# 基于 *NK* 模型下对$(n,k)$网络容错性的研究

## 摘　要

本文针对网络故障点判断和$(n,k)$网络的特性判断，建立模型求得故障点数以及子网络的数量。

针对问题一，首先利用 Excel 表格筛选故障顶点；找出被报告故障最多的五个测试顶点，假设这些为故障顶点，然后利用 Python 程序找出其他顶点对标记顶点的报告结果，筛选出被报告次数最多的故障顶点重复标记。

针对问题二，同问题一利用 Excel 表格筛选故障顶点；找出被报告故障最多的六个测试顶点，将其假设为故障顶点，同样利用程序确定其他无故障顶点对这些顶点的报告做出进一步推测。

针对问题三，通过题目已知$(n,k)$网络的特性以及相邻点的特性，利用公式计算出子网络的总顶点数，通过整个网络的规律结构判断子网络总数为 30，详细过程见模型建立与求解，由于子网络的连通性，破坏顶点即可使某些功能不完善，在已知子网络总数的情况下，求出至少破坏 30 个顶点。

针对问题四，同问题三的规律可直接求出子网络数量为 4080，判断子网络顶点数和连通度确定至少要破坏多少顶点。

针对问题五，通过子网络的规律性对模型进一步优化。

**关键词：** *NK* 模型　Python 技术　$(n,k)$ 星图

# 一、问题重述

问题一和问题二，在自诊断测试的过程中，首先需要确定哪些已识别的测试顶点可能存在故障。这一步骤的具体操作是：使用 Excel 对每个测试顶点对应的列数据进行求和操作，进而筛选出结果为-5 的顶点，将这些顶点作为潜在的故障候选。然后对于报告中提到的故障测试顶点，重点关注出现故障次数最多的前五个顶点。通过将这些故障频繁的顶点与那些未出现故障的顶点进行对比分析，可以识别出最有可能成为故障源的顶点。

问题三和问题四，基于冒泡排序网络的基本特性和原理，可以直接计算出该网络中包含的不同子网络的数量。通过对网络连通性的理解以及题目信息，我们可以确定破坏一个子网络至少需要破坏的顶点数量，从而得出破坏所有子网络所需的最少顶点数。

# 二、问题分析

## 2.1 问题一的分析

使用 Excel 表格找出有可能故障的顶点。具体步骤如下：

1. 通过 Excel 筛选，对数据进行排序，从而找出那些被报告故障次数最多的五个顶点。

2. 将这五个测试顶点单独记录。

然后判断其他测试顶点与这些被报告的故障顶点的关系，推断它们的报告的可靠性。采取以下步骤：

1. 对于剩余的每个顶点，收集它们针对已知故障顶点的报告记录。

2. 分析每个顶点的报告一致性和准确性，考虑以下因素：

   a. 如果一个顶点频繁地报告已知故障顶点有问题，则它可能是可靠的。

   b. 如果一个顶点很少报告已知故障顶点有问题，或者经常误报正常顶点为故障，则它可能是不可靠的。

3. 根据上述分析，判断每个顶点的报告是否可靠。

4. 使用这些判断作为依据，进一步推测那些未直接被报告为故障顶点的状态。即可具体分析确定有多少故障顶点。

### 2.2 问题二的分析

同问题一的步骤，在 Excel 中对数据集的每一列执行求和操作，以筛选出和为-6 的测试顶点。将这些疑似故障的测试顶点与记录中的其他未出现故障的顶点进行对比分析，从中挑选出报告中故障次数最多的前六个顶点作为主要候选。然后，通过进一步审查正常顶点对疑似故障顶点的报告情况，精准地识别出所需的关键测试顶点。

### 2.3 问题三的分析

针对问题三，利用邻近顶点的特性，从网络的任意一顶点着手。首先，随机选择六个数字中的两个数字予以移除，随后将剩余的四个数字组合成 24 个不同的四位数。接着，我们依据顶点间的邻接关系绘制出子网络模型。借助冒泡排序算法的基本属性和原理，我们能够直接计算出整个网络中的所有子网络。最后，通过分析这些子网络的特征，我们能确定至少需要破坏多少个顶点，以便破坏所有子网络。

### 2.4 问题四的分析

直接通过相邻顶点的关系式计算包含多少子网络。

## 三、模型假设

### 3.1 问题一和问题二假设

对于问题一和问题二，我们可以基于以下合理的假设来进行分析和处理：

1. 不考虑网络中边发生故障的概率。

2. 假设网络的连通性不是影响故障诊断的因素，即不考虑网络各部分之间的连接程度对故障判断的影响。

### 3.2 问题三和问题四假设

对于问题三和问题四，假设每个顶点发生故障的概率是一致的，且每个顶点相互独立

# 四、符号说明

| 符号 | 含义 |
| --- | --- |
| $u$ | 行的测试顶点 |
| $v$ | 列的测试顶点 |
| $B_{n,k}$ | $(\mathrm{n},k)$冒泡排序网络集 |
| $k$ | $(\mathrm{n},k)$网络的边 |
| $B_{n-j,k-j}$ | $(\mathrm{n},k)$冒泡排序网络的子网络集 |

# 五、模型的建立与求解

## 5.1 问题一和问题二模型的建立与求解

通过自诊断测试工作原理以及附件 3 和附件 4 的说明：$u$ 行 $v$ 列为 0 说明 $u$ 顶点与 $v$ 顶点不相邻，$u$ 行 $v$ 列为 1 表示 $u$ 顶点报告说 $v$ 顶点是无故障的，$u$ 行 $v$ 列为-1 表示 $u$ 顶点报告说 $v$ 顶点是故障顶点。

经过仔细分析附件 3，并运用 Excel 对各列数据进行求和，找出顶点数之和为-5 的测试顶点和其他测试顶点 $u$ 对 $v$ 无故障报告的顶点个数，总结出如图 5-1 和图 5-2 所示的数据表，由图 5-1 中数据可表示出测试顶点 $u$ 报告的测试顶点 $v$ 有故障的所有顶点数：

| | 1426 | 3416 | 4216 | 5416 | 6412 |
| --- | --- | --- | --- | --- | --- |
| -1 | 2416 | 1436 | 1246 | 1456 | 1462 |
| | 3426 | 2416 | 2416 | 2416 | 2416 |
| | 4126 | 4316 | 3216 | 3416 | 3412 |
| | 5426 | 5416 | 5216 | 4516 | 4612 |
| | 6421 | 6413 | 6214 | 6415 | 5412 |

图 5-1

由图 5-2 中数据可表示出测试顶点 $u$ 评价的测试顶点 $v$ 无故障的所有个数：

| 其它u对v评价无故障的个数 | | | |
|---|---|---|---|
| 2416 | 5 | 1456 | 3 |
| 3426 | 3 | 2416 | -5 |
| 4126 | 3 | 3416 | -3 |
| 5426 | 3 | 4516 | 3 |
| 6421 | 3 | 6415 | 3 |
| 1436 | 3 | 1462 | 3 |
| 2416 | -5 | 2416 | -5 |
| 4316 | 3 | 3412 | 3 |
| 5416 | -1 | 4612 | 3 |
| 6413 | 3 | 5412 | 3 |
| 1246 | 3 | | |
| 2416 | -5 | | |
| 3216 | 3 | | |
| 5216 | 3 | | |
| 6214 | 3 | | |

图 5-2

根据题意，我们可知故障顶点在这些和为-5 的顶点中，且是报错频率最高的，即可进行下一步，在这些顶点中找出故障报告数量最多的五个。利用 Python 编写的程序将这些顶点与其他所有顶点报告的故障次数进行了对比，并输出结果，程序如下：

```
# 计算几行几列

height = len(zhenlie)

width = len(zhenlie[0])


# 定义结果字典

map = dict()
```

```
# 每一行都作为 u 顶点，去遍历每一列，如果出现-1，记录故障的坐标
for i in range(height):
    noAdjacent = list()  # 不相邻的列号列表
    noFault = list()  # 无故障的列号列表
    fault = list()  # 故障的列号列表
    # 这一行的结果，也就是 u 行的结果
    result = dict()
    for j in range(width):
        # if zhenlie[i][j] == -1:
        #         fault.append(j)
        match zhenlie[i][j]:
            case 0:
                noAdjacent.append(j)
            case 1:
                noFault.append(j)
            case -1:
                fault.append(j)
    result["不相邻"] = noAdjacent
    result["无故障"] = noFault
    result["故障"] = fault
    map[i] = result
```

```
for key in map:
    print("结果分隔符========================")
    print(f"{key} 顶点的结果是")
    print(f"{key} 顶点报告的不相邻顶点:", map[key].get("不相邻"))
    print(f"{key} 顶点报告的无故障顶点:", map[key].get("无故障"))
    print(f"{key} 顶点报告的故障顶点:", map[key].get("故障"))
    fo = open("result.txt", "a")
    fo.write("结果分隔符========================\n")
    fo.write(f"{key} 顶点的结果是\n")
    fo.write(f"{key} 顶点报告的不相邻顶点: {map[key].get('不相邻')}\n")
    fo.write(f"{key} 顶点报告的无故障顶点: {map[key].get('无故障')}\n")
    fo.write(f"{key} 顶点报告的故障顶点: {map[key].get('故障')}\n")
```

输出结果如下图示例

```
结果分隔符========================
0 顶点的结果是
0 顶点报告的不相邻顶点: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 196, 197, 198, 199, 200, 201, 202, 203, 204,
205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240,
241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277,
278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310]
0 顶点报告的无故障顶点: [60, 132, 195, 256]
0 顶点报告的故障顶点: []
结果分隔符========================
1 顶点的结果是
1 顶点报告的不相邻顶点: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 132, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 197, 198, 199, 200, 201, 202, 203, 204,
205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240,
241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277,
278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310]
1 顶点报告的无故障顶点: [61, 133, 196, 255]
1 顶点报告的故障顶点: []
结果分隔符========================
2 顶点的结果是
2 顶点报告的不相邻顶点: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
131, 132, 133, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 198, 199, 200, 201, 202, 203, 204,
205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240,
241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277,
```

图 5-2　Python 输出结果

根据输出结果以判断是否其他顶点也针对该顶点提交了故障报告。

随后，我们对这五个故障顶点分别构建了两个假设情境：一是假设该测试顶点是故障的，二是假设该顶点是正常的。在每个假设下，我们追溯并分析了与该顶点相邻的测试顶点，并反向考察这些相邻顶点所报告的故障数量。

通过比较两种假设情况下的数据，我们能够计算出各种情况发生的概率，并最终确定哪种情况的可能性更高。这样，我们就可以根据概率大小来判断该测试顶点是否真正存在故障，进而找出与其相关的相邻测试顶点。

### 5.3 问题三和问题四模型的建立与求解

$(\mathrm{n},k)$ 冒泡排序网络在 $k\geq 2$ 时具有如下性质[1]：

通过删除 $B_{n,k}$ 中的所有 $k$ -边，可以将 $B_{n,k}$ 划分为 $n$ 个不相交的 $B_{n-1,k-1}$ 子网络。

由此性质，可以将 $B_{n,k}$ 划分为 $n(n-1)...(n-j)$ 个不相交的 $B_{n-j,k-j}$ 子网络，所以，所求的 $(6,4)$ 包含着 6*5=30 个不同的 $(4,2)$ 子网络，$(17,8)$ 网络中包含 17*16*15=4080 个 $(14,5)$ 子网络，根据实践，一个 $(\mathrm{n},k)$ 网络所包含的所有 $(n-j,k-j)$ 子网络都被破坏时，它的一些功能就不能实现。一个 $(n-j,k-j)$ 子网络被认为受到破坏，那么它的至少一个顶点被破坏，所以想要破坏所有 $(4,2)$ 子网络至少需要破坏 30 个顶点，要破坏所有的 $(14,5)$ 子网络需要破坏至少 4080 个顶点。

## 六、模型的分析与检验

### 6.1 模型分析

在 $NK$ 模型中，问题的解表示为一个由 $N$ 个位组成的二进制串，每个位的值为 0 或 1，代表一个特征或决策变量。这个二进制串构成了解空间，即所有可能的解的集合。其中 $N$ 个位之间存在相互作用。每个位的适应性受到它本身以及 $K$ 个其他位的影响。这种相互作用通过网络的连接来建模，其中每 $N$ 个位与其他 $K$ 个位相连。对于 $(n,k)$ 网络容错性的模型分析和检验，需要综合考虑多个方面，包括网络拓扑结构、可靠性、故障检测和恢复、容错策略、网络结构优化以及容错算法研究等。

### 6.2 模型检验

1. 模拟实验：通过不同的实验来测试$(n,k)$网络的性能。

2. 实际测试：在实际例题中使用$(n,k)$网络模型进行判断$(n,k)$网络的实用性。

3. 数据分析：将模拟实验和实际测试的数据作为依据进行进一步分析，整体评估$(n,k)$网络的优异之处。

4. 对比分析：将$(n,k)$网络的与其他网络的优缺点进行对比分析，进一步完善模型的可用性。

## 七、模型的评价与改进

### 7.1 模型的优点

1. 灵活性：*NK* 模型能够通调整参数 *N* 和 *K* 来模拟不同的问题

2. 简洁性：模型简单，易于学习和理解，可以快速地进行计算和分析。

3. 普适性：*NK* 模型广泛应用于生物学、经济学、工程学等多个领域，具有很强的普适性和跨学科的研究价值。

### 7.2 模型的缺点

1. 过于简化：由于 *NK* 模型是一个抽象的数学模型，它可能无法完全捕捉到现实世界复杂系统的所有细节和特性。

2. 参数限制：模型的性能很大程度上依赖于参数 *N* 和 *K* 的选择，不恰当的参数设置可能导致模型失效或得出错误的结论。

### 7.3 模型的改进

1. 参数优化：通过算法如遗传算法或者贝叶斯优化等方法来寻找最优的 *N* 和 *K* 参数，以提高模型的预测准确性。

2. 结合其他模型：可将 *NK* 模型尝试与其他模型结合，进一步利用各自的优点提

高模型的整体利用率。

3. 引入动态机制：对 *NK* 模型进行扩展，加入时间维度或者状态变量，使其能够描述系统的动态变化。

4. 实证研究：通过更多的实证研究来验证和细化模型，确保模型能够在特定领域内提供准确的预测和解释。

5. 多目标优化：在适应度景观中考虑多个目标的优化，而不仅仅是单一目标，这样可以更好地模拟现实中的复杂情况。

# 八、参考文献

[1]冯凯,马鑫玉.$(n,k)$-冒泡排序网络的子网络可靠性[J].计算机科学,2021,48(04)

[2]张国珍,杨伟丽.$(n,k)$-冒泡排序网络的结构连通度和子结构连通度[J].山西大学学报(自然科学版),2022,45(02)

[3]张国珍,杨伟丽 $(n,k)$-星图网络的$t/m$-诊断度及诊断算法[J].高校应用数学学报 A 辑,2018,33(03)

[4]冯凯,杨崙迦.$(n-m,k-m)$-星图子网络的可靠性评估[J].山西大学学报(自然科学版),2024,0506

[5]郭玉红.(n,k)-排列图的自同构群[J].北京交通大学,2021 年第 03 期