## **一、问题重述**

​ 某医院每天各时间段内需要的值班护士数如表 1 所示：

|  |  |
| --- | --- |
| 时间区段 | 护士数量 |
| 6：00-10：00 | 18 |
| 10：00-14：00 | 20 |
| 14：00-18：00 | 19 |
| 18：00-22：00 | 17 |
| 22：00-6：00(次日) | 12 |

每名护士每周上 5 个班，并被安排在不同的日子，由一名总护士长负责护士的值班安排。值班方案要做到在人员或经济上比较节省，又做到尽可能合情合理。下面是一些正在考虑中的值班方案：

​方案 1：每名护士连续上班 5 天，休息 2 天，并从上班第一天起按从上第一班到第五班顺序安排。

​方案 2：考虑到方案 1 中每名护士在周末（周六、周日）两天内休息安排不均匀，于是规定每名护士在周六、周日两天内安排一天、且只安排一天休息，再在周一至周五期间安排 4 个班，同样上班的 5 天内分别顺序安排 5 个不同班次。

​在对方案 1、2 建立线性规划模型并求解后发现，方案 2 虽然在安排周末休息上比较合理，但所需值班人员要比方案 1 有较多增加，经济上不太合算，于是又提出了第 3 方案。

​方案 3：在方案 2 的基础上，动员一部分护士放弃周末休息，即每周在周一至周五间由总护士长给安排三天值班，加周六周日共上五个班，同样五个班分别安排不同班次。作为奖励，规定放弃周末休息的护士，其工资和奖金总额比其他护士增加 a%。

​ 根据上述方案，帮助总护士长分析研究：

​ （1）对方案 1、2 建立使值班护士人数为最少的线性规划模型并求解。

​（2）对方案 3，同样建立使值班护士人数为最少的线性规划模型并求解，然后回答 a 的值为多大时，第 3 方案较第 2 方案更经济。

**2 问题分析**

* **排班约束复杂**：每名护士必须在一周内恰好工作 5 天，每天只能上一班，且要依次轮完 5 个班次(1→2→3→4→5)。
* **周末休息方式**：

方案1 不作规定；

方案2 要求周末只休 1 天、上 1 天；

方案3 则允许部分护士“周末都上”，并给出额外补偿。

* **覆盖需求**：每天分 5 个时间段，各自最少需要若干护士；由于班次有重叠，需要将相应班次的人数之和达到最低需求。

为求解这一问题，可采用整数线性规划：

1. **模式枚举**(列出所有可行的“7 天排班方式”)；
2. 决策每种模式使用多少名护士；
3. 通过对每日时段的覆盖约束，满足需求；
4. 最小化总人数（或综合成本）。

**3 模型假设**

1.每名护士一周工作 5 天、休 2 天，总天数按 7 天计(周一到周日)。

2.一周内工作 5 天分别对应 5 个不同班次(依次 1~5)，无跨日接续问题。

3.周末限定规则仅影响“周六、周日”的休息或上班方式：

* 方案2：周六、周日休 1 天、上 1 天；
* 方案3：部分护士主动周六、周日都上(2 天都工作)。

4.薪资模型简化：

* 方案1、2 护士的成本(工资)统一记为 1(基准)；
* 方案3 中，“放弃周末休息”的护士成本为 1+a/100。

5.不考虑换班间隔、个人最大连续工作时长等更细节的限制（可在后续扩展）。

**4 模型建立和求解**

4.1 模型概念与变量设置

1.每日需求

医院每天需要覆盖 5 个主要时段，分别是：

* 06:00~10:00（需求：18 人）
* 10:00~14:00（需求：20 人）
* 14:00~18:00（需求：19 人）
* 18:00~22:00（需求：17 人）
* 22:00~06:00（次日）（需求：12 人）  
  这些时段的覆盖需要依赖 5 个 8 小时班次的交叠：

1. 班次1：02:00~10:00
2. 班次2：06:00~14:00
3. 班次3：10:00~18:00
4. 班次4：14:00~22:00
5. 班次5：18:00~02:00（次日）

**2.一周工作周期**

· 将一周视为 7 天（周一～周日）。每名护士每周工作 5 天，休息 2 天，且在工作 5 天中**依次**执行班次 1→2→3→4→5，不允许跳班或重复班次。

这样一来，当我们考虑某个具体护士在一周 7 天如何排班，就可以看成“在 7 天中选出 5 天、按照班次1→2→3→4→5 上班，另 2 天休息”。

**3.模式（排班方式）**  
由于“5 天上班、2 天休息”且“5 天分别是班次 1~5”，我们可以列举所有**合法**的 7 天序列，并称每种序列为一个“模式（pattern）”。举例：

· 模式1：

* + 周一：班次1
  + 周二：班次2
  + 周三：班次3
  + 周四：班次4
  + 周五：班次5
  + 周六：休息
  + 周日：休息

模式2：

* + 周三：班次1
  + 周四：班次2
  + 周五：班次3
  + 周六：班次4
  + 周日：班次5
  + 周一：休息
  + 周二：休息

这样枚举可得到一系列模式 {p}，每个模式规定了“该护士在一周 7 天里某天上哪个班次”，以及哪些天休息。

**4.决策变量**  
令xp=“选择（采用）模式 p 的护士数量”,p=1,2,…,Nx\_其中 N表示“所有可行模式”的总数。

* 每个 xp 都是一个非负整数，表示有多少护士同时使用了该模式。
* 模式库中包含所有满足“5天排班、2天休、顺序班次1~5”以及“周末是否有特殊要求”的排法。

**5.覆盖系数（COV）**  
 为了把“模式 p 在第 d 天是否上班、上哪个班次”与“时段需求”关联起来，需要定义一个覆盖系数矩阵。

·COV(p,d,s)∈{0,1}，表示“模式 p 是否在第 d 天上班次 s”。若是，则为 1，否则为 0。

·因为一个班次 s 对应一定时间范围（比如班次1=02:00-10：00）可能需同时依赖两个班次叠加覆盖。因此在实际写约束时，常用类似

p∑​[COV(p,d,班次1或2)⋅xp​]≥需求d,(6:00−10:00)​,

或简化为“对于 (d, s) 自己的需求”时，

∑p​COV(p,d,s)xp​≥DEM(d,s)。

4.2 不同方案的额外限制

1.**方案1**：

· 不对周末的休息方式作额外限制。只要保证 5 天上 1~5 班次、2 天休息即可。故所有符合“5 上班+2 休”顺序排班的模式都在模式库内。

· 最终目标：min∑p​xp​。

1. 方案2：

在方案1的基础上，要求“周六、周日只休 1 天、工作 1 天”。因此任何模式 p 若周六、周日都休或都上，则不符合方案2，须排除出模式库。只保留那些“周六上班、周日休”或者“周六休、周日上”的模式。

·目标同样是min∑p​xp​，但模式库更小、更严格

1. 方案3

允许一部分护士(A类)“周六、周日都上班”，另一部分(B类)仍按方案2(周末只上一天)。

将两类模式分为 A 类模式集、B 类模式集：

· A 类成本 = 1+a/100 (因为需要额外补贴)；

· B 类成本 = 1。

若各类模式变量分别记为xpA​,xqB，目标则是

minp∈A∑​(1+100a​)xpA​+q∈B∑​1⋅xqB​.

同理构造覆盖约束：对每天每班次(或时段)，A类与B类模式一起叠加，满足需求

4.3 覆盖需求约束

**核心覆盖约束**可以写成：对每个“日 d（d=1\ldots7）”及“班次 s（s=1\ldots5）”，满足：

p=1∑N​[COV(p,d,s)]⋅xp​≥DEM(d,s),

其中 DEM(d,s)表示当天d、班次s最少需要多少名护士。若对时段需求与班次要合并(如“06:00~10:00”需求18人=班次1+班次2叠加)，可以写：

(p∑​COV(p,d,1)xp​)+(p∑​COV(p,d,2)xp​)≥18,

以表达“在06:00~10:00这个时段，来自班次1和班次2的护士数之和≥18”。

4.4 目标函数

**方案1、2**：若仅以最少护士人数为目标，则

minp=1∑N​xp​,并且所有

xp∈Z≥0x\_p \in \mathbb{Z}\_{\ge 0}xp​∈Z≥0​。

**方案3**（混合成本）：

min[p∈A∑​(1+100a​)xpA​+q∈B∑​1⋅xqB​].

仍保持整数非负约束：xpA,xqB∈Z≥0。

4.5 整体整数规划形式

总结而言，每个方案都可写成一个**整数线性规划**(ILP)。以方案1 为例：

1决策变量：

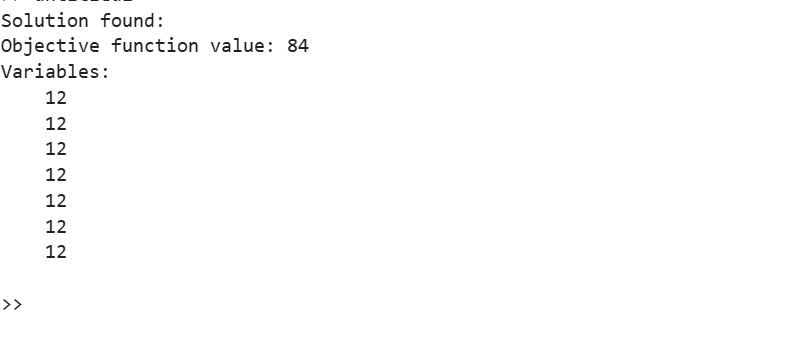
xp​∈Z≥0​,p=1,…,N.

1. **约束**：p∑​COV(p,d,s)xp​≥DEM(d,s),∀d=1,…,7,s=1,…,5.

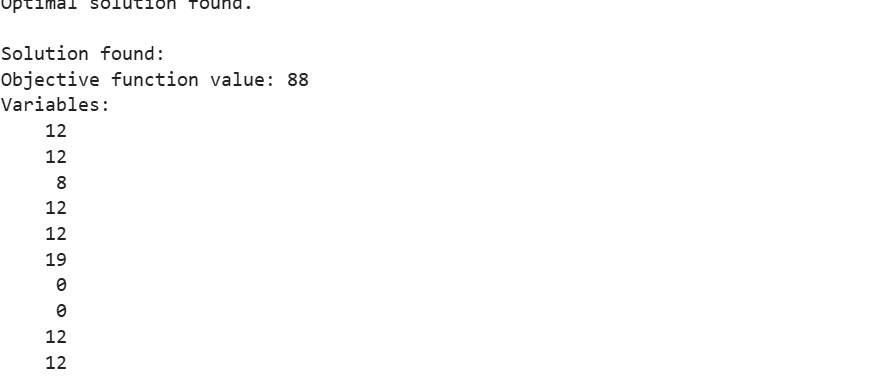
3.**目标函数**：minp∑​xp​.

**5 结果分析**

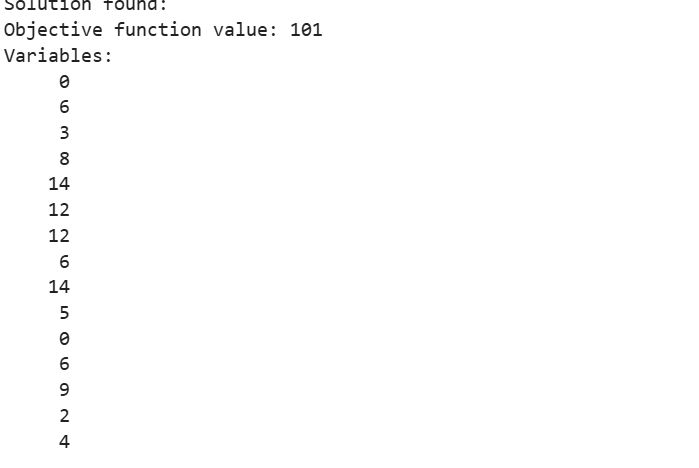
5.1 方案一的结果



5.2 方案二的结果



5.3 方案三的结果



3\*(1+a%)+64<=101

得a<=23.333%

6 附录

6.1 方案一代码：

%================= 方案一 =================

% Objective function coefficients

f = [1 1 1 1 1 1 1];

% Inequality constraints

A = [

-1 0 0 0 0 0 -1;

-1 -1 0 0 0 0 0;

0 -1 -1 0 0 0 0;

0 0 -1 -1 0 0 0;

0 0 0 -1 -1 0 0;

0 0 0 0 -1 -1 0;

0 0 0 0 0 -1 -1

];

b = [-20; -20; -20; -20; -20; -20; -20];

% Bounds

lb = [12; 12; 12; 12; 12; 12; 12];

ub = [];

% Integer constraints (all variables are integers)

intcon = 1:7;

% Options (optional)

options = optimoptions('intlinprog','Display','off');

% Solve the problem

[x, fval, exitflag, output] = intlinprog(f, intcon, A, b, [], [], lb, ub, options);

% Display the results

if exitflag == 1

disp('Solution found:');

disp(['Objective function value: ', num2str(fval)]);

disp('Variables:');

disp(x);

else

disp('No feasible solution found.');

end

6.2 方案二代码

%================= 方案二 =================

A = [

0, 0, 0, 0, 0, 1, 0, 0, 0, 1;

0, 0, 0, 0, 0, 1, 1, 0, 0, 0;

0, 0, 1, 1, 0, 0, 0, 0, 0, 0;

0, 0, 0, 1, 1, 0, 0, 0, 0, 0;

1, 0, 0, 0, 1, 0, 0, 0, 0, 0;

1, 1, 0, 0, 0, 0, 0, 0, 0, 0;

0, 1, 1, 0, 0, 0, 0, 0, 0, 0;

0, 0, 0, 0, 0, 0, 1, 1, 0, 0;

0, 0, 0, 0, 0, 0, 0, 1, 1, 0;

0, 0, 0, 0, 0, 0, 0, 0, 1, 1;

1, 1, 0, 0, 0, 1, 0, 0, 0, 1;

1, 0, 0, 0, 1, 0, 0, 0, 1, 1;

0, 0, 0, 1, 1, 0, 0, 1, 1, 0;

0, 0, 0, 1, 0, 0, 0, 1, 0, 0;

0, 0, 1, 0, 0, 0, 1, 0, 0, 0;

% ... 若图中还有更多行，请补充

];

b = [18, 20, 20, 19, 17, 18, 20, 20, 19, 17, 20, 20, 20, 12, 12];

% （若图中 b 含更多或更少元素，请按原图完整补齐）

lb = [12, 12, 0, 0, 12, 12, 0, 0, 12, 12];

% 同理，若原图中 lb 长度与变量数不同，请补齐

f = [1; 1; 1; 1; 1; 1; 1; 1; 1; 1; 1];

% 维度同变量数一致

% Solve the problem (用 linprog 做线性松弛示例；若需整数解可改 intlinprog)

[x, fval] = linprog(f, -A, -b, [], [], lb);

disp('Solution found:');

disp(['Objective function value: ', num2str(fval)]);

disp('Variables:');

disp(x);

6.3 方案三代码

%================= 方案三 =================

% 定义约束矩阵 A 和向量 b

A = [

-1, 0, 0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1;-1, 0, 0, 0, -1, 0, 0, 0, 0, 0, -1, -1, 0, 0, 0;

0, 0, 0, -1, -1, -1, -1, 0, 0, 0, -1, 0, 0, 0, -1;-1, 0, 0, 0, -1, 0, -1, -1, 0, 0, -1, -1, 0, 0, 0;

0, 0, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1, -1;

0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, -1, -1, 0;

0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1, -1;

0, 0, 0, 0, 0, -1, -1, 0, 0, 0, -1, 0, 0, 0, -1;

0, 0, 0, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0;

0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0;

0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0;

0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0;-1, -1, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0;

0, -1, -1, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0;-1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0;

0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0;

0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1;

0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0;

0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0;

0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0;

0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0;

0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0;

0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1;

0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0;

0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0;

% (请将截图中剩余行完整粘贴，保持行列对应)

];

b = -[

18, 20, 20, 19, 20, 20, 19, 17, 18, 20, 19, 17, 18, 20, 19,20,17, 12, 12, 12, 12, 12, 12, 12, 12

% (按截图中的数据补齐)

]';

% 定义目标函数系数向量 f

f = ones(15, 1);

% 定义变量下界 lb (因为所有 x\_j >= 0)

lb = zeros(15, 1);

lb(6) = 12; % x\_6 >= 12

lb(7) = 12; % x\_7 >= 12

% (如需要上界, 可用 ub, 否则可留空)

ub = [];

% 设置 intcon (若需整数解)

intcon = 1:15;

% Options (optional)

options = optimoptions('intlinprog','Display','off');

% Solve the problem

[x, fval, exitflag, output] = intlinprog(f, intcon, A, b, [], [], lb, ub, options);

disp('Solution found:');

disp(['Objective function value: ', num2str(fval)]);

disp('Variables:');

disp(x);