哈尔滨工业大学深圳校区

毕业设计（论文）中期报告

题　　目　基于信息新鲜度的边缘缓存更新机制研究

姓　　名　邹清林

学　　号　180210405

学　　院　电子与信息工程

专　　业　通信工程

指 导 教 师　罗晶晶

日　　期　2022/3/18

目录

1 课题主要研究内容及进度

1.1 课题主要研究内容

1.1.1 研究目的与背景

缓存和边缘计算在支持为数十亿用户提供现代内容交付网络和电信服务提供方面发挥着关键作用。尤其是新兴的边缘计算技术，其是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供服务，产生更快的网络服务响应。传统的缓存系统拥有一个数据服务器存储着庞大的数据，在用户请求时将内容发送给用户，其存在成本高，网络负载大的缺点，面对内容繁杂的网络，企业需要一种新的缓存技术。边缘计算的应用推动了一种新的缓存技术的产生，即边缘缓存。

边缘缓存利用在传统或超大规模数据中心与访问资源的最终用户之间使用中间存储的操作来减少不必要的下载活动，提高缓存性能。某些基站已经采用了这项技术：将传统的缓存方法和机制集成到基站中，将内容存储移到更靠近最终用户的地方。相较于传统的服务器-基站-用户系统, 使用边缘缓存的基站系统能够让网络资源更接近用户，使用更方便。实际上，不仅基站，笔记本电脑、移动设备、物联网设备和小型分支机构的文件服务器，都可以采用这种缓存机制。然而，部分研究表明，该机制会使得缓存中的文件相对传统模式新鲜度更低。

新鲜度表示消息的新鲜程度，刚刚产生的信息新鲜程度高，产生很久的信息新鲜程度低。研究中为了量化消息的新鲜度，采用信息年龄的概念，定义为接收的信息从生成到被接收经过的时间，如果这段时间很长，说明信息从生成到被接受经过了很久，信息的新鲜度很低，通常需要更新或者从缓存中删除。

如何改进现有的边缘缓存技术，已经有十数篇文献提出了相关方案。最初的方案在考虑策略时只关注了内容的流行度。随着研究的深入，缓存更新策略开始考虑到内容的新鲜度。目前的研究方向是将内容的新鲜度和流行度结合起来考虑。但有的研究忽略了从服务器下载内容的成本，有的研究忽略了缓存的快速更新能力，有的研究则假设有无限的缓存容量。可以看出，考虑实际应用的缓存优化要同时考虑很多因素,如新鲜度与流行度、新鲜度和流行度的时变性、基站下载内容的成本、基站的容量限制等，而之前的研究未考虑全面，实际性能不理想。

1.1.2 主要内容

项目主要内容为边缘缓存更新机制的研究，拟设计一种缓存更新机制，在保证

内容的新鲜度的前提下减小网络负载。该机制面向一个缓存容量受限的边缘缓存系统，通过控制边缘缓存的更新来减少负载。在实际的边缘缓存模型中，该机制的探究要考虑到信息的新鲜度以及流行度，以及有限的缓存容量。发现该机制探究中存在一个优化问题，在一系列约束下进行优化，得到最优的更新策略。

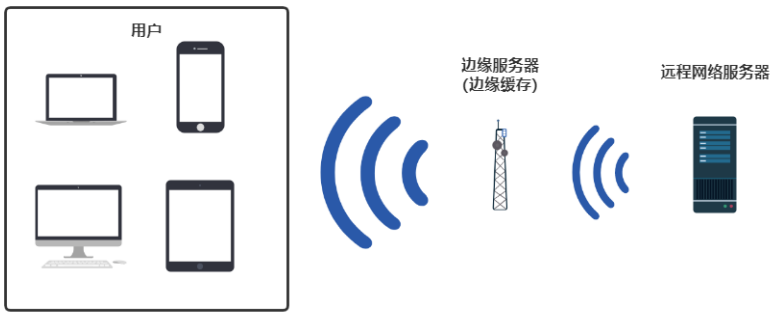

图 1-1-2 边缘缓存模型

最终得到的更新机制需要能在给定文件数，用户数等因素的条件下得到最优更新策略。项目的步骤如下：首先对实际模型进行建模，推导出需要优化的问题，并设计更新机制，将设计的缓存更新机制在数学软件中进行仿真并与已知的算法进行性能比较。

相较于以往研究，本课题分析了一个容量有限的边缘缓存系统的更新策略，推导了该更新策略的数学规划模型，综合考虑了内容新鲜度和流行度以及有限的边缘缓存容量，得到的结果更具有实际意义。

1.2 进度介绍

本课题目前完成的主要工作如下：

1. 问题建模。
2. 问题分析。
3. 列生成算法设计。

2　已完成的研究工作及结果

2.1　问题建模

2.1.1 网络模型

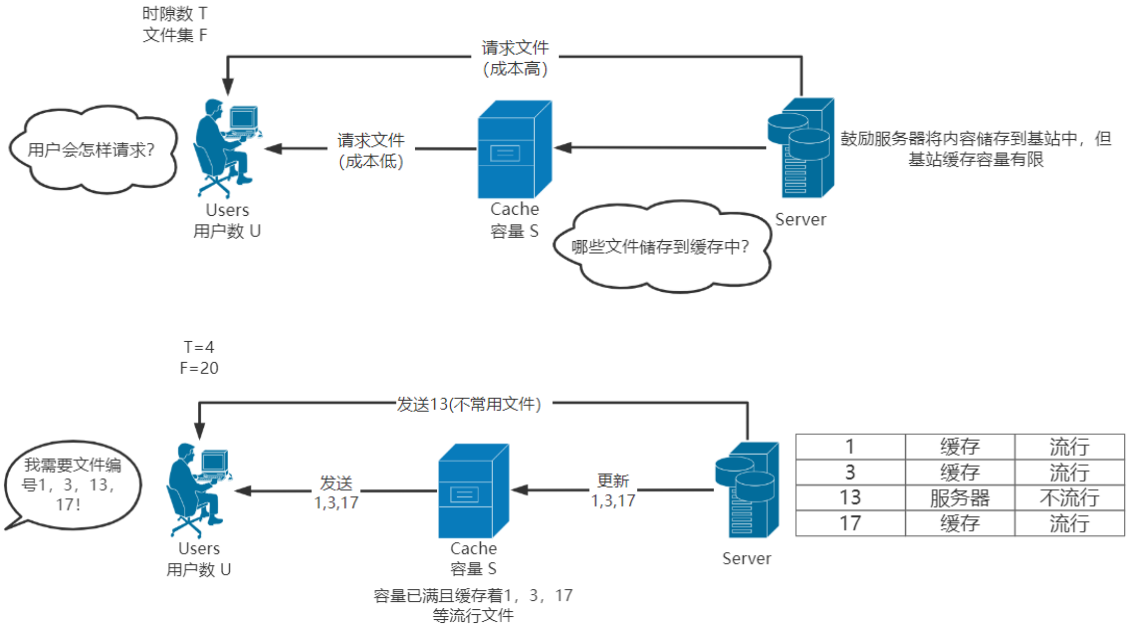

图 2-1-1 网络模型

如图是一个简化的实际服务器-边缘缓存系统。

该系统由三部分组成：一组用户集，一个边缘缓存节点与一个服务器。用U表示数量为$|U|$用户的集合，在每个时隙会向边缘缓存节点发出对内容的请求。先前的研究表明，用户的请求可以用数学模型表示。

边缘缓存节点(下简称缓存)可以低成本的方式将文件发送给用户，但容量有限为S，因此不可以将所有内容都储存在缓存中。

内容储存服务器储存着所有文件，也具有将文件发送给用户的能力，但成本较高。因此，流行的文件应优先通过缓存发送。

文件集记为F，且每个文件都有其自己的编号。将所有时隙记为T，在分析时，只考虑单一时隙。

在每个时隙，收到用户的请求后，缓存节点进行如下工作：

1. 先从缓存查询文件。
2. 如果在缓存中没有查询到，则从服务器查询。

3. 如果缓存中没有，则从服务器将文件发给用户，并且将该文件写入缓存；
   如果缓存中有，则直接将文件发给用户，并且将文件更新。

采用信息年龄量化信息的新鲜度，定义为缓存的中的一个文件从下载后到下一次下载(更新)之间经过的时间。设在时隙t根据用户请求，缓存中刚刚下载了内容f，如果该在t'时隙对缓存中的该文件进行更新，信息年龄 AoI 的表达式为：

$$AoI = t' - t$$

信息的流行度服从齐夫分布，齐夫分布是由哈佛大学的语言学家 George Kingsley Zipf 于 1949 年发表的实验定律。在本课题中对于任意一个请求，请求第f个内容的概率是：

$$p(f) = \frac{f^{-\gamma}}{\sum_{i \in F} i^{-\gamma}}, \gamma \text{为常数}$$

通过上式，可以得到每个请求与文件的概率关系。

2.1.2 目标问题建模

每个用户的所有请求都在每个时隙t初发生，且这些请求是随机变化的，为了进行建模，需要了解用户每个请求的内容与在哪个时隙发出请求。用$h(u,r)$和$o(u,r)$两个函数分别表示用户u的第r个请求与内容和时间的关系。结合两个函数，可以计算出每个文件在哪几个时隙被请求，进而计算其AoI。用m_{tf}表示在t时隙请求内容f的用户数。

缓存还受到容量的限制，由于实际文件大小不等，讨论缓存容量与文件大小的相对大小更有意义。用l_f表示每个文件的大小(单元)，设缓存大小为S，其值为所有文件的总大小的ρ倍：

$$S = \rho \sum l_f, \rho \text{为常数。}$$

使用C_s表示从服务器直接向用户提供文件的单元成本，使用C_b表示从缓存中向用户提供文件的单元成本。

现在考察每个时隙该系统的所有行为，涉及网络传输负载的部分有：服务器向用户发送文件，缓存向用户发送文件与服务器将内容在边缘缓存中更新，共三种行为。其中前两种都是为了满足用户的文件下载请求。如果满足所有用户的所有文件请求的行为带来的网络负载定义为一种成本，记为$C_{downlaod}$，则可以写出它的表达式:

$$C_{downlaod} = \sum_{u=1}^{U} \sum_{r=1}^{R_u} l_{h(u,r)} [C_b x_{o(u,r)h(u,r)} + C_s (1 - x_{o(u,r)h(u,r)})],$$

x_{tf} 是二进制变量，表示内容f是否在t时刻存储在缓存中。当$x = 1$时，括号内右边一项为0，表示内容在缓存中，当$x = 0$时，括号内左边一项为0，表示内容不在缓存中。

用户的下载请求还伴随着缓存中的内容更新，记更新成本为C_{update}。由于研究了内容的新鲜度，还需要引入一个消息过时所造成的成本，在优化时应使得该成本尽可能小。将内容过时的成本记为AoI成本，C_{AoI}。为了细化每个时刻的决策，引入另一个二进制变量a_{tfi}，表示内容f是否在t时刻存储在缓存中且具有i的AoI。

$$C_{update}=\sum_{t=1}^{T}\sum_{f=1}^{F} l_f(C_s-C_b)\, a_{tf0}$$

$$C_{AoI}=\sum_{f=1}^{F}\sum_{t=1}^{T}\sum_{i=1}^{t=1} p_f(i) m_{tf} a_{tfi}$$

a_{tf0}表示内容刚刚从服务器下载。内容储存成本$p_f(i)$定义为内容f为i的AoI与其大小l_f的乘积。

分析引入的两个二进制变量，x_{tf} 表示内容f是否在t时刻存储在缓存中，a_{tfi}表示内容f是否在t时刻存储在缓存中且具有i的AoI。对一个内容f，其(x_{tf}, a_{tfi})对可以表示所有时隙中它在缓存中的情况。所有(x_{tf}, a_{tfi})对即是整个系统的更新策略。在保证内容新鲜度的前提下降低网络负载，优化目标应为使总成本最小，该缓存优化问题表示为：

$$\min_{\boldsymbol{x},\boldsymbol{a}}\ C_{downlaod}+C_{update}+\lambda C_{AoI}.$$

λ为权重系数，$\lambda \in [0,1]$，λ越小，系统越不更新。$\lambda=0$时，系统不更新缓存。

将上式展开并添加约束，有：

$$\min_{\boldsymbol{x},\boldsymbol{a}}\ \sum_{u=1}^{U}\sum_{r=1}^{R_u} l_{h(u,r)}[\,C_b x_{o(u,r)h(u,r)}+C_s(1-x_{o(u,r)h(u,r)})]+$$
$$\sum_{t=1}^{T}\sum_{f=1}^{F} l_f(C_s-C_b)\, a_{tf0}+\lambda\sum_{f=1}^{F}\sum_{t=1}^{T}\sum_{i=1}^{t=1} p_f(i) m_{tf} a_{tfi}$$

约束为:
$$\begin{cases} \sum_{f=1}^{F} x_{tf}\ l_f \le S, t\in T \\ \sum_{i=0}^{i=t-1} a_{tfi}=x_{tf}, t\in T, f\in F \\ a_{tfi}\ \ge\ x_{tf}+a_{(t-1)f(i-1)}-a_{tf0}-1, t\in T, t\neq 1 \\ a_{tfi}\ \le\ a_{(t-1)f(i-1)}, t\in T, t\neq 1 \end{cases}$$

分别表示缓存容量约束，AoI约束。后两个式子的意义为:时隙t中的内容f具有 $AoI=i$的条件：当且仅当内容f的AoI在从时隙2到时隙t中不为0，并且在时隙 $t-1$ 中具有$AoI=i-1$。

2.2 问题分析

对于一个内容f，考察它从0到时隙T里所有的决策对(x_f, a_f)

$$x_f=[x_{1f}, x_{2f}, x_{3f}, x_{4f}, \dots\dots, x_{Tf}]'$$

$$a_f=[a_{1f0}, a_{2f0}, \dots\dots, a_{Tf(t-1)}]'$$

在一个时隙，根据两个二进制变量的意义，(x_f, a_f)可能有三种情况：(0，0)，(1，0)和(1，1)，T个时隙内总共有3^T个这样的二进制变量对。对某一个时隙t以及

i 的AoI，(0，0)表示该内容不在缓存中，(1，0)表示该内容在缓存中，但不具有为i的AoI，(1，1)表示都两个条件具备。总体上看，这3^T个二进制变量对包括了t从1到T，i从0到$t-1$的所有情况，也就是该内容f的所有缓存可能。

为了简化问题，定义一个索引集$K=\{1,2,3,4,\dots\dots,3^T\}$,表示所有的$(x_f, a_f)$对。每一个$(x_f, a_f)$对为该索引集$K$中对应的一列，$k \in K$表示一种可能的解。

对于内容f，其3^T个缓存可能中只有一部分是实际发生的，用一个二进制变量矩阵w_{fk}来表示其选择情况，当且仅当内容f的决策选择了第k种可能时，该变量为1，其余情况为0。通过联合使用w_{fk}和K，仅使用两个变量就可以表示内容f的所有可能选择策略，做到了简化问题。

使用w_{fk}来将问题重构，设内容f在策略为k下的花费为C_{fk}:

$$C_{fk}=\sum_{t=1}^{T} l_f m_{tf}[C_b x_{tf}^k + C_s(1-x_{tf}^k)]$$

$$+\sum_{t=1}^{T} l_f (C_s - C_b)\, a_{tf0}^k + \lambda \sum_{t=1}^{T} \sum_{i=1}^{t=1} p_f(i) m_{tf} a_{tfi}^k$$

目标问题可以简化为：

$$\min_{w} \; C_{fk}\, w_{fk}$$

$$\text{约束为：} \begin{cases} \sum_{f\in F} \sum_{k\in K} l_f\, x_{tf}^k w_{fk} \le S, & t \in T \\ \sum_{k\in K} w_{fk} = 1, & f \in F \\ w_{fk} \in \{0,1\}, f \in F, k \in K \end{cases}$$

分别表示缓存容量约束与二元变量约束。

2.3 列生成算法设计

上文表明，边缘缓存的问题可以被公式化为一个线性规划(ILP)问题，并且是一个整数规划问题。考虑将该问题使用数学软件求解，如 MATLAB。提出求解问题并找到求解方法。将求解方法运用编程语言写出来，并在 MATLAB 中运行。通过调用数学求解器来降低代码难度。

通常情况下，求解整数规划问题的方法有很多，如分枝定界法、割平面法、穷举法等。但本问题中变量数很多，上述算法计算的时间较长，不适用于该问题，需要找到一种在变量很多的线性规划问题下也可以具有快速计算能力的算法。

另外，求解原问题还有两个难点。首先，在原问题中，目标w为$f \times K$维的矩阵，其中K=3^T，目前为了简化令其为一个较小值。实际情况下，如果考察一天为长度，一小时为一个时隙，那么$T=24$，K将会非常大。其次，C_{fk}的计算涉及内容f从0到时隙T里所有的决策对(x_f, a_f)，计算量很大，求解时间很长。

考虑一种算法，从小规模的问题出发，且尽量避免对决策对(x_f, a_f)的计算。因此，本项目采用基于单纯型法提出的列生成算法(Column Generation Algorithm，CGA)。

其思路如下：先把原问题 MP 简化到一个规模更小（即变量数比原问题少的）的简化问题 RMP 下，在 RMP 上求最优解，但是此时求得的最优解只是该情况下的，并不是原问题 MP 的最优解。此时，需要通过构造子问题(Subproblem, SP)去检验在那些未被考虑的变量中是否对原问题有改进的。如果有，那么就把这个变量加入到 RMP 问题的变量中，经过反复的迭代，直到找不到改进原问题 MP 的变量，此时得到原问题 MP 的最优解。

在求解这个线性规划问题中，CGA 算法的优点即得以体现：一方面，在$K' \in K$的子集中，K'可以设置的比较小，能快速得到初始解。另一方面，在初始化时如果对于所有的$f \in F$,将其所有(x_f, a_f)都设为(0,0)，则C_{fk}=$\sum_{t=1}^{T} l_f m_{tf}\, C_s$，在这种情况下 C_{fk}为一个容易求得的常数。由于 CGA 算法是迭代的，在迭代循环中，总能找到最优解，可以从简单的初始解开始，因而避免了上述难点。

2.3.1 参数设置

目前程序输入的参数如下：

名称	意义	值	备注
U	用户集	20	$u \in U$表示某一个用户
F	文件集	20	$f \in F$表示某一个内容
T	总时隙	4	$t \in \{1,2,\dots,T\}$表示第t时隙
C_s	从服务器下载一单元内容的花费	10	$C_s < C_b$
C_b	从缓存中下载一单元内容的花费	1	
l_f	每个文件的大小(单元)		[1,10]均匀分布
S	缓存大小		
$p_f(i)$	内容储存成本		
m_{tf}	t时隙请求内容f的用户数		

λ	更新权重系数	0.5	
ρ	缓存大小因子	0.5	缓存大小为文件总大小的ρ倍
γ	齐夫分布系数	0.54	典型值

表格 2.3.1 参数设置

在编写 MATLAB 程序时，先使用较小的用户数，文件数与时隙数便于简化计算。认为每个时隙，每个用户一定发出且只发出一个请求。则每个用户在每个时隙所请求的文件分布是可求的，进而可以得出每个文件的请求在所有时隙内的分布，即得到了$h(u,r)$和$o(u,r)$。在后续的改进中，会令模型更加实际，增加用户请求行为的分布的函数，如随机分布。

2.3.2 RMP 问题

根据一个比原问题小的简化的K'矩阵来构造 RMP 问题：

RMP 问题：

$$\min_{w} \ C_{fk}\, w_{fk}$$

$$\text{约束为：} \begin{cases} \sum_{f\in F}\sum_{k\in K'} l_f\, x_{tf}^{k} w_{fk} \le S, \quad t \in T \\ \sum_{k\in K} w_{fk} = 1, \ f \in F \\ 0 \le \ w_{fk} \le 1\,, f \in F, k \in K' \end{cases}$$

在该算法中没有整数约束，w_{fk}可以得到分数结果。式中K'表示K的子集。在初始情况下，C_{fk}为一个常数，计算非常简单。

2.3.3 SP 问题

由于 CGA 算法是迭代的，每次在 RMP 下求得解$\boldsymbol{w}^*$后，都需要检查$\boldsymbol{w}^*$是否是 RMP 的最优解。这可以通过为$f \in F$ 的每个内容找到具有最小降低成本(Reduced Cost)的$(x_f\,, a_f)$来确定。如果所有这些值都是非负的，则当前解决方案是最优的。否则，说明添加该变量对问题有优化作用，将具有负的降低成本的$(x_f\,, a_f)$添加到 RMP 问题中，因此需要构建一个 SP 问题。

SP 问题使用对偶变量,定义与$\sum_{f\in F}\sum_{k\in K'} l_f\, x_{tf}^{k} w_{fk} \le S$与$\sum_{k\in K} w_{fk} = 1$两式对应的最优对偶变量为$\pi = [\pi_1, \pi_2\,, \pi_3, \dots\dots, \pi_T\ ]'$ 与$\beta = [\beta_1, \beta_2, \dots\dots, \beta_F]'$。则对于内容$f$，决定是否有新变量加入 RMP 问题的降低成本为

$$C_f - \sum_{t=1}^{T} l_f \pi_t\, x_{tf} - \ \beta_f \text{。}$$

式中，C_f=$\sum_{t=1}^{T} l_f m_{tf}[C_b x_{tf} + C_s(1 - x_{tf})]$+$\sum_{t=1}^{T} l_f (C_s - C_b)\, a_{tf0}$ +

$\lambda \sum_{t=1}^{T} \sum_{i=1}^{t=1} p_f(i) m_{tf} a_{tfi}$

用(x^*_f, a^*_f)表示子问题SP_f的最优解。如果(x^*_f, a^*_f)的降低成本为负，就将其添加到 K'中。

SP 问题:

$$\min_{(x^*_f, a^*_f)} C_f - \sum_{t=1}^{T} l_f \pi_t x_{tf} - \beta_f$$

约束为：

$$\begin{cases} \sum_{i=0}^{t-1} a_{tfi} = x_{tf}, t \in T, f \in F \\ a_{tfi} \geq x_{tf} + a_{(t-1)f(i-1)} - a_{tf0} - 1, \ t \in T, t \neq 1 \\ a_{tfi} \leq a_{(t-1)f(i-1)}, \end{cases}$$

2.3.4 CGA 算法

Column Generation Algorithm

1: 输入变量：$U, F, T, C_{s,} C_b, \lambda, \rho, \gamma$,生成l_f, S,

计算$h(u,r)$和$o(u,r)$

2: 初始化：

对于所有的$f \in F$，令其$(x_f, a_f) = 0$　设置标志变量 stop←0

3: **while** stop=0 **do**

4: 求解 RMP 问题

5: stop←0

6: **for** f =1 to F do

7: 求解 SP 问题,得到(x,a)

8: 计算 reduced cost

9: **if** reduced cost <0

10: 向 RMP 中加入 (x,a)

11： stop←1

2.4 舍入算法设计

考虑到列生成算法只能得到最优解，但不能保证得到整数解，还需要设计一个舍入算法(Rounding Algorithm，RA)，将其与 CGA 算法一起使用来得到线性规划的最优整数解。

首先使用列生成算法来得到最优的缓存更新方案，对该方案进行检验，如果

得到的解并不是一个整数解，则采用 RA 算法进行舍入。设计 RA 算法的功能是在时隙上反复修复内容的缓存决策，直到构造出整数解。考虑到列生成与舍入算法，算法总思路如下：

1. 输入参数，包括时间，用户，文件以及相关系数等。每个文件的大小随机产生。
2. 对输入参数进行处理，计算缓存容量以及内容请求与用户与时间的分布。
3. 进行建模，产生一个用于储存决策的变量集。
4. 应用 CGA 算法求得最优解.
5. 如果该最优解不是整数解，则调用 RA 算法进行反复修改直到求出整数解。

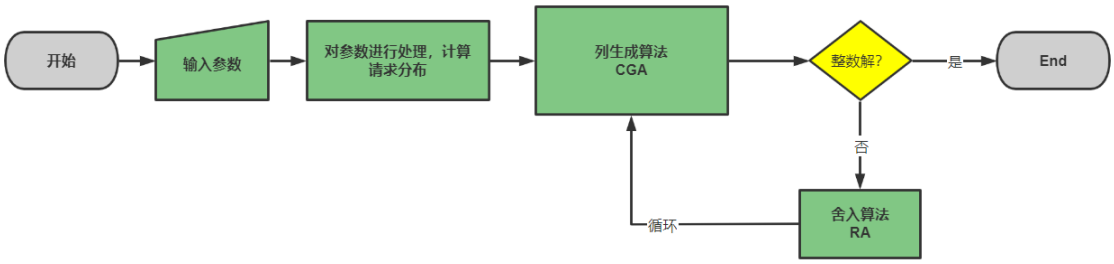

图 2-4 包含 RA 算法的程序设计

3 后期拟完成的研究工作及进度安排

3.1 后期拟完成的研究工作

后续需要完成的工作一是 CGA 算法的实现，目前已经完成了第一次计算的编写，还需要计算 SP 问题并且将两个问题写在一个循环中，添加迭代语句。二是进行 RA 算法设计，在成功运行后，与目前现有的算法进行性能对比。算法改进方面，主要是进行更真实的模拟，比如运用统计学上得到的用户的请求模型。最后，进行课题总结工作，撰写并完成毕业论文。

3.2 进度安排

具体进度安排如下：

2022 年 3 月至 4 月：仿真实现 CGA 算法。

2022 年 4 月中旬：仿真实现 RA 算法，与现有方法进行对比。

2022 年 4 月中旬至 5 月末：进行模型改进，整理实验数据并撰写论文。

4　存在的困难及解决方案

4.1　存在的困难

1.同时考虑新鲜度和流行度的边缘缓存问题资料较少。

2.编程能力有限。

3.相比通常的约束模型，该 CGA 算法相比常见的线性规划问题目标变量 w 是一个 F×K 维的矩阵，需要先对矩阵进行扁平化处理。

4.实际问题变量太多。

4.2　解决方案

针对目前存在的问题和困难，提出以下解决方案：

1. 主动去查阅有关资料并且及时总结不懂的部分及时请教老师和同组学长、同学。学姐在算法编写上给了我很大的启发，帮我进行了建模。

2. 进行简化，某些变量先设置成小值，简化部分模型。先完成核心内容，在后续改进中解决这些问题。

3. 联系上了在该领域发表过有关论文的作者，通过邮件向他请教问题。

5 论文按时完成的可能性

由于在编程上将复杂的线性规划转换成代码遇到了困难，目前课题进度稍慢，略微慢于按照开题报告定下的进度安排。目前的主要目标是下尽快做出 CGA 与 RA，之后与现有算法比较加改进模型。在了解我的困难后，老师和学长学姐给我了很多帮助，给了我许多启发，让项目能得以推进，攻克难点。相信在他们的帮助下，我可以按质按量按时完成毕业设计（论文）。