算法归约：现实世界调度问题中的网络流与 NP 完全性

1st Anay Sinhal佛罗里达大学盖恩斯维尔，美国
sinhal.anay@ufl.edu

2nd Arpana Sinhal*Manipal University斋浦尔斋浦尔，印度
arpana.sinhal@jaipur.manipal.edu

3rd Amit Sinhal*JKLakshmipat University*斋浦尔，印度
amit.sinhal@jklu.edu.in

4th Amit Hirawat*Poornima University*斋浦尔，印度
hiramit05@gmail.com

arXiv:2601.06737v1 [cs.DS] 11 Jan 2026

摘要—**本文通过多项式时间归约，介绍了两个现实世界调度问题及其算法解决方案。首先，我们研究医院患者分配病床问题，展示如何将其归约为最大二分图匹配问题，并通过网络流算法求解。其次，我们处理大学课程安排问题，通过从图着色进行归约证明其 NP 完全性，并提出贪心近似算法。两个问题均使用 Python 实现，实验结果验证了理论复杂度分析。我们的网络流解决方案达到了 $O(n^{2.51})$ 经验复杂度，而贪心着色算法表现出 $O(n^2)$ 特性，其近似比始终低于理论 $\Delta + 1$ 界。**

索引词—**网络流、NP完全、图着色、二分图匹配、调度算法、多项式归约**

I. 引言

资源分配与调度是医疗保健、教育和物流等众多领域面临的基本挑战。许多这类问题具有某些结构特性，因此可以通过成熟的算法范式加以解决；而另一些问题则已被证明难以求解。理解这些问题的计算复杂性，对于制定有效的解决策略至关重要。

本文考察了源自实际应用的两个不同调度问题：

1) **医院患者床位分配**：这是一个医疗保健运营问题，需要根据科室兼容性约束，将患者匹配到可用的医院床位。我们证明，该问题可以归约为最大二分图匹配，并且能够在多项式时间内使用网络流算法求得最优解。2) **大学课程安排**：这是一个学术排课问题，需要将课程分配到时间段，使选课学生有重叠的课程之间不发生冲突。我们证明，该问题是通过从图着色归约得到的NP完全问题，并提供贪心近似算法。

本文其余部分安排如下：第II节介绍网络流问题，第III节讨论NP完全问题，第IV节详细介绍实验验证，第V节总结并讨论研究结果。

通讯作者：Arpana Sinhal，Manipal University Jaipur，印度斋浦尔。
电子邮箱：arpana.sinhal@jaipur.manipal.edu。

II. 问题 1：通过网络流进行医院患者排程

A. 现实世界问题描述

考虑一家拥有多个专科科室的医院（例如心脏病学、骨科和神经病学），每个科室都配备数量有限的病床。需要住院的患者具有特定的疾病状况，而这些状况只能由某些科室进行治疗。医院管理部门必须为患者分配病床，并满足以下条件：

- 每位患者最多分配一张病床
- 每张病床最多容纳一位患者
- 患者只能被分配到兼容科室中的病床
- 入院患者总数达到最大

这个问题每天都会出现在医院运营管理中，并直接影响患者护理质量和资源利用 [1]。

B. 问题抽象

定义 1（医院分配问题）。 给定：

- 一组患者 $P = \{p_1, p_2, \ldots, p_n\}$
- 一组床位 $B = \{b_1, b_2, \ldots, b_m\}$
- 一组科室 $D = \{d_1, d_2, \ldots, d_k\}$
- 一个函数 $dept$： $B \to D$，将每张病床映射到所属科室
- 一个兼容关系 $C \subseteq P \times D$，其中 $(p, d) \in C$表示患者 p 可以由科室 d治疗

求一个分配 A： $P \to B \cup \{\emptyset\}$，满足：

1) 对于所有 $p \in P$：如果 $A(p) = b \neq \emptyset$，那么 $(p$，科室$(b))$ $\in C$2) 对于所有 $b \in B$： $|\{p \in P : A(p) = b\}| \leq 1$3) $|\{p \in P : A(p) \neq \emptyset\}|$ 达到最大值

这一抽象将医院排班问题转化为二分图匹配问题，我们要在患者与兼容床位之间寻找基数最大的匹配。

C. 归约为网络流

我们将医院分配问题归约为最大流问题，该问题可使用 Ford-Fulkerson 或 Edmonds-Karp 等算法在多项式时间内求解。

1) 构造：给定医院分配问题的一个实例（P, B, D,dept C），我们按如下方式构造一个流网络$G = (V, E, c, s, t)$：

顶点：

$$V = \{s, t\} \cup P \cup B \tag{1}$$

其中 s是源点，t 是汇点。

边和容量：

1) **源点到患者：** 对于每位患者 $p_i \in P$:

$$(s, p_i) \in E \text{ with } c(s, p_i) = 1 \tag{2}$$

2) **患者到兼容床位：**对于每个 $p_i \in P$ 和$b_j \in B$:

$$(p_i, b_j) \in E \text{ iff } (p_i, \text{dept}(b_j)) \in C \tag{3}$$

通过 $c(p_i, b_j) = 1$。

3) **床位到汇点：** 对于每张病床 $b_j \in B$:

$$(b_j, t) \in E \text{ with } c(b_j, t) = 1 \tag{4}$$

该构造如图1所示。

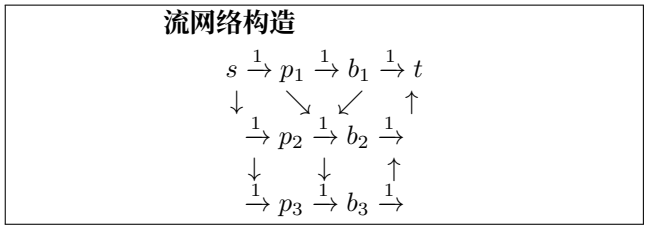

图1. 流网络构造示意图。只有当患者与床位的科室分配兼容时，患者与床位之间才存在边。所有容量均为1。

2）正确性证明：

定理1. 网络 G中的最大流等于医院分配问题中可接收的患者最大数量。

证明。 我们证明这种对应关系的两个方向：

(分配 $\Rightarrow$ 流)：设 A 是一个可接收 k 名患者的有效分配。我们按如下方式构造流 f：

- 对于每位已分配的患者 p_i，满足 $A(p_i) = b_j$:
  - Set $f(s, p_i) = 1$
  - Set $f(p_i, b_j) = 1$
  - Set $f(b_j, t) = 1$
- 所有其他边的流量均为 0

此流是有效的，因为：

1)容量约束： 所有非零流量均为 1，与边容量一致。

2) 流量守恒： 每个患者顶点 p_i 从 s 接收流量 1，并将流量 1 发送至恰好一张病床（若已分配）。每个床位顶点 b_j 接收的流量至多为 1（来自至多一名患者），并将其转发至 t。 3) **边的存在性：** 边 (p_i, b_j) 存在，因为 A 有效，这意味着 $(p_i, \text{dept}(b_j)) \in C$。

总流量等于 k，即入院患者的数量。

(流量 $\Rightarrow$ 分配)：令 f 为值为 k 的整数最大流。由于所有容量均为 1， f 的取值为 0 或 1。定义分配 A:

- For each patient p_i: if $\exists b_j$ such that $f(p_i, b_j) = 1$, set $A(p_i) = b_j$; otherwise, $A(p_i) = \emptyset$

此分配是有效的，因为：

1) 兼容性：如果 $f(p_i, b_j) = 1$，那么边 (p_i, b_j) 存在，这意味着 $(p_i$,dept$(b_j)) \in C$。2) **每位患者一张病床：**在 p_i 处的流量守恒确保流向病床的流量至多为 1 单位。

3) **每张病床一位患者：**在 b_j 处的流量守恒确保来自患者的流入量至多为 1 单位。

入院患者的数量等于流量值 k。

由于有效流与基数相等的有效分配一一对应，因此最大流等于最大分配。

□

多项式时间归约： 该构造在多项式时间内创建$O(|P| + |B|)$个顶点和 $O(|P| \cdot |B|)$条边，满足有效多项式时间归约的要求。

D. 完整算法

算法 1 使用 Ford-Fulkerson 的 Edmonds-Karp 变体，给出了完整的解决方案。

算法 1医院患者分配

要求： 患者集合 P，病床集合 B，兼容性 C

确保：最大 p患者分配g配

1：按说明构造流网络 G 2：初始化残余图 $G_f \leftarrow G$ 3：最大流 $\leftarrow 0$ 4：当存在从 s 到 t 的增广路径 p 于 G_f 中（通过广度优先搜索）时，执行 5：$\delta \leftarrow$ 最小值 $_{(u,v) \in p} c_f(u, v)$ 6：对于每条边 $(u, v) \in p$，执行 7：$c_f(u, v) \leftarrow c_f(u, v) - \delta$ 8：$c_f(v, u) \leftarrow c_f(v, u) + \delta$ 9：结束循环 10：最大流 $\leftarrow$ 最大流 $+ \delta$ 11：结束循环 12：从饱和的患者—床位边中提取分配 13：返回分配和最大流

时间复杂度： Edmonds-Karp 算法的运行时间为$O(VE^2)$。然而，对于单位容量二分图，其时间复杂度可降至 $O(VE) = O((|P| + |B|) \cdot |P| \cdot |B|)$。当 $|P| \approx |B| = n$时，这意味着 $O(n^3)$ [2]。

III. 问题 2：课程安排作为 NP 完全问题

A. 现实问题描述

大学面临着一项挑战：安排课程的时间段，使学生能够参加课程体系中的所有课程。如果两门课程有共同选课学生，它们就会冲突；冲突课程不能同时安排。目标是用尽可能少的时间段制定出可行的课表。

该问题直接影响学术机构中的学生满意度、毕业时间安排和资源利用 [3]。

B. 问题抽象

定义 2（课程安排问题（约束满足问题））。 给定：

- 一组课程 $C = \{c_1, c_2, \ldots, c_n\}$
- 冲突关系 *Conflict* $\subseteq C \times C$，其中 $(c_i, c_j) \in$ *Conflict* 表示课程 c_i 和 c_j 有共同的学生

求一个分配 σ： $C \to \{1, 2, \ldots, k\}$（其中 k 最小化），使得：

$$(c_i, c_j) \in \textit{Conflict} \Rightarrow \sigma(c_i) \neq \sigma(c_j) \tag{5}$$

这种抽象揭示了课程安排等价于图着色：课程是顶点，冲突是边，时间段是颜色，而我们要寻求色数。

C. NP 完全性证明

定理 2. 课程安排问题是 *NP* 完全的。

证明。 我们通过证明 (1) 课程安排问题 $\in$ 属于 NP，以及 (2) 图着色问题 $\leq_p$ 课程安排问题，来证明 NP 完全性。

第 1 部分：CSP $\in$ NP

给定一个排程 σ： $C \to \{1, \ldots, k\}$，我们可以通过检查是否有两门冲突的课程共用一个时间段，在多项式时间内验证其有效性：

$$\forall (c_i, c_j) \in \text{Conflict} : \sigma(c_i) \neq \sigma(c_j) \tag{6}$$

这需要进行 $O(|C|^2)$ 次比较，属于多项式时间。

第 2 部分：从图着色归约

图着色（判定图 G 是否可用 k 种颜色着色）是一个著名的 NP 完全问题 [4]。我们展示图着色 $\leq_p$ CSP。

构造：给定一个图着色实例（$G = (V, E)$ k）：

1）创建课程集合 $C = V$（每个顶点对应一门课程）2）定义冲突 $= E$（边转化为冲突）3）询问：能否在 $\leq k$ 个时间段内安排课程?

正确性：

断言 1。 G 可进行 k着色 $\Leftrightarrow$ C 可以使用 $\leq k$个时间段进行安排。

($\Rightarrow$) 设 χ： $V \to \{1, \ldots, k\}$ 是一个有效的 k-着色。对对应的顶点定义$\sigma(c_i) = \chi(v_i)$。由于 χ 有效，相邻顶点的颜色不同，因此相冲突的课程安排在不同的时间段。

($\Leftarrow$) 设 σ： $C \to \{1, \ldots, k\}$ 是一个有效的课程安排。定义$\chi(v_i) = \sigma(c_i)$。由于 σ 有效，相冲突的课程（相邻顶点）安排在不同的时间段（颜色）内。

多项式时间归约： 该构造耗时 $O(|V| + |E|)$，为每个顶点创建一门课程，并为每条边创建一个冲突。

由于图着色是NP完全问题，并且可在多项式时间内归约到约束满足问题，而约束满足问题 $\in$ NP问题，因此我们得出约束满足问题是NP完全问题。 □

D. 贪心近似算法

由于约束满足问题是 NP完全的，除非 P = NP，否则我们无法期待在多项式时间内得到最优解。我们提出了两种贪心启发式算法，可以高效地求得近似解。

1) Welsh-Powell 算法：Welsh-Powell算法按度数降序排列顶点，并为每个顶点分配当前可用的最小颜色。

算法 2 Welsh-Powell贪心着色

要求： 图 $G = (V, E)$
确保： 有效着色 $\chi : V \to \mathbb{Z}^+$
1: 按以下条件对顶点排序 度： $v_1, v_2, \ldots, v_n$ 其中 $\deg(v_1) \geq \deg(v_2) \geq \ldots$
2: **对于** $i = 1$ 至 n **执行**
3: 已使用 $\leftarrow \{\chi(u) : (v_i, u) \in E \wedge \chi(u)$ 已定义$\}$
4: $\chi(v_i) \leftarrow$ 最小值$\{c \in \mathbb{Z}^+ : c \notin$已使用$\}$
5: **结束循环**
6:**返回** χ

近似保证：Welsh-Powell 算法使用的颜色不超过$\Delta + 1$ 种，其中 Δ 是最大度数 [5]。

时间复杂度： $O(n^2+m)$ 其中 $n = |V|$ 和 $m = |E|$。

*2) DSatur*算法：DSatur（饱和度）通过动态选择饱和度最高的顶点（其邻域中不同颜色的数量），改进了 Welsh-Powell 算法。

算法 3 DSatur着色

要求： 图 $G = (V, E)$
确保： 有效着色 $\chi : V \to \mathbb{Z}^+$
1: 初始化所有 v **的饱和度 sat**$[v] \leftarrow 0$ **2:** $U \leftarrow V$ {**未着色顶点**}**3: 当** $U \neq \emptyset$ **时执行 4:** $v \leftarrow$ **arg max**$_{u \in U}$ **(sat**$[u]$**, deg(**u**))** {**按度数打破平局**}**5: used** $\leftarrow \{\chi($**u**$) :$ **(**v, u**)** $\in E \wedge \chi$**已定义**}**6:** χ**(**v**)** $\leftarrow$**min**$\{c \in \mathbb{Z}^+ : c \notin$ **used**$\}$**7:** $U \leftarrow U \setminus \{v\}$**8: 对** u **的每个邻居** v **执行** U **9: 使用新颜色更新 sat**$[u]$ **10: 结束 for 循环 11: 结束 while 循环 12: 返回** χ

时间复杂度： $O(n^2 + m)$，与 Welsh-Powell 算法相同。

DSatur 在实践中通常能产生更好的着色结果，尽管两者具有相同的最坏情况下的近似保证。

IV. 实验验证

我们用 Python 实现了这两种算法，并进行了计时实验，以验证理论复杂度分析。

A. 网络流实验

设置：我们生成了包含 n名患者、 n 张病床和5个科室的随机医院排班实例。每名患者都被随机分配至1—3个符合条件的科室。对于输入规模 $n \in \{20, 40, 60, \ldots, 400\}$，我们测量了执行时间，每种规模平均进行5次试验。

结果：表I总结了实验结果。所有实例均实现了完美匹配（所有 n 名患者均已分配），表明我们的随机实例具有充足的床位可用性。

表I 网络流实验结果

输入规模（n）	平均时间（秒）	平均匹配数
20	0.000434	20.0
40	0.001628	40.0
60	0.003998	60.0
80	0.008268	80.0
100	0.012558	100.0
150	0.035787	150.0
200	0.083915	200.0
250	0.157401	250.0
300	0.284309	300.0
350	0.500918	350.0
400	0.669159	400.0

复杂度分析： 双对数回归得出的多项式度约为**2.51**。对于单位容量二分图匹配，理论复杂度为 $O(VE)$。在我们的构造中：

- $V = O(n)$（患者 + 床位 + 源点 + 汇点）
- $E = O(n^2/d)$，其中 d 为科室数量

因此，预期复杂度为 $O(n \cdot n^2) = O(n^3)$。测得的指数 2.51 低于此界限，这可能是由于：1) 稀疏兼容图（每位患者仅与 1–3 个科室兼容）2) 找到完美匹配后提前终止 3) Edmonds-Karp 实现中的高效广度优先搜索遍历

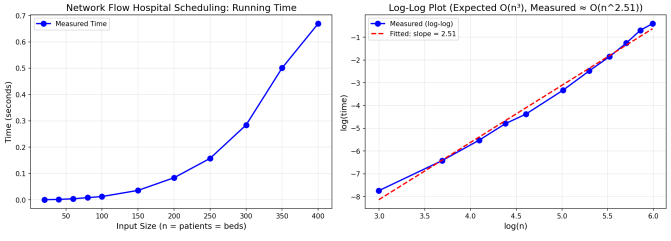

图 2. 网络流计时结果。左：显示多项式增长的原始计时数据。右：带拟合直线的双对数图（斜率 ≈ 2.51）。

B. NP 完全问题实验

设置：我们生成了包含n 门课程、边密度为 0.3（任意两门课程之间发生冲突的概率为 30%）的随机课程冲突图。输入规模从 50 门课程到 1000 门课程不等，每种规模进行 5 次试验。

结果：表 II 比较了两种贪心算法。

表 II 贪心着色实验结果

n	贪心（秒）	DSatur（秒）	G. 颜色	D. 颜色	Δ
50	0.0001	0.0005	8	7	22
100	0.0004	0.0016	13	12	42
200	0.0018	0.0068	21	19	78
300	0.0039	0.0157	29	26	114
400	0.0056	0.0240	35	32	144
500	0.0083	0.0346	42	39	181
600	0.0127	0.0513	49	46	215
700	0.0160	0.0674	56	51	252
800	0.0207	0.0919	61	57	281
900	0.0256	0.1149	67	63	315
1000	0.0330	0.1473	73	68	347

复杂度分析：双对数回归得出的指数为**1.85** （Greedy）和**1.92** （DSatur），与稠密图中理论上的 $O(n^2)$ 复杂度一致，其中$m = O(n^2)$。

近似质量： 两种算法使用的颜色都显著少于 $\Delta + 1$ 上界：

- Greedy： ≈ 73 种颜色，相较于 $n = 1000$的上界 348
- DSatur： ≈ 68 种颜色，相较于 $n = 1000$的上界 348

这表明，贪心启发式算法在实践中表现良好，远胜于最坏情况下的保证。

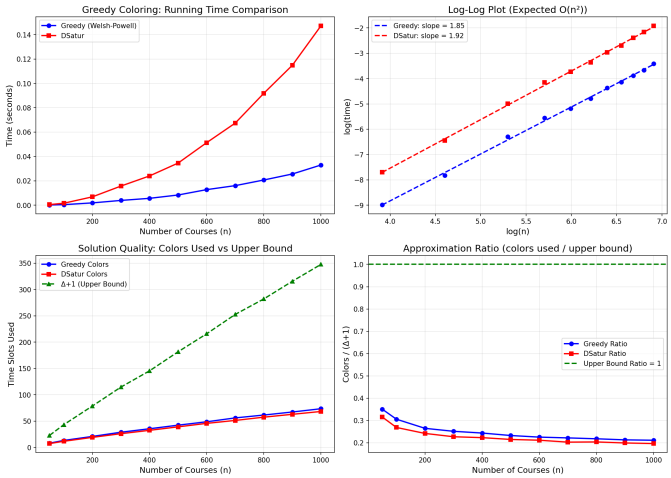

图 3. 贪心着色结果。左上：运行时间比较。右上：双对数复杂度分析。左下：所用颜色数与 $\Delta + 1$ 界限的比较。右下：近似比。

V. 结论

我们介绍了两个现实世界调度问题，并对其进行了严格的算法分析：

1) **医院患者床位分配**问题被归约为最大二分图匹配并得到解决

通过网络流实现。实验结果证实其复杂度接近立方级（$n^{2.51}$），对于 400 名患者/个床位，可在不到 0.7 秒内得出最优解。

2) 大学课程安排 问题已通过从图着色问题归约证明为NP完全问题。贪心近似算法（Welsh-Powell 算法和 DSatur）展现出二次复杂度，且其实用近似比远优于理论界。

这些结果展示了多项式时间归约的强大之处：识别问题结构，既能找到最优的多项式时间解，也能在求解最优解难以处理时获得有效的近似解。

参考文献

[1] M. L. Brandeau、F. Sainfort 和 W. P. Pierskalla，《运筹学与医疗保健：方法与应用手册》，施普林格科学与商业媒体，2004。 [2] T. H. Cormen、C. E. Leiserson、R. L. Rivest 和 C. Stein，《算法导论》，第 3 版，麻省理工学院出版社，2009。 [3] S. Even、A. Itai 和 A. Shamir，《关于时间表与多商品流问题的复杂性》，《SIAM 计算杂志》，第 5 卷，第 4 期，第 691–703 页，1976。 [4] R. M. Karp，《组合问题之间的可归约性》，载于《计算机计算的复杂性》，施普林格，1972，第 85–103 页。 [5] D. J. A. Welsh 和 M. B. Powell，《图的色数上界及其在编排课程表问题中的应用》，《计算机杂志》，第 10 卷，第 1 期，第 85–86 页，1967。

声明

资金支持： 作者未获得本研究的外部资金支持。

利益冲突： 作者声明不存在任何竞争性利益。

作者贡献： Anay Sinhal：概念化、方法论、软件、验证、实验以及撰写（初稿）。Arpana Sinhal：监督、概念化以及撰写（审阅与编辑）。Amit Sinhal：形式化分析、证明开发以及撰写（审阅与编辑）。Amit Hirawat：软件支持、文献支持以及撰写（审阅与编辑）。所有作者均审阅并批准了最终稿件。

伦理声明/批准： 不适用。本研究不涉及人类参与者、人类数据或动物。

知情同意： 不适用。

数据可用性： 未使用任何专有数据集。用于生成合成实例和复现实验的代码可在 https://github.com/anayy09/ Algorithmic-Reductions 获取。

数据共享： 该代码仓库包含用于生成合成实例和复现报告中计时结果的脚本。作者可在合理请求下提供其他材料。

附录

A. 网络流：*Edmonds-Karp* 算法

```
类 Ford-Fulkerson：定义 __初始化__(自身, n)：自身.graph = defaultdict(lambda：defaultdict(整数))定义 添加_边(自身, u, v, 容量):
        自身.graph[u][v] += 容量

定义 广度优先搜索(self,                源点          汇点, parent):
    已访问 = 集合([源点])
    队列 = 双端队列([源点])
    当 队列:
        u = 队列.pop左端()
        对于 v 在 self.graph[u]:
            如果v不在已访问且self.graph[u][v] >
                0:
                已访问.add(v)
                父节点[v] = u
                如果 v == 汇点:
                    返回 真
                队列.append(v)
    返回 假

定义 最大_流(self , 源点 , 汇点):
    父节点 = {}
    最大流量值 = 0
    当 self.bfs(source, sink, parent):
        路径_流 = 浮点数('inf')
        s = 汇点
        当 s != 源点:
            路径_流 = 最小(路径_流, self.graph[
                父节点[s]][s])
            s = 父节点[s]
        v = 汇点
        当 v != 源点:
            u = 父节点[v]
            自身的图[u][v] -= 路径_流
            自身的图[v][u] += 路径_流
            v = 父节点[v]
        最大_流_值 += 路径_流
        父节点 = {}
    返回 最大值 _flow_值
```

清单 1. 使用广度优先搜索的 Ford-Fulkerson（Edmonds-Karp）

B. 贪心图着色：*Welsh-Powell* 算法

```
类贪心课程调度器:
    定义__初始化__(自身，图):
        自身.图 = 图
        self.颜色 = {}
    定义 solve(self):
        n = self.graph.num_课程
        # 按度降序排序（戚尔士-鲍威尔启发式算法
            ）
        课程_按_度 = 排序(
            范围(n),
            键=lambda x: self.graph.get_度(x),
            reverse=True)

        对于 课程 在 课程_按_度:
            邻居_颜色 = 集合()
            对于 邻居 在 self.graph.get_邻居(课程
                ):
                如果 邻居 在 self.colors:
                    邻居_颜色.add(self.颜色[
                        邻居])
            # 分配最小的可用颜色
            颜色 = 0
            当 颜色 在 邻居_颜色:
                颜色 += 1
            self.颜色[课程] = 颜色

        返回 最大值(self.colors.values()) + 1
```

代码清单 2. Welsh-Powell贪心着色