“华为杯”第十五届中国研究生数学建模竞赛

“华为杯”第十五届中国研究生
数学建模竞赛

题 目 **光传送网建模与价值评估**

摘 要

在通信行业迅速发展的时代，光传送网为网络中的底层。本文首先对光传输链路模型进行了建模，研究了不同调制方案的 BER 与 SNR 关系曲线；其次对因网络的连接给两地带来的价值进行评估，并建立了相应模型，通过算法使网络价值最大；最后通过算法改进调制方案，提高了通信系统的容忍噪声能力。本文所做的光传送网络建模与价值评估，对现实中核心网建设提供了一定参考依据，并能进一步加快中国 5G 的通信建设。

针对问题一，在子问题-1 中，对于三种调制格式（QPSK、8QAM、16QAM）编码方案的纠前误码率（BER）与信噪比（SNR）的关系曲线，通过建立基本**数字通信传输模型**，了解了二进制序列传输过程中误码率的产生原理。同时，利用星座点噪声的公式定义，确定工程意义上的信噪比公式。采用 MATLAB 实现数字传输模型的建立，最终得到三种调制格式方案的 BER-SNR 关系曲线图，并得出当 BER=0.02 时，三种调制格式（QPSK、8QAM、16QAM）SNR 的容限点分别为 6.35、10.9 和 13.29。在子问题-2 中，在前一问题中得出的数字传输模型基础上，**引入放大器噪声和光纤噪声**。通过分析单跨信号衰减的幅度，由给定的噪声公式得出叠加的噪声，进而得出噪声与传输距离的关系曲线图。由他们的关系可得，单跨为 80km 时，QPSK 调制格式最远传输距离为 9600km；8QAM 调制格式最远传输距离为 4800km；16QAM 调制格式最远传输距离为 4000km。单跨为 100km 时，QPSK 调制格式最远传输距离为 10000km；8QAM 调制格式最远传输距离为 5000km；16QAM 调制格式最远传输距离为 4000km。

针对问题二，子问题-1 和子问题-2 均属于单目标优化问题，子问题-3 属于多目标优化问题，三个子问题均采用**遗传算法**求解。**在子问题-1 中**，无需考虑中间节点的问题，在 12 个城市点均连接在网络上的前提下，给定网络中的连接数量，求解当网络价值最大时，城市点的规划。采用 **Dijkstra 算法**的遍历思想，当网络连接线遍历每个城市点时，进行下一个步骤，否则采用**惩罚手段**使此时的适应度函数值为 0。最终得出：n=16 最大网络价值为 4068.4214mTb/s，约迭代 50 次左右网络价值趋于最优；n=33 最大网络价值为 6854.7631mTb/s，约迭代 150 次左右网络价值趋于最优。**在子问题-2 中**，需要**考虑中间节点和设备容量**问题，在不超过每个城市设备点最大容量的前提下，求解最大价值网络。同样采用遗传

算法，得出：n=16 最大网络价值为 2731.5505mTb/s，约迭代 250 次左右时网络价值趋于最优；n=33 最大网络价值为 7318.3339mTb/s，约迭代 40 次左右时的网络价值趋于最优。当城市点由市扩大到省时，关键是人口数量发生了改变，此时得出的结果为：n=16 最大网络价值为 2731.5505mTb/s，约迭代 250 次左右时网络价值趋于最优；n=33 最大网络价值为 7318.3339mTb/s，约迭代 40 次左右时的网络价值趋于最优。**在子问题-3 中**，若要满足连接发达城市以获得更高的收入，则用 **GDP 衡量每条路径的权重**，从而使网络价值随权重变化而变化；若要保障发展相对滞后的地区的通信，则使网络规划路线中**网孔的个数尽可能多**。此时结果为：n=16 最大网络价值为 2359.2138mTb/s，约迭代 450 次左右时网络价值趋于最优；n=33 最大网络价值为：4351.0578mTb/s，约迭代 240 次左右时，网络价值波动虽然没有稳定在一定的数值，但是波动情况较小，即将趋于稳定。

针对问题三，对于保持调制格式的信息熵为 3bit，我们分析信息熵只与星座点的数量和每个点的概率有关。我们通过将星座点的数量减少为 8 个，且令每个点的概率相同，实现了这一目标。对于降低 SNR 容限点的目标，我们考虑改变星座点的位置，采用遗传算法，根据通信原理的相关知识，设置适应度函数选出使相邻星座点的**欧式距离较大**的星座图，逐步迭代求得 SNR 容限点小于 8QAM 的星座图。问题三的最优星座图呈现心形，如正文中所示。

关键字：遗传算法、适应度函数、Dijkstra 算法、光传送链路模型

目 录

一、问题重述

1.1 问题背景

光纤通信通信系统是一种以光为传输载波，以光纤作为传输媒介，通过一系列电光、光电变换来传输信息的通信系统。光纤以其大容量、远距离传送等特性，以及光在传输过程中速率高、介质可靠性高、业务透明等诸多优点，光纤通信自诞生以来已经成为信息时代中主要通信技术之一。

光纤通信从诞生至今，50 多年里基于数字光纤通信技术的光传送网构建起了全球通信的骨架。从城市内一家一户间的传输，到城市与城市之间的传输，甚至于跨越大洋的传输和跨越国界的传输都有光纤通信的踪迹。光传送网为人类提供了大容量、高可靠性和低能耗的信息传输管道。光纤通信技术的崛起，光传送网的不断扩大，使得相关研究者们逐渐将注意力集中到了光传送网的规划和建设。

光传送网是整个光通信的承载网络，所有基于光通信的各类业务网络都依靠光传送网提供数据传送服务，因此传送网网络性能直接关系到上层各类业务网络的通信质量。光传送网的合理规划和建设将从根本上影响传送网的传送性能，改善网络质量。同时，随着人类对通信容量、通信质量追求的不断扩大，光传送网的规划逐渐成为光通信领域研究的重要方向之一。各大运营商、设备制造商、政府相关部门及研究机构都将光传送网的规划与建纳入改善网络质量的重要考虑对象。

光纤通信相比于其它通信方式有自己独有的特性，因此在对光传送网进行规划时，不能照搬其它通信传送网方式。光通信最大的特点之一是，光在传输过程中会逐渐衰减。在长距离传输中，为克服光的这一特性，经常需要在一段距离点将光信号放大后再传输。也因为光的这一特性，使得在相同技术条件下信号传输容量会随着传输距离增加而减小。对于光传送网的规划者而言，需要充分考虑到光通信的各种特性，从而提升网络质量。

就目前的研究来看，为使传送网的价值最大化，光传送网规划者往往需要考虑传输容量、传输距离、网络拓扑等多种因素的影响。优化光传送网、规划光传送线路已然是光纤通信领域研究热点之一。

1.2 需解决的问题

本课题中，为研究光传送网的规划，需从底层物理出发建立光传送链路的模型；基于网络价值与光传送链路模型，制定光传送网的规划；改进调制格式以改善传送网性能，延长链路总长度。故需要解决以下三方面的内容：

（1）问题-1：光传送链路建模

现代数字传输系统就是对 0101 二进制序列进行编码并传输的系统，一个二进制的 0 或 1 称为 1 个比特。众所周知，无论是语音消息、视频消息还是各种其它类型的消息，都可以转化为一串像“0101…”这种数字型的二进制比特序列。传统的通信方式中，该序列在发送端由调制器编码调制成某个“载体信号”，通过信道传送到达目的地，接收端通过解调器解调载体信号为发送端的二进制序列。图 1.1 是上述过程的简化模型。

光纤通信同样也沿用该模型，传输信道就是光纤，信息的载体是光波。在该模型中，信道中出现噪声是不可避免的，一旦有噪声干扰信号传输，就可能导致接受端最终接受的二进制序列与发送端发送的二进制序列不一致。显然这是任何

通信中都不愿看到的，因此通信系统抗干扰能力也成为重要的衡量指标之一。

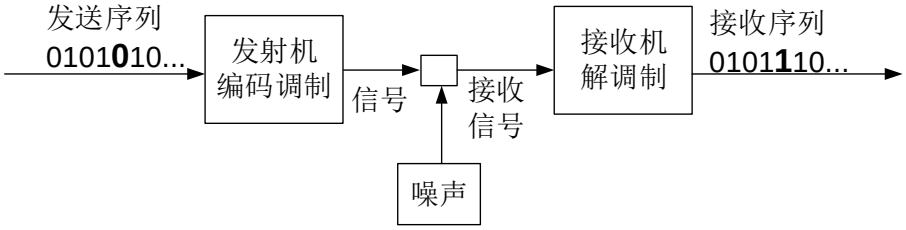


图 1.1 简化后的数字传输模型

在信号传输中，二进制序列通常不是一个个传输，而是以块为单位传输。假定二进制序列将 K 个比特作为一个“符号”传输，那么该符号中就有 2^K 个不同状态。光纤通信中，通常采用光波的复振幅来承载传输信号。因此，复平面上不同的点正好可以描述二进制序列“符号”的不同状态，复平面上的点称为“星座点”，这些星座点构成的图为“星座图”。图 1.2(a)是发送端的四个点构成的星座图，这些信号经过 QPSK（Quadrature Phase Shift Keying）调制，信道叠加噪声和接收机处理后，接收端的星座图不再是理想的四个点，而是相对扩散的点。之所以会这样，是因为当接收机接收到一个符号时，由于噪声的影响，会将这个符号判定为离这个符号最近的星座点。显然，存在噪声且对符号影响足够大，就会导致接收到的符号可能被判错到其它地方，这样就产生了误码。图 1.2(b)是接受端接受到发送端的星座图，其中蓝色的点就是一个误码。BER 是衡量通信系统性能的最根本指标，采用纠错编码，只要纠前 BER 小于 BER 容限点，纠错编码后就能实现纠后误码率为零的传输。

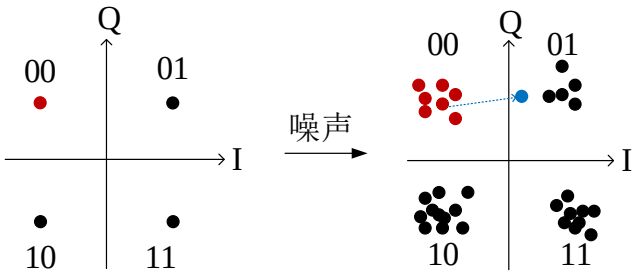


图 1.2 星座图与噪声导致误码的示意图

图 1.3 中理想星座点用s_k表示，接收到的符号用r_k表示，则噪声为：

$$n_k = r_k - s_k \tag{1.1}$$

噪声通常服从均值为 0 的正态分布。噪声的方差等于噪声的平均功率，定义为：

$$P_{\mathrm{n}} = \frac{1}{N}\sum_{k=1}^{N}|n_k|^2 \tag{1.2}$$

其中 N 为总共传输的符号数。信号平均功率定义为发送符号绝对值平方的均值：

$$P_s = \frac{1}{N}\sum_{k=1}^{N}|s_k|^2 \tag{1.3}$$

定义信号和噪声功率的比值为信噪比（Signal-to-Noise Ratio, SNR）：

$$SNR = P_{\mathrm{s}}/P_{\mathrm{n}} \tag{1.4}$$

工程上通常用 dB 作为 SNR 的单位，定义为：

$$SNR(\text{dB}) = 10log10(P_s/P_n) \quad (1.5)$$

增大十倍为加 10dB，减小 0.5 倍为减去 3dB。本题中功率单位统一为毫瓦（mW），星座图实部和虚部单位为$\sqrt{\text{mW}}$。

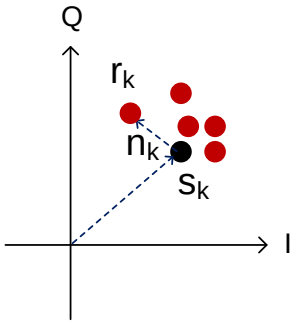


图 1.3 信号和噪声的相关定义示意图

图 1.4 为基本的光传输链路模型，在远距离的信号传输中，光传输链路由多个相同的跨段级联形成，其中跨段为一段距离的光纤和一个放大器。信号通过光纤传输时，每传输 15km，其功率就会衰减一半。因此，信号在光纤上每传输一段距离，就需要利用放大器对信号进行光功率补偿。同时，由于光纤传输过程中会不可避免的引入噪声，信号和噪声经过放大器同时被放大，就会使放大器本身也产生一个自发辐射噪声，其公式为：

$$P_n = 2\pi hfB(NF + 1/Gain) \quad (1.6)$$

其中是h是普朗克常数（$6.62606896 \times 10^{-34}$ J·S），f是光波频率（可定为193.1THz），B 为带宽（设为 50GHz），NF 为噪声指数（可设为 4），Gain 为补偿光纤衰减所对应的功率增益。

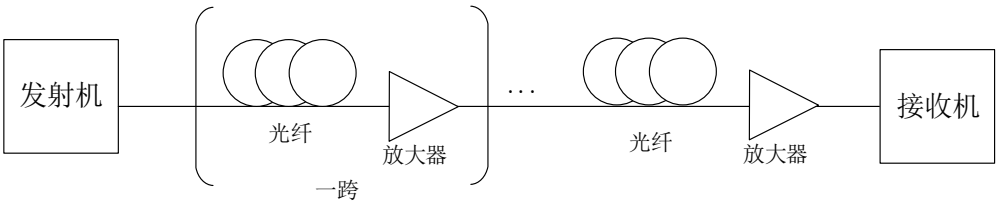


图 1.4 基本的光传输链路模型

另一方面，由于光纤是一种传输介质，本身传输过程中就会产生损耗，在这里表现的方式就是光纤的非线性效应，被等效为上文传输过程中的噪声。其等效噪声功率与光纤功率近似呈平方关系，光纤功率为 1mW 时的非线性噪声约等于单个放大器噪声的 2/3。

子问题-1）纠前误码率与信噪比计算

星座图的编码分布模式也称为调制格式，对于给定的调制格式，BER 和 SNR 呈一一对应的关系，纠前 BER 门限对应的 SNR 记做“SNR 容限点”。给出图 1.5 中所示的三种调制格式及编码方式（相邻星座点距离相等），每个符号等概率出现，分别称为 QPSK，8QAM (Quadrature Amplitude Modulation, QAM)，16QAM。请给出 BER 与 SNR 的关系曲线，BER=0.02 时 SNR 容限点分别为多少？

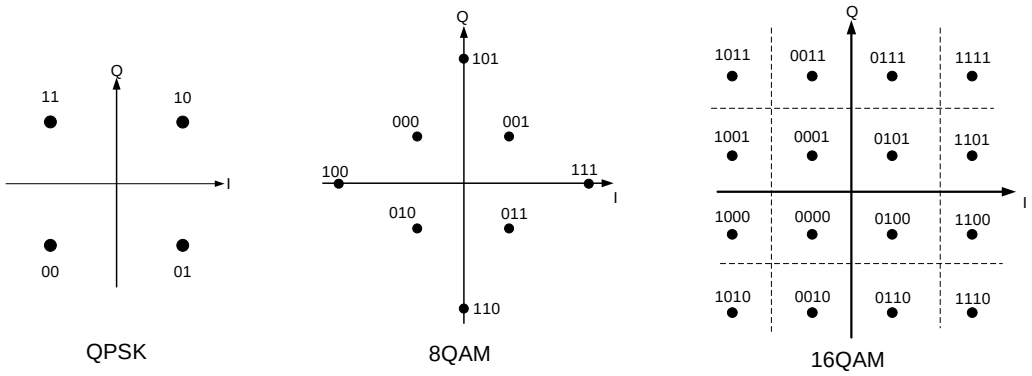


图 1.5 三种调制格式的编码方案

子问题-2）光链路性能计算

当单跨传输距离为 80km 和 100km 两种情况，以纠前误码率 0.02 为门限，图 5 给出的传输格式最远的传输距离（每跨距离×跨段数量）是多少？

（2）问题-2 光传送网规划

如表 1 所示，是在更深层次优化升级后，三种不同传输格式的传输距离。而现代通信网的建立，就是为了使更多的人能够通过网络更好地连接在一起，故我们定义如下方式的网络的价值：

1）每条直接连接两个城市/区域的链路当做 1 个连接，每个连接的价值定义为传输的容量与连接区域人口数的乘积（取两区域人口数乘积的 0.5 次方）

2）网络的价值则是所有连接价值的加权和

$$\text{网络价值} = \sum \text{权重} * \text{容量} * \text{人口} \tag{1.6}$$

表 1 不同传输格式的传输距离

单波传输容量	最大传输距离	总容量
100 Gb/s	3000 km	8 Tb/s
200 Gb/s	1200 km	16 Tb/s
400 Gb/s	600 km	32 Tb/s

如图 1.6 所示为北京、上海、南京的相对位置示意图，以此为例，若三个城市之间均互相连接，根据两城市之间的距离得出传输容量。假设每条连接链路的权重为 1，通过人口算出网络价值（Network Value, NV）为：

$$\text{NV} = \sqrt{21 \times 24}\text{m} \times 16\text{Tb/s} + \sqrt{21 \times 8}\text{m} \times 16\text{Tb/s} + \sqrt{24 \times 8} \times 32\text{Tb/s} \approx 1010\text{mTb/s} \tag{1.7}$$

其中 m 代表百万人(million)，$\text{Tb/s} = 10^{12}\text{bit/s}$, 该网络的连接数为 3。

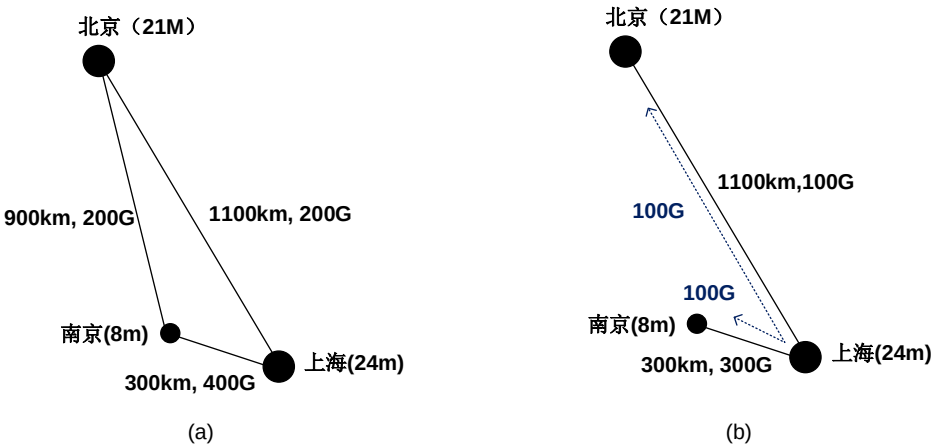


图 1.6 三个节点网络示意图

实际情况往往会略有差异，诸如资源等影响因素，城市网络中并不是所有网络节点都能直接连接。图 1.6(b)即表示不能直接连接的两个节点需要辅助节点建立连接。北京和南京为不能直接的两个节点，需要通过上海建立连接，此时该网络的连接数为 2。当北京上海之间的传输仅保留一半容量(100Gb/s)，而另一半容量用于南京到北京的信号传输(100Gb/s)，相应地南京与上海之间的直接传输容量也会降低至 300Gb/s，此时网络的价值为：

$$\mathrm{NV} = \sqrt{21 \times 24}\mathrm{m} \times 8\mathrm{Tb/s} + \sqrt{21 \times 8}\mathrm{m} \times 8\mathrm{Tb/s} + \sqrt{24 \times 8} \times 24\mathrm{Tb/s} \approx 616\mathrm{mTb/s} \quad (1.8)$$

根据需要两个节点之间也可以有多个连接，考虑网络价值和需求为图 1.7 中的我国城市群制定光传送网规划，图中共有 12 个区域(其中北京/天津，深圳/广州均按 1 个区域对待)

图 1.7 需要考虑的城市群

子问题-1:如果连接数从 16 增加到 33 条时，不考虑中间节点，给出你们的两个网络规划及其价值。网络价值最多是多少？

子问题-2：存在中间节点，且两个节点之间可以有多个连接的情况下，重新解决子问题-1 并给出所有中间节点传输容量的分配，假定每条链路容量可任意分配，只要总容量不超过表 1 的规定。如果由市扩大为省（区）影响如何？（人口请从网上查找）

子问题-3:光传送网络价值有多个侧面，例如从运营商的角度，连接经济发达的地区会带来更多的收入，从政府的角度保障发展相对滞后地区的通信是均衡发展的要求等。你队认为制定光传送网络规划的目标函数应该是什么？前面制定的规划有无变化？

问题-3 改善星座图

由第一问可知，纠前 BER 不变时，降低 SNR 容限点可以提高系统容忍噪声的能力，从而延长链路的总长度。请尝试任意改变16QAM方案中星座点的位置、数量或每个点的概率，探索产生比图 5 中 8QAM（相邻各星座点之间距离相等）具有更低 SNR 容限点的调制方案？调制格式的信息熵需保持为 3bit。

信息熵定义为：

$$\Omega = -\sum_1^{\mathrm{N}} p_k log_2(p_k) \tag{1.9}$$

其中p_k为每个符号状态出现的概率，N 为状态数。图 1.5 所示的等概率情况下，QPSK、8QAM 和 16QAM 的信息熵分别为 2bit, 3bit 和 4bit。

二、模型假设

为了方便建模，作出如下一些简化假设：

1、假设传播过程中，只存在光纤的非线性效应等效噪声和放大器的自发辐射噪声，忽略其他损耗的影响；

2、在光传输链路的每一跨中，放大器对光功率的补偿，假设能将其补偿到单位“1”；

3、假设问题一中发送 10000 个符号对模型进行仿真；

4、假设在问题二中前两小问不考虑连接线路权重的影响，令其为 1；

5、假设在求解不同规划的网络价值时，前提保证每个城市点都在网络中；

6、在问题二的子问题-3 中，假设两城市点的连接线的 GDP 值为两城市点 GDP 总数乘积的 0.5 次方。

三、符号说明

表 2 符号说明

符号	意义
$\mathbf{NV_{ij}}$	i、j 两地连接线的网络价值
$\boldsymbol{Cap_{ij}}$	i、j 两地连接线的容量
NF	噪声指数
B	带宽
n	城市点的个数
N	城市点两两相连的总连接数
$\boldsymbol{Num_i}$	i 地的人口数量
$\mathbf{NV_{Total}}$	整个网络的价值
$\boldsymbol{W_{ij}}$	i、j 两地连接线的权重
$\mathbf{P}(\boldsymbol{x_i})$	个体x_i被遗传到下一代种群中的概率
$\boldsymbol{q_i}$	个体x_i的累积概率
$\boldsymbol{Eco_i}$	i 地的经济发展权重

四、问题一：光传送链路建模

4.1 子问题-1：纠前信噪比与误码率的计算

4.1.1 问题分析

在光传送过程，信号以光纤为载体，从发送端传输到接收端，在此过程中计入噪声对信号影响所产生的损耗，再通过放大器对损耗的光功率进行补偿。但在信号传输过程中，由于噪声的影响，使信号不再是以前的理想状态而出现偏差，因此本题探讨误码率（BER）与信噪比（SNR）在三种不同的调制格式下的关系曲线。在求解 SNR 容限点时，结合上一步得出的关系曲线图，当 BER=0.02 时，得到各个调制格式下的 SNR 容限点即为所求。

4.1.2 问题求解

在光纤通信[1]中，其链路传输基本模型如图 4.1 所示。二进制序列先通过相位转化，将“符号”表现在星座图上，便于观察，初始时刻是理想的星座图，经过噪声的影响后，重新将这个序列逆映射在星座图上。通过比较前后两个星座图中星座点的位置，得出需要计算的误码率。

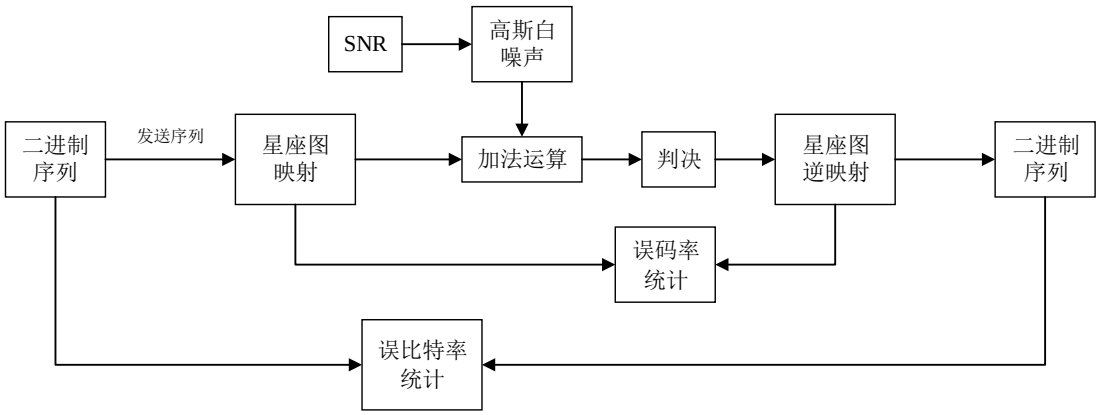
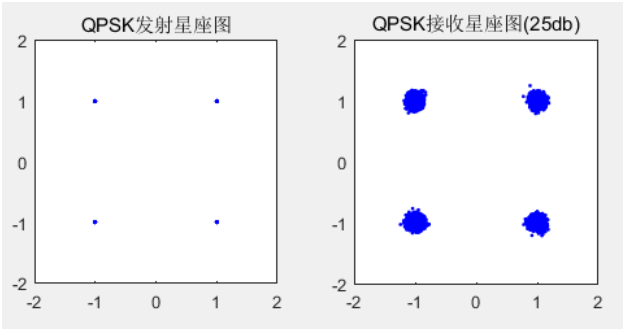
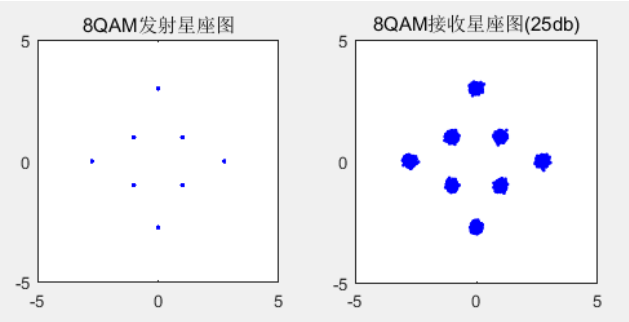


图 4.1 传输链路模型

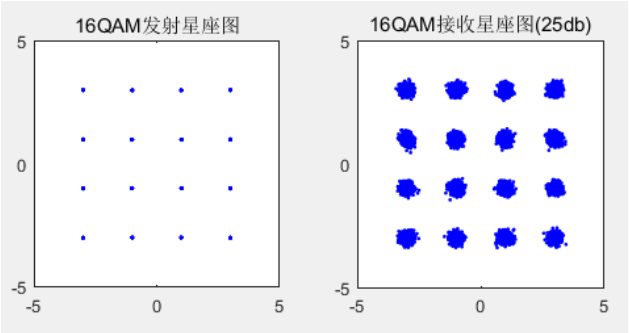
在第一小问中，假设信噪比（SNR）在 1-25（dB）中变化。在 MATLAB 中生成三种调制格式的理想星座图，每种序列经过调制器变换后，再计入链路传输中噪声的影响（加入信噪比为 25 的噪声），得到如图 4.2 所示的可能存在误码率的星座图。



(a) QPSK 星座图



(b) 8QAM 星座图



(c) 16QAM 星座图

图 4.2 三种调制格式的星座图

在讨论 QPSK、8QAM、16QAM 的误码率（BER）与信噪比(SNR)之间的关系时[2]，首要任务是如何判断计入噪声影响的符号是否属于误码。根据题意，在 QPSK 中，当某一象限的符号出现在其它象限，就属于误码情况；在 8QAM 中，当经过噪声影响后的符号出现在原始信号一定范围之外的范围，属于误码情况；16QAM 与 8QAM 同理。

在接收设备受到某个信号后，解码设备对其进行解码，由于通信链路上有噪声干扰，解码设备对接收到信号的解码需要一定的容错，我队对接收到信号的星座图解码算法如下：

1) 计算接收到的符号与编码方案中星座图中每个星座点的距离；
2) 找到所有距离中，最小的一个距离；
3) 查找该最小距离是与哪个星座点的距离；
4) 确定该接收到符号即为该星座点编码；
5) 按该星座点对该符号进行解码。

另一方面，根据题中所给的噪声服从均值为 0 的正态分布，此时加入的噪声属于高斯白噪声。根据题中给定的调制解调方式，并在传输链路中加入高斯白噪声，结合上一步得出的星座图，通过 MATLAB 编程可得出三种调制方式 BER 与 SNR 的关系曲线，如图 4.3 所示，其具体代码见附录。

由图 4.3 可得，在 BER=0.02 这一条件下，当调制格式为 QPSK 时，SNR 容限点为 6.35；当调制格式为 8QAM 时，SNR 容限点为 10.9；当调制格式为 16QAM 时，SNR 容限点为 13.29。

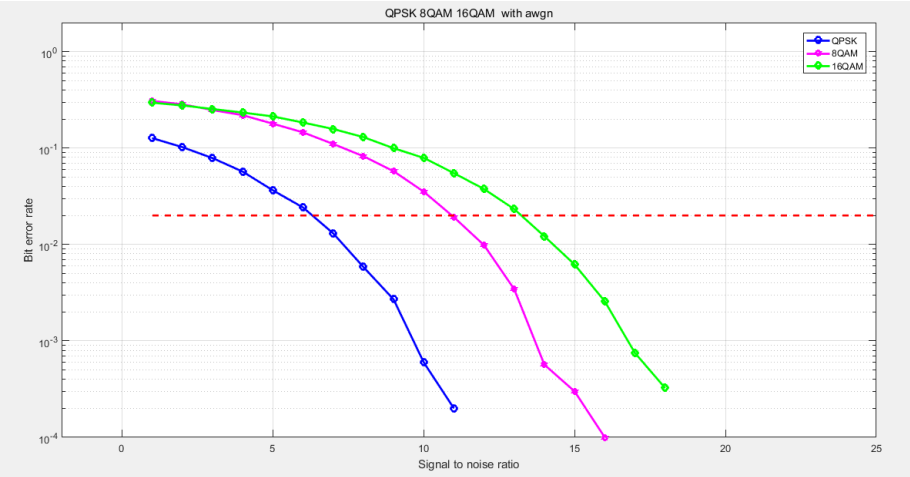


图 4.3 三种调制格式 BER 与 SNR 的关系曲线

4.2 子问题-2：光链路性能计算

4.2.1 问题分析

本小题要求考虑单跨传输距离为 80km 和 100km 两种情况下，以纠前误码率（BER）0.02 为门限，计算给定的三种调制格式[3]（QPSK、8QAM、16QAM）最远的传输距离。由于放大器噪声和非线性噪声会逐跨叠加，导致信噪比（SNR）降低，故这一问的本质是求解：经过一段距离的光纤传输之后，达到纠前 BER 为 0.02 为门限时对应的 SNR 容限点时的传输距离，此距离就是最远的传输距离。

4.2.2 问题求解

光传输链路是由多个相同跨段级联而成，几十千米的光纤与一个放大器构成 1 个跨段。为了直观地了解光传输过程，首先做出基本的光纤传输链路模型，如图 4.5 所示。可以看到，在光纤传输过程中，噪声来自光纤本身的非线性效应引入的等效噪声和放大器的自发辐射噪声。其中，前者在每跨开头叠加，后者在每跨末位叠加。每跨中间信噪比保持不变，每跨叠加的噪声间相互独立。

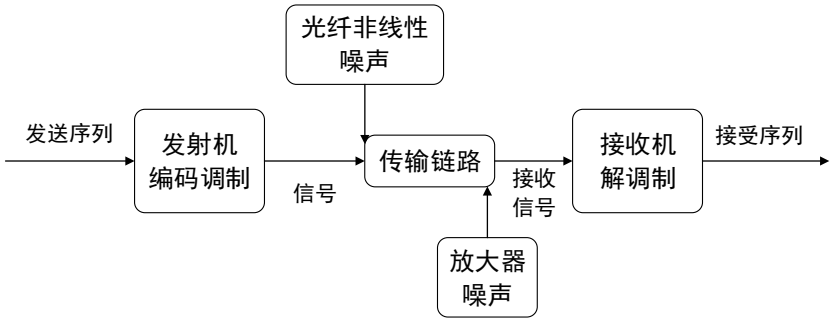


图 4.4 光纤传输链路模型

光纤传输模型[4]中，放大器噪声的公式为：

$$P_n = 2\pi hfB(NF + 1/Gain) \tag{4.1}$$

其中是$\boldsymbol{h}$是普朗克常数（$6.62606896 \times 10^{-34}\,\mathrm{J \cdot S}$），$\boldsymbol{f}$是光波频率（可定为

193.1THz），B 为带宽（设为 50GHz），NF 为噪声指数（可设为 4），Gain 为补偿光纤衰减所对应的功率增益。另一方面，光纤本身非线性噪声功率与光纤功率近似呈平方关系[5]，光纤功率为 1mW 时的非线性噪声约等于单个放大器噪声的 2/3。

假设噪声功率与光纤功率的关系式为 $y=kx^2$，当 $x=1$ 时，$y=2/3*P_n=k*1^2$，则 $k=\frac{2}{3}P_n$。故光纤传输过程中本身的非线性效应等效噪声：

$$P_{非}=\frac{2}{3}P_n \tag{4.2}$$

本文将问题分为两个部分进行分析，具体求解过程如下：

（1）当单跨传输距离为 80km 时各调制格式的最大传输距离为：

步骤 1：在第一小问的基础上，将传输链路模型中的高斯白噪声替换为光纤的非线性等效噪声和放大器噪声。光纤的非线性等效噪声在每跨开头叠加，放大器噪声在每跨末尾叠加；

步骤 2：由于每隔 15km，光功率衰减一半；故当单跨传输距离为 80km 时，光功率衰减为原来的 $\left(\frac{1}{2}\right)^{\frac{16}{3}}$ 倍；

步骤 3：在 MATLAB 代码中设置循环语句，重复光功率衰减与补偿的过程。在此过程中，叠加放大器的噪声 P_n 和光纤噪声 $\frac{2}{3}P_n$；

步骤 4：当 SNR 数值达到 BER=0.02 门限所对应的 SNR 容限点时，代码循环结束。此时得到跨段数量与 BER 的关系曲线图，如图 4.5 所示。

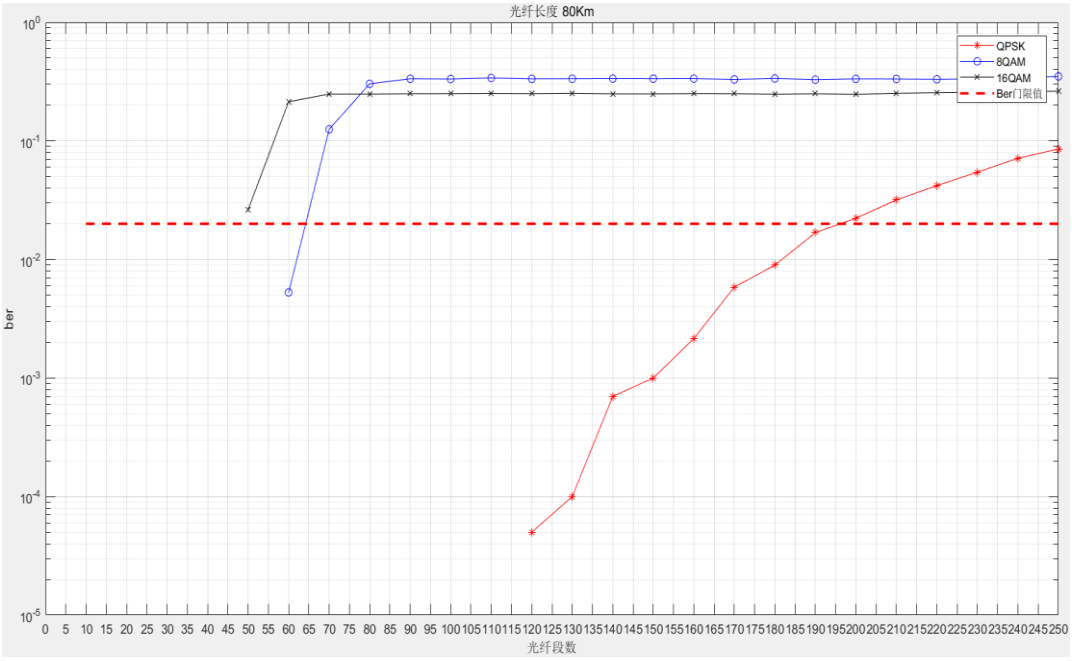


图 4.5 跨段数量与 BER 的关系曲线图（单跨为 80km）

（2）当单跨传输距离为 100km 时各调制格式的最大传输距离

此问题与 1 相比，仅仅改变了距离因素，所以在（1）的解题步骤基础上，

改变步骤 3 中光的衰减率，由$\left(\frac{1}{2}\right)^{\frac{16}{3}}$倍变为$\left(\frac{1}{2}\right)^{\frac{20}{3}}$倍。因此放大器的噪声$P_n$和光纤噪声$\frac{2}{3}P_n$也随之增加。得到跨段数量与 BER 的关系曲线图，如图 4.6 所示。

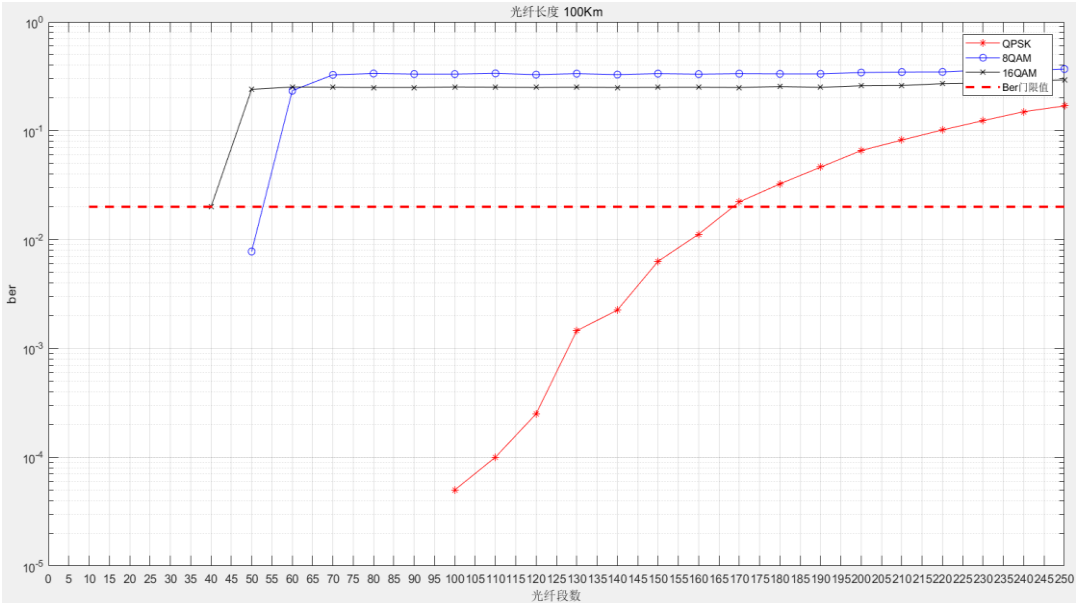


图 4.6 跨段数量与 BER 的关系曲线图（单跨为 100km）

4.2.3 结果分析

由图 4.5 可知，单跨为 80km 时，QPSK 调制格式最远传输距离为 9600km；8QAM 调制格式最远传输距离为$60\times80=4800km$；16QAM 调制格式最远传输距离为$50\times80=4000km$。

由图 4.6 可知，单跨为 100km 时，QPSK 调制格式最远传输距离为 10000km；8QAM 调制格式最远传输距离为$50\times100=5000km$；16QAM 调制格式最远传输距离为$40\times100=4000km$。

五、问题二：光传送网规划

5.1 子问题-1：单目标网络规划及其价值

5.1.1 问题分析

子问题-1 是在城市连接点分别为 16 和 33，且不考虑城市中间节点的情况下，求解网络价值最多时，城市点的网络规划，并求此时网络的最大价值。在本小题中，根据题中式 1.6 可知，网络价值与每条连接的权重有关，此问的求解与连接的权重无很大关系，因此在本题求解中，假设每条连接的权重为 1。综上所述，本小问**关键问题是对每条线路容量的计算**。

根据题意及参考文献，本题为**典型的单目标优化问题**，因此我队采用**遗传算法**[6]来规划网络中城市的连接。

5.1.2 数据处理

在网上对题中给定的城市点进行搜索，得出各个城市点的经纬度及人口如表 3 所示。

表 3 城市经纬度及人口数量

城市	经度（°）	纬度（°）	人口（百万）
哈尔滨	126.53	45.80	9.9526
北京&天津	116.46	39.92	34.8245
郑州	113.66	34.75	9.03
上海	121.47	31.23	23.8043
武汉	114.3	30.60	10.12
广州&深圳	113.27	23.13	23.3863
西安	108.93	34.27	8.8321
乌鲁木齐	87.62	43.82	3.80
拉萨	91.11	29.97	0.9025
成都	104.7	30.67	16.0447
昆明	102.72	25.05	7.2131
重庆（主城区）	106.55	29.57	8.6506

来源：经纬度在线查询、国家统计局

利用数据画出城市的二维坐标图，如图 5.1 所示：

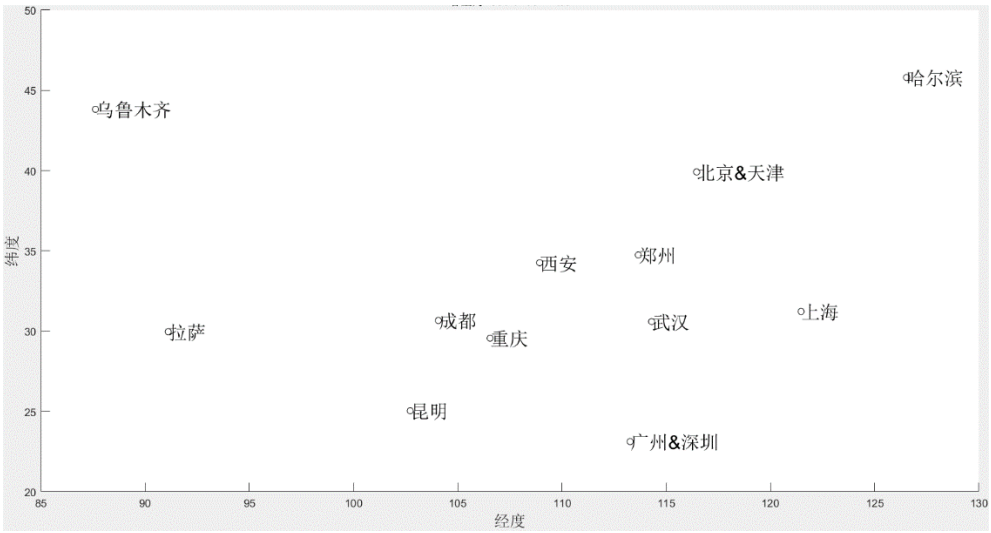


图 5.1 城市点位置图

5.1.3 模型建立

1、 遗传算法简介：

遗传算法（Genetic Algorithm）是 1975 年 Michigan 大学的 J.Holland 教授提出的一种算法，这种算法是基于达尔文的生物进化论中“适者生存，优胜劣汰”和孟德尔遗传学的理论基础，模拟生物进化过程提出来的一种计算模型，并利用随机搜索来求取问题的最优解。在遗传算法中，将二进制或自然数等简单编码方式来比拟遗传学中的染色体，模拟其选择、交叉、变异等生物进化机制来解决复杂的优化问题，其程序流程图如图 5.2 所示。

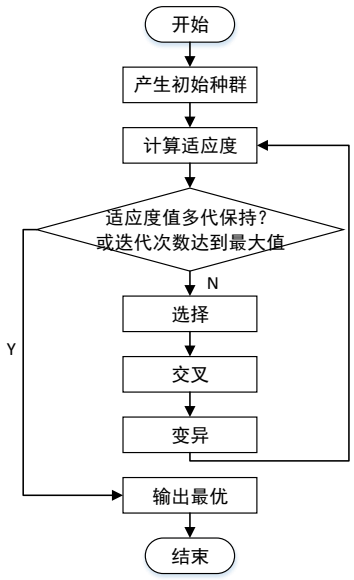


图 5.2 遗传算法流程图

以下为遗传算法中需要重点考虑的问题：

(1) 编码策略

生物中的遗传性状是由每个生物的遗传基因决定的，在使用遗传算法求解优化问题时，把优化问题中每一个可行解通过浮点编码或者二进制编码，将其转化为基因串，成为遗传问题中的一个个体。针对不同问题有不同的编码、解码方式，一般的数学优化问题通常将个体编码成二进制的形式。

(2) 初始种群

遗传算法中的初始种群是由若干个遗传个体组成，其中一个个体代表优化问题的可行解。遗传算法主要是对种群进行操作，因此需要根据问题的可行解，随机生成若干个个体，组成初始种群的数据。

(3) 适应度函数

遗传算法中的适应度函数通常是目标函数，它体现了遗传个体对环境的适应程度，适应程度高的个体就能继续在种群中生存，这充分体现了达尔文“优胜劣汰”的物种进化思想。

(4) 遗传算子

遗传算法的进化是基于遗传算子共同作用而实现的。遗传算子有选择（Selection）、交叉（Crossover）、变异（Mutation）三种，这三种遗传算子相互作用于初始种群中，使其产生下一代种群，推动种群的进化。

对于选择算子，是按照遗传个体的适应度值的大小有选择的将一些优秀个体直接放入下一代种群中，当然也可以忽略这一步骤直接进入交叉和变异这两个步骤。常用的选择算法有：轮盘赌选择法、随机联赛选择法和最佳个体保留选择法。

对于交叉算子，是将种群中两个随机的遗传个体按照一定的规则交换部分基因，从而形成新的两个个体，常用的交叉方法有：单点交叉、双点交叉、多点交叉和均匀交叉等。

对于变异算子，这一算子类似于生物遗传中的基因突变：给种群中某一个体的一个或者多个基因小概率的扰动，使其成为一个新的个体的过程。这种算子也可以保持种群的多样性，进而增强遗传算法的全局能力。

根据遗传算法的算法流程图以及算法中关键因子的确定可得遗传算法的伪代码实现如下：

算法：遗传算法

输入：随机种群
输出：适应度较好的个体

```
1 Begin
2    I=0;//进化种群代数
3    Initialize P(I);//初始化种群
4    Fitness P(I);// “适者生存”遗传选择
5    While(not Terminate-condition) //不满足终止条件时循环
6    {
7        Fitness P(I);//“适者生存”遗传选择
8        GA-Operation P(I);//遗传算字交叉 or 变异
9        I++;//下次循环
10   }
11 End
```

2、 模型建立

问题二的第一小问属于单目标优化问题，给定城市点的位置以及城市之间的连接线的数量，来求解能够使网络价值最大时，该网络的城市点应该如何连接。对于单目标问题，可以采用蚁群算法、粒子群算法、模拟退火算法、遗传算法等，但是前三种算法对初始种群的依赖较大，若初始种群选择不恰当，就可能导致算法得出的结果不是较理想的。对于遗传算法，虽然可能陷入局部最优，但遗传算法是从群体出发进行搜索，最终收敛到个体，且个体之间可以多个同时比较，具有潜在的并行性和较高的全局性等优点。另一方面，要保证所有城市点均连接在所求网络中，可将此问题当做是一个小的 **TSP 问题**，采用 Dijkstra 算法，利用遍历的思想解决这一问题。

本题模型的具体框架如图 5.3 所示，以**遗传算法**作为模型的基本框架，在适应度函数中嵌入 **Dijkstra 算法**，其作用是：判断通过遗传算法得出网络规划路线是否满足遍历每个点，若是，则适应度函数保持不变；否则，采用**惩罚手段**使此时的适应度函数为 0。用此手段处理后的网络规划不会迭代到下一代种群中，还可以保证每一个城市点连接在网络连线图中。

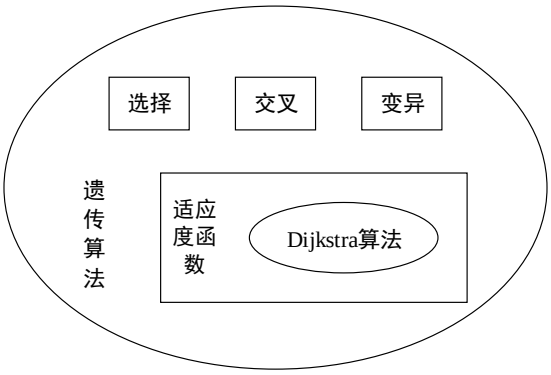


图 5.3 算法框架图

在本题中，遗传算法模型[7]建立如下：

(1) 染色体编码

12 个城市点两两相连，由公式 5.1 可得总连接数为 66 条。

$$\mathrm{N}=\frac{n\times(n-1)}{2} \tag{5.1}$$

其中，n 表示城市点的个数，N 表示 n 个城市两两相连的总连接数。

利用遗传算法解决光传输线路规划问题时，采用的编码方式是：

1) 对 66 条连接链路进行 1-66 的编号；

2) 将编好号的 66 个数随机地放入一个个体中的 66 个染色体空位中，形成一个种群中的个体，若假设初始种群中个体数量为 50，则余下的 49 个个体采用同样的方法随机生成。

(2) 初始种群

初始种群的产生依赖于随机函数，在该模型中一个个体是由 66 个线路基因组成，而这 66 基因是采用链路序号进行编码随机生成的。

(3) 适应度函数

本题中适应度函数就是目标函数，根据题中的定义，每个连接线价值为：

$$\mathrm{NV}_{ij}=Cap_{ij}\times\sqrt{Num_i\times Num_j} \tag{5.2}$$

其中，NV_{ij}表示 i、j 两地连接线的网络价值（Network Value），Cap_{ij}表示 i、

j 两地连接线的容量，Num_i、Num_j分别表示 i、j 两地的人口数量。

整个网络的目标函数就是所有连接线价值的加权和：

$$MAX\ \mathrm{NV_{Total}} = \sum W_{ij} * \mathrm{NV}_{ij}\ ,\ (\mathrm{i,j} = 1,2,\dots,12) \tag{5.3}$$

其中，W_{ij}表示 i、j 两地连接线的权重。

(4) 遗传算子

对于选择算子，在该问中，采用的是**轮盘赌选择法**，其基本思想为：种群中每个个体被选中的概率与其适应度大小成正比。其基本步骤为：

1) 计算 M 个种群中每个个体的适应度，在这里即是计算每个个体中被选中的 16（或 33）条连接线路的总的网络价值；

2) 计算出每个个体被遗传到下一代种群中的概率（利用第一步求得的适应度来求解）：

$$\mathrm{P}(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_i)} \tag{5.4}$$

3) 计算每个个体的累积概率

$$q_i = \sum_{j=1}^{i} P(x_j) \tag{5.5}$$

其中，q_i表示个体x_i的累积概率。

4) 在区间[0,1]中产生随机数，若这个随机数刚好落在累加概率的某个体的区域内，则选择该区域的个体进入下一代；

5) 一直运行直到个体的数目达到初始种群的数目。

对于交叉和变异算子，在本题中是对产生的初始种群而言，随机选择两个个体，将个体中的基因进行交叉和变异操作，直到生成够一个新的种群为止。

Dijkstra 算法步骤如下：

1) 选择城市 i 作为起点，再选择另外一个城市 j 作为终点，采用贪心法的算法策略，运用 Dijkstra 算法以起始点为中心向外层层扩散，直到扩散到终点；

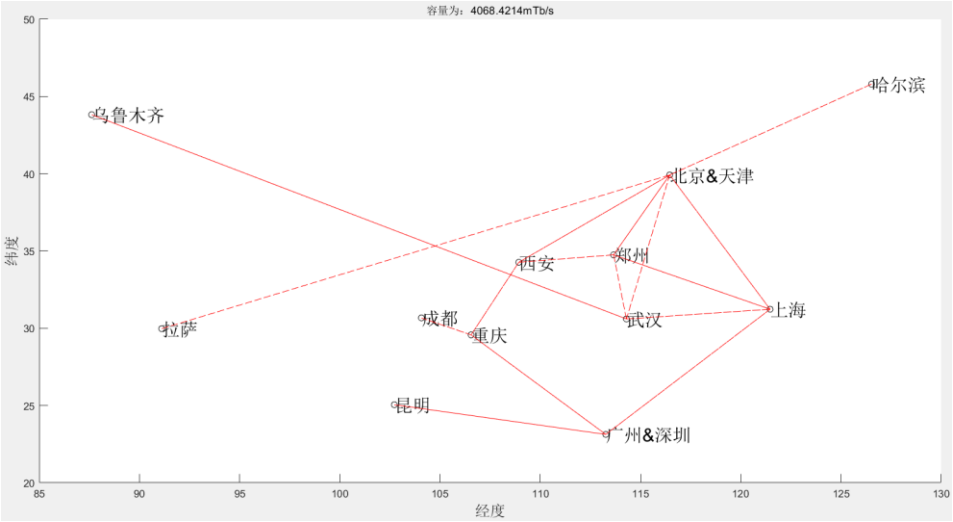
2) 此次遍历得到一个最短路径，也即最大的网络价值（城市距离与链路价值呈负相关）；

3) 更换终点城市，判断得到的遍历路径是否在遗传算法得出的路径范围中，若是，跳出循环；否则继续；

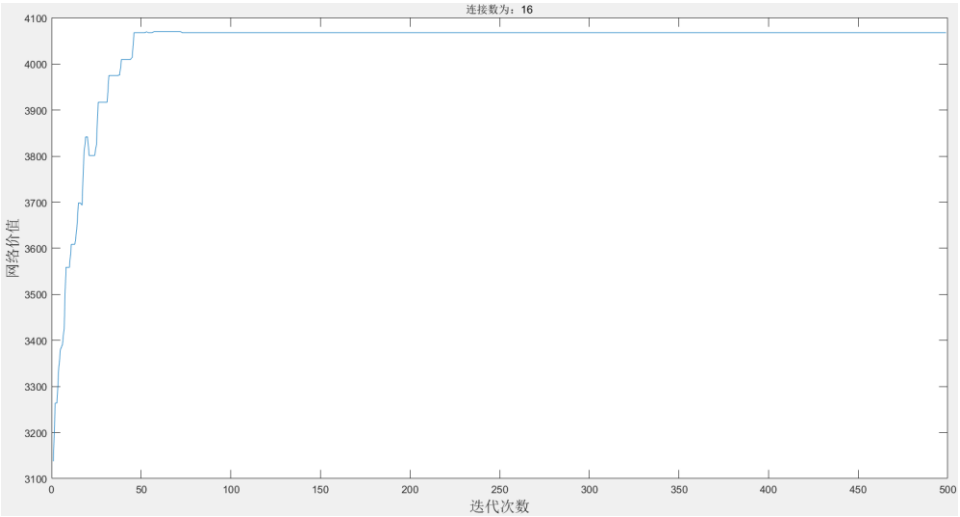
4) 更换起点城市，重复步骤 1-3。

5.1.4 模型求解

根据网络价值的模型，可以在 MATLAB 中实现对光传送网的规划。当城市点连接数 n=16 时，其规划具体路线如图 5.3(a)所示，此时最大网络价值为：4068.4214mTb/s；而随着迭代次数的增加，城市网络价值的变化曲线如图 5.3(b)所示，迭代次数在 50 次左右时，网络价值趋于最优。



(a) 网络线路规划图



(b) 迭代次数与网络价值的关系

图 5.3 n=16 时网络规划

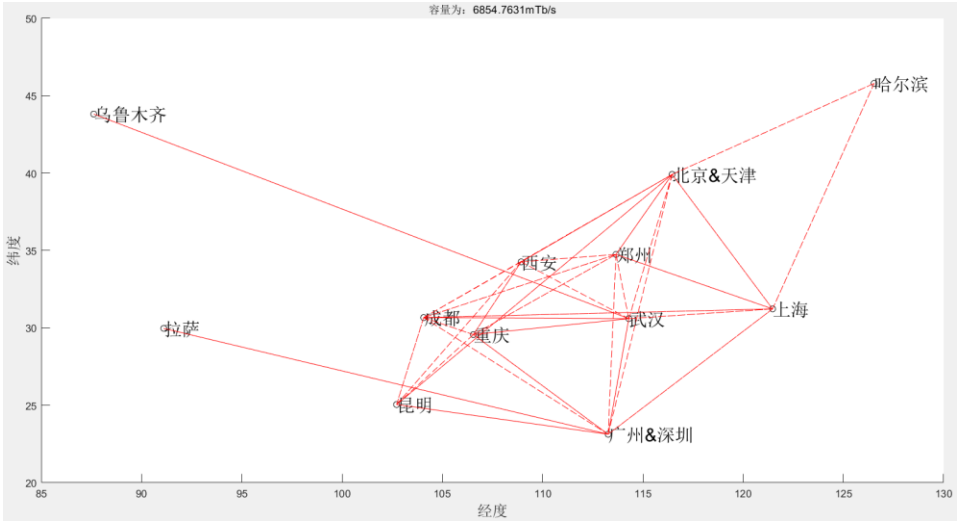
同时，根据定义的变量得出各个城市点的邻接矩阵，如表 4 所示。结合表格的横纵数值对比分析，1 代表城市点间有连接，0 代表城市点间没有连接，可以分析出网络规划图的路线。(城市点编号：1-哈尔滨，2-北京&天津，3-郑州，4-上海，5-武汉，6-广州&深圳，7-西安，8-乌鲁木齐，9-拉萨，10-成都，11-昆明，12-重庆)

表 4 n=16 各城市点的邻接矩阵

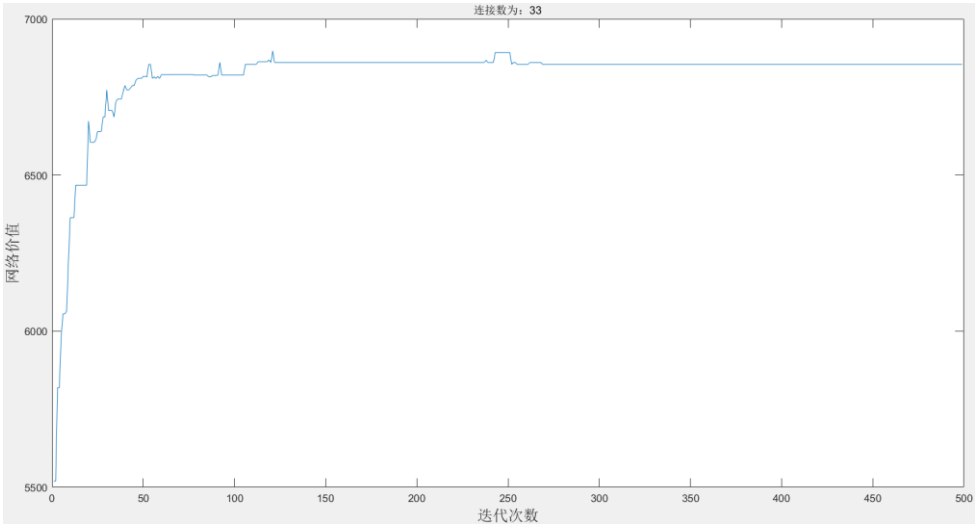
	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	0	0	0	0	0	0	0	0	0
2	1	0	1	1	1	0	1	0	1	0	0	0
3	0	1	0	1	1	0	1	0	0	0	0	0
4	0	1	1	0	1	1	0	0	0	0	0	0
5	0	1	1	1	0	0	0	1	0	0	0	0

6	0	0	0	1	0	0	0	0	0	0	1	1
7	0	1	1	0	0	0	0	0	0	0	0	1
8	0	0	0	0	1	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	1	0	0	0	0	0	0
12	0	0	0	0	0	1	1	0	0	1	0	0

当城市点连接数 n=33 时，其规划具体路线如图 5.4(a)所示，此时最大网络价值为：6854.7631mTb/s。而随着迭代次数的增加，城市网络价值的变化曲线如图 5.3(b)所示，迭代次数在 150 次左右时的网络价值波动虽然没有稳定在一定的数值，但是波动情况较小，即将趋于稳定。



(a) 网络线路规划图



(b) 迭代次数与网络价值的关系

图 5.4 n=33 时网络规划

同理，根据定义的变量得出各个城市点的邻接矩阵，如表 5 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。

表 5 n=33 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	0	1	0	0	0	0	0	0	0	0
2	1	0	1	1	1	1	1	0	0	1	1	0
3	0	1	0	1	1	1	1	0	0	1	0	1
4	1	1	1	0	1	1	0	0	0	1	0	0
5	0	1	1	1	0	1	1	1	0	1	0	1
6	0	1	1	1	1	0	0	0	1	1	1	1
7	0	1	1	0	1	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0	0	0
10	0	1	1	1	1	1	1	0	0	0	1	1
11	0	1	0	0	0	1	1	0	0	1	0	1
12	0	0	1	0	1	1	1	0	0	1	1	0

5.2 子问题-2：多约束单目标网络规划及其价值

5.2.1 问题分析

子问题-2 在 1 的基础上，要多考虑代表城市的两个节点之间，存在可以多个连接的情况，即节点 1 要到节点 3，可以先经过节点 2 再转到节点 3，而不是直接到达。在这种情况下，就需要考虑节点处设备容量的问题。因此，在这子问题-2 中，需要解决的是在不超过每个城市设备点最大容量的前提下，使光传送网络的价值达到最大，并分别考虑连接数为 16 和 33 时网络的规划情况及容量分配。另一方面，当城市点由市扩大为省时，相较于市不同的是省包含的人口数量更大，从而会使网络中的价值更高。因此参考各个省的人口数量，用同样的方法可以得到新的网络规划图及容量分配。

5.2.2 数据处理

当城市点由市扩大到省（区）时，主要的变化就是区域内人口的变化，通过百度百科的统计，可以得到各个省（区）的人口数量如表 6 所示。

表 6 各个省（区）人口的数量

省（区）	人口（百万）	省（区）	人口（百万）
黑龙江	37.99	陕西	38.13
北京&天津	37.35	新疆	23.98
河南	95.32	西藏	3.31
上海	24.20	四川	82.62
湖北	58.85	云南	47.71
广东	109.99	重庆	30.48

来源：经纬度在线查询、国家统计局

5.2.3 模型建立

考虑中间节点的连接情况， 将中间节点涉及到的链接标记为特殊链接，故染色体编码将会增加一倍。此外，每个节点网络的单波传输容量是有限制的，构成选择下一代的约束条件。利用遗传算法解决光传输线路规划问题时，模型建立如下：

(1) 染色体编码

12 个城市点两两相连，且有可能存在中间节点，此时单个个体基因总数为 276 个，基因段的编码可分为三个部分，如图 5.5 所示。

1) 对 66 条连接链路进行 1-66 的编号，将 66 个编号随机放入线路编码中的 66 位置中；

2) 假设最高重复度为 4，第二段的编码的 66 位表示 1-66 条线路对应的重复度，随机生成 0-4 的整数放入基因位中，就表示一个基因位的线路重复的次数；

3) 12 条线路有 122 个容量，随机生成 0-1 的数，再经过一定的处理，可以表示每个城市点的容量分配以及每条线路的速率分配。

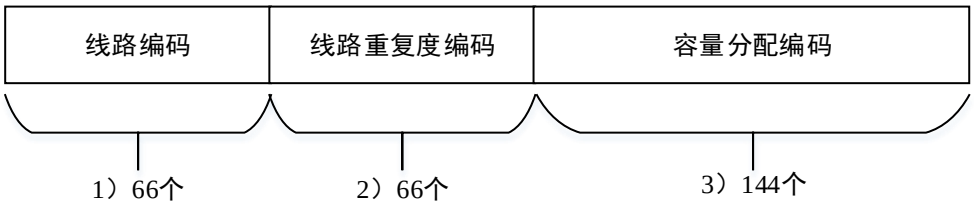


图 5.5 基因段编码分段

(2) 初始种群

初始种群的产生依赖于随机函数，在该模型中一个个体是由 276 个基因组成。假设该模型中初始种群的数量为 100，则剩下的 99 个个体用同样的方式生成。

(3) 适应度函数

增加中间节点后的适应度函数比较复杂，在原基础求解网络价值的基础上，要加上关于中间节点的约束条件，此时的适应度函数基本分为四个步骤：

1) 利用 Dijkstra 算法对种群进行筛选，使满足所有城市点遍历的个体进入下一步；

2) 若连接数为 16，累加基因编码中的第二部分，直到累加和大于 16 时停止。根据线路连接，若非两点直连，则寻找最近的中间节点，使这两点实现连接；

3) 根据两点之间的距离分配线路速率以及城市点的设备容量

4) 利用公式（5.3）计算此时网络的价值。

(4) 遗传算子

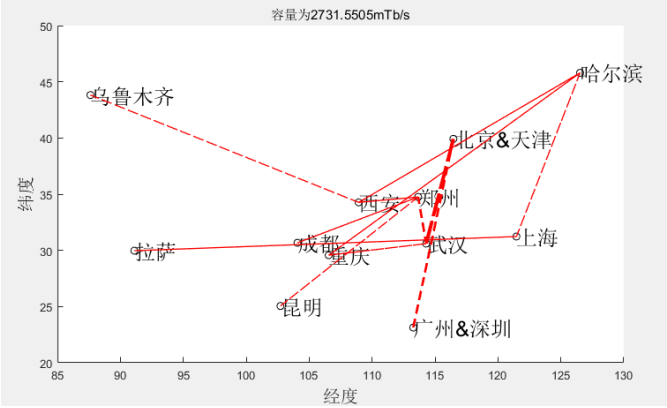
对于选择算子，同样采用轮盘赌选择法；

对于交叉算子，由于每个个体的基因数量较多，因此采用多点交叉的方法使三个对应的编码区域实现对应的交叉；

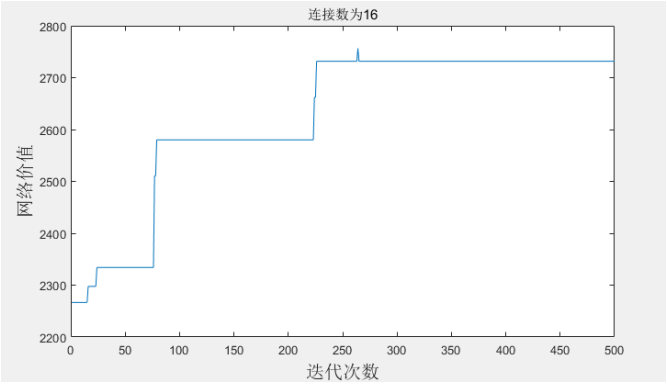
对于变异算子，同样要使三个区域内的基因均可以发生变异。

5.2.4 模型求解

根据网络价值的模型，可以在 MATLAB 中实现对光传送网的规划。当城市点连接数 n=16 时，其规划具体路线如图 5.6(a)所示，此时最大网络价值为：2731.5505mTb/s；而随着迭代次数的增加，城市网络价值的变化曲线如图 5.6(b)所示，迭代次数在 250 次左右时，网络价值趋于最优。



(a) 网络线路规划图



(b) 迭代次数与网络价值的关系

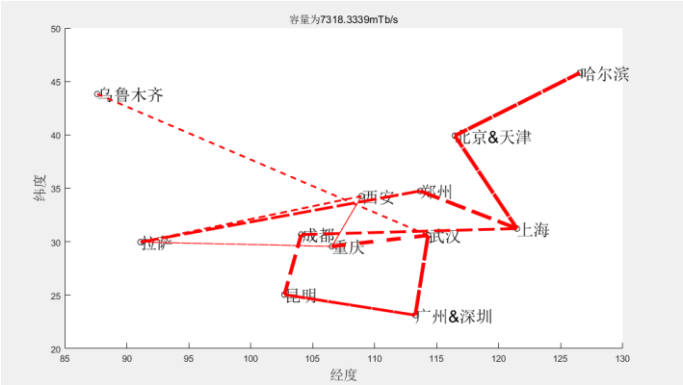
图 5.6 n=16 时网络规划

根据定义的变量得出各个城市点的邻接矩阵，如表 7 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。

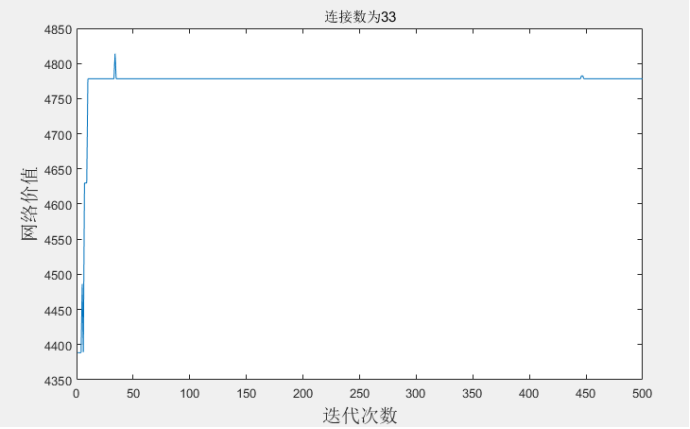
表 7　n=16 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	0	0	1	0	0	0	0	1
2	0	0	0	0	3	2	0	0	0	0	0	0
3	0	0	0	0	2	0	1	0	0	1	1	0
4	1	0	0	0	0	0	0	0	1	0	0	0
5	0	3	2	0	0	0	0	0	0	0	0	1
6	0	2	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0
12	1	0	0	0	1	0	0	0	0	0	0	0

当城市点连接数 n=33 时，其规划具体路线如图 5.7(a)所示，此时最大网络价值为：7318.3339mTb/s。而随着迭代次数的增加，城市网络价值的变化曲线如图 5.7(b)所示，迭代次数在 40 次左右时的网络价值趋于最优。



a)　网络线路规划图



b)　迭代次数与网络价值的关系

图 5.7 n=33 时网络规划

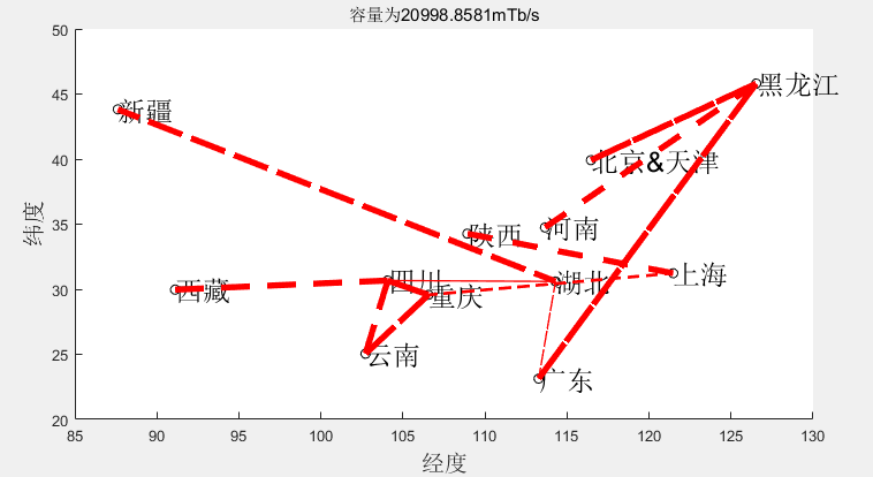
同理，根据定义的变量得出各个城市点的邻接矩阵，如表 8 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。表格中的“2”、“3”表示该

条线路是一个中间路段，该线路被利用了两次或三次。

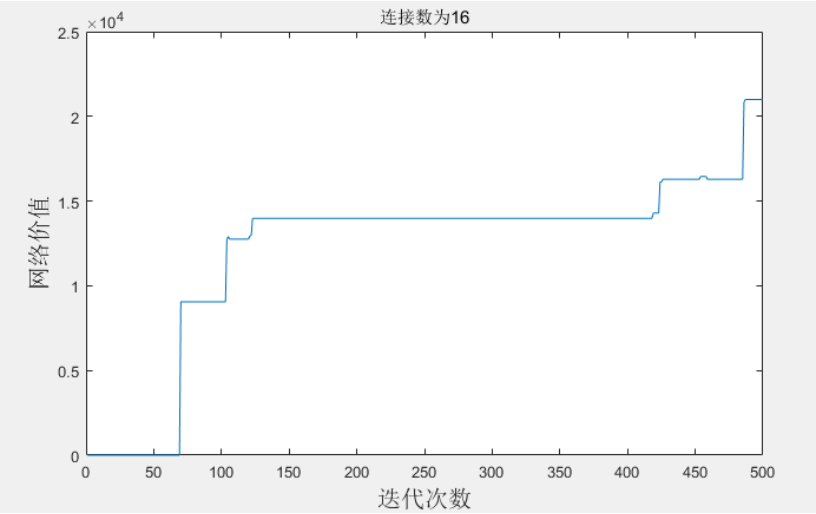
表 8　n=33 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	1	0	0	1	0	0	0	0	1
2	0	0	0	0	3	2	0	0	0	0	0	0
3	0	0	0	0	2	0	1	0	0	1	1	0
4	1	0	0	0	0	0	0	0	1	0	0	0
5	0	3	2	0	0	0	0	0	0	0	0	1
6	0	2	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	1	0	0	0	0	0	0	0	0	0
12	1	0	0	0	1	0	0	0	0	0	0	0

当城市点由市扩大到省（区）时，关键改变的是人口数量。当城市点连接数 n=16 时，其规划具体路线如图 5.8(a)所示，此时最大网络价值为：20998.8581mTb/s。而随着迭代次数的增加，城市网络价值的变化曲线如图 5.8(b)所示，迭代次数在 480 次左右时的网络价值趋于最优。



a)　网络线路规划图



b)　迭代次数与网络价值的关系

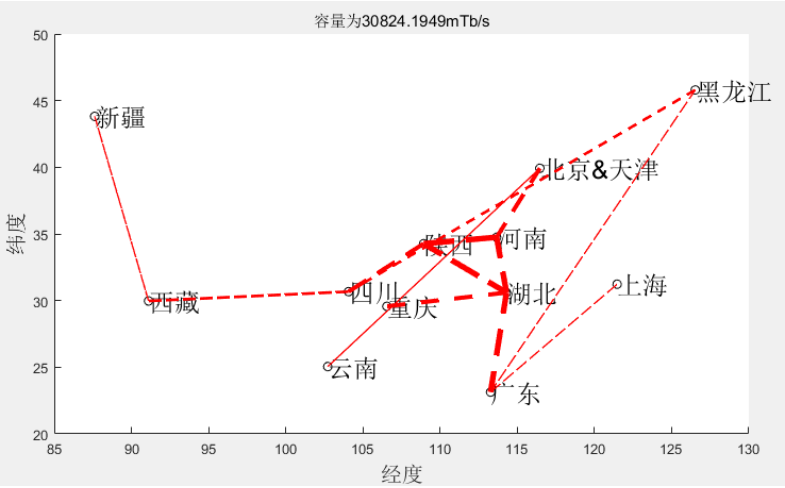
图 5.8 n=16 时网络规划

根据定义的变量得出各个城市点的邻接矩阵，如表 9 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。

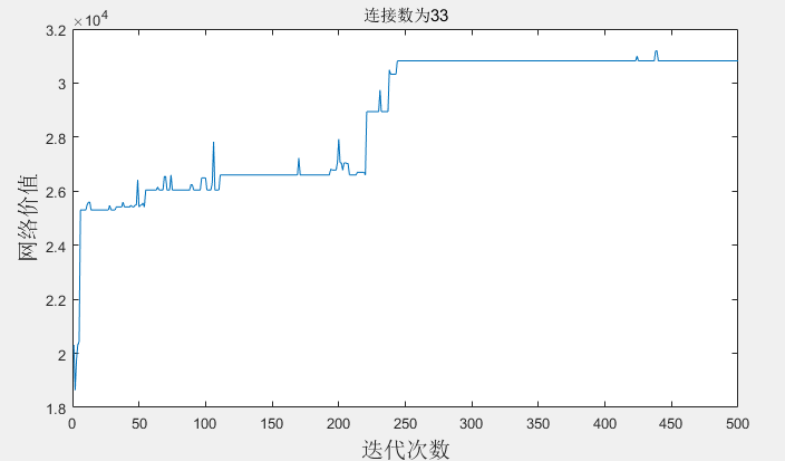
表 9　n=16 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	4	0	0	4	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	0	0	0	0	2
5	0	0	0	0	0	1	0	4	0	1	0	0
6	4	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	4	0	0	0	0	0	0	0	0
8	0	0	0	4	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	4	0	0
10	0	0	0	0	1	0	0	0	4	0	4	4
11	0	0	0	0	0	0	0	0	0	4	0	4
12	0	0	0	2	0	0	0	0	0	4	4	0

当城市点连接数 n=33 时，其规划具体路线如图 5.9(a)所示，此时最大网络价值为：20998.8581mTb/s。而随着迭代次数的增加，城市网络价值的变化曲线如图 5.9(b)所示，迭代次数在 480 次左右时的网络价值趋于最优。



a)　网络线路规划图



c)　迭代次数与网络价值的关系

图 5.9 n=33 时网络规划

根据定义的变量得出各个城市点的邻接矩阵，如表 10 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。

表 10　n=33 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	4	0	0	4	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	4	0	0	0	0	2
5	0	0	0	0	0	1	0	4	0	1	0	0
6	4	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	4	0	0	0	0	0	0	0	0
8	0	0	0	0	4	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	4	0	0
10	0	0	0	0	1	0	0	0	4	0	4	4
11	0	0	0	0	0	0	0	0	0	4	0	4
12	0	0	0	2	0	0	0	0	0	4	4	0

5.3 子问题-3：多约束多目标网络规划及其价值

5.3.1 问题分析

本问题要求考虑不同角度分析光传输网络价值，属于多目标优化问题。连接经济发达地区会带来更多的收入，连接发展相对滞后地区可以保障国家通信均衡发展。我队认为，从运营商角度考虑时，可以将城市的 GDP 来衡量其经济发展状况，从而产生不同的权重。我队选择 GDP 的理由是，某城市 GDP 根据**收入法**，其计算公式如下：

GDP=工资+利息+利润+租金+简介税和企业转移支付+折旧 (5.6)

这一等式计算的得来的，基本代表某城市经济状况，可以用做对不同城市经济发展状况的概括。从政府角度出发，为了使发展相对滞后地区的通信状况能够均衡发展，我队拟优先保证 12 个城市均连接在网络传输路线中。因此我队拟制定光传送网络规划的目标应该是保障国家通信均衡发展的前提下，再考虑如何取得更多的网络价值。故可以将本问题多目标优化问题分成两个单目标优化问题：**第一部分**是将保障国家通信均衡发展问题，从另一个角度思考，转化为在保证网络**连接网中网孔数量尽可能多**的情况下，求解网络价值最大；**第二部分**是在第一部分的基础上，结合第二问建立的模型，增加不同城市点之间**经济状况的得出的权值**，再使网络价值达到最大的网络规划。

5.3.2 模型建立

第一部分的问题：网络连接网中网孔数量的求解。该问题可以类比电路图中网孔数量的计算方法，得到网络图中网孔的数量。使下代网孔数量不减少，同时保证每个节点都被联通，可以解决题目中提到的保障发展相对滞后地区的通信是均衡发展的。

第二部分问题的解决前提是先确定地区经济发展程度。类比题中所定义的两城市的之间的连接线路人口的计算，我队拟假设两城市点连接线路的权重取两城市点 GDP 总数乘积的 0.5 次方，再通过归一化处理后作为该城市的经济发展权重，以此来衡量城市经济发展程度。例如：城市 i 和城市 j 的经济发展权重分别为Eco_i和Eco_j，则两座城市链路连接价值为：

$$NV_{ij} = \sqrt{Eco_i \times Eco_j} \times Cap_{ij} \times \sqrt{Num_i \times Num_j} \quad \mathrm{i,j} = 1,2,\dots,12 \tag{5.7}$$

因此，在前两问的基础上，结合此问中的权重，该问中光传送网络回话的目标函数为：

$$\mathrm{MAX}\ NV_{Total} = \sum NV_{ij} \tag{5.8}$$

从而网络目标函数是所有链路连接价值的加权和：

$$MAX\ NV_{\mathrm{Total}} = \sum NV_{ij}, \qquad (\mathrm{i,j} = 1,2,\dots,12) \tag{5.9}$$

5.3.3 模型求解

本题中求解出的城市链路权重如表 11 所示。

表 11 城市链路权重表

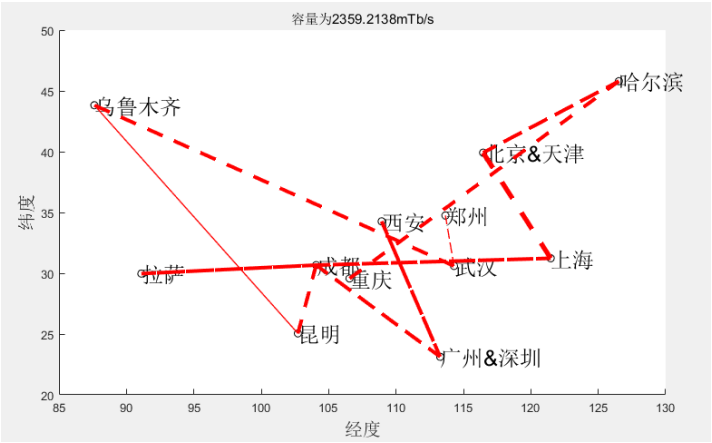
	北京 天津	哈尔 滨	上海	郑州	武汉	广州 深圳	重庆	西安	昆明	成都	拉萨	乌鲁 木齐

北京 天津		0.3766	0.8042	0.4396	0.5363	0.9694	0.6474	0.3933	0.3228	0.5460	0.1013	0.2451
哈尔 滨			0.3029	0.1655	0.2020	0.3651	0.2438	0.1481	0.1216	0.2056	0.0381	0.0923
上海				0.3535	0.4313	0.7796	0.5206	0.3163	0.2596	0.4391	0.0815	0.1971
郑州					0.2357	0.4261	0.2846	0.1729	0.1419	0.2400	0.0445	0.1077
武汉						0.5199	0.3472	0.2109	0.1731	0.2928	0.0543	0.1314
广州 深圳							0.6276	0.3813	0.3129	0.5293	0.0982	0.2376
重庆								0.2547	0.2090	0.3535	0.0656	0.1587
西安									0.1270	0.2148	0.0398	0.0964
昆明										0.1763	0.0327	0.0791
成都											0.0553	0.1338
拉萨												0.0248
乌鲁 木齐												

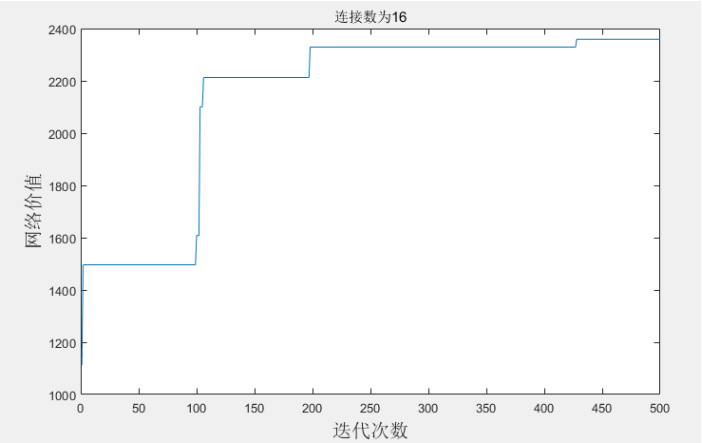
数据来源：国家统计局，经过归一处理完成

得到城市链路经济权重表之后，更新网络价值的公式。满足第一部分问题的网络连接图的约束条件，记录前一代个体的网孔数量，当当前代的网孔数量比前一代数量减少时，对该个体进行惩罚，使其适应度为 0，网孔数量相等或增加时进行保留。采用遗传算法，并应用该适应度函数后，得到最大的网络价值及网络连接图。

当城市点连接数 n=16 时，其规划具体路线如图 5.8(a)所示，此时最大网络价值为：2359.2138mTb/s；而随着迭代次数的增加，城市网络价值的变化曲线如图 5.8(b)所示，迭代次数在 450 次左右时，网络价值趋于最优。



(a) 网络线路规划图



(b) 迭代次数与网络价值的关系

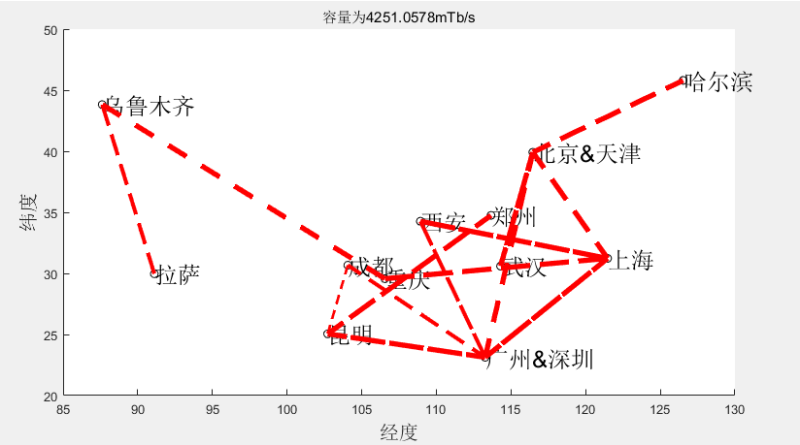
图 5.8 n=16 时网络规划

同理，根据定义的变量得出各个城市点的邻接矩阵，如表 12 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。表格中的“2”、“3”表示该条线路是一个中间路段，该线路被利用了两次或三次。

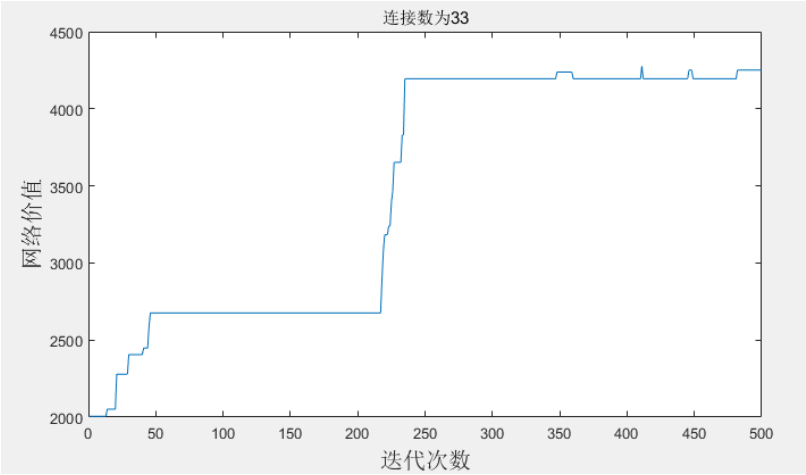
表 12　n=16 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	3	0	0	0	0	0	0	0	0	0	3
2	3	0	0	4	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0
4	0	4	0	0	0	0	0	0	0	3	0	0
5	0	0	1	0	0	0	0	3	0	0	0	0
6	0	0	0	0	0	0	3	0	0	0	0	0
7	0	0	0	0	0	3	0	0	2	0	1	0
8	0	0	0	0	3	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	3	0	0
10	0	0	0	3	0	3	0	0	3	0	3	0
11	0	0	0	0	0	0	0	1	0	3	0	0
12	3	0	0	0	0	0	0	0	0	0	0	0

当城市点连接数 n=16 时，其规划具体路线如图 5.8(a)所示，此时最大网络价值为：4351.0578mTb/s；而随着迭代次数的增加，城市网络价值的变化曲线如图 5.8(b)所示，迭代次数在 240 次左右时，网络价值波动虽然没有稳定在一定的数值，但是波动情况较小，即将趋于稳定。



(a) 网络线路规划图



(b) 迭代次数与网络价值的关系

图 5.8 n=33 时网络规划

同理，根据定义的变量得出各个城市点的邻接矩阵，如表 13 所示。结合表格的横纵数值对比分析，可以分析出网络规划图的路线。表格中的“2”、“3”表示该条线路是一个中间路段，该线路被利用了两次或三次。

表 13　n=16 各城市点的邻接矩阵

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	4	0	0	0	0	0	0	0	0	0	0
2	4	0	0	4	4	4	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	4	0
4	0	4	0	0	0	4	4	0	0	0	0	4
5	0	4	0	0	0	0	0	0	0	0	0	0
6	0	4	0	4	0	0	3	0	0	3	4	0
7	0	0	0	4	0	3	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	3	0	0	4
9	0	0	0	0	0	0	0	3	0	0	0	0
10	0	0	0	0	0	3	0	0	0	0	2	0
11	0	0	4	0	0	4	0	0	0	2	0	0
12	0	0	0	4	0	0	0	4	0	0	0	0

六、问题三：改进调制格式

6.1 问题分析

根据第一问的结果，光纤传输链路的长度与 SNR 容限点有关。在纠前 BER 不变的情况下，若 SNR 容限点越低，链路系统容忍噪声的能力越强，那么链路的长度越长。因此在问题三中，探讨改变 16QAM 方案中星座点的位置、数量或每个点的概率，来获得一个比问题一中 8QAM 中具有更低 SNR 容限点的调制方式。同时该问题还引入了信息熵的概念，信息熵[8]是计算机中的概念，将信源的平均不定度用概率分布来度量。根据式(1.9)信息熵的定义，要使信息熵保持不变，可以通过减少星座点数量或者减小星座点的概率来实现。但在本问题中，假定使信息熵保持为 3bit 时星座点的个数是 8 个，由信息熵定义可知每个点的概率也恒定保持不变。因此，要使 SNR 容限点更低，只需要考虑星座点的位置，通过遗传算法寻找星座点的最优位置，使新的调制方案的 SNR 容限点最低[9]。

6.2 模型建立

问题三模型的建立相较于问题二比较容易，首先 8QAM 的星座点可直接作为一个初始种群个体的基因，其次适应度函数满足相邻两点的距离最大。遗传算法建模如下：

(1) 染色体编码

8QAM 的 8 个星座点即可当做遗传算法中的基因，在坐标平面内，分别以 x、y 轴上[-1,1]组成的正方形为星座点随机生成的区域空间，在此区间随机生成的一个复数就代表一个基因，8 个星座点对应一个个体的 8 个基因。

(2) 初始种群

遗传算法的初始种群是 20 个个体，每个个体的基因是需要改善的 8QAM 的 8 个星座点，在上述区域中随机生成 8 个复数即为一个个体的 8 个基因，随机产生的 20 个个体构成一个初始种群。

(3) 适应度函数

该模型的适应度函数是对相邻两点间的距离作出评价，具体步骤可以分为：

1) 求八个点中两两个节点的距离，并找到最小距离；

2) 将这八个点的坐标除以最小距离，得出新的八个坐标点，对这八个新坐标点取模→平方→取均值→求倒数，得出一个价值数值；

3) 依次类推，产生 20 个价值数值，即为所求的适应度。

(4) 遗传算子

对于选择算子，同样采用轮盘赌选择法，择出最高价值的个体进行交叉变异进入下一代；

对于交叉算子，由于个体中的基因数量较少，一次采用单点交叉较合理；

对于变异算子，采用基本变异，只要能够保持种群基因的多样性就能够满足条件。

6.3 模型求解

如下图 6.1 所示为遗传算法迭代产生的新的 8QAM 调制方案。

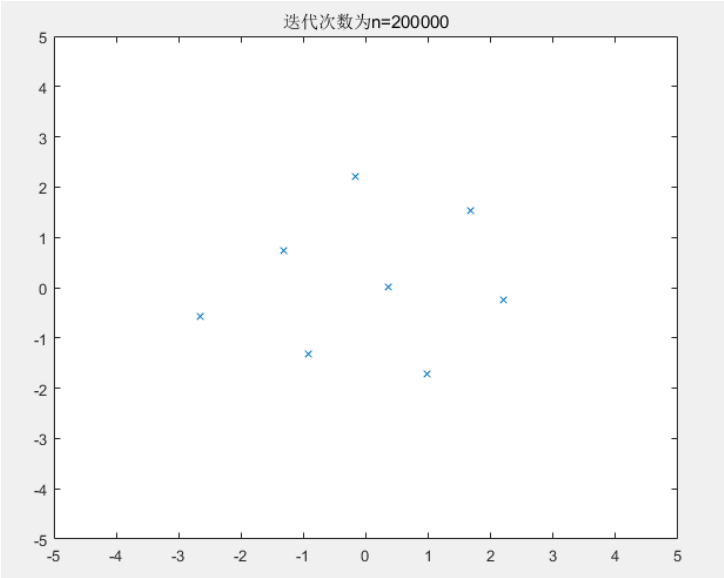


图 6.1 改善后 8QAM 调制方案

图 6.2、6.3 为改善前后的发送与接收星座图，从下图可以看出，改善后的 8QAM 星座图构成一个心形(横型)。

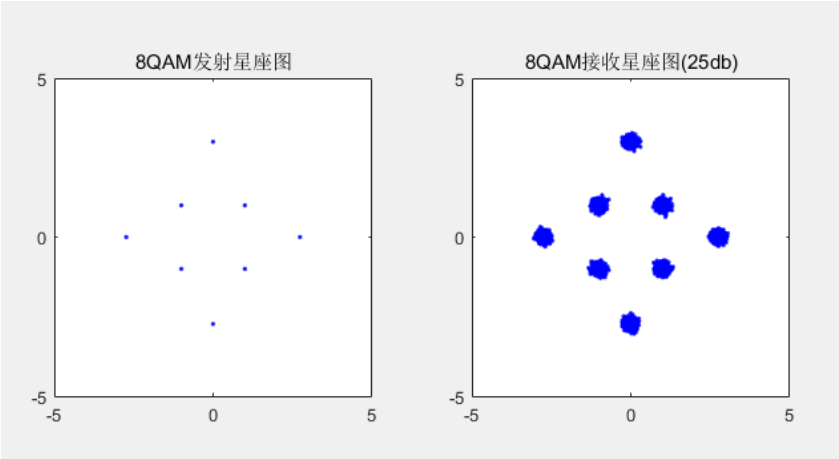


图 6.2 改善前 8QAM 星座图

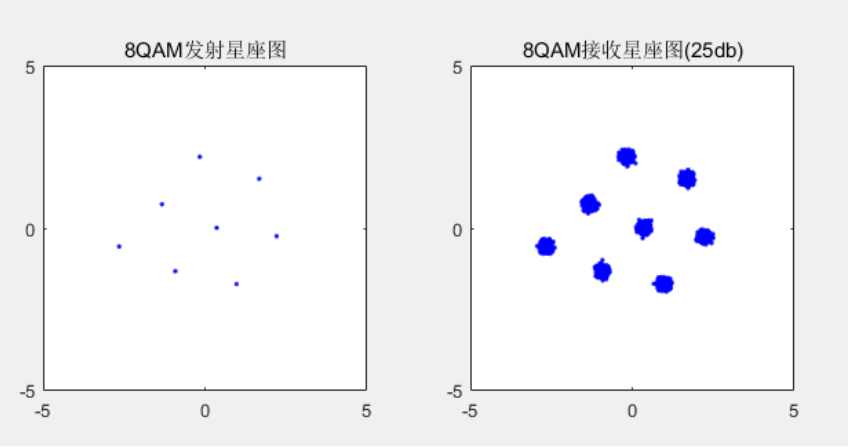


图 6.3 改善后 8QAM 星座图

迭代次数与适应度关系曲线如图 6.4 所示，经过两万次迭代后，适应度基本稳定，并随着迭代次数逐渐缓慢增长，并且评价值稳定在 0.7 以上。表明遗传算法对该问题有效。

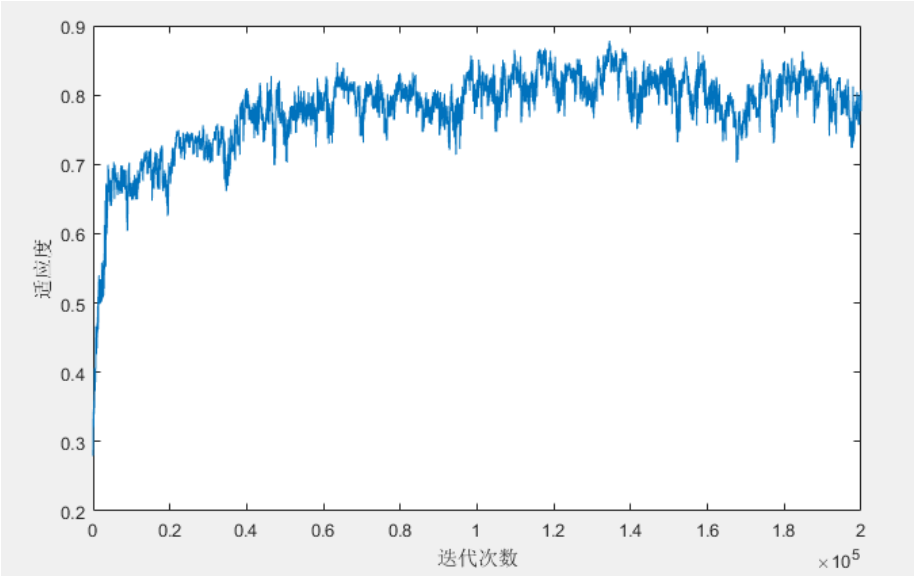


图 6.4 迭代次数与适应度关系曲线

如图 6.5 所示，改善后的 8QAM 调制方案比改善前具有更低的 SNR 容限点，可以提高系统容忍噪声的能力，从而延长链路的总长度。

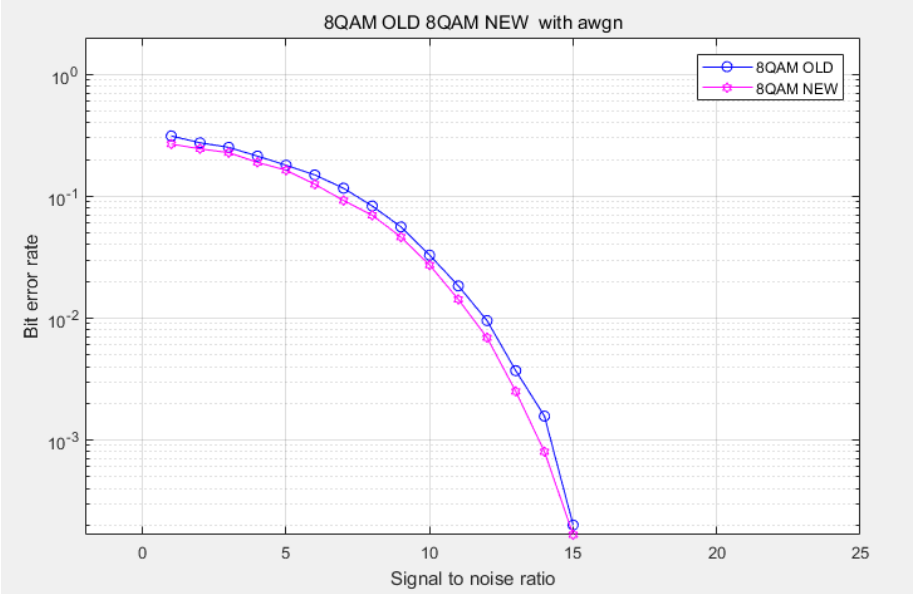


图 6.5 改善前后 8QAM 调制方案 BER 与 SNR 关系曲线

参考文献

[1]石悦，邱雪松等人.基于改进遗传算法的电力光传输网规划方法[J].通信学报：2016 , 37 (1) :116-122.

[2]张公礼,刘俊霞,罗宏杰.采用 8QAM 调制的 GSM/EDGE 移动通信系统[J].电子器件,2008(03):887-889.

[3] 褚衍杰,胡年福,巢凯今,彭晓成.通带 MQAM 信号调制方式识别方法[J].西安电子科技大学学报,2008(05):932-937.

[4] 孔五艳. MQAM 信号调制识别技术研究[D].苏州大学,2013.

[5]刘继红，李佳泯，梁猛.16-QAM 相干光纤通信系统星座图的优化与选择[J].半导体光电：2012 , 33 (1) :110-112.

[6]魏全新，刘贤锋等人.遗传算法选择方法的比较分析[J].《通讯和计算机:中英文版》, 2008 (8) :61-65.

[7]卢金. 基于遗传算法的光传送网络空闲资源优化设计[D].吉林大学，2006.

[8]百度百科：https://baike.so.com/doc/6616826-6830620.html.

[9]豆丁网：http://www.docin.com/p-977362927-f2.html

附录

问题 1-1 代码

```
clear all;
close all;
%假设传输10000个符号
N_symbol=10000;
for snrdb=1:1:25
%%%%%%%%%%%%%%%%%%%%qpsk原始信号%%%%%%%%%%%%%%
    %qpsk原始二进制位流
    si_qpsk=(1/(2^(1/2)))*2*(round(rand(1,N_symbol))-0.5);
    sq_qpsk=(1/(2^(1/2)))*2*(round(rand(1,N_symbol))-0.5);
    s_qpsk=si_qpsk + 1i*sq_qpsk;
%%%%%%%%%%%%%%%%%%%%8qam原始信号%%%%%%%%%%%%%%%%%%
    dot_8qam = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -
(1+3^(1/2))];   %8qam星座映射点
    %产生符号序列%  映射16qam
    rand0_7 = randi(8,1,N_symbol);
    s_8qam = zeros(1,N_symbol);
    for a=1:N_symbol
        s_8qam(1,a)=dot_8qam(1,rand0_7(1,a));
    end
    %8qam原始二进制位流
    s_8qam_oct=qamdemod_8qam(s_8qam,8); %此时解调出来的是8进制信号
    s_8qam_bit_stream=de2bi(s_8qam_oct,3,'left-msb');     %转化为对应的二进制比特流
s_8qam_bit_stream=reshape(s_8qam_bit_stream.',numel(s_8qam_bit_stream),1');
%%%%%%%%%%%%%%%%%%%%16qam原始信号%%%%%%%%%%%%%%%%
    dot_16qam = [1+1j -1+1j -1-1j 1-1j 3+1j 3+3j 1+3j -1+3j -3+3j -3+1j -3-1j -3-3j -1-3j 1-
3j 3-3j 3-1j];   %16qam星座映射点
    %产生符号序列%  映射16qam
    rand0_15 = randi(16,1,N_symbol);
    s_16qam = zeros(1,N_symbol);
    for a=1:N_symbol
        s_16qam(1,a)=dot_16qam(1,rand0_15(1,a));
    end
    %16qam原始二进制位流
    s_16qam_hex=demodulate(modem.qamdemod(16),s_16qam); %此时解调出来的是16
进制信号
    s_16qam_bit_stream=de2bi(s_16qam_hex,4,'left-msb');   %转化为对应的二进制比特
流
s_16qam_bit_stream=reshape(s_16qam_bit_stream.',numel(s_16qam_bit_stream),1');
%%%%%%%%%%%%%%%%%%%%加高斯白噪声%%%%%%%%%%%%%%%%%%
    %qpsk加高斯白噪声
    w_qpsk = awgn(s_qpsk,snrdb,'measured');
    r_qpsk = w_qpsk;
    %8qam加高斯白噪声
    w_8qam = awgn(s_8qam,snrdb,'measured');
    r_8qam = w_8qam;
    %16qam加高斯白噪声
    w_16qam = awgn(s_16qam,snrdb,'measured');
    r_16qam = w_16qam;
    %对加噪声之后的信号计算误码率
    %qpsk BER
```

```
        si__qpsk=sign(real(r_qpsk));
        sq__qpsk=sign(imag(r_qpsk));
        ber1_qpsk=(N_symbol-sum((si_qpsk/(1/(2^(1/2))))==si__qpsk))/N_symbol;
        ber2_qpsk=(N_symbol-sum((sq_qpsk/(1/(2^(1/2))))==sq__qpsk))/N_symbol;
        ber_qpsk(snrdb)=mean([ber1_qpsk ber2_qpsk]);
        %8qam BER
        r_8qam_oct=qamdemod_8qam(r_8qam,8);     %此时解调出来的是8进制信号
        r_8qam_bit_stream=de2bi(r_8qam_oct,3,'left-msb');       %转化为对应的二进制比特
流
    r_8qam_bit_stream=reshape(r_8qam_bit_stream.',numel(r_8qam_bit_stream),1');
        ber_8qam(snrdb)=biterr(s_8qam_bit_stream,r_8qam_bit_stream)/(N_symbol*3);
            %16qam BER
        r_16qam_hex=demodulate(modem.qamdemod(16),r_16qam);   %此时解调出来的是
16进制信号
        r_16qam_bit_stream=de2bi(r_16qam_hex,4,'left-msb');     %转化为对应的二进制比特
流
        r_16qam_bit_stream=reshape(r_16qam_bit_stream.',numel(r_16qam_bit_stream),1');
    ber_16qam(snrdb)=biterr(s_16qam_bit_stream,r_16qam_bit_stream)/(N_symbol*4);
    end
    %QPSK星座图
    figure(1);clf;hold on;
    subplot(1,2,1);plot(s_qpsk,'b.');title('QPSK发射星座图');axis equal;axis([-2 2 -2 2])
    subplot(1,2,2);plot(r_qpsk,'b.');title('QPSK接收星座图(25db)');axis equal;axis([-2 2 -2 2])
    %8qam星座图
    hold on;
    figure(2);clf;hold on;
    subplot(1,2,1);plot(s_8qam,'b.');title('8QAM发射星座图');axis equal;axis([-5 5 -5 5])
    subplot(1,2,2);plot(r_8qam,'b.');title('8QAM接收星座图(25db)');axis equal;axis([-5 5 -5 5])
    %16qam星座图
    figure(3);
    clf; hold on;
    subplot(1,2,1);plot(s_16qam,'b.');title('16QAM发射星座图');axis equal;axis([-5 5 -5 5])
    subplot(1,2,2);plot(r_16qam,'b.');title('16QAM接收星座图(25db)');axis equal;axis([-5 5 -5
5])
    %%%%%%%%%%%%%%%qpsk 8qam 16qam BER-SNR图%%%%%%%%%
    figure(4);
    snrdb=1:1:25;
    semilogy(snrdb,ber_qpsk,'-bo',snrdb,ber_8qam,'-mh',snrdb,ber_16qam,'-
gd',snrdb,0.02*ones(size(snrdb)),'r--','linewidth',2)
    title('QPSK 8QAM 16QAM    with awgn');
    xlabel('Signal to noise ratio');
    ylabel('Bit error rate');
    legend('QPSK','8QAM','16QAM');
    axis([-2 25 -5 2]);
    grid on;
    function [out] = qamdemod_8qam(In,mod)
    switch mod
        case 8
            Symmbol_8qam = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -
(1+3^(1/2))];
            for a=1:length(In)
                distance = abs(Symmbol_8qam-In(a));
                [~,Ind] = min(distance);
                out(a) = Ind-1;
            end
```

```
end
end
```

问题 1-2 代码

```
clear all;
close all;
clc;
%假设传输10000个符号
N_symbol=10000;
%放大器自发辐射噪声
%常数
h = 6.62606896*(1e-34);
f = 193e12;
B = 50e9;
NF = 4;
%符号单位功率
Transmit_Power    = 1e-3;
Transmit_Amp = sqrt(Transmit_Power);
Span_num = (10:10:250);
%%%%%%%%%%%%%%%%%%%%%%%%%%80km为一跨%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
One_span_distance = 80;      %单位km
for span_i=1:length(Span_num)
%%%%%%%%%%%%%%%%%%%%%%%%%qpsk原始信号%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %qpsk原始二进制位流
    si_qpsk=(1/(2^(1/2)))*2*(round(rand(1,N_symbol))-0.5);
    sq_qpsk=(1/(2^(1/2)))*2*(round(rand(1,N_symbol))-0.5);
    s_qpsk=si_qpsk + 1i*sq_qpsk;
    s_qpsk = s_qpsk*Transmit_Amp;
%%%%%%%%%%%%%%%%%%%%%%%%%8qam原始信号%%%%%%%%%%%%%%%%%%%%%%%%%%
    dot_8qam = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -(1+3^(1/2))];    %8qam星座映射点
    %产生符号序列%  映射16qam
    rand0_7 = randi(8,1,N_symbol);
    s_8qam = zeros(1,N_symbol);
    for a=1:N_symbol
        s_8qam(1,a)=dot_8qam(1,rand0_7(1,a));
    end
    %8qam原始二进制位流
    s_8qam_oct=qamdemod_8qam(s_8qam,8); %此时解调出来的是8进制信号
    s_8qam_bit_stream=de2bi(s_8qam_oct,3,'left-msb');                        %转化为对应的二进制比特流
s_8qam_bit_stream=reshape(s_8qam_bit_stream.',numel(s_8qam_bit_stream),1');
    s_8qam = s_8qam*Transmit_Amp;
%%%%%%%%%%%%%%%%%%%16qam原始信号%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    dot_16qam = [1+1j -1+1j -1-1j 1-1j 3+1j 3+3j 1+3j -1+3j -3+3j -3+1j -3-1j -3-3j -1-3j 1-3j 3-3j 3-1j];    %16qam星座映射点
    %产生符号序列%  映射16qam
    rand0_15 = randi(16,1,N_symbol);
    s_16qam = zeros(1,N_symbol);
    for a=1:N_symbol
```

```
            s_16qam(1,a)=dot_16qam(1,rand0_15(1,a));
        end
        %16qam原始二进制位流
        s_16qam_hex=demodulate(modem.qamdemod(16),s_16qam); %此时解调出来的是16
进制信号
        s_16qam_bit_stream=de2bi(s_16qam_hex,4,'left-msb');                %转化为对应
的二进制比特流
    s_16qam_bit_stream=reshape(s_16qam_bit_stream.',numel(s_16qam_bit_stream),1');
        s_16qam = s_16qam*Transmit_Amp;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%经跨传
输%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        r_qpsk = span(s_qpsk,One_span_distance,Span_num(span_i),h,f,B,NF)/Transmit_Amp;
        r_8qam =
span(s_8qam,One_span_distance,Span_num(span_i),h,f,B,NF)/Transmit_Amp;
        r_16qam =
span(s_16qam,One_span_distance,Span_num(span_i),h,f,B,NF)/Transmit_Amp;
        %经跨段传输之后的信号计算误码率
        %qpsk BER
        si__qpsk=sign(real(r_qpsk));
        sq__qpsk=sign(imag(r_qpsk));
        ber1_qpsk=(N_symbol-sum((si_qpsk/(1/(2^(1/2))))==si__qpsk))/N_symbol;
        ber2_qpsk=(N_symbol-sum((sq_qpsk/(1/(2^(1/2))))==sq__qpsk))/N_symbol;
        ber_qpsk(span_i)=mean([ber1_qpsk ber2_qpsk]);
        %8qam BER
        r_8qam_oct = qamdemod_8qam(r_8qam,8);     %此时解调出来的是8进制信号
        r_8qam_bit_stream=de2bi(r_8qam_oct,3,'left-msb');                %转化为对应的
二进制比特流
    r_8qam_bit_stream=reshape(r_8qam_bit_stream.',numel(r_8qam_bit_stream),1');
    ber_8qam(span_i)=biterr(s_8qam_bit_stream,r_8qam_bit_stream)/(N_symbol*3);
        %16qam BER
        r_16qam_hex=demodulate(modem.qamdemod(16),r_16qam); %此时解调出来的是16
进制信号
        r_16qam_bit_stream=de2bi(r_16qam_hex,4,'left-msb');   %转化为对应的二进制比特
流
    r_16qam_bit_stream=reshape(r_16qam_bit_stream.',numel(r_16qam_bit_stream),1');
    ber_16qam(span_i)=biterr(s_16qam_bit_stream,r_16qam_bit_stream)/(N_symbol*4);
    end
    figure(1);clf;hold on;
    plot(Span_num,ber_qpsk,'r-*');
    plot(Span_num,ber_8qam,'b-o');
    plot(Span_num,ber_16qam,'k-x');
    plot(Span_num,0.02*ones(size(Span_num)),'r--','linewidth',2);
    legend('QPSK','8QAM','16QAM','Ber门限值');
    xlabel('光纤段数');ylabel('ber');
    title('光纤长度  80Km');
    set(gca,'yscale','log');
    grid on; box on;
    function [Out] = qamdemod_8qam(In,mod)
    switch mod
        case 8
            ModSymmbol = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -
(1+3^(1/2))];
            for a=1:length(In)
                Dis = abs(ModSymmbol-In(a));
                [~,Ind] = min(Dis);
```

```
                Out(a) = Ind-1;
            end
        end
        end
        function out = span(sgl,One_span,Span_num,h,f,B,NF)
        sgl_tmp=sgl;
        for a=1:Span_num
            sgl_tmp=span_opt(sgl_tmp,One_span,h,f,B,NF);
        end
        out = sgl_tmp;
        end
        function out = span_opt(sgl,One_span,h,f,B,NF)
        %计算光纤的衰减和Gain值
        Loss_dB = 3*(One_span/15000);
        Loss = 10^(Loss_dB/10);
        Gain = 1/Loss;
        %线性噪声功率
        Pn = 2*pi*h*f*B*(NF+1/Gain);
        InPow = mean(sgl.^2)/1e3;
        Pno = 2*pi*h*f*B*NF;
        %非线性噪声功率
        Pnnl = (InPow/(1e-3))^2*Pno*2/3;
        %生成入跨信号
        NLNoise = (randn(size(sgl))+randn(size(sgl))*1i)/sqrt(2)*sqrt(Pnnl);
        In2 = sgl + NLNoise;
        %进行衰减和放大
        Out1 = In2/Loss*Gain;
        %放大器噪声
        LnaNoise = (randn(size(sgl))+randn(size(sgl))*1i)/sqrt(2)*sqrt(Pn);
        out = Out1+LnaNoise;
        end
```

问题 2-1 主要代码

```
        clear;clc;close all;
        遗传函数参数
        Link_Num = 16;
        P_cross = 0.3; %交叉概率0.3-0.9
        P_mutate = 0.01; %变异概率0.01-0.2
        Link_Num = 33;
        P_cross = 0.3; %交叉概率0.3-0.9
        P_mutate = 0.02; %变异概率0.01-0.2
        Initial_population_individual_num = 400; %初始种群的个体数量
        Max_generation = 500;    %最大代数
        %目标连接数
        Link_Num = 16;
        %城市信息
        City_Num = 12;
        %总可连接数
        Link_Num_Total=(City_Num-1) * City_Num/2;
        City_Name = ["哈尔滨","北京&天津", "郑州", "上海", "武汉",   "广州&深圳", "西安", "
乌鲁木齐", "拉萨", "成都", "昆明", "重庆"];
        City_Population = [9.9526 34.8245 9.03 23.8043 10.12 23.3853 8.83 3.8 0.9025 16.0477
7.2131 8.56];
        City_Locaiton=[45.8 126.53;39.92 116.46;34.75 113.66;31.23 121.47;30.6 114.3;23.13
113.27;34.27 108.93;43.82 87.62;29.97 91.11;30.67 104.07;25.05 102.72;29.57 106.55];
```

```
%City_coordinate为经纬度转换后的坐标
City_coordinate=city_information(City_Locaiton);
%%%%%%%%%%%%%%%%遗传算法参数%%%%%%%%%%%%%%%%%%
Initial_population_individual_num = 400; %初始种群的个体数量
Max_generation = 500;   %最大代数
P_cross = 0.3; %交叉概率0.3-0.9
P_mutate = 0.01; %变异概率0.01-0.2
%%%%%%%%%%%%%%%%%%产生初始种群%%%%%%%%%%%%%%%%%%
Initial_population = zeros(Initial_population_individual_num,Link_Num_Total);
for i=1:Initial_population_individual_num
    %采用tsp编码，即对每条线路进行编码
    Initial_population(i,:) = randperm(Link_Num_Total);
end
%%%%%%%%%%%%%%%%%%计算种群的适应度%%%%%%%%%%%%%%%%%%
[~,P_popualtion_distri]=population_fitness(Initial_population,City_coordinate,City_Populati
on,Link_Num);
%%%%%%%%%%%%%%%%%%进化阶段%%%%%%%%%%%%%%%%%%
Evolution_popualtion = Initial_population;
generation=1;
Generations_of_best_individual=zeros(Initial_population_individual_num,Link_Num_Total)
;
Population_cross_new=zeros(Initial_population_individual_num,Link_Num_Total);
Population_mutate_new=zeros(Initial_population_individual_num,Link_Num_Total);
while generation<Max_generation
%%%%%%%%%%%%%%%%%%产生新种群%%%%%%%%%%%%%%%%%%
    for j=1:2:Initial_population_individual_num
        %根据种群概率分布进行选择操作
        selected_num=select(P_popualtion_distri);
        %交叉
        Population_cross=crossover(Evolution_popualtion,selected_num,P_cross);
        Population_cross_new(j,:)=Population_cross(1,:);
        Population_cross_new(j+1,:)=Population_cross(2,:);
        %变异
        Population_mutate_new(j,:)=mutate(Population_cross_new(j,:),P_mutate);
      Population_mutate_new(j+1,:)=mutate(Population_cross_new(j+1,:),P_mutate);
    end
    Evolution_popualtion=Population_mutate_new;
%%%%%%%%%%计算新种群中个体的适应度%%%%%%%%%%%%%%%%%%
[Individual_fitness,P_popualtion_distri]=population_fitness(Evolution_popualtion,City_coor
dinate,City_Population,Link_Num);
    %记录当代最佳个体的适应度和个体在群落中的位置
[Best_fitness_individual_value,Best_fitness_individual_num]=max(Individual_fitness);
%找出适应度最大的个体
    %记录当前代的最佳个体
    Best_individual=Evolution_popualtion(Best_fitness_individual_num,:);
    %记录每代最佳个体
    Generations_of_best_individual(generation,:)=Best_individual;
    %记录每代最佳个体的适应度，即总网络价值
    Generations_of_best_individual_fitness(generation) =
Single_Individual_fitness(Best_individual(end,1:Link_Num),City_coordinate,City_Population);
    generation=generation+1; %进入下一代
end
function[Network_total_value,City_connection,City_Capacity]=Single_Individual_fitness(Si
ngle_individual_gene,City_coordinate,City_Population)
%解码
```

```
    %从66条连接里面找到Link_Num（16或33）个连接并解码为城市间连接矩阵
    City_Num = size(City_coordinate,1);   %坐标个数%即城市个数
    City_connection = 1-tril(ones(City_Num,City_Num),0);        %取下三角矩阵
    City_connection = City_connection(:);                          %组成一个向量
    Ind = find(City_connection==1);                    %下三角矩阵元素为1位置索引
    City_connection(Ind)=1:length(Ind);               %右上三角递增
    for ii=1:length(Single_individual_gene)
        City_connection(City_connection==(Single_individual_gene(ii)))=Inf;
    end
    City_connection=(City_connection==Inf);
    City_connection = reshape(City_connection,City_Num,City_Num);
    City_connection = City_connection + City_connection';
    %判断是否连接了所有结点
    if (length(TSP(1,City_connection)) < City_Num)
        Network_total_value =0;
        return;
    end
    %计算总的网络价值，权值为1
    Network_total_value = 0;
    Max_trans_dist_Req = [Inf 3000 1200 600];
    Capacity_Table = [0 8 16 32];
    City_Capacity = zeros(City_Num,City_Num);
    for ii =1:City_Num
        for jj=ii+1:City_Num
            if (City_connection(ii,jj)>=1)                %如果有连接
                Dist= norm(City_coordinate(ii,:) - City_coordinate(jj,:));     %两城市间距离
                Ind = find(Max_trans_dist_Req>Dist);
                City_Capacity(ii,jj) = Capacity_Table(Ind(end));%选择某城市的网络容量
                Network_total_value = Network_total_value +
sqrt(City_Population(ii)*City_Population(jj))*City_Capacity(ii,jj);

            end
        end
    end

    end
    %%%%%%%%%%%%%%%%%%%%%表达该网络%%%%%%%%%%%%
    %计算总价值，城市连接，城市容量
    [Individual_fitness,City_connection,City_Capacity]=Single_Individual_fitness(Generations_
of_best_individual(end,1:Link_Num),City_coordinate,City_Population);
    %%%%%%%%%%%%%%%%作图%%%%%%%%%%%%%%%%%%%%%%
    figure(1);
    clf;hold on;
    %画城市坐标点
    plot(City_Locaiton(:,2),City_Locaiton(:,1),'ko');
    %显示城市名字
    for i= 1:City_Num
        text(City_Locaiton(i,2),City_Locaiton(i,1),City_Name{i},'fontsize',18)
    end
    %画城市间连线
    for ii=1:City_Num
        for jj=1:City_Num
            if (City_connection(ii,jj)==1)
                plot([City_Locaiton(ii,2) City_Locaiton(jj,2)],...
                    [City_Locaiton(ii,1) City_Locaiton(jj,1)],'r--');
```

```
            end
        end
    end
    title(['容量为：' num2str(Individual_fitness) 'mTb/s'])
    ylabel('纬度','fontsize',15);
    xlabel('经度','fontsize',15);
    figure(2);
    plot(Generations_of_best_individual_fitness);
    title(['连接数为：' num2str(Link_Num)]);
    ylabel('网络价值','fontsize',15);
    xlabel('迭代次数','fontsize',15);
```

问题 2-2 主要代码

```
    clear ;clc;close all;
    %目标连接数
    %Link_Num = 16;
    Link_Num = 33;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%城市信
息%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    City_Num = 12;
    Link_Num_Total=(City_Num-1)*City_Num/2;
    City_Name = ["哈尔滨","北京&天津", "郑州", "上海", "武汉",  "广州&深圳", "西安", "
乌鲁木齐", "拉萨", "成都", "昆明", "重庆"];
    City_Population = [9.9526 34.8245 9.03 23.8043 10.12 23.3853 8.83 3.8 0.9025 16.0477
7.2131 8.56];
    City_Locaiton=[45.8 126.53;39.92 116.46;34.75 113.66;31.23 121.47;30.6 114.3;23.13
113.27;34.27 108.93;43.82 87.62;29.97 91.11;30.67 104.07;25.05 102.72;29.57 106.55];
    %City_coordinate为经纬度转换后的坐标
    City_coordinate=city_information(City_Locaiton);
    City_Dist = zeros(City_Num,City_Num);
    for ii=1:City_Num
        for jj=1:City_Num
            City_Dist(ii,jj) = norm(City_Locaiton(ii,:)-City_Locaiton(jj,:));
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%遗传算法参
数%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Initial_population_individual_num=100; %初始种群大小
    Max_generation=100;   %最大代数
    P_crossover=0.3; %交叉概率
    P_mutate=0.02; %变异概率
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%产生初始种
群%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    max_line = 4;
    Code_Num = Link_Num_Total+Link_Num_Total+City_Num*City_Num;
    Initial_population=zeros(Initial_population_individual_num,Code_Num);
    for i=1:Initial_population_individual_num
        si=randperm(Link_Num_Total);
        sid=randi([1 max_line],1,Link_Num_Total);
        xx = rand(City_Num,City_Num);
        xx = xx/sum(xx(:));
        Initial_population(i,:) = Code(si,sid,xx);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%计算种群的适应
```

```
度%%%%%%%%%%%%%%%%%%%%%%%%
    [~,P_popualtion_distri]=population_fitness(Initial_population,City_coordinate,City_Dist,Cit
y_Population,Link_Num);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%进化阶
段%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    generation=1;
    Evolution_popualtion = Initial_population;
    Generations_of_best_individual=zeros(Initial_population_individual_num,Code_Num);
    Population_cross_new=zeros(Initial_population_individual_num,Code_Num);
    Population_mutate_new=zeros(Initial_population_individual_num,Code_Num);
    while generation<Max_generation+1
        for j=1:2:Initial_population_individual_num
            %选择
            selected_num=select(P_popualtion_distri);
            %交叉
    Population_cross=crossover(Evolution_popualtion,selected_num,P_crossover,City_Num,Lin
k_Num);
            Population_cross_new(j,:)=Population_cross(1,:);
            Population_cross_new(j+1,:)=Population_cross(2,:);
            %变异

Population_mutate_new(j,:)=mutate(Population_cross_new(j,:),P_mutate,City_Num,Link_Num);
Population_mutate_new(j+1,:)=mutate(Population_cross_new(j+1,:),P_mutate,City_Num,Link_N
um);
        end
        Evolution_popualtion=Population_mutate_new;  %产生了新的种群
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%计算新种群中个体的适应
度%%%%%%%%%%%%%%%%%%%%%%%%
[Individual_fitness,P_popualtion_distri]=population_fitness(Evolution_popualtion,City_coordinat
e,City_Dist,City_Population,Link_Num);
        %记录当代最佳个体的适应度和个体在群落中的位置
[Best_fitness_individual_value,Best_fitness_individual_num]=max(Individual_fitness);
        %记录当前代的最佳个体
    Best_individual=Evolution_popualtion(Best_fitness_individual_num,:);
        %记录每代最佳个体
        Generations_of_best_individual(generation,:)=Best_individual;
        [~,~,XFp,si2,sid2] = Decode(Best_individual,12,Link_Num);
        XId = [si2;sid2]';
        %记录每代最佳个体的适应度，即总网络价值
    Generations_of_best_individual_fitness(generation)=Single_Individual_fitness(XId,XFp,Cit
y_Dist,City_coordinate,City_Population);
        generation=generation+1;
        generation
    end
    function[Individual_fitness,City_connection,Xload2]=Single_Individual_fitness(XId,XFp,Cit
y_coordinate,City_Locaiton,City_Population)
    City_num = size(City_Locaiton,1);
    %计算连接矩阵
    City_connection = 1-tril(ones(City_num,City_num),0);
    City_connection = City_connection(:);
    Ind = find(City_connection==1);
    City_connection(Ind)=1:length(Ind);
    Xc = zeros(1,City_num*City_num);
    for ii=1:length(XId)
        Xc(City_connection==XId(ii,1)) = 1;
        City_connection(City_connection==XId(ii,1))=XId(ii,2);
```

```
end
City_connection(Xc==0)=0;
City_connection = reshape(City_connection,City_num,City_num);
City_connection = City_connection+City_connection';
%判断是否连接了所有结点的问题
Xd = City_connection>0;
result=TSP(1,Xd);
if (length(result)<City_num)
    Individual_fitness =0;
    return;
end
XInt = City_connection>0;
City_coordinate= City_coordinate.*XInt;
City_coordinate(XInt==0) = Inf;
%路径速率计算
Xload = ones(City_num,City_num);
SnrReq = [Inf 3000 1200 600];
Capacity_Table = [0 8 16 32];
for ii =1:City_num
    for jj=1:City_num
        if (City_connection(ii,jj)>=1)
            Dist= norm(City_Locaiton(ii,:)-City_Locaiton(jj,:));
            Ind = find(SnrReq>Dist);
            ModRate = Capacity_Table(Ind(end));
            Xload(ii,jj) = City_connection(ii,jj)*ModRate;
        end
    end
end
%计算含一个中间结点情况下的速率分配
Xload2 = zeros(City_num,City_num);
XFp = XFp*sum(Xload(:))/4;
for ii =1:City_num
    for jj=ii+1:City_num
        [shortestPath, totalCost] = Dijkstra(City_coordinate, ii, jj);
        if (totalCost==Inf || length(shortestPath)~=3)
            Xload2(ii,jj) = 0;
            Xload2(jj,ii) = 0;
        else
            Xload2(jj,ii) = XFp(ii,jj);
            Xload2(jj,ii) = XFp(ii,jj);
            Xload(ii,shortestPath(2))=Xload(ii,shortestPath(2))-XFp(ii,jj);
            Xload(shortestPath(2),ii)=Xload(shortestPath(2),ii)-XFp(ii,jj);
            Xload(shortestPath(2),jj)=Xload(shortestPath(2),jj)-XFp(ii,jj);
            Xload(jj,shortestPath(2))=Xload(jj,shortestPath(2))-XFp(ii,jj);
        end
    end
end
if (min(Xload(:))<0)
    Individual_fitness= 0;
    return;
else
    for ii =1:City_num
        for jj=ii+1:City_num
            Xload2(ii,jj)= Xload(ii,jj);
            Xload2(jj,ii)= Xload(jj,ii);
        end
    end
```

```
    end
    % 计算总的价值
    Individual_fitness = 0;
    for ii =1:City_num
        for jj=ii+1:City_num
            if (City_connection(ii,jj)>=1)
                Dist= norm(City_Locaiton(ii,:)-City_Locaiton(jj,:));
                Ind = find(SnrReq>Dist);
                ModRate = Capacity_Table(Ind(end));
                Individual_fitness = Individual_fitness+
sqrt(City_Population(ii)*City_Population(jj))*Xload2(ii,jj);
            end
        end
    end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%表达该网
络%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %计算总价值，城市连接，城市容量
    [~,~,XFp,si2,sid2] = Decode(Generations_of_best_individual(end,:),12,Link_Num);
    XId = [si2;sid2]';
    [Individual_fitness,City_connection]=Single_Individual_fitness(XId,XFp,City_Dist,City_co
ordinate,City_Population);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%作
图%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(1);
    clf;hold on;
    plot(City_Locaiton(:,2),City_Locaiton(:,1),'ko');
    %显示城市名字
    for i= 1:City_Num
    text(City_Locaiton(i,2),City_Locaiton(i,1),City_Name{i},'fontsize',18)
    end
    %画城市间连线
    for ii=1:City_Num
        for jj=1:City_Num
            if (City_connection(ii,jj)>=1)
                plot([City_Locaiton(ii,2) City_Locaiton(jj,2)],...
                    [City_Locaiton(ii,1) City_Locaiton(jj,1)],'r--
','linewidth',double(City_connection(ii,jj)));
            end
        end
    end
    title(['容量为' num2str(Individual_fitness) 'mTb/s'])
    ylabel('纬度','fontsize',15);
    xlabel('经度','fontsize',15);
    figure(2);
    plot(Generations_of_best_individual_fitness);
    title(['连接数为' num2str(Link_Num)]);
    ylabel('网络价值','fontsize',15);
    xlabel('迭代次数','fontsize',15);
```

问题 2-3 主要代码

```
clear ;clc;close all;
%目标连接数
% Link_Num = 16;
Link_Num = 33;
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%城市信
息%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    City_Num = 12;
    Link_Num_Total=(City_Num-1)*City_Num/2;
    City_Name = ["哈尔滨","北京&天津", "郑州", "上海", "武汉",  "广州&深圳", "西安", "
乌鲁木齐", "拉萨", "成都", "昆明", "重庆"];
    City_Population = [9.9526 34.8245 9.03 23.8043 10.12 23.3853 8.83 3.8 0.9025 16.0477
7.2131 8.56];
    City_Locaiton=[45.8 126.53;39.92 116.46;34.75 113.66;31.23 121.47;30.6 114.3;23.13
113.27;34.27 108.93;43.82 87.62;29.97 91.11;30.67 104.07;25.05 102.72;29.57 106.55];
    %City_coordinate为经纬度转换后的坐标
    City_coordinate=city_information(City_Locaiton);
    City_Dist = zeros(City_Num,City_Num);
    for ii=1:City_Num
        for jj=1:City_Num
            City_Dist(ii,jj) = norm(City_Locaiton(ii,:)-City_Locaiton(jj,:));
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%遗传算法参
数%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Initial_population_individual_num=300; %初始种群大小
    Max_generation=300;   %最大代数
    P_crossover=0.3; %交叉概率
    P_mutate=0.01; %变异概率
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%产生初始种
群%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    max_Link_num = 2;
    Code_Num = Link_Num_Total+Link_Num_Total+City_Num*City_Num;
    Initial_population=zeros(Initial_population_individual_num,Code_Num);
    for i=1:Initial_population_individual_num
        si=randperm(Link_Num_Total);
        sid=randi([1 max_Link_num],1,Link_Num_Total);
        xx = rand(City_Num,City_Num);
        xx = xx/sum(xx(:));
        Initial_population(i,:) = Code(si,sid,xx);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%计算种群的适应
度%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [~,P_popualtion_distri]=population_fitness(Initial_population,City_coordinate,City_Dist,Cit
y_Population,Link_Num);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%进化阶
段%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    generation=1;
    Evolution_popualtion = Initial_population;
    Generations_of_best_individual=zeros(Initial_population_individual_num,Code_Num);
    Population_cross_new=zeros(Initial_population_individual_num,Code_Num);
    Population_mutate_new=zeros(Initial_population_individual_num,Code_Num);
    while generation<Max_generation+1
        for j=1:2:Initial_population_individual_num
            %选择
            selected_num=select(P_popualtion_distri);
            %交叉
    Population_cross=crossover(Evolution_popualtion,selected_num,P_crossover,City_Num,Lin
k_Num);
            Population_cross_new(j,:)=Population_cross(1,:);
```

```
            Population_cross_new(j+1,:)=Population_cross(2,:);
            %变异
    Population_mutate_new(j,:)=mutate(Population_cross_new(j,:),P_mutate,City_Num,Link_N
um);
    Population_mutate_new(j+1,:)=mutate(Population_cross_new(j+1,:),P_mutate,City_Num,Li
nk_Num);
        end
        Evolution_popualtion=Population_mutate_new;  %产生了新的种群
    %%%%%%%%%%%%%%%%%%%%%%%%%%计算新种群中个体的适应
度%%%%%%%%%%%%%%%%%%%%%%
[Individual_fitness,P_popualtion_distri]=population_fitness(Evolution_popualtion,City_coordinat
e,City_Dist,City_Population,Link_Num);
        %记录当代最佳个体的适应度和个体在群落中的位置
[Best_fitness_individual_value,Best_fitness_individual_num]=max(Individual_fitness);
        %记录当前代的最佳个体
Best_individual=Evolution_popualtion(Best_fitness_individual_num,:);
        %记录每代最佳个体
        Generations_of_best_individual(generation,:)=Best_individual;
        [~,~,XFp,si2,sid2] = Decode(Best_individual,12,Link_Num);
        XId = [si2;sid2]';
        %记录每代最佳个体的适应度，即总网络价值
    Generations_of_best_individual_fitness(generation)=Single_Individual_fitness(XId,XFp,Cit
y_Dist,City_coordinate,City_Population);
        generation=generation+1;
        generation
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%表达该网
络%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %计算总价值，城市连接，城市容量
    [~,~,XFp,si2,sid2] = Decode(Generations_of_best_individual(end,:),12,Link_Num);
    XId = [si2;sid2]';
    [Individual_fitness,City_connection]=Single_Individual_fitness(XId,XFp,City_Dist,City_co
ordinate,City_Population);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%作
图%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure(1);
    clf;hold on;
    plot(City_Locaiton(:,2),City_Locaiton(:,1),'ko');
    %显示城市名字
    for i= 1:City_Num
    text(City_Locaiton(i,2),City_Locaiton(i,1),City_Name{i},'fontsize',18)
    end
    %画城市间连线
    for ii=1:City_Num
        for jj=1:City_Num
            if (City_connection(ii,jj)>=1)
                plot([City_Locaiton(ii,2) City_Locaiton(jj,2)],...
                    [City_Locaiton(ii,1) City_Locaiton(jj,1)],'r--
','linewidth',double(City_connection(ii,jj)));
            end
        end
    end
    title(['容量为' num2str(Individual_fitness) 'mTb/s'])
    ylabel('纬度','fontsize',15);
    xlabel('经度','fontsize',15);
```

```
figure(2);
plot(Generations_of_best_individual_fitness);
title(['连接数为' num2str(Link_Num)]);
ylabel('网络价值','fontsize',15);
xlabel('迭代次数','fontsize',15);
%计算所有种群的适应度
function
[Individual_fitness,P_popualtion_distri]=population_fitness(Evolution_popualtion,City_coordinat
e,City_Dist,City_Population,Link_Num)
City_num = size(City_coordinate,1);
Individual_num=size(Evolution_popualtion,1);  %读取种群大小
Individual_fitness=zeros(Individual_num,1);
for i=1:Individual_num
    [~,~,XFp,si2,sid2] = Decode(Evolution_popualtion(i,:),City_num,Link_Num);
    XId = [si2;sid2]';
Individual_fitness(i)=Single_Individual_fitness(XId,XFp,City_Dist,City_coordinate,City_Po
pulation);  %计算函数值，即适应度
end
Individual_fitness=Individual_fitness';
%根据个体的适应度计算其被选择的概率
All_individual_fitness_sum=0;
for i=1:Individual_num
All_individual_fitness_sum=All_individual_fitness_sum+Individual_fitness(i)^15;% 让适
应度越好的个体被选择概率越高
end
P_individual_for_live=zeros(Individual_num,1);
for i=1:Individual_num
P_individual_for_live(i)=Individual_fitness(i)^15/All_individual_fitness_sum;
end
%计算累积概率
P_popualtion_distri=zeros(Individual_num,1);
P_popualtion_distri(1)=P_individual_for_live(1);
for i=2:Individual_num
    P_popualtion_distri(i)=P_popualtion_distri(i-1)+P_individual_for_live(i);
end
P_popualtion_distri=P_popualtion_distri';
End
```

问题3代码

```
clear all;clc;
close all;
New_constellation=Genetic_algorithm();
%假设传输10000个符号
N_symbol=10000;
for snrdb=1:1:25
%%%%%%%%%%%%%%%%%%%%%%%%8qam原始信
号%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    dot_8qam = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -
(1+3^(1/2))];  %8qam星座映射点
    %产生符号序列% 映射16qam
    rand0_7 = randi(8,1,N_symbol);
    s_8qam_old = zeros(1,N_symbol);
    for a=1:N_symbol
        s_8qam_old(1,a)=dot_8qam(1,rand0_7(1,a));
    end
```

```
%8qam原始二进制位流
s_8qam_oct_old=qamdemod_8qam(s_8qam_old,8); %此时解调出来的是8进制信号
s_8qam_bit_stream_old=de2bi(s_8qam_oct_old,3,'left-msb');                    %转化为对应的二进制比特流
s_8qam_bit_stream_old=reshape(s_8qam_bit_stream_old.',numel(s_8qam_bit_stream_old),1');
%%%%%%%%%%%%%%%%%%%%%%%%8qam原始信号%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s_8qam_oct_new = randi(8,1,N_symbol)-1;
%8qam_new原始二进制位流
s_8qam_new = qammod_new_constellation(s_8qam_oct_new,New_constellation);%%%%%%%%%%%用新的星座图，发送符号
s_8qam_bit_stream_new=de2bi(s_8qam_oct_new,3,'left-msb');                    %转化为对应的二进制比特流
s_8qam_bit_stream_new=reshape(s_8qam_bit_stream_new.',numel(s_8qam_bit_stream_new),1');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%8qam加高斯白噪声
w_8qam_old = awgn(s_8qam_old,snrdb,'measured');
r_8qam_old = w_8qam_old;
%8qam加高斯白噪声
w_8qam_new = awgn(s_8qam_new,snrdb,'measured');
r_8qam_new = w_8qam_new;
 %8qam BER
r_8qam_oct_old=qamdemod_8qam(r_8qam_old,8);                                  %此时解调出来的是8进制信号
r_8qam_bit_stream_old=de2bi(r_8qam_oct_old,3,'left-msb');    %转化为对应的二进制比特流
r_8qam_bit_stream_old=reshape(r_8qam_bit_stream_old.',numel(r_8qam_bit_stream_old),1');
ber_8qam_old(snrdb)=biterr(s_8qam_bit_stream_old,r_8qam_bit_stream_old)/(N_symbol*3);
%8qam BER
r_8qam_oct_new=qamdemod_new_constellation(r_8qam_new,New_constellation); %此时解调出来的是8进制信号
r_8qam_bit_stream_new=de2bi(r_8qam_oct_new,3,'left-msb');    %转化为对应的二进制比特流
r_8qam_bit_stream_new=reshape(r_8qam_bit_stream_new.',numel(r_8qam_bit_stream_new),1');
ber_8qam_new(snrdb)=biterr(s_8qam_bit_stream_new,r_8qam_bit_stream_new)/(N_symbol*3);
end
%8qam_old星座图
hold on;
figure(4);clf;hold on;
subplot(1,2,1);plot(s_8qam_old,'b.');title('8QAM发射星座图');axis equal;axis([-5 5 -5 5])
subplot(1,2,2);plot(r_8qam_old,'b.');title('8QAM接收星座图(25db)');axis equal;axis([-5 5 -5 5])
%8qam_new星座图
hold on;
figure(2);clf;hold on;
subplot(1,2,1);plot(s_8qam_new,'b.');title('8QAM发射星座图');axis equal;axis([-5 5 -5 5])
```

```
subplot(1,2,2);plot(r_8qam_new,'b.');title('8QAM接收星座图(25db)');axis equal;axis([-5 5 -5 5])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%8qam BERvsSNR%%%%%%%%%%%%%%%%%%%%%%%
figure(5);
snrdb=1:1:25;
semilogy(snrdb,ber_8qam_old,'-bo',snrdb,ber_8qam_new,'-mh')
title('8QAM OLD 16QAM NEW    with awgn');
xlabel('Signal to noise ratio');
ylabel('Bit error rate');
legend('8QAM OLD','8QAM NEW');
axis([-2 25 -5 2]);
grid on;
function New_constellation = Genetic_algorithm()
%种群大小
population_individual_num = 50;
%二进制编码长度
code_length=8;
%交叉概率
p_crossover = 0.1;
%变异概率
p_mutae = 0.02;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%初始种群
Initial_population = Init_population(population_individual_num,code_length);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%存储最佳星座点
save_fit_value =0;
New_constellation = [];
Evolution_popualtion = Initial_population;
for generation = 1:200000
    %计算适应度值（函数值）
    evaluate_value = cal_evaluate_value(Evolution_popualtion);
    fit_value = evaluate_value;
    %选择
    new_population = select(Evolution_popualtion,fit_value);
    %交叉
    new_population = crossover(new_population,p_crossover);
    %变异
    new_population = mutate(new_population,p_mutae);
    %更新
    Evolution_popualtion = update_popualtion(new_population);
    %寻找最优解
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %计算个体适应度
    [bestindividual,best_fit_value] = Find_Best_Individual(Evolution_popualtion,fit_value);
    if (save_fit_value<best_fit_value)
       save_fit_value = best_fit_value;
       New_constellation = bestindividual.*3;
    end
    Fitness(generation) = (best_fit_value);
    if mod(generation,100) == 0
        figure(1);clf
```

```
            plot(New_constellation,'x');
            axis([-5 5 -5 5]);
            title(['迭代次数为n=' num2str(generation)]);
            pause(0.01);
        end
end
figure(2);clf
plot(New_constellation,'x');
figure(3);
plot(Fitness)
end
function Initial_population=Init_population(popsize,chromlength)
Initial_population = (rand(popsize,chromlength)*2-1)+(rand(popsize,chromlength)*2-1)*1i;
end
function [population] = update_popualtion(population)
population_real=real(population);
population_imag=imag(population);
population_real = max(min(population_real,1),-1);
population_imag = max(min(population_imag,1),-1);
population = population_real+population_imag*1i;
end
function [evaluate_value] = cal_evaluate_value(population)
[individual,len] = size(population);
for kk=1:individual
        Dis = inf(len,len);
        for ii=1:len
            for jj=ii+1:len
                Dis(ii,jj) = abs(population(kk,ii)-population(kk,jj));
            end
        end
        minDis = min(Dis(:));
        if (minDis~=0)
            population(kk,:) = population(kk,:)/minDis;
            evaluate_value(kk) = 1/(mean(abs(population(kk,:)).^2));
        else
            evaluate_value (kk)= 0;
        end
end
end
function [new_population] = select(population,individual_fitness)
[px,~] = size(population);
total_fitness = sum(individual_fitness);
p_fitness = individual_fitness/total_fitness;
p_fitness = cumsum(p_fitness);%概率求和排序
ms = sort(rand(px,1));%从小到大排列
fitin = 1;
newin = 1;
while newin<=px
        if(ms(newin))<p_fitness(fitin)
            new_population(newin,:)=population(fitin,:);
            newin = newin+1;
        else
            fitin=fitin+1;
        end
end
end
function [new_population] = crossover(population,pc)
```

```
    [population_x,population_y] = size(population);
    new_population = ones(size(population));
    for i = 1:2:population_x-1
        if(rand<pc)
            cross_num = round(rand*population_y);
            new_population(i,:) =
[population(i,1:cross_num),population(i+1,cross_num+1:population_y)];
            new_population(i+1,:) =
[population(i+1,1:cross_num),population(i,cross_num+1:population_y)];
        else
            new_population(i,:) = population(i,:);
            new_population(i+1,:) = population(i+1,:);
        end
    end
    end
    function [new_populatin] = mutate(population,p_mutate)
    [population_x,population_y] = size(population);
    new_populatin = ones(size(population));
    for i = 1:population_x
        mutate_num = ceil(rand*population_y);
        if(rand<p_mutate && ~isempty(mutate_num))
            new_populatin(i,:) = population(i,:);
            new_populatin(i,mutate_num) =    new_populatin(i,mutate_num)+...
       (randn(1,length(mutate_num))+randn(1,length(mutate_num))*1i)*0.05;
        else
            new_populatin(i,:) = population(i,:);
        end
    end
    end
    function [best_individual, best_fit_value] =
Find_Best_Individual(popualtion,individual_fitness_value)
    [population_x,~] = size(popualtion);
    best_individual = popualtion(1,:);
    best_fit_value = individual_fitness_value(1);
    for i = 2:population_x
        if individual_fitness_value(i)>best_fit_value
            best_individual = popualtion(i,:);
            best_fit_value = individual_fitness_value(i);
        end
    end
    end
    function [out] = qamdemod_8qam(In,mod)
    switch mod
        case 8
            Symmbol_8qam = [1+1j -1+1j -1-1j 1-1j 1i*3 -1i*(1+3^(1/2)) (1+3^(1/2)) -
(1+3^(1/2))];
            for a=1:length(In)
                distance = abs(Symmbol_8qam-In(a));
                [~,Ind] = min(distance);
                out(a) = Ind-1;
            end
    end
    end
    function Out = qammod_new_constellation(xIn,ModSymmbol)
    Out = ModSymmbol(xIn+1);
    end
    function Out = qamdemod_new_constellation(In,ModSymmbol)
```

```
for i=1:length(In)
          Dis = abs(ModSymmbol-In(i));
        [~,Ind] = min(Dis);
             Out(i) = Ind-1;
end
end
```