j)$	节点 i 和 j 之间的估计跳数

5.2 消息投递跳数预测

在本节中，定义了用于路由决策的效用函数，即投递消息所需跳数的预测值。首先讨论如何从网络中收集所需信息。然后基于收集到的信息提出启发函数。路由协议的转发决策将由该函数决定。

5.2.1 消息收集

在提出的算法中，利用了消息跳数指标来做出路由决策。跳数信息有很多方式获取，本章中的方法，即是令网络中的消息数据包头部携带有该消息所进过的节点的记录。当某消息数据包到达一个节点时，该节点将获得该消息所经过的节点到该节点所经过的条数值。例如，若消息 M_k 产生于节点 s ，并沿路径 $s \rightarrow p \rightarrow q \rightarrow r$ 转发，则从该消息的头部，节点 r 可以获知：节点 s 与节点 r 之间为 3 跳， p 与 r 之间为 2 跳， q 与 r 之间为 1 跳。这些记录是针对消息 M_k 而言的，若对节点 r 所收到的所有消息的对应记录求平均值，则得到这些节点到 r 的平均跳数；在本章中，利用滑动窗口方法实现该目的。每个节点维护一个矩阵，记为 \mathbf{A} ，用于记录其它所有节点到该节点的跳数值。每个矩阵元素将对应一个滑动窗口。

图 5-1 为矩阵 \mathbf{A} 及其每个元素所对应的滑动窗口工作方式。对于矩阵 \mathbf{A} 中的每个元素，都有一个对应的滑动窗口记录了之前收到的消息对应的跳数信息。滑动窗口越长，则对网络状态的变化越不敏感，反之则越敏感。通过对滑动窗口中记录的跳数值，即从 $t - r + 1$ 到 t 位置取平均，则得到该矩阵元素对应节点到当前节点的平均跳数。由此得到跳数预测值，记为 $a(t + 1)$ 。

维护矩阵 \mathbf{A} 以及滑动窗口的过程如算法 8 所示。算法需要的信息如下：消息 M_k ，其头部记录着消息所经过的跳数信息。每当数据包到达时，算法即运行一次。在第 1 行，从消息 M_k 中获得其源节点的 ID。在第 2 行， M_k 所经过的所有节点按顺序存在 *Sequence* 数组中。在算法 8 中有两个循环，分别用于更新滑动窗口以及矩阵 \mathbf{A} 的信息。通过运行第一个循环，如 4-7 行所示，跳数信息存于滑动窗口对应的 t 位置。8-11 行为算法第二个循环，通过对滑动窗口所有记录求平均的方式，计算出矩阵 \mathbf{A} 对应位

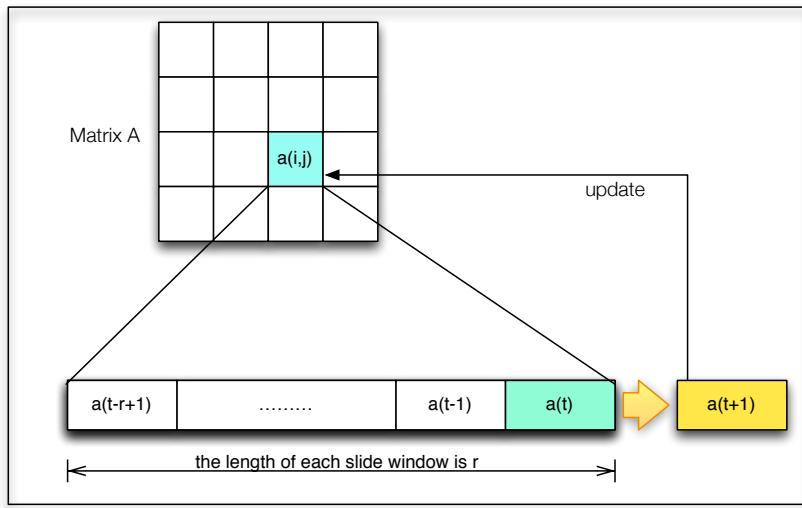


图 5-1 矩阵 A 及其对应的滑动窗口

置的新的元素值。由此，矩阵 A 的每一个元素，反映了某一对节点之间的平均跳数。在 5.3 小节中，基于矩阵 A 维护的跳数信息，设计了用于路由决策的启发函数。

5.2.2 启发函数

公式 (5-1) 为启发函数，其参数为当前节点索引号 i 及消息 M_k ；其中 $hop(k)$ 表示了 M_k 经过的跳数值， $h(i, d)$ 表示从当前节点 i 到消息 M_k 目的节点 d 的估计跳数。公式 (5-1) 所示的函数 $\mathcal{H}(i, k)$ 由两部分组成。第一部分即反应了该消息经过的实际跳数，第二部分估计了该消息在到达目的节点之前还需经过的跳数。

$$\mathcal{H}(i, k) = hop(k) + h(i, d) \quad (5-1)$$

公式 (5-1) 定义了启发函数 $h(i, d)$ 。其中符号 $path_c[i \rightarrow d]$ 代表了从节点 i 到节点 d 的路径，符号 m 代表从节点 i 到节点 d 的总的可能路径数。由此， $h(i, d)$ 即为从节点 i 到节点 d 的平均跳数。

$$h(i, d) = \frac{\sum_{c=1}^{c=m} path_c[i \rightarrow d]}{m} \quad (5-2)$$

接下来给出 $h(i, d)$ 的计算过程，将引入自定义的矩阵运算 \odot ，如下。

定义 17. 假设 M 和 N 都是 $n \times n$ 的方阵， $O = M \odot N$ ，则对于矩阵 O 的任意元素 $o_{i,j}$ ，有

$$o_{i,j} = \sum_{k=1}^{k=n} (m_{i,k} + n_{k,j}) / w$$

其中有

$$w = |\{c | c = m_{i,k} + n_{k,j} \text{ and } c > 0\}|$$

Algorithm 8 Maintaining the matrix A and its slide-windows**Require:** packet M_k , current time $t + 1$ **Ensure:** matrix A , slide-window win When packet M_k comes**local variables:** i, j, c

```

1:  $i \leftarrow M_k.getSourceNodeID()$ 
2:  $Sequence \leftarrow M_k.getPassedNodes()$ 
3:  $c \leftarrow 1$ 
4: for  $j \in Sequence$  do
5:    $win[i, j, t] \leftarrow c$ 
6:    $c \leftarrow c + 1$ 
7: end for
8: for  $i \leftarrow 1$  to  $n$  do
9:   for  $j \leftarrow 1$  to  $n$  do
10:     $a_{i,j} \leftarrow \left( \sum_{k=t-r+1}^{k=t} win[i, j, k] \right) / r$ 
11:   end for
12: end for
13: return  $A, win$ 

```

算法 9 用于计算函数值 $h(i, d)$ 。算法的输入为任意节点 i 所维护的矩阵 A 。通过调用该算法，最终可对节点 i 所持有的任意一条消息获知其对应的 $h(i, *)$ 的值 ($*$ 为该消息的目的节点)。算法的外层循环遍历节点 i 缓存中的所有消息，如第 1 行所示。第 2–4 行初始化三个变量 h, c 和 M ，其中 M 为矩阵变量。该三个变量将在内层循环的每次迭代中更新。变量 c 为局部变量，用于在每次循环迭代中累加总的路径数。矩阵变量 M 初始化为单位矩阵 Λ ，在每次循环迭代中都会乘以矩阵 A 。第 5 行与第 6 行用于获取当前节点的 ID 和消息 M_k 目的节点的 ID。内层循环直到矩阵 M 的元素 $m_{i,d}$ 降为 0 时结束；该元素为 0 即意味着节点 i 与节点 d 之间不存在 h 跳的路径。最终， $h(i, d)$ 设为节点 i 与节点 d 之间所有可能路径跳数的平均值，如算法第 13 行所示。

图 5–2 展示了消息从源节点到目的节点跳数的估计过程。假设消息 M_k 在源节点 s 产生，目的节点为 d ， i 是当前运行算法的节点。对于消息 M_k 而言，从节点 s 到 i 一共经过了 6 跳，因此有 $hop(k) = 6$ 。为了简便起见，在图中，当前节点 i 标号为 1，目

Algorithm 9 Heuristic value calculation

Require: Matrix A of node i

Ensure: $h(i, *)$

local variables: i, h, c, d

1: **for** $M_k \in i.messages$ **do**

2: $h \leftarrow 0$

3: $c \leftarrow 0$

4: $M \leftarrow \Lambda$

5: $i \leftarrow getHostID()$

6: $d \leftarrow M_k.getDestinationID()$

7: **repeat**

8: $M \leftarrow M \odot A$

9: $h \leftarrow h + m_{i,d}$

10: $c \leftarrow c + 1$

11: **until** $m_{i,d} = 0$

12: $h(i, d) = h/c$

13: **end for**

14: **return** $h(i, *)$

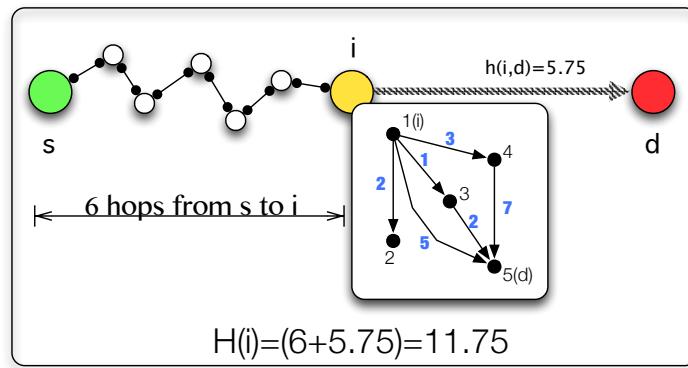


图 5-2 跳数计算过程

的节点 d 标号为 5，如图 5-2 所示。对于内层循环的首次迭代，有

$$M = \Lambda \odot A = \begin{pmatrix} 0 & 2 & 1 & 3 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

表 5-2 机会路由决策

策略：持有该消息的节点	对应情况
节点 v 和节点 u	向网络中增加一个新的消息副本
节点 v	节点 u 不如节点 v 优
节点 u	节点 u 比节点 v 优

其中，有

$$m_{i,d} = m_{1,5} = 5$$

$$h = 0 + 5 = 5$$

$$c = 0 + 1 = 1$$

对于第二次迭代，有

$$M = A \odot A = \begin{pmatrix} 0 & 0 & 0 & 0 & 6.5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

其中，有

$$m_{i,d} = m_{1,5} = [0 + 0 + (1 + 2) + (3 + 7) + 0] / 2 = 6.5$$

$$h = 5 + 6.5 = 11.5$$

$$c = 1 + 1 = 2$$

循环将在第三次迭代开始之前结束，因为元素 $m_{i,d} = m_{1,5} = 0$ ，最终得到如下结果

$$h(i, d) = \frac{h}{c} = \frac{11.5}{2} \approx 5.75$$

注意上述计算结果 $h(i, d)$ 是公式 (5-2) 的一个近似值。在图 5-2 所示例子中，通过公式 (5-2) 可得平均跳数值为 6，接近于计算结果 5.75。

5.3 路由协议

表 5-2 展示了两个相遇的节点可选的转发策略。在本章路由中，第一条原则是不刻意减少在网络中产生的副本数量。第二条原则旨在将网络中副本数量控制在一定范围之内，即当且仅当相遇的两个节点 u 和 v 都难以将数据包转发到目的节点时，才向网络中注入新的副本。在这种情况下，多拷贝策略被用来提高路由性能表现；在其它情况，只会在 u 和 v 之间选取唯一节点作为中继节点。

Algorithm 10 Routing**For** each node $v \in V$ **do**

```

1:  $neighbors \leftarrow v.getNeighbors()$ 
2: for  $M_k \in v.messages$  do
3:   for  $u \in neighbors$  do
4:      $s \leftarrow M_k.getSourceID()$ 
5:      $d \leftarrow M_k.getDestinationID()$ 
6:     if  $\mathcal{H}(v, k), \mathcal{H}(u, k) > \overline{hop}(s, d)$  then
7:        $extra \leftarrow \text{true}$ 
8:     else
9:        $extra \leftarrow \text{false}$ 
10:    end if
11:    if  $extra = \text{true}$  then
12:       $v$  sends an extra copy of  $M_k$  to  $u$ 
13:    else if  $\mathcal{H}(u, k) \leq \overline{hop}(s, d)$  then
14:       $v$  turns over its own copy of  $M_k$  to  $u$ 
15:    end if
16:  end for
17: end for

```

算法 10 为路由细节过程。如第 1 行所示，首先从当前节点 v 取得其邻居节点集合。然后，如 2–16 行所示，对存于节点 v 缓存中的每条消息，及所有其邻居节点做出路由决策。决策基于小节中的启发函数做出。考虑消息 M_k ，设其源节点为 s 目的节点为 d 。在 $\mathcal{H}(v, k), \mathcal{H}(u, k)$ 的值皆大于 $\overline{hop}(s, d)$ 时，当前节点 v 及其邻居节点 u 其估计跳数皆大于 s 与 d 之间的平均跳数，这意味着对于节点 v 和 u 将消息在平均跳数之内传输至目的节点都很困难，此时多拷贝策略将被触发。

在 6–10 行，根据 $extra$ 指示变量选择是否加入新的消息 M_k 的拷贝。若 $extra = true$ ， v 对节点 u 产生一份额外的 M_k 的拷贝，如第 12 行所示，这对应表 5–2 的第一行策略。在第 13–15 行，若 $\mathcal{H}(u, k) \leq \overline{hop}(s, d)$ ，节点 v 将把 M_k 移交给节点 u ，并将其从 v 中缓存移除。在这种情况下，有 $\mathcal{H}(u, k) \leq \mathcal{H}(v, k) \leq \overline{hop}(s, d)$ ；由此认为节点 u 足以将消息 M_k 递交于目的节点；对应表 5–2 中第三条表目。剩下的唯一一种情况即 $\mathcal{H}(v, k) \leq \mathcal{H}(u, k) \leq \overline{hop}(s, d)$ ，意味着节点 u 不是优于 v 的中继节点选择（针对消息 M_k 而言），且节点 v 自身足以将消息在平均时延内传递至目的节点。在该情况下， v 不向 u 转发 M_k ，对应表 5–2 条目二。

表 5-3 Helsinki City 场景仿真设置

parameter name	range(default value)
number of nodes	120
world size($m \times m$)	4500×3000
tickets for S & W	13
message TTL(min)	200–500 (300)
simulation time(hours)	12
message size(KB)	500–1024
pedestrian buffer(MB)	15–55 (15)
tram buffer(MB)	500
bluetooth range(m)	10
highspeed range(m)	1000
bluetooth bandwidth(KBps)	250
highspeed bandwidth(MBps)	10
pedestrian speed(m/s)	0.5–1.5
message interval(s)	35–40

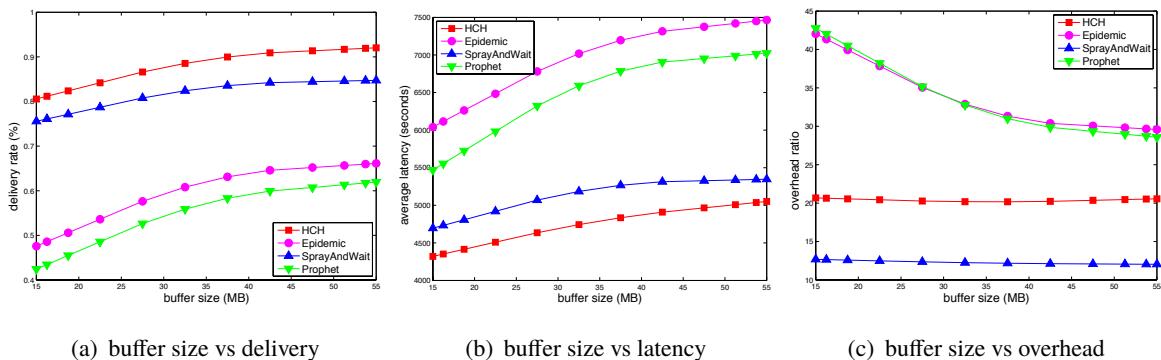


图 5-3 [Helsinki City Scenario] 改变节点缓存大小的仿真结果

5.4 仿真实验

仿真实验基于机会网络模拟器 ONE [7]。本章将提出的算法与 Epidemic, Spray-and-Wait (S & W) 以及 PRoPHET 相比较，仿真实验场景包含 Helsinki City Model 以及 Cambridge-iMote。实验可分为如下几组：Helsinki City Model 下改变节点缓存大小；Cambridge-iMote 下改变节点缓存大小；Helsinki City Model 下改变消息 TTL；Cambridge-iMote 下改变消息 TTL。仿真评估标准分为消息投递率，平均时延，网络开销三个。

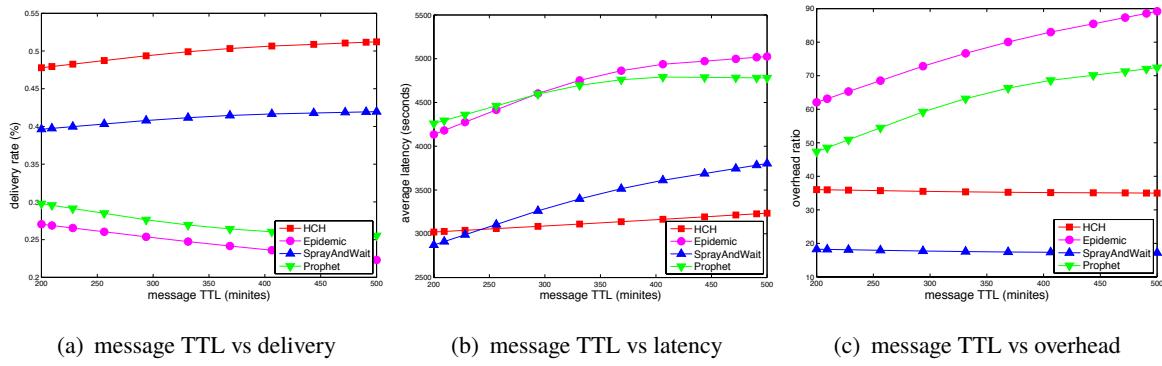


图 5-4 [Helsinki City Scenario] 改变消息 TTL 的仿真结果

表 5-4 Simulation settings of Cambridge-iMote trace

parameter name	range
number of nodes	36
tickets for S & W	5
message TTL(min)	600–2000(1200)
simulation time(days)	11.5
message size(KB)	500–1024
device buffer(MB)	50–150(100)
interface bandwidth(KBps)	250
message interval(s)	35–40

5.4.1 Helsinki City 场景

5.4.2 Cambridge-iMote 场景

5.5 本章小结

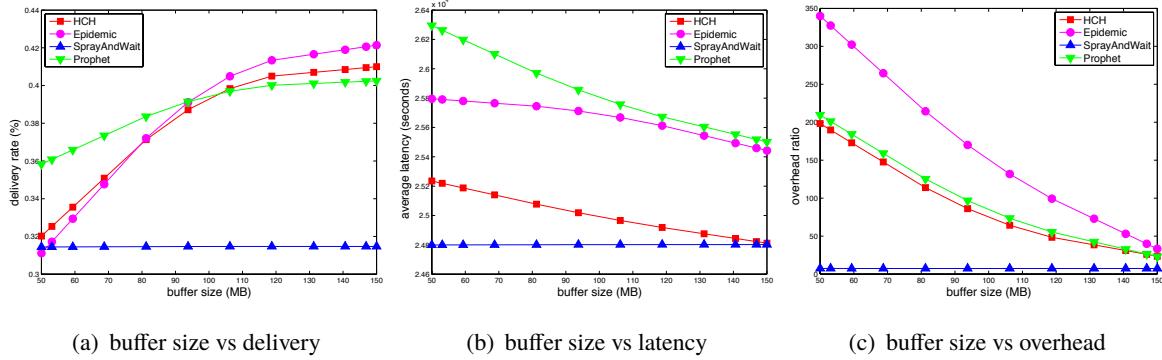


图 5-5 [Cambridge-iMote] Buffer size vs delivery ratio, average latency and overhead ratio.

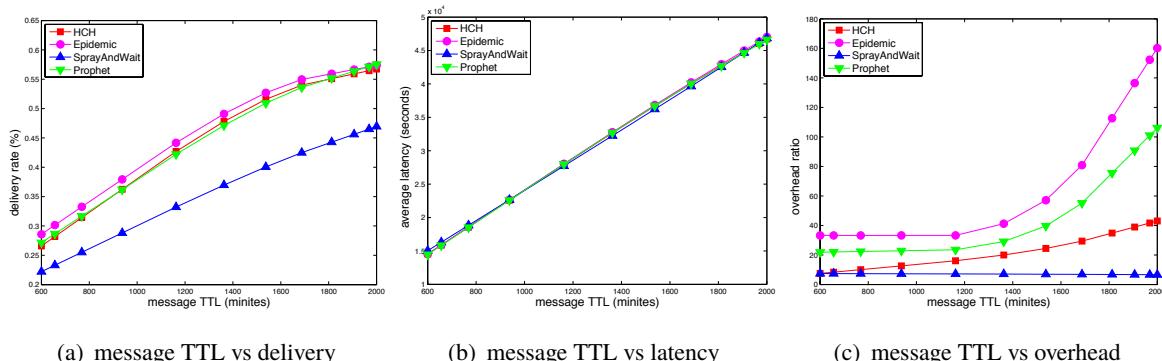


图 5-6 [Cambridge-iMote] Message TTL vs delivery ratio, average latency and overhead ratio.

全文总结

这里是全文总结内容。

参考文献

- [1] CONTI M, GIORDANO S. Mobile ad hoc networking: milestones, challenges, and new research directions[J]. Communications Magazine, IEEE, 2014, 52(1):85–96.
- [2] HENDERSON T, KOTZ D, ABYZOV I. The changing usage of a mature campus-wide wireless network[J]. 2004:187–201. <http://dl.acm.org/citation.cfm?id=1023739>.
- [3] IBRAHIM M, NAIN P, CARRERAS I. Analysis of relay protocols for throwbox-equipped dtns[J]. ... in Mobile, 2009. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5291625.
- [4] JEDARI B, XIA F. A Survey on Routing and Data Dissemination in Opportunistic Mobile Social Networks[J]. arXiv.org, 2013. <http://arxiv.org/abs/1311.0347>.
- [5] VAHDAT A, BECKER D. Epidemic Routing for Partially Connected ad hoc Networks[J]. Acta Electronica Sinica, 2000, Cs-2000-06(Tech. Rep.). http://scholar.google.com/scholar?q=related:6bSunIRK3e0J:scholar.google.com/&hl=en&num=30&as_sdt=0,5<http://publication/livfe/id/65938>.
- [6] SPYROPOULOS T, PSOUNIS K, RAGHAVENDRA C S. Spray and wait[C]//Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05. New York, New York, USA: ACM Press, 2005:252–259. <http://dl.acm.org/citation.cfm?id=1080143><http://publication/livfe/id/67848><http://portal.acm.org/citation.cfm?doid=1080139.1080143>.
- [7] KERÄNEN A, OTT J O R, KÄRKÄINEN T. The ONE simulator for DTN protocol evaluation[C]//Proceedings of the Second International ICST Conference on Simulation Tools and Techniques.[S.l.]: ICST, 2009:1–10. <http://dl.acm.org/citation.cfm?id=1537683>[file:///Users/lei/Library/ApplicationSupport/Papers2/Articles/2009/Ker%C3%A4nen/2009Ker%C3%A4nen.pdf](http://Users/lei/Library/ApplicationSupport/Papers2/Articles/2009/Ker%C3%A4nen/2009Ker%C3%A4nen.pdf)<http://publication/doi/10.1109/WD.2011.6098142><http://eudl.eu/doi/10.4108/ICST.SIMUTOOLS2009.5674>.
- [8] EKMAN F, KERÄNEN A, KARVO J, et al. Working day movement model[J]. MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models, 2008:1–8. <http://www.netlab.tkk.fi/~jo/papers/2008-mobmod-working-day-model.pdf>.

- [9] ERRAMILLI V, CHAINTREAU A, CROVELLA M, et al. Delegation forwarding[C]//Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '08. New York, New York, USA: ACM Press, 2008:251. <http://dl.acm.org/citation.cfm?id=1374653>papers2://publication/livfe/id/67810<http://portal.acm.org/citation.cfm?doid=1374618.1374653>.
- [10] DALY E M, HAAHR M. Social network analysis for routing in disconnected delay-tolerant MANETs[C]//Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing - MobiHoc '07. New York, New York, USA: ACM Press, 2007:32–40. <http://portal.acm.org/citation.cfm?id=1288107.1288115&coll=DL&dl=GUIDE&CFID=105151204&CFTOKEN=34046091>papers2://publication/livfe/id/65951<http://portal.acm.org/citation.cfm?doid=1288107.1288115>.
- [11] LINDGREN A, DORIA A, SCHEL E N O. Probabilistic routing in intermittently connected networks[J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2003, 7(3):19–20. http://download.springer.com/static/pdf/410/chp%3A10.1007%2F978-3-540-27767-5_24.pdf?auth66=1354693724_8a4222e585bb8b3a99efd48511d18650&ext=.pdfpapers2://publication/uuid/5B1B05F2-4EF0-4071-A037-8F19CA620A96<http://portal.acm.org/citation.cfm?doid=961268.961272>.
- [12] KERÄNEN A, KÄRKKÄINEN T, OTT J. Simulating Mobility and DTNs with the ONE[J]. Journal of Communications, 2010, 5(2):92–105. <http://www.ojs.academypublisher.com/index.php/jcm/article/viewArticle/050292105>.

表格索引

3–1 网络场景图 3–3 边权值	11
3–2 图 3–3 中的局部搜索过程示例	19
3–3 图 3–3 例子对应的禁忌搜索过程	25
3–4 Local-MPAR 中的节点状态迁移	27
3–5 Tabu-MPAR 中的节点状态迁移	28
4–1 数学符号	42
4–2 节点 n_a 缓存中的 5 条消息	44
4–3 动态规划计算过程	47
4–4 路由信息表	47
4–5 Simulation settings of Cambridge-iMote trace	52
4–6 Simulation settings of Helsinki City Scenario	52
5–1 数学符号定义	58
5–2 机会路由决策	62
5–3 Helsinki City 场景仿真设置	64
5–4 Simulation settings of Cambridge-iMote trace	65

插图索引

3–1 利用投掷盒完成消息从节点 a 到节点 d 的传递操作	8
3–2 周期 T 划分为 h 个时间槽	9
3–3 网络场景举例	10
3–4 超立方体解空间	22
3–5 Local-MPAR 节点状态迁移图.	26
3–6 Tabu-MPAR 节点状态迁移图.	27
3–7 Local-MPAR 路由过程.	30
3–8 Tabu-MPAR 路由过程.	32
3–9 Working day movement 模型, 结合 Manhattan blocks.	33
3–10 消息投递率 vs. 消息生存时间	35
3–11 端到端平均时延 latency vs. 消息生存时间	35
3–12 网络开销 vs. 消息生存时间	36
3–13 平均跳数 vs. 消息生存时间	36
3–14 消息投递率 vs. 节点缓存	37
3–15 端到端平均时延 vs. 节点缓存	37
3–16 网络开销 vs. 节点缓存	38
3–17 平均跳数 vs. 节点缓存	38
4–1 带宽预测过程	44
4–2 [Cambridge-iMote] 不同仿真时间, 改变节点缓存大小	50
4–3 [Cambridge-iMote] 不同仿真时间, 改变消息 TTL	51
4–4 [Helsinki City Scenario] 不同节点数目, 改变节点缓存大小	53
4–5 [Helsinki City Scenario] 不同节点数目, 改变消息 TTL	54
5–1 矩阵 A 及其对应的滑动窗口	59
5–2 跳数计算过程	61
5–3 [Helsinki City Scenario] 改变节点缓存大小的仿真结果	64
5–4 [Helsinki City Scenario] 改变消息 TTL 的仿真结果	65
5–5 [Cambridge-iMote] Buffer size vs delivery ratio, average latency and overhead ratio.	66
5–6 [Cambridge-iMote] Message TTL vs delivery ratio, average latency and overhead ratio.	66

List of Algorithms

1	Tabu Search Framework.	20
2	Local-MPAR Algorithm.	28
3	Tabu-MPAR Algorithm.	31
4	Updating the τ value	45
5	Get the solution by dynamic programming	46
6	Information exchange protocol	48
7	Data transmission protocol	48
8	Maintaining the matrix A and its slide-windows	60
9	Heuristic value calculation	61
10	Routing	63

攻读学位期间发表的学术论文目录

- [1] CHEN H, CHAN C T. Acoustic cloaking in three dimensions using acoustic metamaterials[J]. Applied Physics Letters, 2007, 91:183518.
- [2] CHEN H, WU B I, ZHANG B, et al. Electromagnetic Wave Interactions with a Metamaterial Cloak[J]. Physical Review Letters, 2007, 99(6):63903.

致 谢

感谢所有测试和使用交大硕士学位论文 L^AT_EX 模板的同学！

感谢那位最先制作出博士学位论文 L^AT_EX 模板的交大物理系同学！

感谢 William Wang 同学对模板移植做出的巨大贡献！

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日期：_____年____月____日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保 密 ，在_____年解密后适用本授权书。

本学位论文属于

不保密 。

(请在以上方框内打“√”)

学位论文作者签名：_____

指导教师签名：_____

日期：_____年____月____日

日期：_____年____月____日

