2019 年中国海洋大学数学建模竞赛

参赛队员信息

参赛题号（从 A/B/C/D 中选择一项填写）： **B**

队名	姓名	学号	年级专业	电子邮箱	联系电话
Model333	黄佳城	16060012014	16 级经济学	305179568@qq.com	17864295856
	林子皓	17010006029	17 级海洋科学	591526170@qq.com	13345027222
	赵国庆	18120014029	18 级经济学	zgq316056529@126.com	17861428256

新一代通信网络设计与规划

摘要

信息产业曾被誉为“工业化的牵引”，近期通信爆发式的发展也被认为是第四次工业革命的一个潜在突破点。“建立一个万物互联的世界”——是指将每家每户都通过通信技术联接起来，共享通信技术发展给我们带来的革新。通信网络可以大体分为四个层次：接入网、汇聚网、核心网和骨干网。骨干网，作为连接多个地区和区域的高速网络，承担了各城市对外网络交互的出入口功能，是信息网络的主动脉。新通信技术的部署离不开新一代骨干网的配合，只有拥有更大信息容量、更广覆盖度的骨干传送网，才能确保用户对新技术的完美体验. 如何在复杂的设计因素约束下，通过科学的规划和设计，让网络的价值最大化，部署成本最低，是一个极有探讨价值的问题。

在本文中，我们从数据出发，在对数据的探索性分析中，我们得到了一些对于附件中天线阵的信息：阵元均为定向天线，具有空间结构和各自的物理倾角等等。随后我们便对数据中存在的信息，由 MaxWell 方程出发，建立了相控阵阵元的信息，旨在获得相控阵中每个阵元的基础信息：包括天线的基座位置、物理倾角以及各个阵元的方向图参数。并将其转化一个规划问题，利用遗传算法解决并求得 32 个阵元的基础信息，为后面微波问题的求解打下坚实的基础。

对于微波问题 1 和微波问题 2 两个多目标规划问题，我们首先从 32 个阵元的基础信息入手，计算得到微波问题 1 和微波问题 2 中所涉及的物理量：如辐射强度、旁瓣电平等等，并对“凹坑”的衡量做出定义，引入 0-1 变量决定阵元的开关。对于多目标问题的求解，我们采取线性加权法生成新的目标函数，并利用智能算法求解出各个阵元的相位配置。

对于骨干网 1 和骨干网 2 问题，我们首先根据题目要求利用带有长短时记忆结构的循环神经网络对时间序列数据——网络接入量以及网民上网时间进行预测，以备后面的问题使用。而对于部署价值函数，我们先将其定义为网络通信的总成本，建立针对于广东深的骨干网络数据流模型，并利用网络流问题的 Ford-Fulkerson 方法计算瞬时网络接入量，将总体网络的瞬时接入量和区域网络的瞬时作为限制条件，构建了成本最小的规划魔球以求得部署价值函数。而后在骨干网 2 问题中，我们重新考虑了每个地区的人口结构，引入在人口资源环境经济学的概念改造了梅特卡夫定理，对于网络的价值进行一个重新的定义。

关键词：Mazwell 方程、方向图、多目标规划、遗传算法、LSTM-RNN、梅特卡夫定理

目录

一、问题重述

相控阵天线具有大空域内的波束扫描、可控波束方向、较大增益等优势，成为现在微波发射机研究的焦点。部署相控天线于微波接入中，能精准地控制基站发出微波的方向，建立接收机与发射机的最佳路径，确保最优通信质量。天线单元可以是单个的波导喇叭天线、偶极子天线、贴片天线等。在每个天线单元后端都设置有移相器，用来改变单元之间信号的相位关系，信号的幅度变化则通过功率分配或者衰减器来实现。

待研究的问题

现有一个 32 单元相控天线阵列，移相器配置有 4 种方式（0°，90°，180°，270°）。采用仪器测量每种相位配置下的 3D 空间分布（数据参见附件一）。使用数据时需要考虑数据中存在的测量误差：移相器配置为 90° 和 180°，两者信号相差（90+/- err）°，其中 err 为测量误差。

微波问题 1

假设：1. 合成功率为 35dBm；2. 旁瓣电平尽可能低；3. 可以选择关闭某些通道；4.在位置 AZ=10°，俯仰 EL=10° 处有一站点，波束设计时避免对该站点的干扰。请给出方位（水平角 AZ=10°，俯仰角 EL=5°）对应的波束配置（移相器配置矢量〖z=[Δ〗_1,…,Δ_N]）。

微波问题 2

波束配置矢量要求：1. 整个覆盖区域内，平均辐射功率 35dBm；2. 波束覆盖的“凹坑”尽可能小；3. 覆盖波束个数尽可能少；4. 用尽可能少的波束，覆盖水平角±30°，俯仰角±15° 的区域。请给出波束配置列表[z_1,z_2,…,z_(M])（M 为波束个数）。

骨干网问题 1

随着众多网络接入技术的发展，以及未来用户数据需求量呈指数形式的增长，城市之间的骨干网输入输出端口带宽与容量需求也将爆炸式增长。使用最新的通信技术升级改造骨干网已成为运营商应对未来通信市场需求的一个迫在眉睫的举动。

现需要在广东省对已有骨干网进行满足电信市场要求的改造。个人信息接入需求量会随着新科技在群众中的接受度，科技自身更迭速度等因素变化；同时各个年龄段，各收入阶层对信息需求量也有不同。请预估未来十年（截止 2028 年）信息时代各类人群所需要的信息量，以满足广东省全部人口的信息接入需求为目标，省会广州市为数据中心，选择性地用最新的通信技术更新从广州到其余主要行政城市的骨干网络。（城市之间连接的城际网可选用的调制格式及其特性见附件二中）。请参考广东省一些相关数据，自我构造约束条件，为广东省未来电信市场建立一个有效的部署价值函数，用这个价值函数寻找到网络价值最大化的部署方案。作为电信从业人员网络部署设计方案的参考。

骨干网问题 2

某通信公司曾提出：“建立一个万物互联的世界”。而在现实社会中，站在运营商角度，每个人的接入价值是不一样的，不同收入、地区、年龄的人接入成本和数据需求都不一样。如果将不同人群的网络接入价值也纳入考虑，并选择性的接入网络价值大的人群，请重新构建网络价值函数，并找到新的最优部署方案。

二、问题分析

对于一个相控阵天线，最重要的无非是以下几点：阵元的位置、激励电流的幅度、相位、以及每个阵元的方向图参数，因而在解决任何问题之前，我们都需要建立起一个数据模型，从附件中获取每个阵元的基础信息。而在建立数据模型前则需要对附件的数据测量方式进行探究，在摸清楚左右情况之后，我们才能动手解决微波问题。

而微波问题 1 和 2 都是多目标规划问题，在解题前则需要理解多阵元的相控阵列方向图是怎样合成的才能继续写出条件。

骨干网问题 1、2 则更像是开放性问题，需要多借鉴经济学等领域关于网络价值、人口模型的研究才能获得更好的解决。

三、模型假设

3.1 未来十年广东省流量接入需要的模型假设

以广东省 2016 年 8 月到 2019 年 2 互联网总流量为基础，假设 10 年间科技技术进步的增长速度和现在的增长速度相似，各类年龄段、收入阶层的比例与现在相似，各类人群对流量的需求量的增长率与现在相似且没有颠覆性的巨大变化，预测 2028 年 1 月份广东省互联网总流量。

3.2 网络价值最大化的部署方案模型

假设网络价值的最大化方案就是在广东省各城市在传输距离和传输流量的限制下，总的接入成本最低。以 2011 年到 2018 年国民平均上网时间为基础，假设各年增长率稳步增长且没有颠覆性的巨大变化，预测 2028 年国民平均上网时间。把 2028 年国民平均上网时间当作 2028 年广东省人均上网时间。以 2018 年广东省各城市的人口占总人口的比率为基础，假设 2028 年时各城市人口占总人口的比重基本不变，用预测出的 2028 年广东省的总流量乘以各城市的人口比例，以此来预测 2028 年广东省各城市的流量接入需求量。以广东省各城市一个月的流量接入量和广东省人均上网时间为基础，计算出各个城市每秒钟流量的需求量。假设三种传输方案的传输线路充足而且可以在一条线路上同时使用多条传输线进行叠加，以此来计算出成本最低的广东省网络部署方案。

3.3 重构网络最大价值的最优方案模型

假定 25-50 岁的人群是利用网络创造价值的真正人群，且网络的价值与用户数的平方成正比，网络用户越多，价值函数就越大。

四、符号与约定

序号	参数	意义
1	J_t	外加电流
2	$\vec{A}$	磁位
3	G_0	天线的最大增益（$\mathrm{d}Bi$）
4	$HPBW$	主瓣的半功率角（°）
5	$HPBW_{back}$	主后瓣的半功率角（°）
6	FS	主瓣的前侧比（$\mathrm{d}B$）
7	FB_{back}	主后瓣的前侧比（$\mathrm{d}B$）
8	n_{side}	侧瓣（旁瓣）的数量（不计算镜像）
9	n_{back}	后侧瓣（后旁瓣）的数量（不计算镜像）
10	ANG	需要计算的角度（三维极坐标系中的方向角或者仰角）
11	$TILT$	场强方向图的偏移度数

五、模型的建立与求解

5.1 天线基础

天线[1]是一种用来发射或接收无线电波的设备，广泛而言为电磁波的电子元件。通俗点来，天线就是一个“转换器”——把输入线上的导行波，变换成在自由空间中传播的电磁波，或者还是进行相反的操作。而输入线中的导行波则是由到导线中的电流产生的，电生磁，磁生电，电磁波就于空间中传播，如图 5.1-1.

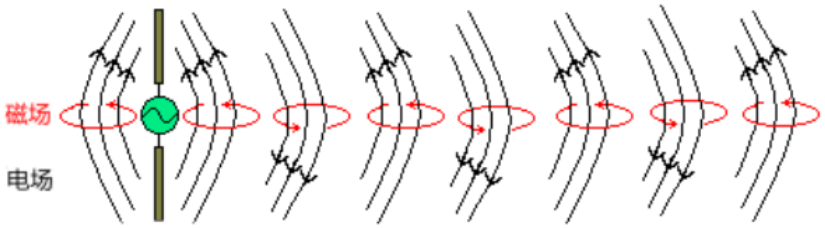


图 5.1-1 电磁波的空间传输

电磁波的产生和传播满足 MaxWell 方程：

$$\begin{cases} \nabla \times H = J_t + \dfrac{\partial D}{\partial t} \\ \nabla \times E = -\dfrac{\partial B}{\partial t} \\ \nabla \times B = 0 \\ \nabla \times D = \rho_t \end{cases}$$

而天线问题则是在电流方程的追加上：

$$\nabla \times J_t = -\frac{\partial \rho_t}{\partial t}$$

通过已知的外加电流 J_t 求解 E 和 H。

以天线基本的模型电流元（基本电振子）为例，我们给出求解 MaxWell 方程组的过程与一些天线基本参数的推导。基本电振子是一段理想的高频电流直导线，长度 $l << \lambda$ ，半径 $a << l$，沿线电流均匀分布，。

假设电流源位于坐标原点，沿着 Z 轴放置，长度为 l，如图 5.1-2 所示。其上电流为等幅同相分布，即 $\vec{I} = I_0 \vec{a_z}$ ，这里 I_0 是常数。

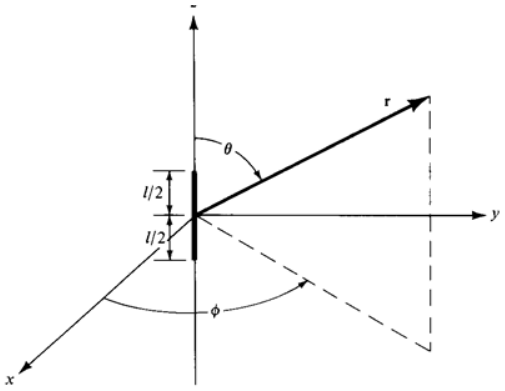


图 5.1-2 基本电振子示意图

为求其空间的场分布，首先求出其矢量磁位 $\vec{A}$，再由 $\vec{A}$ 求出电场 E 和磁场 H。根据电磁场理论，电流分布 $\vec{I}(x',y',z') = I_0 \vec{a_z}$ 的电流源，其矢量磁位 $\vec{A}$ 可以表示为：

$$\vec{A}(x,y,z) = \frac{\mu}{4\pi} \int_l \vec{I}_e(x',y',z') \frac{e^{-jkr}}{r} dl'$$

其中，(x,y,z) 为观察点坐标，(x',y',z') 为源点坐标，r 为源点到观察点的距离。

由于基本电振子的长度l远小于波长λ和距离r，因矢量磁位$\vec{A}$可以表示成：

$$\vec{A}(x,y,z)=\vec{a}_z\frac{\mu I_0}{4\pi}e^{-jkr}\int_{-\frac{l}{2}}^{\frac{l}{2}}dz'=\vec{a}_z\frac{\mu I_0 l}{4\pi}e^{-jkr}$$

引用直角坐标与球坐标的变换关系，将其变化为球坐标形式：

$$A_r=A_z\cos\theta=\vec{a}_z\frac{\mu I_0 l}{4\pi}e^{-jkr}\cos\theta$$
$$A_\theta=-A_z\sin\theta=\vec{a}_z\frac{\mu I_0 l}{4\pi}e^{-jkr}\sin\theta$$
$$A_\varphi=0$$

依据$\vec{H}=\frac{1}{\mu_0}\nabla\times\vec{A}=\vec{a}_\varphi\frac{1}{\mu_0 r}[\frac{\partial}{\partial r}(rA_\theta-\frac{\partial A_r}{\partial\theta})]$，可以得到磁场表达式：

$$H_r=0$$
$$H_\theta=0$$
$$H_\varphi=\frac{I_0 l\sin\theta}{4\pi}[j\frac{k}{r}+\frac{1}{r^2}]e^{-jkr}$$

同理有$\vec{E}=\frac{1}{j\omega\varepsilon}\nabla\times\vec{H}$，可以得到电场表达式：

$$E_r=\frac{I_0 l\cos\theta}{2\pi\omega\varepsilon_0}[\frac{k}{r^2}+\frac{1}{jr^3}]e^{-jkr}$$
$$E_\theta=\frac{I_0 l\sin\theta}{4\pi\omega\varepsilon_0}[j\frac{k^2}{r}+\frac{1}{r^2}-j\frac{1}{r^3}]e^{-jkr}$$
$$E_\varphi=0$$

由此求解 MaxWell 方程的工作就结束了。接下来我们将从电场E和磁场H出发推导天线的两个基本参数：方向性函数与方向图。

由于任何天线辐射的电磁波都不是均匀平面波，其辐射场都具有方向性。所谓的方向性函数，就是在相同距离的条件下天线的辐射场的相对值与空间方向(θ,φ)的关系，一般用$f(\theta,\varphi)$来表示。

以基本电振子为例，其辐射电场强度可以表示成：

$$\left|E(r,\theta,\varphi)\right|=\frac{60\pi Il}{\lambda r}\sin\theta=\frac{60I}{r}f(\theta,\varphi)$$

方向性函数定义为：

$$f(\theta,\varphi)=\frac{|E(r,\theta,\varphi)|}{\frac{60I}{r}}=\frac{\pi l}{\lambda}|\sin\theta|$$

为便于比较，通常采用归一化方向性函数$F(\theta,\varphi)$来表示，即：

$$F(\theta,\varphi)=\frac{f(\theta,\varphi)}{f_{\max}(\theta,\varphi)}=\frac{|E(r,\theta,\varphi)|}{|E_{\max}|}$$

基本电振子的归一化方向性函数为：

$$F(\theta,\varphi)=|\sin\theta|$$

而将方向性函数以曲线方式描绘出来，称之为方向图。它是描述天线辐射场在空间相对分布随方向 变化的图形。通常指归一化方向图。一般有场强方向图、功率方向图、相位方向图等等。如图 5.1-3 为基本电振子的方向图。

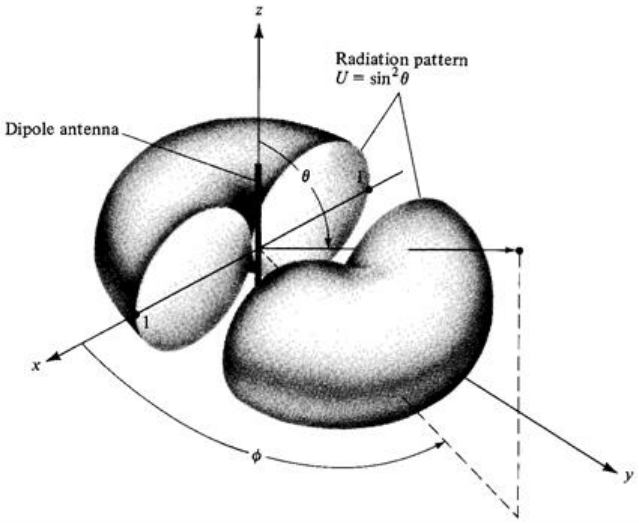


图 5.1-3 基本电振子的方向图

由于三维方向图比较复杂，不利于工程使用，于是一般使用E-方向图和H-方向图来表示整体方向图。E-方向图是指包含最大辐射方向的电场矢量所在的平面。用E面去截取立体方向图，则得到E-方向图。H-方向图则是指包含最大辐射方向的磁场矢量所在的平面。用H面去截取立体方向图，则得到H-方向图。基本电振子的E-方向图和H-方向图如图 5.1-3 所示。

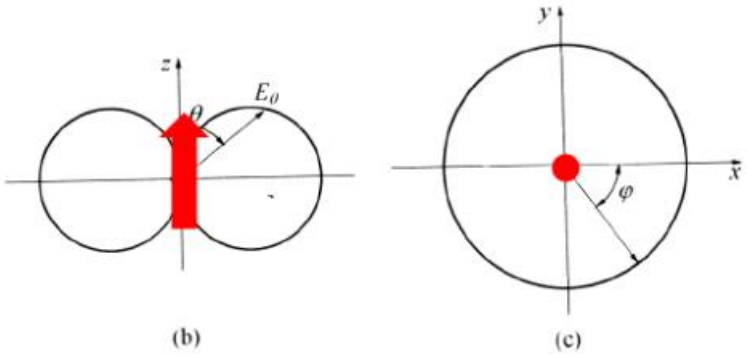


图 5.1-3 基本电振子的 E -方向图和 H -方向图

由 MaxWell 方程组求解出的电场 E 和磁场 H 我们还可以得出坡印廷量与辐射功率密度方向图的计算方式。

其中坡印廷量的定义为：

$$S(\theta,\varphi)=E(\theta,\varphi)\times H^{*}(\theta,\varphi)=\frac{1}{2\eta}\left|E(\theta,\varphi)\right|^{2}$$

归一化的辐射功率密度方向图定义为：

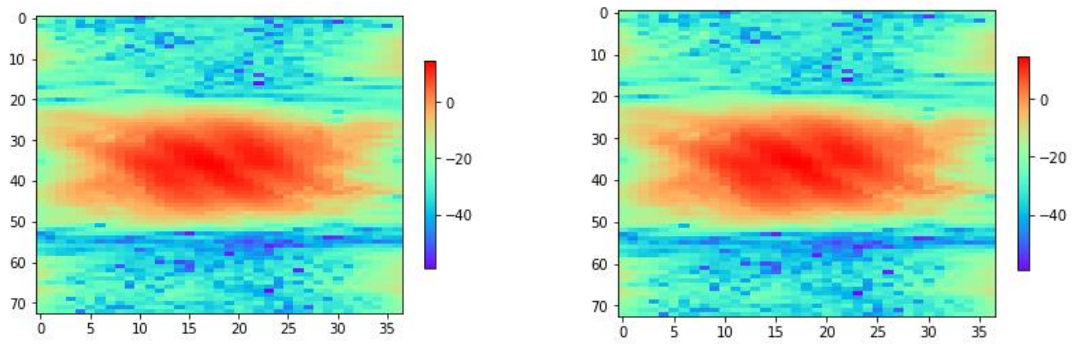
$$P(\theta,\varphi)=\frac{S(\theta,\varphi)}{S_{\max}(\theta,\varphi)}=\frac{\left|E(\theta,\varphi)\right|^{2}}{\left|E_{\max}(\theta,\varphi)\right|}=F(\theta,\varphi)^{2}$$

由此可知，方向图，无论是场强方向图还是功率方向图，与空间距离是无关的.

5.2 探索性数据分析

附件中有的数据主要是每一座天线的辐射功率与接收到的信号相位，在不同的移项器配置下。我们对附件的数据进行了探索性数据分析。

首先我们绘制的是不同移相器相位下同一天线的信号强度（以天线为 1 号天线），如图 5.2-1 所示。



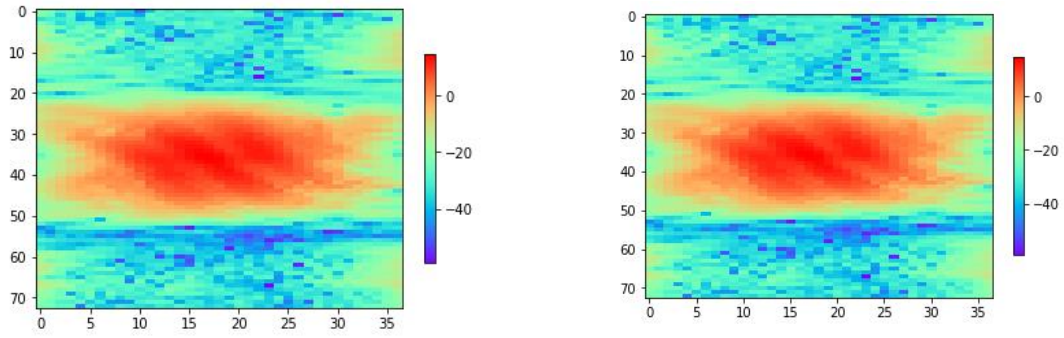


图 5.2-1 不同移相器相位下的信号强度（1 号天线）

我们能从图像中得到的是，天线的信号强度集中于某一个空间区域内，我们可以推断出该相控阵的中的阵元中均为定向天线，并且阵元有各自的物理朝向：区别于通过相位移动造成的波束移动（扫描现象）。

接下来我们绘制了不同天线在相同相位下的空间信号强度图，这里在图 5.2-2 中放出 1 号天线（左）和 2 号天线（右）的图。

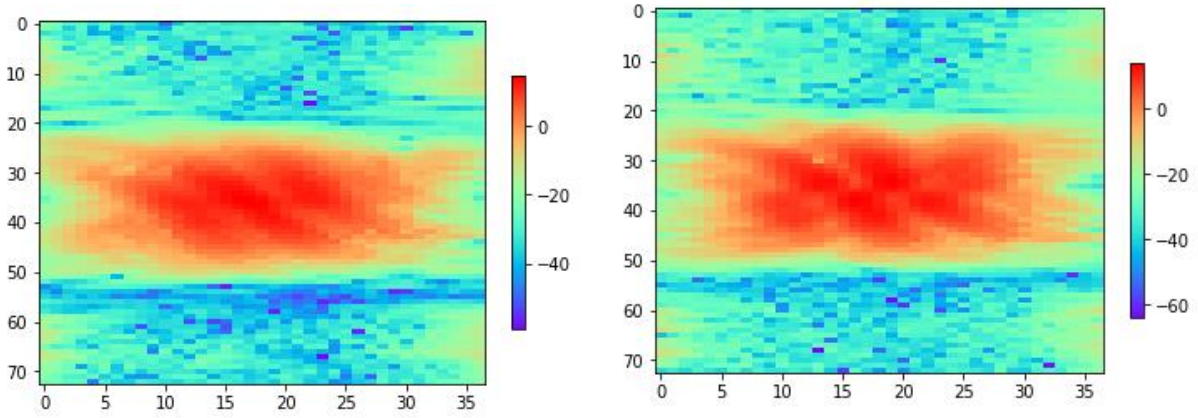


图 5.2-2 不同天线在相同相位下的空间信号强度图

这里为了编排我们不会放出所有天线的图，但是对比所有图形之后我们可以得出一个简单的猜想：所有阵元的物理朝向都是一致的，这点我们会在后面的模型计算中进行验证。

在探究完空间信号强度之后，我们也对空间相位进行了探究，我们现实绘制了不同移相器相位下同一天线的空间信号相位，与图 5.2-1 类似，得到的也是相同的结论，在这里不进行图片的累赘展示。接下来我们绘制了不同天线在相同相位下的空间信号，这里在图 5.2-3 中放出 1 号天线（左）和 4 号天线（右）的图。

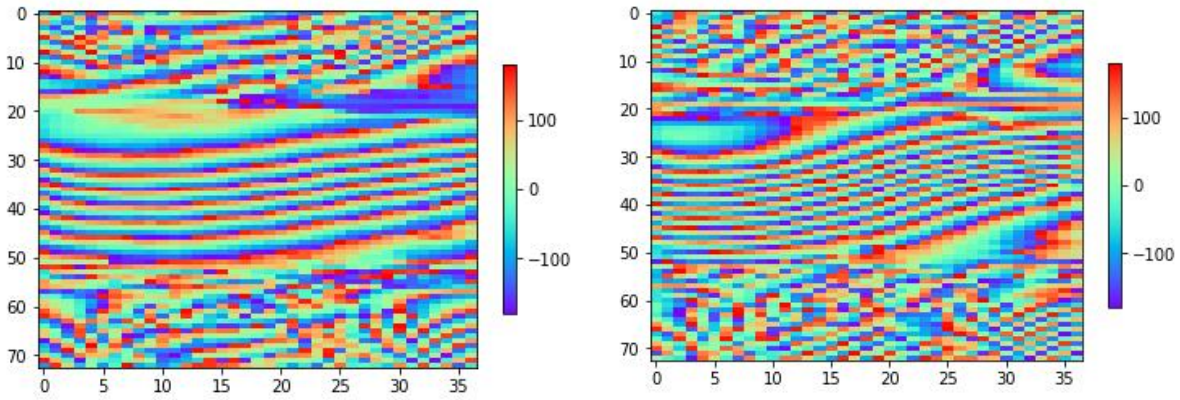


图 5.2-3 不同天线在相同相位下的空间信号相位图

不同天线在相同相位下的空间信号相位图大家都是不一样的，这暗示了一个问题：阵列中的天线阵元具有自己独立的位置，这也是后期我们需要确定的。

最后我们关注的是附件数据中的测量方式，其实主要有两种：一种是离该天线固定距离进行空间图测量，相当于是收集球体数据；另一种是离地图上某个点固定距离测量所有天线的空间数据。两种测试方式获得的数据是不一样的。

于是我们探究了同一个天线在相同水平水平角，不同俯仰角的空间信号相位。以 1 号天线为例，空间相位随不同俯仰角的变化图如图 5.2-3 所示。

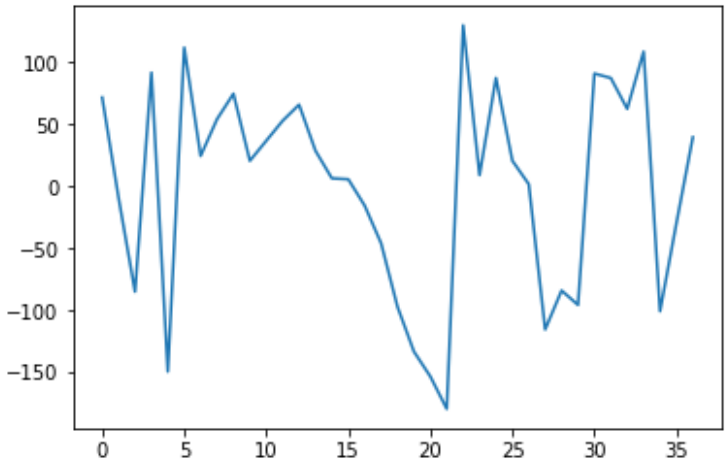


图 5.2-3 空间相位随不同俯仰角的变化（1 号天线）

我们会发现大部分时候这个相位角度是正负叫交替的，如果第一种测量方式，数据应该趋于平稳，仅受一部分物体的阻挡，不会出现大规模正负交替的情况。第二种方式，由于距离固定的某个点测量，而不是距离固定天线固定距离，那我们得到的就不是球体数据，而是这个数据和天线的分布是有关系的。图 5.2-4 所示的 1 号天线的空间相位随不同水平角的变化图同样佐证了我们的观点。

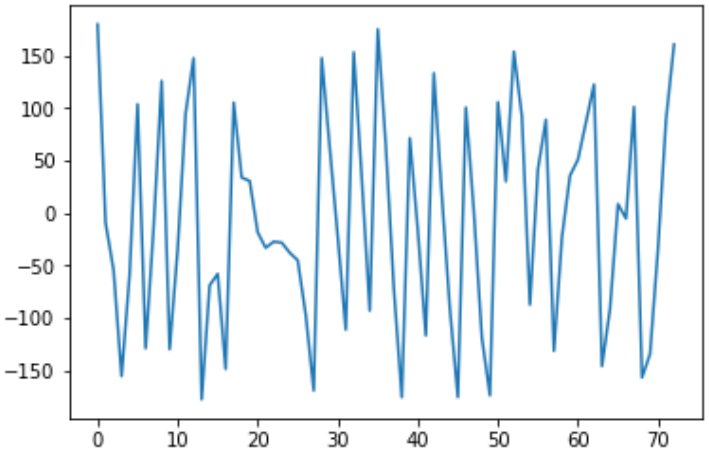


图 5.2-4 空间相位随不同俯仰角的变化（1 号天线）

对此我们对数据的探索性分析我们主要获得了一些关于线控阵的基本信息，接下来我们就要从数据中获得 32 个阵元的基础数据。

5.3 相控阵阵元的场强方向图与其数值算法 *NmAntPat* 模型

我们在 5.2 探索性数据分析中已经获得相控阵的 32 个阵元均为定向天线，定向天线的方向图比较复杂，一般如图 5.3-1 所示。

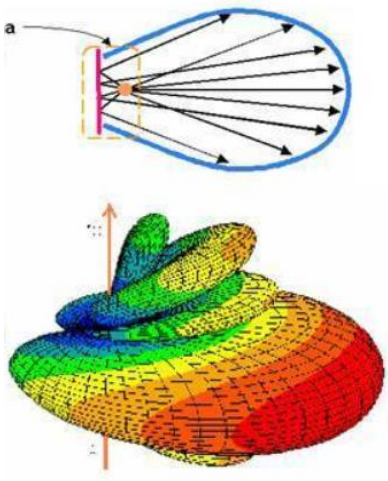


图 5.3-1 定向天线的方向图

定向天线的场强方向图分有多个波束：主瓣、旁瓣等等。一般其方向函数是不能被推导而出，只能通过某些描述方向的参数：半功率角等，结合数值算法求解。我们这里使用 *NmAntPat* 算法来实现我们题中阵元的场强方向图的数值求解。

对于一个广义的定向天线的场强方向图如图 5.3-2 所示。

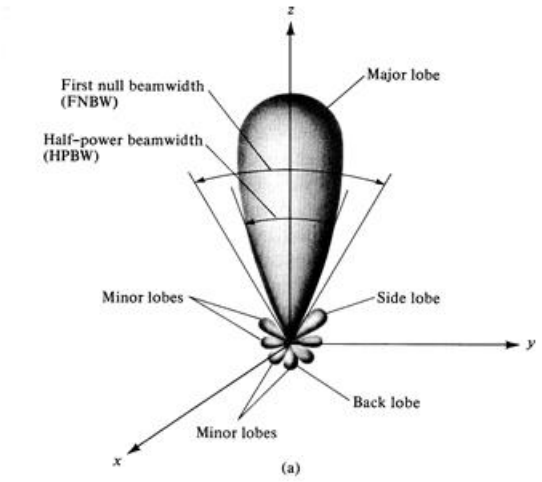


图 5.3-2 广义场强定向天线的方向图

由于该场强方向图为一旋转体，所以我们使用 E-场强方向图来表示该三维方向图，并认为该三维方向图为 E-方向图旋转生成。广义定向天线的 E-场强方向图如图 5.3-3 所示。

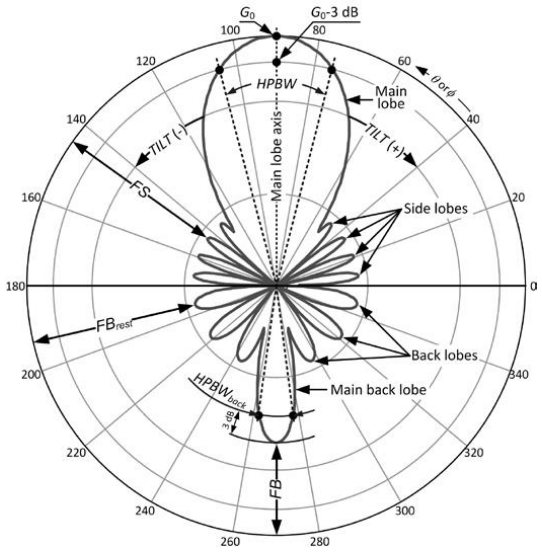


图 5.3-2 广义定向天线的 E-场强方向图

NmAntPat 算法假设该场强方向图可由有 NMAntPat 函数表示：

$$NmAntPat = f(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, ANG, TILT)$$

其中有关的十个参数是与天线和方向图相关的属性：

G_0：天线的最大增益（ d*Bi* ）

HPBW：主瓣的半功率角（°）

$HPBW_{back}$：主后瓣的半功率角（°）
FS：主瓣的前侧比（dB）
FB：主后瓣的前后比（dB）
FB_{back}：主后瓣的前侧比（dB）
n_{side}：侧瓣（旁瓣）的数量（不计算镜像）
n_{back}：后侧瓣（后旁瓣）的数量（不计算镜像）
ANG：需要计算的角度（三维极坐标系中的方向角或者仰角）
$TILT$：场强方向图的偏移度数

这 10 个参数在图 5.3-2 均有标注。而图 5.3-3 则给出了该数值算法的伪代码。

$$\text{NmAntPat}\left(G_0, \text{HPBW}, \text{HPBW}_{\text{back}}, \text{FS}, \text{FB}, \text{FB}_{\text{rest}}, n_{\text{side}}, n_{\text{back}}, \text{ANG}, \text{TILT}\right) :=$$

for $i \in 0.1, 0.2 .. 500$

$N \leftarrow i \text{ if } \left(\sqrt{\sin\left(90\text{deg} - \frac{\text{HPBW}}{2}\right)}\right)^{i} \geq 0.5$

for $i \in 0.1, 0.2 .. 500$

$N_{\text{back}} \leftarrow i \text{ if } \left(\sqrt{\sin\left(90\text{deg} - \frac{\text{HPBW}_{\text{back}}}{2}\right)}\right)^{i} \geq 0.5$

$q_{\text{side}} \leftarrow 10^{\frac{\text{FS}}{10}}$

$q_{\text{back}} \leftarrow 10^{\frac{\text{FB}}{10}}$

$q_{\text{back.rest}} \leftarrow 10^{\frac{\text{FB}_{\text{rest}}}{10}}$

$G_{\text{cal}} \leftarrow 10^{\frac{G_0}{10}}$

$\text{ANG} \leftarrow \text{ANG} + \text{TILT}$

if $\text{ANG} \leftarrow \text{ANG} - 360\text{deg} \text{ if } \text{ANG} + \text{TILT} \geq 360\text{deg}$
$\text{TILT} \leq \text{ANG} + \text{TILT} \leq \text{TILT} + 180\text{deg}$

$a \leftarrow \left(\sqrt{\sin(\text{ANG})}\right)^{N} \text{ if } \left(\sqrt{\sin(\text{ANG})}\right)^{N} > \frac{\left|\sin\left[n_{\text{side}}(\text{ANG})\right]\right|}{q_{\text{side}}}$

$a \leftarrow \frac{\left|\sin\left[n_{\text{side}}(\text{ANG})\right]\right|}{q_{\text{side}}} \text{ if } \left(\sqrt{\sin(\text{ANG})}\right)^{N} < \frac{\left|\sin\left[n_{\text{side}}(\text{ANG})\right]\right|}{q_{\text{side}}}$

otherwise

$a \leftarrow \frac{\left[\sqrt{\sin((-\text{ANG}))}\right]^{N_{\text{back}}}}{q_{\text{back}}} \text{ if } a \leftarrow \frac{\left(\sqrt{\sin(-\text{ANG})}\right)^{N_{\text{back}}}}{q_{\text{back}}} > \frac{\left|\sin\left[n_{\text{back}}(\text{ANG})\right]\right|}{q_{\text{back.rest}}}$

$a \leftarrow \frac{\left|\sin\left[n_{\text{back}}(\text{ANG})\right]\right|}{q_{\text{back.rest}}} \text{ if } \frac{\left(\sqrt{\sin(-\text{ANG})}\right)^{N_{\text{back}}}}{q_{\text{back}}} < \frac{\left|\sin\left[n_{\text{back}}(\text{ANG})\right]\right|}{q_{\text{back.rest}}}$

return $G_{\text{cal}} \cdot a$

图 5.3-3 *NmAntPat* 算法

5.4 附件数据的量纲分析

在我们进行定向天线的方向图分析后，我们就要将我们的附件中的数据与定向天线的方向图联系在一起。这里我们采用的是量纲分析。

附件数据中给出信号强度的单位是 $\mathrm{d}BM$，我们通过一下公式进行转换：

$$P(W)=\frac{10^{\frac{P(dBm)}{10}}}{1000}$$

由此我们可以得到单位 $\mathrm{d}BM$ 的量纲为 W。由 5-1 **天线基础**中我们有：

$$P(\theta,\varphi)=\frac{S(\theta,\varphi)}{S_{\max}(\theta,\varphi)}=\frac{|E(\theta,\varphi)|^2}{|E_{\max}(\theta,\varphi)|}=F(\theta,\varphi)^2$$

即辐射功率密度由坡印廷量计算得来，并且为归一化的方向图函数的平方。而坡印廷量的量纲为 P/m^2。因此，对坡印廷量进行曲面积分即可获得信号强度（辐射功率）。

因此我们由量纲建立了以下过程：

$$f(\theta,\varphi)\to F(\theta,\varphi)\to F(\theta,\varphi)^2\to S(\theta,\varphi)\to\int S(\theta,\varphi)dS\to W\to dBm$$

接下来我们将利用这个数据量纲关系建立相控阵阵元位置模型并求解出 32 个阵元的位置。

5.5 相控阵阵元信息模型

在 5-2 的探索性数据分析中我们已经获得数据的测量方式：离地图上某个点固定距离测量所有天线的空间数据。

现在我们假设该固定的测试点为 O，并且这个用来测试数据的坐标系为 $x_0y_0z_0$，其有直角坐标形式 (x_0,y_0,z_0)，球坐标形式 (r_0,θ_0,φ_0)。我们还不清楚采集的对于第 i 个天线，这个固定距离是否一样：于是我们先假设距离为 R_i。

对于第 i 个天线，我们假设它在坐标系 $x_0y_0z_0$ 的位置坐标的直角坐标形式为 $(x_0^{i-ap},y_0^{i-ap},z_0^{i-ap})$，球坐标为 $(r_0^{i-ap},\theta_0^{i-ap},\varphi_0^{i-ap})$。我们还有一个属于第 i 个天线的坐标系 $x_iy_iz_i$，其有直角坐标形式 (x_i,y_i,z_i)，极坐标形式 (r_i,θ_i,φ_i)：这个坐标系的原点在第 i 个天线的基座上。而第 i 个天线有物理倾角，这个倾角使得主瓣的方向发生了偏移，我们假这个倾角为（$\theta^{i-ap},\varphi^{i-ap}$）。

对于每个坐标系下的直角坐标形式与求坐标形式，我们都有一下坐标转换公式：

$$\begin{cases} x = r\sin\theta\cos\varphi \\ y = r\sin\theta\sin\varphi \\ z = r\cos\theta \end{cases} \quad \text{或} \quad \begin{cases} r = \sqrt{x^2+y^2+z^2} \\ \theta = \arccos\dfrac{z}{r} \\ \varphi = \arctan\dfrac{y}{x} \end{cases}$$

附件数据中给出的水平角 AZ 和俯仰角 EL 与球坐标系中的 θ 与 φ 不同，转换公式如下：

$$\begin{cases} \theta = \mathrm{EL} + 90° \\ \varphi = AZ + 180° \end{cases}$$

做好以上假设后，我们开始进行天线信息的推导。对于第 i 个天线，我们有以下信息需要求取：

$$G^{i-ap}{}_0, HPBW^{i-ap}, HPBW^{i-ap}{}_{back}, FS^{i-ap}, FB^{i-ap}, FB^{i-ap}{}_{rest}, n^{i-ap}{}_{side}, n^{i-ap}{}_{back}, TILT^{i-ap}$$
$$x_0^{i-ap}, y_0^{i-ap}, z_0^{i-ap}, \quad \theta^{i-ap}, \varphi^{i-ap}$$

我一般假设 $TILT^{i-ap}=0°$，这里方向图的偏移我们后面使用 $\theta^{i-ap}, \varphi^{i-ap}$ 和坐标变换来计算。整个空间体系可以由图 5.5-1 描述。红色为我们真正天线的方向图。

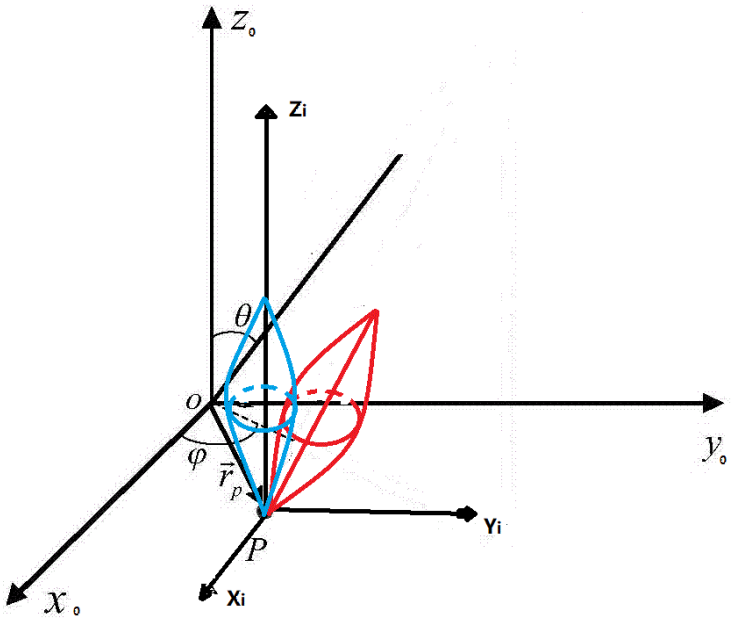


图 5.5-1 空间示意图

现在我们利用 5.4 中发现的过程：

$$f(\theta,\varphi)\to F(\theta,\varphi)\to F(\theta,\varphi)^2\to S(\theta,\varphi)\to\int S(\theta,\varphi)dS\to W\to dBm$$

搭建从天线基础信息到附件数据的关系。

首先由 $G^{i-ap}{}_0, HPBW^{i-ap}, HPBW^{i-ap}{}_{back}, FS^{i-ap}, FB^{i-ap}, FB^{i-ap}{}_{rest}, n^{i-ap}{}_{side}, n^{i-ap}{}_{back}, TILT^{i-ap}$ 这几个数据以及 *NmAntPat* 算法，我们可以得出图 5.5-1 中的蓝色方向图;

$$NmAntPat=f(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, ANG, TILT)$$

将其归一化有：

$$NmAntPat^{'}=F(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, ANG, TILT)$$

对于第 i 个天线的中的某一条附件数据，我们表示为 (EL, AZ, dBm)；获得该数据的点在坐标系 $x_0y_0z_0$ 中的球坐标点可以被表示为 $D\mathrm{ata}\quad (R_i, \mathrm{EL}+90°, AZ+180°)$ 。

由于实际上天线自身的物理角度 $\theta^{i-ap}, \varphi^{i-ap}$ 使得真实的天线方向图是图 5.5-1 中的红色的方向图。于是我们需要将 $(R_i, \mathrm{EL}+90°, AZ+180°)$ 转化为坐标系 $x_iy_iz_i$ 的坐标，在利用旋转，从红色的方向图定位回蓝色的方向图中。

坐标系 $x_0y_0z_0$ 与坐标系 $x_iy_iz_i$ 的转化方程如下：

$$\begin{cases} x_i = x_0 + x_0^{i-ap} \\ y_i = \mathrm{y}_0 + y_0^{i-ap} \\ z_i = \mathrm{z}_0 + z_0^{i-ap} \end{cases}$$

联合此前的极坐标与球坐标的转换公式，因此 $D\mathrm{ata}$ 在坐标系 $x_iy_iz_i$ 的直角坐标为：

$(R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}, R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}, R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap})$

接下去利用旋转公式将红色的方向图定位回蓝色的方向图中：由于天线的自身物理角度为（$\theta^{i-ap}, \varphi^{i-ap}$），所想要复位可以被认为是先围绕 Z_i 轴顺时针旋转 φ^{i-ap}，再围绕 Y_i 轴顺时针旋转 θ^{i-ap}.

围绕 Z 轴的旋转公式为：

$$\begin{cases} x^{'} = x\cos\gamma - y\sin\gamma \\ y^{'} = x\sin\gamma + y\cos\gamma \\ z^{'} = z \end{cases}$$

围绕 Y 轴旋转的公式为：

$$\begin{cases} x^{'} = z\sin\gamma + x\cos\gamma \\ y^{'} = \mathrm{y} \\ z^{'} = z\cos\gamma - x\sin\gamma \end{cases}$$

以上两个公式均已逆时针为正方向。由此复位后的 *D*ata 坐标应为：

$$\begin{aligned}
&([R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap}]\sin(-\theta^{i-ap})+\\
&\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\cos(-\varphi^{i-ap})-[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\sin(-\varphi^{i-ap})\}\cos(-\theta^{i-ap}),\\
&[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\sin(-\varphi^{i-ap})+[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\cos(-\varphi^{i-ap}),\\
&[R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap}]\cos(-\theta^{i-ap})-\\
&\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\cos(-\varphi^{i-ap})-[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\sin(-\varphi^{i-ap})\}\sin(-\theta^{i-ap}))
\end{aligned}$$

由于方向图是一个旋转图，我们就直接采用俯仰角处的方向图数据做积分，此时有该数据测量点的方向图函数为：

$$F(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, \arccos\frac{z}{r}+90^{\circ}, 0^{\circ})$$

其中，

$$z = R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap}]\cos(-\theta^{i-ap})-\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\cos(-\varphi^{i-ap})-[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\sin(-\varphi^{i-ap})\}\sin(-\theta^{i-ap})$$

，

$$r = \sqrt{\begin{aligned}
&\{[R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap}]\sin(-\theta^{i-ap})+\\
&\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\cos(-\varphi^{i-ap})-[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\sin(-\varphi^{i-ap})\}\cos(-\theta^{i-ap})\}^2\\
&+\\
&\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\sin(-\varphi^{i-ap})+[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\cos(-\varphi^{i-ap})\}^2\\
&+\\
&\{[R_i\cos(\mathrm{EL}+90°)+z_0^{i-ap}]\cos(-\theta^{i-ap})-\\
&\{[R_i\sin(\mathrm{EL}+90°)\cos(AZ+180°)+x_0^{i-ap}]\cos(-\varphi^{i-ap})-[R_i\sin(\mathrm{EL}+90°)\sin(AZ+180°)+y_0^{i-ap}]\sin(-\varphi^{i-ap})\}\sin(-\theta^{i-ap})\}^2
\end{aligned}}$$

。

接下来我们就对可以得到归一化的坡印廷量为：

$$S = F(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, \arccos\frac{z}{r}+90^{\circ}, 0^{\circ})^2$$

接下来对球面进行积分，我们的测试数据集将整个球面分成 73*37 份，每一份的 $dS = \frac{4\pi R^2{}_i}{73\times 37} = \frac{4\pi R^2{}_i}{2701}$ 。理论上在数值算法中应给对此面积微元进行无线分割，这里为了方便，不再进行分割，直接求取数值积分可得

$$W = \int S(\theta,\varphi)dS = \frac{4\pi R^2{}_i}{2701}\cdot F(G_0, HPBW, HPBW_{back}, FS, FB, FB_{rest}, n_{side}, n_{back}, \arccos\frac{z}{r}+90^{\circ}, 0^{\circ})^2$$

现在我们回到数据集中的一条数据(EL, AZ, dBm)，可由 5-4 中的公式求得：

$$P(W)=\frac{10^{\frac{P(dBm)}{10}}}{1000}$$

归一化后有：

$$W^{'}=\frac{P(W)}{P_{\max}(W)}$$

于此，求取相控阵的阵元的信息可以变成一个规划问题：

$$\min \quad \sum_{i=1}^{2701}(W_{\mathrm{i}}-W_{i}^{'})^{2}$$

其中，i 的取值范围为 1-2701 来源于在当移相器相位相同时，每一座阵元有 2701 条空间辐射强度数据。

5.6 遗传算法算法与相控阵阵元信息模型的求解

现在我们已经将整个相控阵阵元信息的求解转化成一个规划问题，我们选择用遗传算法求解该规划问题。

在这里简单说一下遗传算法的配置：一共遗传 10 代，每一代有 20 个个体。编码方式为二进制编码，选择方式为留下每一代中最好的个体（优选），交叉方法是均匀交叉，变异方法是均匀变异。

在使用粒子群算法求解该问题之后，我们获得 32 个阵元的基础信息如下表所示。

阵元编号	（$G^{i-ap}{}_{0}, HPBW^{i-ap}, HPBW^{i-ap}{}_{back}, FS^{i-ap}, FB^{i-ap}, FB^{i-ap}{}_{rest}, n^{i-ap}{}_{side}, n^{i-ap}{}_{back}$， $x_0^{i-ap}, y_0^{i-ap}, z_0^{i-ap}$，$\theta^{i-ap}, \varphi^{i-ap}$，$\mathrm{R}_i$）
1	[4.24352097e+00 8.59765110e+01 3.64227165e+01 2.42871039e+01 6.82616885e+00 1.88291252e+01 1.37484205e+00 4.63620151e+00 7.60007629e+02 8.52577069e+02 -1.39273776e+02 4.44742436e+01 3.11847565e+02 5.04494195e-01]
2	[1.23100398e+00 2.21988413e+01 7.35104499e+01 3.71966717e+00 2.28593091e+01 3.45650049e+00 2.59870777e+00 3.97473237e+00 2.85325322e+02 9.68801469e+02 5.62606394e+02 1.27346771e+02 2.37952764e+02 2.89917269e-01]
3	[1.93257516e+00 2.64884915e+01 2.54314856e+01 4.00982285e+00 3.25933290e+01 3.91354457e+01 7.82452376e-01 2.25204206e+00 1.48487233e+00 -3.28250244e+02 -9.06078249e+02 1.70587969e+02

	2.30462485e+01 1.24073147e+00]
4	[7.73478292e-01 7.61745512e+01 4.63818325e+01 1.47830150e+01 2.97045037e+01 6.49572038e+00 2.30924350e+00 3.87653244e+00 7.10015974e+02 -6.38328207e+02 5.68112915e+02 8.39919319e+01 1.28395623e+02 3.81470090e-03]
5	[1.28431443e+00 7.99688720e+01 2.74279951e+01 2.62427103e+01 4.82347948e+01 2.59798775e+01 2.09357461e+00 4.77462747e+00 9.00745297e+02 -7.66850249e+00 8.41436235e+02 1.66552893e+01 2.39217567e+02 3.66211287e-01]
6	[5.53799204e-01 7.88052643e+01 1.13412488e+01 1.70185251e+01 4.13245595e+01 3.88689412e+00 1.46766326e+00 1.05856520e+00 5.48810529e+02 -8.20627041e+02 6.09427080e+01 1.54858107e+02 2.57614191e+02 4.35829578e-01]
7	[5.71570942e+00 4.47776554e+01 3.58903083e+01 4.30108957e+01 2.65413061e+01 4.53249887e+01 6.77304914e-01 3.86418234e+00 -4.71158477e+02 -7.65512243e+02 -5.50170469e+02 1.57911623e+02 2.55145011e+02 4.68254536e-01]
8	[1.81865866e+00 2.51434375e+01 1.51956512e+01 2.59257087e+01 2.56514794e+00 2.08230217e+00 4.75228286e+00 2.13653768e+00 -8.07431991e+02 5.02300741e+00 9.14939799e+02 3.62846148e+01 2.38869094e+02 5.06401545e-01]
9	[5.69153375e-01 5.93336957e+01 1.59710464e+01 2.71490833e+01 1.04349713e+01 1.52830270e+01 5.47681377e-01 2.42148630e+00 -6.50170946e+02 8.28748540e+02 2.38911857e+02 1.08720807e+02 1.39447536e+01 4.10080347e-01]
10	[1.86014353e-01 5.10184965e+01 3.32457764e+01 1.41320363e+00 6.09422311e+00 2.97993467e+01 1.39768734e+00 4.64570965e+00 -5.33626112e+02 -3.29747514e+02 9.21382829e+02 9.08364209e+01 1.49792547e+02 4.88281716e-01]
11	[1.75047088e-01 1.71822044e+01 6.01415349e+01 2.53993277e+01 4.75908733e+01 2.70118971e+00 3.22532485e+00 1.15613571e+00 -7.40485898e+02 9.97763632e+01 3.80208378e+02 6.91715328e+01 2.81281854e+02 1.07097728e+00]
12	[1.33180745e+00 8.32664902e+01 8.09134397e+01 1.45648618e+01 6.14176382e+00 2.27060535e+01 6.86011969e-01 8.45189901e-01 1.65038266e+02 -7.90618697e+02 -4.49754190e+02 7.61986696e+01 7.16628567e+01 1.99127387e+00]
13	[2.34589800e+00 1.41140119e+00 1.94136328e+01 3.48811482e+01 2.71636745e+01 3.26716258e+01 2.79372005e+00 2.63349784e+00 5.72089741e+02 8.40440598e+02 -4.18879908e+02 1.62707846e+02 3.43234847e+02 4.19617099e-01]
14	[2.16641633e+00 8.98876475e+01 1.64981332e+01 3.68390435e+01 4.58230932e+01 2.17574804e+01 6.86441122e-01 2.72533200e+00 7.08078106e+02 -4.20103474e+01 7.61150132e+02 3.74397253e+01

	2.53475431e+01 3.49998808e-01]
15	[1.99570846e+00 4.09871969e+01 7.49904299e+01 1.93059629e+01 2.40463963e+01 1.81988890e+01 1.27254131e+00 3.02987388e+00 3.96207234e+02 4.41682283e+02 -3.83477577e+02 1.66914584e+02 7.99630355e+01 4.21524450e-01]
16	[1.33199819e+00 6.47869537e+01 4.85527502e+01 3.44292015e+01 3.19508380e+01 3.01868250e+01 4.06927020e+00 3.33113988e-01 1.32229931e+02 -8.35371814e+01 -4.91420261e+02 8.52783635e+01 2.82583392e+02 4.48227356e-01]
17	[1.65147939e+00 1.74818396e+01 1.45181031e+01 3.23584388e+01 2.00834466e+00 3.94977946e+01 3.46933219e+00 3.70291109e+00 4.52604725e+01 5.53460649e+02 -3.27208831e+02 1.70073500e+02 2.49624376e+02 4.31061202e-01]
18	[2.30937224e+00 1.14239039e+01 6.28381661e+01 3.19524116e+01 4.41441957e+00 1.77550962e+01 1.78636244e+00 1.59583721e+00 9.81990797e+02 -8.07035262e+02 6.10723124e+02 9.87395465e+01 1.27893341e+02 4.24385476e-01]
19	[8.64077438e-01 2.67111365e+01 6.56734997e+01 3.72866032e+01 2.36533391e+01 3.81995089e+01 2.67787235e+00 3.57740267e+00 -3.44863267e+02 -1.69688387e+02 7.76502396e+02 3.07747944e+01 2.25998941e+02 5.94139666e-01]
20	[3.25899435e+00 3.22550318e+01 7.86931693e+01 1.93911261e+01 4.37039315e+01 4.55969768e+01 2.08265026e+00 1.73658060e+00 2.12842191e+02 6.84476552e+02 -1.23322604e+02 3.49521780e+01 2.96439225e+02 1.11580001e-01]
21	[8.39186515e-01 7.81055718e+01 1.70419569e+01 4.70899077e+01 1.22420905e+01 1.60019550e+01 4.36964452e+00 4.04671578e+00 9.15006557e+02 -3.12853158e+01 3.45147462e+02 1.46655432e+02 2.94648108e+02 2.90870944e-01]
22	[5.66025320e+00 1.09736356e+01 3.04880338e+01 3.73617052e+01 2.94871611e+01 2.51867535e+01 1.16413704e+00 1.17637746e+00 -8.69861479e+02 4.81870157e+02 1.88039005e+02 1.68890752e+02 7.53731493e+01 4.31061202e-01]
23	[3.43036979e-01 5.84722123e+00 2.37690294e+01 1.57590063e+01 3.49048471e+01 1.38336314e+01 2.54845385e+00 3.10816584e+00 1.96107098e+02 -9.88967885e+02 -6.25367761e+02 5.34200415e+01 2.49052056e+02 4.95911117e-01]
24	[1.13072503e+00 6.89657964e+01 7.16393391e+01 3.99105453e+01 3.58462676e+00 4.77459886e+01 8.41875879e-01 4.83088954e-01 -5.08658894e+02 -7.32648595e+02 -8.04673962e+02 1.50614634e+02 8.28936414e+01 1.57451780e+01]
25	[1.45573755e+00 3.73029111e+01 2.04663472e+00 4.58752116e+01 8.96778962e+00 1.71097442e+01 1.73583196e-01 1.18449801e+00 -2.36867177e+02 1.17959135e+02 -8.78091696e+02 3.00744153e+01

	2.24000801e+02 1.54495387e-01]
26	[6.33097299e-01 2.25369287e+01 5.76562192e+01 8.33035310e-02 4.39426841e+01 5.92704385e+00 3.38574732e+00 2.80215531e+00 -5.07205493e+02 -9.36433731e+02 9.78098848e+01 5.91810028e+01 4.89286889e+01 4.60625134e-01]
27	[1.48658894e+00 1.89356412e+01 5.91557685e+01 1.39446392e+00 1.77961996e+01 1.93651384e+01 1.51389266e+00 1.19263763e+00 -5.85654817e+02 9.01546384e+02 -7.62606394e+01 7.93289178e+01 2.02604907e+02 4.73976587e-01]
28	[1.49774694e+00 2.84847436e+01 4.27715328e+01 2.04504685e+01 4.20404358e+01 4.16476170e+01 3.97292516e+00 3.56106621e+00 7.64800801e+02 9.31266719e+02 2.89628305e+02 1.21851541e+02 2.30433989e+02 5.41687528e-01]
29	[2.81787187e+00 5.25074029e+01 6.01632501e+01 2.55604511e+01 4.12346279e+01 7.40690938e+00 4.26403452e+00 2.02902511e+00 5.67290847e+02 5.59320983e+01 8.30810386e+02 5.25989843e+01 2.43716472e+02 1.20163078e-01]
30	[9.19295234e-01 4.67182891e+01 7.84240040e+01 2.36305462e+01 4.35656963e+01 9.82075674e+00 1.20689507e+00 1.88490571e+00 -1.91527549e+02 3.06748683e+02 -3.44477028e+01 1.33825620e+01 1.95323711e+02 2.07901199e-01]
31	[1.47647998e+00 6.94089407e+01 6.24950147e+01 8.85554205e+00 2.09239206e+01 1.11416923e+01 7.67246024e-01 2.08307942e+00 -6.64325394e+02 2.46878859e+02 1.76739861e+02 2.62712538e+01 2.76523739e+02 1.98364447e-01]
32	[7.36141907e-01 8.56340462e+01 8.34806380e+01 1.23877643e+00 3.23467563e+01 3.35044704e+01 1.59087333e+00 3.06626135e+00 1.79300479e+01 -9.80594617e+02 -5.62005579e+02 8.63651527e+01 2.53833517e+02 3.75748039e-01]

在我们处理完之后，我们从每一座发现 32 个阵元统一在一个倾斜的平面上面，形成一个 4*8 的平面相控阵，并带有一定物理倾角，但是显然我们可以知道每个阵元的倾角是不同的。

5.7 相控阵的方向图函数与辐射功率

通过 5-4 中的模型我们已经获得每个阵元的位置，并且知道了这是一个 4*8 的平面阵。图下图 5.7-1 所示。

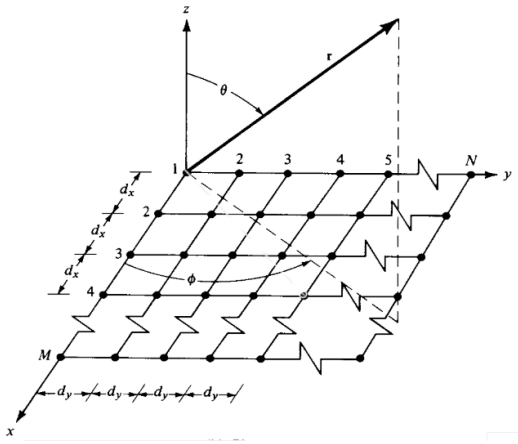


图 5.7-1 平面天线阵

图 5.7-1 的坐标系以 1 号天线的基座坐标为原点，将所有的相控阵阵元放置于 xy 平面中，该坐标系可以看坐标系 $x_1y_1z_1$ 旋转而成，我们称这个坐标系为 xyz 。在这个坐标系中对该相控阵进行分析。由天线阵的方向图乘积定理，对于相控阵的中的任意阵元 i ，其阵因子有：

$$AF(\theta,\varphi)=\sum_{m=1}^{4} I_{mn} e^{j\xi_m} \cdot \sum_{m=1}^{8} I_{mn} e^{j\xi_n}$$

其中， $\xi_m = \beta x_m^{'} \sin\theta\cos\varphi$ ， $\xi_n = \beta y_n^{'} \sin\theta\cos\varphi$ ， θ 和 φ 为波束的方向。

对于相控阵生成的新的相控阵的方向图，我们有

$$f_{Array}(\theta,\varphi)=\sum AF(\theta,\varphi)\cdot f_i(\theta,\varphi)$$

$f_i(\theta,\varphi)$ 是每个阵元的方向图。

接下来我们就可以通过得到的 $f_{Array}(\theta,\varphi)$ 求得所有的参数，包含辐射功率。过程与 5.5 中的推导相似，这里不再赘述。过程大体流程如下：

$$f_{\text{Array}}(\theta,\varphi)\rightarrow F_{\text{Array}}(\theta,\varphi)\rightarrow F_{\text{Array}}(\theta,\varphi)^2\rightarrow S_{\text{Array}}(\theta,\varphi)\rightarrow\int S_{\text{Array}}(\theta,\varphi)dS\rightarrow W_{\text{Array}}\rightarrow dBm_{\text{Array}}$$

5.8 微波问题 1 的模型建立与求解

微波问题 1 显然是一个多目标规划问题。需要满足一下条件：

1. 合成功率为 35dBm；2. 旁瓣电平尽可能低；3. 可以选择关闭某些通道；4.在位置 AZ=10°，俯仰 EL=10°处有一站点，波束设计时避免对该站点的干扰。

我们一一分解限制：

第一个限制条件是：$\begin{cases}\max(|\theta|)\\ \max(|\varphi|)\end{cases}$

$$if \quad dBm_{\text{Array}}(\theta,\varphi)\geq 35$$

第二个目标是：$\min \quad FS_{\text{Array}}$

第三条件则是允许关闭吗某个阵元，于是我们对每个阵元引入 p^i 这个 0-1 变量，来确定其是否打开。并且将移相器状态扩张为 $(0,1,2,3,4)$，0 值极为关闭状态。

第四个条件是：$\min \quad aBm_{Array}(10°,10°)$

对于多目标规划问题，我们采用线性加权和法转化为以下问题：

$$\max \quad \frac{1}{4}(|\theta|)+\frac{1}{4}(|\varphi|)+\frac{1}{4}(-FS_{\text{Array}})+\frac{1}{4}(dBm(10°,10°))$$
$$st.dBm_{\text{Array}}(\theta,\varphi)\geq 35$$

采用遗传算法求解，我们获得的相位配置是[1,3,2,3,2,3,1,0,3,2,4,1,3,3,2,0,1,1,4,0,3,2,4,2,2,2,1,1,3,0,0,1]

5.9 微波问题 2 的模型建立与求解

微波问题 2 任然是一个多目标规划,要求满足四个条件：

整个覆盖区域内，平均辐射功率 35dBm；2. 波束覆盖的“凹坑”尽可能小；3. 覆盖波束个数尽可能少；4. 用尽可能少的波束，覆盖水平角±30°，俯仰角±15°的区域。请给出波束配置列表[z_1,z_2,…,z_(M])（M 为波束个数）。

我们一一分解限制：

$$dBm_{\text{Array}}(\theta,\varphi)\geq 35$$

第一个条件与第四个条件：$st. \quad -30<\theta<30$

$$-15<\varphi<15$$

第二个条件：我们对于区域内的每一个点，计算其余附近的临近点的 dBm 差值来定义凹坑，这是由于凹坑是指‘功率最大值与最小值的偏差’。也即：

$$\min \sum_{i=1}^{2701}\begin{matrix}[(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta+5,\varphi))^2+(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta,\varphi+5))^2++(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta+,\varphi+5))^2\\ +(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta-5,\varphi+5))^2+(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta+5,\varphi-5))^2+(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta-5,\varphi))^2\\ +(\text{dbm}(\theta,\varphi)\text{-dbm}(\theta,\varphi-5))^2]\end{matrix}$$

第三个条件：$\min \quad p^i$

我们样适用线性加权法与遗传算法求解，解得相位配置为[3,0,1,1,2,1,1,1,3,0,1,1,3,0,0,0,4,2,3,2,3,3,1,2,4,2,1,2,4,3,3,3].

5.10 骨干网问题分析

骨干网问题 1 中需要我们请预估未来十年（截止 2028 年）信息时代各类人群所需要的信息量，以满足广东省全部人口的信息接入需求为目标，省会广州市为数据中心，选择性地用最新的通信技术更新从广州到其余主要行政城市的骨干网络，核心是建立一个部署价值函数来描述这一件事情。而骨干网问题 2 中则是要求我们考虑不同人群的价值，将价值考虑进去以获得新的网络价值函数。我们对于骨干网问题的处理过程是：

1. 首先我们对广东省的互联网接入量和网民上网时间进行预测。
2. 建立广州的骨干网络数据流模型，设计成本规划的模型并求解。
3. 从人口资源环境经济学出发，思考不同人群的价值与年龄结构。

5.11 互联网接入量预测与网民上网时间预测

我们从网上寻找到有关广东省 2016 年 8 月到 2019 年 2 月，各月的互联网接入量 C_t（万 G）[8] 与 广东省 2011 年到 2018 年网民平均每周上网时间 $Net-hours$（小时）[9]，我们接下来将利用具有长短时记忆结构的循环神经网络[5][6][7]来对网络接入量与网民平均上网时间进行预测。

循环神经网络（RNN）是深度学习中常用的来解决序列性问题：如文本生成、翻译、还有摘要等问题的利器，循环神经网路一般结构如图 5.11-1 所示。

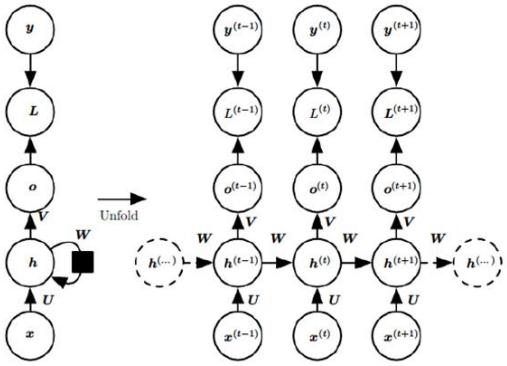


图 5.11-1 RNN 的标准结构

我们可以看到，由于循环神经网络的每一层神经网络都共享了参数W、U、V，因此它具有记忆和学习序列特征的能力。但是缺点也很明显：由于结构的复杂性收到序列长度的影响，当 BP 反向传播算法使用时，就容易发生梯度消失或者时梯度爆炸的情况。于是针对这个问题，便有了长短时记忆结构（LSTM）来解决这一问题。LSTM 结构一般如图 5.11-2 所示。

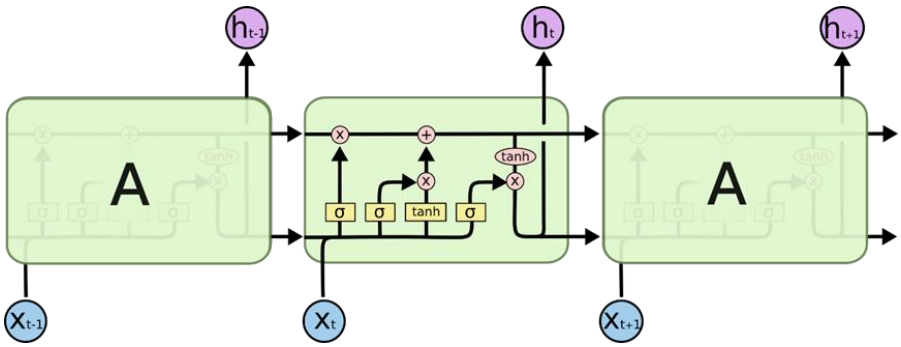


图 5.11-2 长短时记忆结构（LSTM）-RNN

长短时记忆结构通过模仿人类的记忆行为，添加入一条记忆线：构造遗忘门、输入门、输出门，来保护和控制神经结构细胞状态。进而来对记忆特征进行控制，解决梯度爆炸和梯度消失的问题。我们接下来将使用这种结构对广东省的互联网连接量和网民上网时间进行预测。主要过程如下：

（1） 将时间序列转化为监督学习

对于时间序列问题，我们可以通过使用上一个时间 C_{t-1} 或 $Net\text{-}hours_{t-1}$ 的观测值作为输入，以当前时间步 C_t 或 $Net\text{-}hours_t$ 的观测值作为输出来实现将问题转化为一个监督学习的问题。

（2）将时间序列转换为平稳的

时间序列数据一般都是不平稳的，具有一定的时间以来结构，如图 5.11-3 所示。经过 ADF 检验，我们使用一节差分将时间序列数据转化为平稳数据。

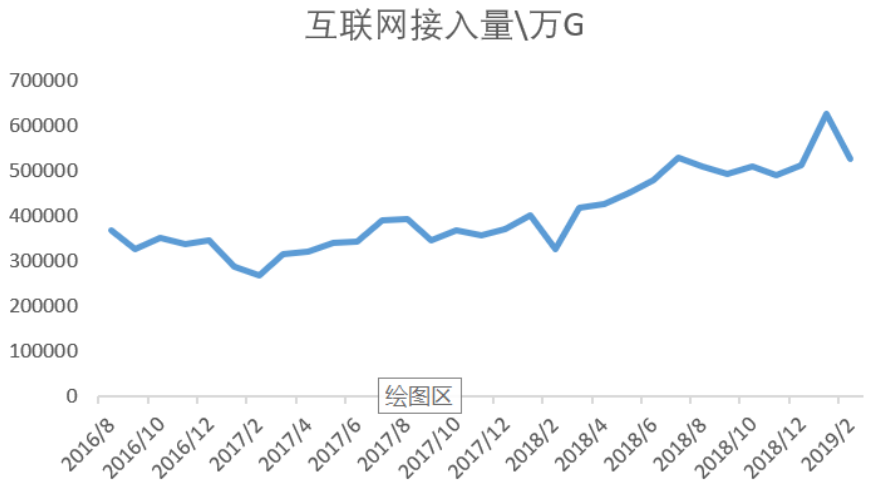


图 5.11-3 互联网接入数据走势图

（3）将时间序列按比例缩放

像其他神经网络一样，LSTM 希望数据大小能控制在神经网络使用的激活函数的范围内。LSTM 的默认激活函数是双曲正切（ tanh ），它输出-1 和 1 之间的值。这是时间序列数据的首选范围。为了使实验公平化，必须在训练数据集上计算缩放系数（最小值和最大值），并将其应用于缩放测试数据集和任何预测。这是为了避免使用来自测试数据集的信息影响实验，这样可能给模型带来一个小优势。

（4）神经网路的训练与预测结果

利用 Python 的 Krea 库我们建立了神经网络模型，并进行了预测，对于网络接入量，模型在测试集上的 $MSE=537.3$；对于网民平均上网时间，模型在测试集上的 $MSE=0.43$。

由该模型我们预测出 2028 年 8 月份广东省的网络接入量 C_{2018} 为 1267721.9 万 G ，2028 年广东网民的某周平均上网时间 $N\text{et-hours}_{2028}$ 为 48 h

5.12 广东骨干网络数据流模型的建立与部署价值函数的求解

我们对广东省的网络结构建立以下模型，如下图 5.12-1 所示。

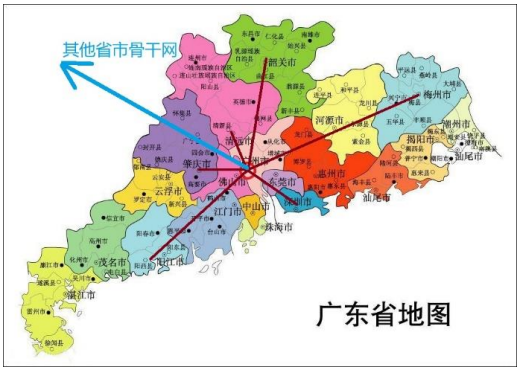


图 5.12-1 广东的骨干网结构

我们建立如下网络结构：整个广东省只有广州一个节点连接有其他省市的骨干网，而其他节点的数据只能通过广州这个节点向外传输数据。于是整个网路的数据必须实时传输到广州并传递出去。每个市之间则可以有有不同类型的网络设备，并允许同时存在有多台设备进行数据传输。我们的目标则是在，满足我们预测的网络接入量的情况下，使得整体的部署成本最小。

我们首先计算出广州作为一个骨干网最大的节点，需要在每秒钟转发多少的数据量，我们假设这个数据量为 TC 。则有：

$$TC=\frac{C_{2018}}{4*60*60*N\text{et-hours}_{2028}}$$

TC 来源于广东省的 21 市，于是我们按照人口比例将每个市发出的数据传输量 TC_i 计算出来来，我们在此我们假设各市的人口占全广东人口的比例与 2017 年保持一致。这里 $i=1,2\cdots21$ ，代表我们对广东 21 个市的编号。于是有:

$$TC_i=\alpha_i\cdot TC$$

其中α_i为人口比例。

第二步我们将进行城际网传输的两个城市连接起来，构成一个网络结构，这显然是一个图。值得指出的是能够进行城际网传输的条件是两座城市之间的距离小于$200km$，即$D(City_i, City_j) < 200km$。我们定义一个 21*21 的矩阵$CC$，里面的矩阵$CC$的元素$CC(i, j)$表示在真实传输环境中每秒$City_i$向$City_j$传输的数据量，显然这是有方向性的。于是有了矩阵$CC$的定义之后，我们就能根据语境构建好的图与矩阵$CC$的数据，依据运筹学的网络流算法——Ford-Fulkerson 方法[10]与之前计算好的TC_i 计算每秒到达广州的数据量$SECTC$，这里对网络流算法进行一定的过程性描述：

首先我们需要了解一些图论概念：

（1）残留网络：容量网络 - 流量网络 = 残留网络。假定一个网络$G=(V,E)$，其源点 s，汇点 t。设 f 为 G 中的一个流，对应顶点 u 到顶点 v 的流。在不超过$C(u,v)$的条件下（C 代表边容量），从 u 到 v 之间可以压入的额外网络流量，就是边（u，v）的残余容量。

(2) 增广路径：这是一条不超过各边容量的从 s 到 t 的简单路径，向这个路径注入流量，可以增加整个网络的流量。我们称在一条增广路径上能够为每条边增加的流量的最大值为路径的残余容量，$cf(p) = \min\{cf(u,v); (u,v) \in 路径p\}$

（3）割：用来证明“当残留网络中找不到增广路径时，即找到最大流”。

算法过程如下：

（1）选定任一一个城市为源点，广州为汇点，对于整个网络上的所有结点$u,v \in V, f(u,v) = 0$，给出的初始流量值为 0。

（2）在每一次迭代中，将 G 的流值增加，方法就是在残留网络 Gf 中寻找一条增广路径（一般用广度有限搜索 BFS 算法遍历残留网络中各个结点，以此寻找增广路径），然后在增广路径中的每条边都增加等量的流值，这个流值的大小就是增广路径上的最大残余流量。

（3）重复这一过程，直到残余网络中不再存在增广路径为止。

（4）选取另外的一个城市为源点，广州仍为汇点，重复上述过程。最终获得整体流量$SECTC$，而矩阵CC中的值就是增广路径中每一次数据的加总。

接下来我们就来构建部署价值函数$Total\,\mathrm{Cost}$，对于每个两个图中连接的城市$City_i$和$City_j$，他们之间每秒所需要传输的总数据$TCC_{ij} = CC(i,j) + CC(j,i)$。这些数据都需要有三种以下基站进行传输，如下表所示。

类型	传输容量	最大传输距离	总容量	标准成本
A	400 Gb/s	200 km	32Tb/s	X 万元
B	600 Gb/s	100 km	48 Tb/s	1.25X 万元
C	800Gb/s	80 km	64Tb/s	1.5X 万元

我们假设城市$City_i$和$City_j$之间有x_{ij} 台 A 型传输设备，y_{ij} 台 B 型传输设备，z_{ij} 台 C 型传输设备。于是有以下约束，对于他们需要每秒顺势传输量，我们需要满足：

$$x_{ij} \cdot p^A{}_{ij} \cdot 400 + y_{ij} \cdot p^B{}_{ij} \cdot 600 + z_{ij} \cdot p^C{}_{ij} \cdot 800 \geq TCC_{ij}$$

即他们建立的链接设备需要满足他们的顺时传送量。其中$p^A{}_{ij}$、$p^B{}_{ij}$、$p^C{}_{ij}$是一个 0-1

规划量，取决于 $D(City_i, City_j)$。以 $p^A{}_{ij}$ 为例，其表达式为：

$$p^A{}_{ij} = \begin{cases} 0 & D(City_i, City_j) > 200\text{km} \\ 1 & D(City_i, City_j) \le 200\text{km} \end{cases}$$

对于每秒到达广州的数据量 $SECTC$，我们有如下限制条件：

$$SECTC \le TC$$

而部署价值函数 $Total\,\mathrm{Cost}$ 的求解就转化成一个规划问题：

$$\min \quad Total\,\mathrm{Cost} = \sum x_{ij} \cdot p^A{}_{ij} + 1.25 \cdot \sum y_{ij} \cdot p^B{}_{ij} + 1.5 \cdot \sum z_{ij} \cdot p^C{}_{ij}$$

5.13 梅特卡夫定理与网络价值函数的构建

骨干网问题 2 中则是要求我们考虑不同人群的价值，将价值考虑进去以获得新的网络价值函数。在这里我们引入资源与环境经济学中的梅特卡夫定理，来构建我们的网络价值函数 $TotalValue$ 。

梅特卡夫定理说明了以下问题：网络价值与用户数的平方成正比。网络使用者越多，价值就越大。也即网络的价值 $NetValue$ 可以被表示为：

$$NetValue = k \cdot N^2$$

其中，k 为价值系数，N 为使用人数。

我们首先承认梅特卡夫定理的正确性，但是这里有一个其他的问题：每个城市的用户阶层数不同的，有的城市属于创新型城市，如广东，年轻人和中年人在网络中创造的价值要比留守老人比例高的人要搞。因此我们提出了一下对于 N 的修正：

$$N' = \beta_i N$$

这里 β_i 是指某个特定城市 $City_i$ 人口结构中，25-50 岁人群所占的比例。这群人才是正在利用网络创造价值的真正人群。因此有网络价值函数 $TotalValue$ 可以写成：

$$TotalValue = NetValue - Total\,\mathrm{Cost} = \sum k \cdot (\beta_i N)^2 - (\sum x_{ij} \cdot p^A{}_{ij} + 1.25 \cdot \sum y_{ij} \cdot p^B{}_{ij} + 1.5 \cdot \sum z_{ij} \cdot p^C{}_{ij})$$

而该网络价值函数的求取也是一个规划问题：

$$\max \quad TotalValue = NetValue - Total\,\mathrm{Cost}$$

六、模型的评价

6.1 2028 年广东省互联网接入量模型评价

利用 LSTM 进行预测，由于统计资源限制，以 2016 年 8 月到 2019 年 2 月的互联网总的流量为基础，预测到 2028 年 1 月的互联网接入流量，数据量有些小，训练集不够，对于预测的数据可能出现较大的偏差。

6.2 网络价值最大化的部署方案模型评价

采用最大流和最小费用模型进行模型的构建，所得出的结果只能是无限趋近于最优结果却不能是最优结果，这个问题也是现在一直困扰着科学家们的问题，对于最优方案以前参数的设置还需继续探索和改进。

6.3 重新构建网络价值函数最优部署方案模型评价

以梅特卡夫定理为基础，进行的最新部署方案的改进，考虑的因素较少，给出的权重也不一定具有很大的成效，还需要继续改进

参考文献

[1] Warren L.Stutzman, Gary A.Thiele 著，朱守正，安同一 译，天线理论与设计，北京，人民邮电出版社，2006 年 10 月

[2] Jankowski-Mihułowicz P, Lichoń W, Węglarski M. Numerical Model of Directional Radiation Pattern Based on Primary Antenna Parameters[J]. International Journal of Electronics and Telecommunications, 2015, 61(2): 191-197.

[3] 了不起的赵队，RNN，https://blog.csdn.net/zhaojc1995/article/details/80572098，2019/5/4

[4]朱小虎 XiaohuZhu，RNN 理解，https://www.jianshu.com/p/9dc9f41f0b29，2019/5/4

[5] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult[J]. IEEE transactions on neural networks, 1994, 5(2): 157-166.

[6] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[7] Graves A. Long short-term memory[M]//Supervised sequence labelling with recurrent neural networks. Springer, Berlin, Heidelberg, 2012: 37-45.

[6] [6]QiqiHe，用 Python 的长短期记忆神经网络进行时间序列预测，https://cloud.tencent.com/developer/article/1040557，2019/5/3

[8] 广东省通信管理局——统计信息——广东省通信发展情况，https://gdca.miit.gov.cn/framenet/web/pgemore.jsp，2019/5/4

[9] 2018 年中国网民数达 8.29 亿 人均每天上网 4 小时，https://www.awtmt.com/articles/3485361?from=wscn，2019/5/4

[10] 2018 年广东省各地级市人口 GDP 人均 GDP 排名，http://www.chamiji.com/201806036052.html，2019/5/4

[11] 廖少少，Ford-Fulkerson 方法—— 最大流问题，https://www.jianshu.com/p/efb2d79e2b0f，2019/5/5

[12] Metcalfe R. Metcalfe’s law[J]. Infoworld, 1995, 17: 40-53.

附录

获得每个阵元的信息参数，包括方向图参数以及其地理位置倾角

```
import math

import numpy as np


def Ant(G0,HPBW,HPBW_back,FS,FB,FB_rest,n_side,n_back,ANG,TILT=0):

    for i in range(1,5001):
        if (((math.sin((90-(HPBW/2))/180*math.pi)))**(1/2))**(i/10) >= 0.5:
            N = i/10
        break

    for i in range(1,5001):
        if (((math.sin((90-(HPBW_back/2))/180*math.pi)))**(1/2))**(i/10) >=
0.5:
            N_back = i/10
        break

    q_side = 10**(FS/10)
    q_back = 10**(FB/10)
    q_back_rest = 10**(FB_rest/10)
    G_cal =  10**(G0)

    ANG_new = ANG + TILT

    if ANG_new >= 360:
        ANG_new = ANG_NEW - 360

    if TILT <= ANG_new + TILT <= TILT + 180:
        if         ((math.sin((ANG_new))/180*math.pi))**(1/2)**(N)         >=
math.fabs(math.sin((n_side*ANG_new))/180*math.pi)/(q_side):
            a = ((math.sin((ANG_new))/180*math.pi))**(1/2) ** (N)
        else:
            a = math.fabs(math.sin((n_side*ANG_new))/180*math.pi)/(q_side)
    else:
        ANG_new = ANG + TILT
        if         ((math.sin((ANG_new))/180*math.pi))**(1/2)**(N)         >=
```

```
math.fabs(math.sin((n_side*ANG_new))/180*math.pi)/(q_side):
            a = ((math.sin((ANG_new))/180*math.pi))**(1/2) ** (N_back)
        else:
            a = math.fabs(math.sin((n_side*ANG_new))/180*math.pi)/(q_side)

    return G_cal * a




def ZhuangHuang(EL,AZ,x,y,z,xieta,fai,r):

x1=((r*math.sin(((EL+90)/180)*math.pi)+z)*math.sin(((-xieta)/180)*math.pi))
+(((r*math.sin(((EL+90)/180)*math.pi)*math.cos(((AZ+90)/180)*math.pi)+x)*ma
th.cos(((-fai)/180)*math.pi))-(r*math.sin(((EL+90)/180)*math.pi)*math.sin((
(AZ+90)/180)*math.pi)+y))*math.cos(((-xieta)/180)*math.pi)

y1=(((r*math.sin(((EL+90)/180)*math.pi)*math.cos(((AZ+90)/180)*math.pi)+x)*
math.sin(((-fai)/180)*math.pi))+(r*math.sin(((EL+90)/180)*math.pi)*math.sin
(((AZ+90)/180)*math.pi)+y))

z1=((r*math.cos(((EL+90)/180)*math.pi)+z)*math.cos(((-xieta)/180)*math.pi))
-(((r*math.sin(((EL+90)/180)*math.pi)*math.cos(((AZ+90)/180)*math.pi)+x)*ma
th.cos(((-fai)/180)*math.pi))-(r*math.sin(((EL+90)/180)*math.pi)*math.sin((
(AZ+90)/180)*math.pi)+y))*math.sin(((-xieta)/180)*math.pi)
    r1=((x1)**2+(y1)**2+(z1)**2)**(1/2)
    return ((math.acos(z1/r1))/math.pi)*180 + 90

'''
x=[G0,HPBW,HPBW_back,FS,FB,FB_rest,n_side,n_back,x,y,z,xieta,fai,r]
'''


def fun1(x):
    #print(x)
    loss=0
    EL_list=np.linspace(-90,90,37)
    AZ_list=np.linspace(-180,180,73)

    for i in range(73):
        for j in range(37):

            EL = EL_list[j]
            AZ = AZ_list[i]
```

```
            F                                                                   =
Ant(x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],ZhuangHuang(EL,AZ,x[8],x[9],x[1
0],x[11],x[12],x[13]),0)
            F = 10**(F/20)
            loss                    =                    loss                    +
(F**2*((4*math.pi*(x[13]**2))/2701)-10**(a[i,j]*(1/10))*(1/1000))**2
    #print(loss)
    return loss*(1/2701)



'''
from time import time
from sopt.util.functions import *
from sopt.util.pso_config import *
from sopt.PSO.PSO import PSO
from sopt.util.constraints import *

class TestPSO:
    def __init__(self):
        self.func = fun1
        self.func_type = 'min'
        self.variables_num = 14
        self.lower_bound                                                    =
np.array([0,0,0,0,0,0,0,0,-10000,-10000,-10000,0,0,0])
        self.upper_bound                                                    =
np.array([100,90,90,100,100,100,10,10,10000,10000,10000,180,360,10000])
        self.c1 = basic_config.c1
        self.c2 = basic_config.c2
        self.generations = 200
        self.population_size = 100
        self.vmax = 1
        self.vmin = -1
        self.w = 1
        self.w_start = 0.9
        self.w_end = 0.4
        self.w_method = pso_w_method.linear_decrease
        #self.complex_constraints = [constraints1,constraints2,constraints3]
        self.complex_constraints = None
        self.complex_constraints_method = complex_constraints_method.loop
        self.PSO = PSO(**self.__dict__)

    def test(self):
        start_time = time()
```

```
        self.PSO.run()
        print("PSO costs %.4f seconds!" %(time()-start_time))
        self.PSO.save_plot()
        self.PSO.show_result()


if __name__ == '__main__':

    Gonglv1=[]
    Gonglv2=[]
    Gonglv3=[]
    Gonglv4=[]

    Xiang1=[]
    Xiang2=[]
    Xiang3=[]
    Xiang4=[]


    for i in range(32):
        Gonglv1.append([])
        Xiang1.append([])
        Gonglv2.append([])
        Xiang2.append([])
        Gonglv3.append([])
        Xiang3.append([])
        Gonglv4.append([])
        Xiang4.append([])


    for i in range(32):
        for j in range(73):
            Gonglv1[i].append([])
            Xiang1[i].append([])
            Gonglv2[i].append([])
            Xiang2[i].append([])
            Gonglv3[i].append([])
            Xiang3[i].append([])
            Gonglv4[i].append([])
            Xiang4[i].append([])


    for i in range(32):
        for j in range(73):
```

```
        for k in range(37):
            Gonglv1[i][j].append([])
            Xiang1[i][j].append([])
            Gonglv2[i][j].append([])
            Xiang2[i][j].append([])
            Gonglv3[i][j].append([])
            Xiang3[i][j].append([])
            Gonglv4[i][j].append([])
            Xiang4[i][j].append([])
```

```
f = open('1.txt',encoding='utf8')
for i in f:
    i = i.replace('\n','').split(',')
    if i[3] == '1':
        Gonglv1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '2':
        Gonglv2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '3':
        Gonglv3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    else:
        Gonglv4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]

f.close()
```

```
TestPSO().test()
```

```
'''



'''



import numpy as np
from matplotlib import pyplot as plt
from matplotlib import cm
from matplotlib import axes

def draw_heatmap(data):
    #cmap=cm.Blues
    cmap=cm.get_cmap('rainbow',1000)
    figure=plt.figure(facecolor='w')
    ax=figure.add_subplot(1,1,1,position=[0.1,0.15,0.8,0.8])



    vmax=data[0][0]
    vmin=data[0][0]
    for i in data:
        for j in i:
            if j>vmax:
                vmax=j
            if j<vmin:
                vmin=j

map=ax.imshow(data,interpolation='nearest',cmap=cmap,aspect='auto',vmin=vmi
n,vmax=vmax)
    cb=plt.colorbar(mappable=map,cax=None,ax=None,shrink=0.7)
    plt.show()


a=np.array(Gonglv1[0],dtype='float64')


a=np.array(Xiang1[4],dtype='float64')


draw_heatmap(a)
```

```
'''



'''

from time import time
from sopt.util.functions import *
from sopt.Optimizers.SA import SA
from sopt.util.sa_config import *
from sopt.util.constraints import *

class TestSA:
    def __init__(self):
        self.func = fun1
        self.func_type = 'min'
        self.variables_num = 14
        self.lower_bound                                                    =
np.array([0,0,0,0,0,0,0,0,-10000,-10000,-10000,0,0,0])
        self.upper_bound                                                    =
np.array([100,90,90,100,100,100,10,10,10000,10000,10000,180,360,10000])
        self.T_start = 100
        self.T_end = 1e-6
        self.q = 0.9
        self.L = 100
        self.init_pos = None
        #self.complex_constraints = [constraints1,constraints2,constraints3]
        self.complex_constraints_method = complex_constraints_method.loop
        self.SA = SA(**self.__dict__)


    def test(self):
        start_time = time()
        self.SA.run()
        print("SA costs %.4f seconds!" %(time()-start_time))
        self.SA.save_plot()
        self.SA.show_result()


if __name__ == '__main__':

    Gonglv1=[]
```

```
Gonglv2=[]
Gonglv3=[]
Gonglv4=[]

Xiang1=[]
Xiang2=[]
Xiang3=[]
Xiang4=[]


for i in range(32):
    Gonglv1.append([])
    Xiang1.append([])
    Gonglv2.append([])
    Xiang2.append([])
    Gonglv3.append([])
    Xiang3.append([])
    Gonglv4.append([])
    Xiang4.append([])


for i in range(32):
    for j in range(73):
        Gonglv1[i].append([])
        Xiang1[i].append([])
        Gonglv2[i].append([])
        Xiang2[i].append([])
        Gonglv3[i].append([])
        Xiang3[i].append([])
        Gonglv4[i].append([])
        Xiang4[i].append([])


for i in range(32):
    for j in range(73):
        for k in range(37):
            Gonglv1[i][j].append([])
            Xiang1[i][j].append([])
            Gonglv2[i][j].append([])
            Xiang2[i][j].append([])
            Gonglv3[i][j].append([])
            Xiang3[i][j].append([])
            Gonglv4[i][j].append([])
            Xiang4[i][j].append([])
```

```
    f = open('1.txt',encoding='utf8')
    for i in f:
        i = i.replace('\n','').split(',')
        if i[3] == '1':
            Gonglv1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
            Xiang1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
        elif i[3] == '2':
            Gonglv2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
            Xiang2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
        elif i[3] == '3':
            Gonglv3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
            Xiang3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
        else:
            Gonglv4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
            Xiang4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]

    f.close()



    TestSA().test()


'''
'''


from time import time
from sopt.util.functions import *
from sopt.util.constraints import *
from sopt.util.random_walk_config import *
from sopt.Optimizers.RandomWalk import RandomWalk

class TestRandomWalk:
    def __init__(self):
        self.func = fun1
        self.func_type = 'min'
```

```
        self.variables_num = 14
        self.lower_bound                                                    =
np.array([0,0,0,0,0,0,0,0,-10000,-10000,-10000,0,0,0])
        self.upper_bound                                                    =
np.array([100,90,90,100,100,100,10,10,10000,10000,10000,180,360,10000])
        self.generations = 100
        self.init_step = 10
        self.eps = 1e-4
        self.vectors_num = 10
        self.init_pos = None
        # self.complex_constraints = [constraints1,constraints2,constraints3]
        self.complex_constraints = None
        self.complex_constraints_method = complex_constraints_method.loop
        self.RandomWalk = RandomWalk(**self.__dict__)

    def test(self):
        start_time = time()
        self.RandomWalk.random_walk()
        print("random walk costs %.4f seconds!" %(time() - start_time))
        self.RandomWalk.save_plot()
        self.RandomWalk.show_result()


if __name__ == '__main__':

    Gonglv1=[]
    Gonglv2=[]
    Gonglv3=[]
    Gonglv4=[]

    Xiang1=[]
    Xiang2=[]
    Xiang3=[]
    Xiang4=[]


    for i in range(32):
        Gonglv1.append([])
        Xiang1.append([])
        Gonglv2.append([])
        Xiang2.append([])
        Gonglv3.append([])
        Xiang3.append([])
        Gonglv4.append([])
```

```
    Xiang4.append([])


for i in range(32):
    for j in range(73):
        Gonglv1[i].append([])
        Xiang1[i].append([])
        Gonglv2[i].append([])
        Xiang2[i].append([])
        Gonglv3[i].append([])
        Xiang3[i].append([])
        Gonglv4[i].append([])
        Xiang4[i].append([])


for i in range(32):
    for j in range(73):
        for k in range(37):
            Gonglv1[i][j].append([])
            Xiang1[i][j].append([])
            Gonglv2[i][j].append([])
            Xiang2[i][j].append([])
            Gonglv3[i][j].append([])
            Xiang3[i][j].append([])
            Gonglv4[i][j].append([])
            Xiang4[i][j].append([])




f = open('1.txt',encoding='utf8')
for i in f:
    i = i.replace('\n','').split(',')
    if i[3] == '1':
        Gonglv1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '2':
        Gonglv2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '3':
        Gonglv3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    else:
        Gonglv4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
```

```
            Xiang4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]

    f.close()
    TestRandomWalk().test()

'''
from time import time


from sopt.GA.GA import GA
from sopt.util.functions import *
from sopt.util.ga_config import *
from sopt.util.constraints import *

class TestGA:
    def __init__(self):
        self.func = fun1
        self.func_type = 'min'
        self.variables_num = 14
        self.lower_bound                                                    =
np.array([0,0,0,0,0,0,0,0,-1000,-1000,-1000,0,0,0])
        self.upper_bound                                                    =
np.array([50,90,90,50,50,50,5,5,1000,1000,1000,180,360,1000])
        self.cross_rate = 0.8
        self.mutation_rate = 0.1
        self.generations = 10
        self.population_size = 20
        self.binary_code_length = 20
        self.cross_rate_exp = 1
        self.mutation_rate_exp = 1
        self.code_type = code_type.binary
        self.cross_code = False
        self.select_method = select_method.keep_best
        self.rank_select_probs = None
        self.tournament_num = 2
        self.cross_method = cross_method.uniform
        self.arithmetic_cross_alpha = 0.1
        self.arithmetic_cross_exp = 1
        self.mutation_method = mutation_method.uniform
        self.none_uniform_mutation_rate = 1
        #self.complex_constraints = [constraints1,constraints2,constraints3]
        self.complex_constraints = None
        self.complex_constraints_method = complex_constraints_method.penalty
        self.complex_constraints_C = 1e6
```

```
        self.M = 1e8
        self.GA = GA(**self.__dict__)

    def test(self):
        start_time = time()
        self.GA.run()
        print("GA costs %.4f seconds!" % (time()-start_time))
        self.GA.save_plot()
        self.GA.show_result()



if __name__ == '__main__':
    Gonglv1=[]
    Gonglv2=[]
    Gonglv3=[]
    Gonglv4=[]

    Xiang1=[]
    Xiang2=[]
    Xiang3=[]
    Xiang4=[]


    for i in range(32):
        Gonglv1.append([])
        Xiang1.append([])
        Gonglv2.append([])
        Xiang2.append([])
        Gonglv3.append([])
        Xiang3.append([])
        Gonglv4.append([])
        Xiang4.append([])


    for i in range(32):
        for j in range(73):
            Gonglv1[i].append([])
            Xiang1[i].append([])
            Gonglv2[i].append([])
            Xiang2[i].append([])
            Gonglv3[i].append([])
            Xiang3[i].append([])
            Gonglv4[i].append([])
```

```
        Xiang4[i].append([])


for i in range(32):
    for j in range(73):
        for k in range(37):
            Gonglv1[i][j].append([])
            Xiang1[i][j].append([])
            Gonglv2[i][j].append([])
            Xiang2[i][j].append([])
            Gonglv3[i][j].append([])
            Xiang3[i][j].append([])
            Gonglv4[i][j].append([])
            Xiang4[i][j].append([])









f = open('1.txt',encoding='utf8')
for i in f:
    i = i.replace('\n','').split(',')
    if i[3] == '1':
        Gonglv1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang1[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '2':
        Gonglv2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang2[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    elif i[3] == '3':
        Gonglv3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang3[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]
    else:
        Gonglv4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[4]
        Xiang4[int(i[2])-1][int(i[0])-1][int(i[1])-1]=i[5]

f.close()


'''
for i in range(32):
    if (i != 3) and (i != 19):
```

```
            print(i+1)
            a = np.array(Gonglv1[i],dtype='float64')
            TestGA().test()



    a = np.array(Gonglv1[0],dtype='float64')
    TestGA().test()

    '''
    a = np.array(Gonglv1[21],dtype='float64')
    TestGA().test()
```

预测总接入量和预测2028年人均上网时间

```
from pandas import DataFrame
from pandas import Series
from pandas import concat
from pandas import read_csv
from pandas import datetime
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from math import sqrt
from matplotlib import pyplot
import numpy

# 用于加载数据集的日期——时间解析函数
def parser(x):
    return datetime.strptime('19'+x, '%Y-%m')

# 把序列建构成一个监督学习问题
def timeseries_to_supervised(data, lag=1):
    df = DataFrame(data)
    columns = [df.shift(i) for i in range(1, lag+1)]
    columns.append(df)
    df = concat(columns, axis=1)
    df.fillna(0, inplace=True)
    return df

# 创建一个差分化序列
def difference(dataset, interval=1):
    diff = list()
```

```
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return Series(diff)

# 反向转换差分化的值
def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]

# 将测试集的数据缩放到[-1,1]范围内
def scale(train, test):
    # fit scale
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaler = scaler.fit(train)
    # transform train
    train = train.reshape(train.shape[0], train.shape[1])
    train_scaled = scaler.transform(train)
    # transform test
    test = test.reshape(test.shape[0], test.shape[1])
    test_scaled = scaler.transform(test)
    return scaler, train_scaled, test_scaled

# 反缩放预测值
def invert_scale(scaler, X, value):
    new_row = [x for x in X] + [value]
    array = numpy.array(new_row)
    array = array.reshape(1, len(array))
    inverted = scaler.inverse_transform(array)
    return inverted[0, -1]

# 将一个LSTM模型拟合到训练数据集上
def fit_lstm(train, batch_size, nb_epoch, neurons):
    X, y = train[:, 0:-1], train[:, -1]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    model = Sequential()
    model.add(LSTM(neurons,    batch_input_shape=(batch_size,    X.shape[1],
X.shape[2]), stateful=True))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
    for i in range(nb_epoch):
        model.fit(X,    y,    epochs=1,    batch_size=batch_size,    verbose=0,
shuffle=False)
        model.reset_states()
    return model
```

```
# 做出一步预测
def forecast_lstm(model, batch_size, X):
    X = X.reshape(1, 1, len(X))
    yhat = model.predict(X, batch_size=batch_size)
    return yhat[0,0]

# 加载数据集
series = read_csv('shampoo-sales.csv')

# 把数据变得平稳
raw_values = series.values
diff_values = difference(raw_values, 1)




# 转换数据变成监督学习问题
supervised = timeseries_to_supervised(diff_values, 1)
supervised_values = supervised.values

# 把数据分成训练数据集和测试数据集
train, test = supervised_values[0:-5], supervised_values[-5:]

# 缩放数据
scaler, train_scaled, test_scaled = scale(train, test)

# 拟合模型
lstm_model = fit_lstm(train_scaled, 1, 3000, 4)
# 预测训练集
train_reshaped = train_scaled[:, 0].reshape(len(train_scaled), 1, 1)
lstm_model.predict(train_reshaped, batch_size=1)

# 在测试数据集上的前向验证
predictions = list()
for i in range(len(test_scaled)):
    # 做出一步预测
    X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
    yhat = forecast_lstm(lstm_model, 1, X)
    # 反向缩放
    yhat = invert_scale(scaler, X, yhat)
    # 反向转换差分化数据
    yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
    # 存储预测值
```

```
    predictions.append(yhat)
    expected = raw_values[len(train) + i + 1]
    print('Month=%d, Predicted=%f, Expected=%f' % (i+1, yhat, expected))

# 报告模型性能
rmse = sqrt(mean_squared_error(raw_values[-5:], predictions))
print('Test RMSE: %.3f' % rmse)
# 绘制观测值和预测值对比的折线图
pyplot.plot(raw_values[-5:])
pyplot.plot(predictions)
pyplot.show()
```