数学建模课程研究性作业
基于 MILP 的 SRTP 选择优化

Abstract

SRTP (Student Research Training Program), 即大学生科研训练计划, 是高等学校培养的人才的一种计划，不仅要学习和掌握本专业的基本知识与技能，而且学习具备创造性地解决所学专业领域内理论和实践问题的基本能力是高等学校培养的人才的一种计划，不仅要学习和掌握本专业的基本知识与技能，而且学习具备创造性地解决所学专业领域内理论和实践问题的基本能力。对于大多数本科生来说，SRTP也是第一次进入科研的机会，那么，找到合适的项目和导师，以树立起对科研的兴趣、培养科研的基本素养就尤为重要，然而每一个老师的时间和精力都是十分有限的，他所擅长的学科子领域也是有限的。但是，往往，同一个学科子领域会有很多不同的老师在研究，但是这里并不是所有老师对于一个特定的老师来说都是满意的。

因此如何进行学生和老师、以及项目之间的匹配就十分重要。但是又由于资源的有限性，我们显然不可能保证所有的学生都能选到心仪的导师或者项目。不同于目前现实中所常用的先到先得的方式，下文中我们将给出一种基于混合整数线性规划 **MILP** 的模型，使得尽可能更多的学生能够选到能让他们心仪的导师或者项目（或者说使得总效用（**Total Utility**）最大化，在下文中我们用满意度来替代效用）。

除了给出了该模型以使得满意度最大化之外，我们还基于现实生活中的选择老师或者课程的规律，参考了一些成熟的权威数据库（例如，教育部教育统计数据，UCAS等等），并且参考了一些关于导师选择、科研项目选择相关的调查研究的结论，生成了一组数据，进行案例研究。

并用 python 对该案例进行了求解得到了结果。最后，我们将对该 $SRTP$ 匹配模型进行评价，并且将其和目前使用较多的其他 $SRTP$ 匹配算法就行比较，以阐释其优越性。同时我们也将对该模型的可优化空间进行进一步的讨论

Contents

1 模型建立

1.1 问题描述

SRTP 是众多学生参与科研的重要形式，其中，导师和研究方向的选择尤其重要。所有的学生都希望能够选到其中意的研究方向以及中意的导师，这在导师人数多于学生时往往是可行的，但是在中国众多的大学中，老师的人数往往少于学生（中国大学的师生比大概在 1：14 左右），这个时候让每个学生都选到中意的研究方向和导师是不现实的，由于每个老师只有一些特定的研究方向 (以浙江大学计算机科学与技术学院为例，某位老师可能的研究方向为机器学习、深度学习、计算机摄影学，显然他的研究方向不能涵盖计算机科学的所有方面，尤其是差异较大的领域，比如上面描述的这位老师基本上不存在如计算机系统、分布式计算、并行算法等等偏向硬件的研究方向)，并且只有有限的精力去指导学生。

因此，需要提出一种分配方案，将学生与导师相匹配，同时尽可能提高总体的学生的整体满意度，但没有超过老师所能指导的最大学生人数。

基于上述考量，该数学问题可以**重新描述为：**

将参加 $SRTP$ 的所有学生进行与导师相匹配，其中，每位导师有一些研究方向，在满足下列所有条件的情况尽可能令更多的学生满意:

(1) 每位老师所带的学生没有超过他能能带的学生的上限（更为严格的，我们可以有下限）

(2) 每位学生尽量能够匹配到中意的项目或者中意的导师其中的一个，以减少有学生最后没有 $SRTP$ 项目的情况

(3) 每位学生只能选择一个 $SRTP$ 项目，即有且仅有一个 $SRTP$ 导师，以及有且仅有一个研究方向（并且该方向也是该 $SRTP$ 导师的研究方向）

1.2 问题分析

在在将学生与导师进行匹配之前，学生应该被要求填写一份调查问卷以了解他们的中意的导师和中意的研究方向（并且这些偏好有已经被排序）。然后根据这两个标准来进行匹配，并且尽可能得保证能够让每个学生都能满足两个标准之一（而不是必须强制满足其中一个）, 以尽可能地提高学生地整体满意度。

另外，一些其他的 $SRTP$ 项目需要考量的因素，例如基于可用资金、$SRTP$ 项目等级、学生的研究能力基础等因素，由于能以获取或者难以量化，我们将不做考虑。

为简单起见，可以提出一个单一目标的数学规划公式，其中数学规划的目标是使得学生的总体满意度最大化。每个老师只能选择有限的学生，每个学生只能选择 1 个研究方向以及 1 位该研究方向的导师，可以通过约束条件来实现。

1.3 模型假设

基于上述对于问题的分析，在本文中，我们针对分配问题提出了一个用于匹配的混合整数线性规划模型，这个混合整数线性规划包括 0-1 变量：学生是否选择该导师，线性约束：学生只能选择一个研究方向以及一个包含该研究方向的导师，目标函数：使得尽可能多的学生能够选到中意的 $SRTP$ 项目或者 $SRTP$ 研究方向中的至少一个所录取（即 1.5 中的数模模型）。更进一步得我们将把目标函数更改为使得更多的学生能够选到满意度排序更高的项目，并将前模型中的结果作为约束条件，对结果进行进一步的优化（1.6 中的二次优化）。

为了方便起见，我们假设学生 i 的满意度 u_i：如果选到了中意的研究方向或者中意的导师中的至少 1 个，则满意度为 1，否则满意度为 0。如此，目标就可能抽象为 $max \quad \sum_{i \in S} u_i$

1.4 符号说明

符号	含义
T	SRTP 导师的集合
S	参加 SRTP 的学生的集合
P	研究方向的集合
a_{ij}	如果学生 i 最终选择老师 j, 则 $a_{ij} = 1$; 否则，$a_{ij} = 0$
α_j	分配给老师 j 的学生数量的下限
β_j	分配给老师 j 的学生数量的下限
TS_i	对于学生 i 所中意的导师的集合
PS_i	对于学生 i 所中意的研究方向的集合
T_P	对于研究方向中有研究方向 P 的老师的集合
$\widehat{\vartheta_i}$	松弛变量, 用于标记学生 i 是否被中意的 SRTP 导师录取
$\overline{\overline{\vartheta_i}}$	松弛变量, 用于标记学生 i 是否被中意的 SRTP 研究方向录取
u_i	学生 i 的满意度：如果选到了中意的研究方向或者中意的导师中的至少 1 个，则满意度为 1，否则满意度为 0
U	总满意度，满足 $U = \sum_{i \in S} u_i$
tw_{ij}	学生 i 选到导师 j 的满意度
tt	对每个学生收集了 tp 个排序的导师志愿
pw_{ip}	为学生 i 选到研究方向 p 的满意度
tp	表示对每个学生收集了 tp 个排序的研究方向志愿

1.5 模型建立

对于每个 $SRTP$ 导师

$$\forall j \in T \quad \alpha_j \leq \sum_{i \in S} a_{ij} \leq \beta_j$$

由于我们定义 a_ij 为如果学生 i 最终选择老师 j, 则 $a_{ij} = 1$; 否则，$a_{ij} = 0$，显然

$$\forall i \in S \quad \sum_{j \in T} a_{ij} = 1$$

又学生只能选一个项目，因此**不引入松弛变量时**，可以建立混合整数线性规划：

$$\max \sum_{i\in S} u_i$$
$$\begin{cases} \sum\limits_{j\in T} a_{ij} = 1 \\ \alpha_j \le \sum\limits_{i\in S} a_{ij} \le \beta_j \\ \sum\limits_{p\in PS_i} \sum\limits_{j\in T_p or j\in TS_i} a_{ij} = 1 \\ \overline{\overline{\vartheta}}, \widehat{\vartheta}, u_i, a_{ij} \in \{0,1\} \end{cases}$$
$$\forall i \in S, \forall j \in T, \forall p \in PS_i$$

为了满足学生只能选择一个 $SRTP$ 项目，即只能有至多一个选到的导师，以及至多一个选到的研究方面，我们可以引入两个松弛变量 $\widehat{\vartheta_i}$ 以及 $\overline{\overline{\vartheta_i}}$

实际上，对于 $\overline{\overline{\vartheta}}$ 和 $\widehat{\vartheta}$，我们可以放宽要求为

$$\overline{\overline{\vartheta}}, \widehat{\vartheta} \in \mathbb{R}^+$$

，但是最终由于 a_ij 为0-1 变量，于 $\overline{\overline{\vartheta}}$ 和 $\widehat{\vartheta}$ 会被约束为0-1 变量。为了方便运算，可以直接让于 $\overline{\overline{\vartheta}} \quad \widehat{\vartheta} \in 0,1$

首先，对于研究方向，有：

$$\begin{cases} \forall i \in S \quad \forall p \in PS_i \quad \overline{\overline{\vartheta}} \in \{0,1\} \quad \overline{\overline{\vartheta}} \le 1 - \sum\limits_{j\in T_p} a_{ij} \\ \forall i \in S \quad \overline{\overline{\vartheta}} \in \{0,1\} \quad \overline{\overline{\vartheta}} \ge 1 - \sum\limits_{p\in PS_i} \sum\limits_{j\in T_p} a_{ij} \end{cases}$$

如果学生没有匹配到 PS_i 中的一个，则 $\sum\limits_{j\in T_p} a_{ij} = 0$, 由于两个不等式的夹逼

$$\begin{cases} \overline{\overline{\vartheta}} \le 1 \\ \overline{\overline{\vartheta}} \ge 1 \end{cases}$$

因此 $\overline{\overline{\vartheta}} = 1$, 另一方面，如果 $\overline{\overline{\vartheta}} = 1$，则由不等式

$$\overline{\overline{\vartheta}} \le 1 - \sum_{j\in T_p} a_{ij}$$

可知，学生 i 一定没有选到中意的研究方向

如果学生匹配到了 PS_i 中的一个，则 $\sum\limits_{j\in T_p} a_{ij} = 1$, 由于两个不等式的夹逼

$$\begin{cases} \overline{\overline{\vartheta}} \le 0 \\ \overline{\overline{\vartheta}} \ge 0 \end{cases}$$

因此 $\overline{\overline{\vartheta}} = 0$, 另一方面，如果 $\overline{\overline{\vartheta}} = 0$，则由不等式

$$\overline{\overline{\vartheta}} \ge 1 - \sum_{p\in PS_i} \sum_{j\in T_p} a_{ij}$$

可知，学生 i 一定选到了中意的研究方向

并且，由于 $\overline{\overline{\vartheta}} \in \{0,1\}$，这就保证了任意一个学生 i 只能选到一个项目，否则，就会出现 $\overline{\overline{\vartheta}}$ 为负数。

其次，对于导师，有：

$$\forall i \in S \quad \widehat{\vartheta} \in \{0,1\} \quad \sum_{j\in TS_i} a_{ij} + \widehat{\vartheta} = 1$$

如果学生 i 没有被分配给任何 TS_i 中的以为，则 $\sum\limits_{j\in TS_i} a_{ij} = 0$, 则显然会有 $\widehat{\vartheta} = 1$, 否则 $\widehat{\vartheta} = 0$

之后，我们定义 u_i 为学生的满意度, 按照模型假设中的那样，如果选到了中意的研究方向或者中意的导师中的至少 1 个，则满意度为 1，否则满意度为 0。我们的目标 (object) 可以变为总满意度 $U = \sum\limits_{i\in S} u_i$ 最大化

为了，得到单个学生的满意度 u_i，可以引入下列的式子来进行夹逼：

$$\begin{cases} 2-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i \\ 1-\overline{\overline{\vartheta}}\leq u_i \\ 1-\widehat{\vartheta}\leq u_i \end{cases} \qquad \forall i\in S$$

由于，只要选到中意的研究方向，$\overline{\overline{\vartheta}}=0$, 由于

$$1-\overline{\overline{\vartheta}}\leq u_i$$

使得其满意度大于等于 1；只要选到了中意的导师，$\widehat{\vartheta}=0$, 由于

$$1-\widehat{\vartheta}\leq u$$

使得其满意度大于等于 1。同时有约束

$$2-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i$$

，这就使得 $1\leq u_i$；如果中意的研究方向和中意的导师同时选到，那么 $1\leq u_i\leq 2$, 在目标为最大化满意度的条件下，$u_i=2$; 如果只选到其中一个，则由夹逼，得到 $u_i=1$。(如果要求只选到其中一个，可以把 $2-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i$ 改为 $1-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i$，但是这种情况下如果某个学生只有一个中意导师和中意方向并且他们重合，由于约束，最终 $u_i=0$, 这样，同时选到中意导师和中意方向但是满意度却降低了，这是不符合现实条件的)

如果都没有选到则 $\widehat{\vartheta}=\overline{\overline{\vartheta}}=0$, 可以由

$$2-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i$$

得到 $u_i=0$

考虑到我们的目标是让学生能够选到中意的导师或者研究方向中的其中一个即可，可以再增加 $u_i\in\{0,1\}$ 作为约束

基于上述描述，**最后可以得到下述混合整数线性规划模型 MILP I：**

$$\max\sum_{i\in S}u_i$$

$$s.t.\begin{cases} \sum\limits_{j\in T}a_{ij}=1 \\ \alpha_j\leq\sum\limits_{i\in S}a_{ij}\leq\beta_j \\ \sum\limits_{j\in TS_i}a_{ij}+\widehat{\vartheta}=1 \\ \overline{\overline{\vartheta}}\leq 1-\sum\limits_{j\in T_p}a_{ij} \\ \overline{\overline{\vartheta}}\geq 1-\sum\limits_{p\in PS_i}\sum\limits_{j\in T_p}a_{ij} \\ 2-\overline{\overline{\vartheta}}-\widehat{\vartheta}\geq u_i \\ 1-\overline{\overline{\vartheta}}\leq u_i \\ 1-\widehat{\vartheta}\leq u_i \\ \overline{\overline{\vartheta}},\widehat{\vartheta},u_i,a_{ij}\in\{0,1\} \end{cases}$$

$$\forall i\in S,\forall j\in T,\forall p\in PS_i$$

1.6 二步优化

上述模型我们可以保证使得尽量多的学生能够被分配到中意的 $SRTP$ 研究方向或者项目导师中的至少其中一个，以更加体现公平性，让尽可能多的学生能提高满意度。

但是，我们在求解该问题的时候，考虑到现实情况中，人的喜好顺序往往是有优劣的，换言之，对某样事务的偏好，实际上是可以排序的——如果能偶匹配到更加偏好的项目或者导师，往往能够进一步

提高满意度，因此，我们在上述模型考虑公平性的情况下，提高 $SRTP$ 项目匹配的总体满意度的前提下，可以对模型进行进一步优化。

正如上所述，可以定义 tw_{ij} 为学生 i 选到导师 j 的满意度，为了简单起见，用 tt (total teacher) 表示对每个学生收集了 tt 个排序的导师志愿，如果老师 j 是该学生的第 1 志愿，则 $tw_{ij} = tt$，如果老师 j 是该学生的第 2 志愿，则 $tw_{ij} = tt - 1$，....., 老师 j 是该学生的第 $tt - 1$ 志愿，则 $tw_{ij} = 1$，如果一个老师都没有选到，则 $tw_ij = 0$

同理，定义 pw_{ip} 为学生 i 选到研究方向 p 的满意度，用 tp (total project) 表示对每个学生收集了 tp 个排序的研究方向志愿，如果研究方向 p 是该学生的第 1 志愿，则 $tw_{ij} = tt$，如果研究方向 p 是该学生的第 2 志愿，则 $pw_{ip} = tp - 1$，....., 研究方向 p 是该学生的第 $tp - 1$ 志愿，则 $pw_{ip} = 1$，如果一个老师都没有选到，则 $pw_{ip} = 0$

此时为了不改变原先上述方程求出的结果，我们可以将原来方程求出的结果作为约束条件，定义 $\widehat{U} = \max \sum_{i \in S} u_i$ 为前一模型计算得到的最大总满意度，我们可以得到一个**新的混合整数线性规划模型，对上述解进行进一步的优化 MILP II:**

$$
\begin{aligned}
&\max \sum_{i \in S} \sum_{j \in TS_i} tw_{ij} a_{ij} + \sum_{i \in S} \sum_{p \in PS_i} \sum_{j \in T_p} pw_{ip} a_{ij} \\
&s.t. \begin{cases}
\widehat{U} = \sum_{i \in S} u_i \\
\sum_{j \in T} a_{ij} = 1 \\
\alpha_j \le \sum_{i \in S} a_{ij} \le \beta_j \\
\sum_{j \in TS_i} a_{ij} + \widehat{\vartheta} = 1 \\
\overline{\overline{\vartheta}} \le 1 - \sum_{j \in T_p} a_{ij} \\
\overline{\overline{\vartheta}} \ge 1 - \sum_{p \in PS_i} \sum_{j \in T_p} a_{ij} \\
2 - \overline{\overline{\vartheta}} - \widehat{\vartheta} \ge u_i \\
1 - \overline{\overline{\vartheta}} \le u_i \\
1 - \widehat{\vartheta} \le u_i \\
\overline{\overline{\vartheta}}, \widehat{\vartheta}, u_i, a_{ij} \in \{0, 1\}
\end{cases} \\
&\forall i \in S, \forall j \in T, \forall p \in PS_i
\end{aligned}
$$

1.7 模型求解

上述问题可以抽象为混合整数线性规划，而求解混合整数线性规划的算法通常使用基于 LP 的分支定界算法 $branch\ and\ bound\ algorithm$ 来解决（算法思路如 Figure 1 所示）。就是不断分支，不断定界的过程。当松弛问题没有可行解或整数解大于当前下界（默认是最小值问题），那么就剪断此支（incumbent 需要分支的节点；fathomed 不必再分支的节点）整数解为上界，不断更换。同时有个有效下界，不断缩小上下界 Gap 的过程，就是寻优的过程。当上下界相等，那么就找到了最优解。

同时对于混合整数线性规划问题，可以通过预处理 (Presolve)，割平面 (Cutting Planes)，启发式算法 (Heuristics)，平行求解 (Parallelism) 来提升求解效率。

其中，精确算法能够求得模型的精确最优解，但其缺点在于在现有计算机技术下、在有限的计算时间内无法处理决策变量较多的问题。而启发式算法虽然能够处理决策变量较多的问题，但其得到的最优解为近似最优解，且比较容易陷入局部最优解，所求得的近似最优解与实际最优解之间的差距无法衡量和估计。

在 MatLab 中，线性规划类问题的求解基本上有两种解决方案，最简单的是直接调用求解器 (solver)

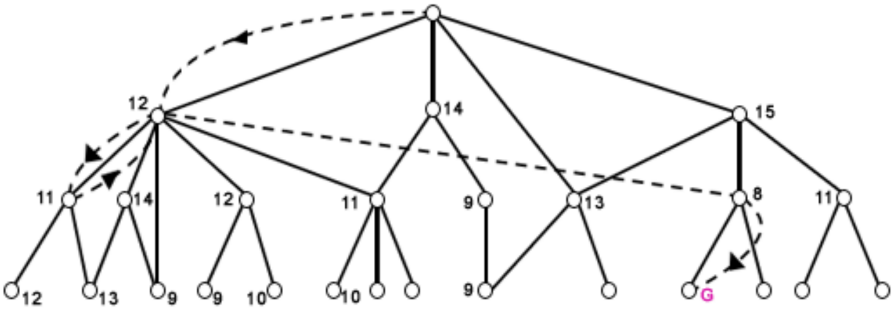

Figure 1: branch and bound algorithm

求解，这种方案简单，但需要我们手动列出所有系数矩阵、向量, 但是当约束增多，这个工作几乎是不可行的。另一种是基于问题的求解方案 (problem based linear programming)。这种方案更加直观，缺点是需要自己一步步实现，它实际上上也是调用了求解器，使用单纯形法、内点法等方法求解 (可以指定)。

但是本模型数据、变量、限制条件往往比较庞大，并且混合了 0-1 规划、整数规划和线性规划，因此采用 **GLPK** (GNU Linear Programming Kit) 求解器是较优的方法，本文第三节的案例分析所给出的求解器也是在 python 中利用了 GLPK 求解器。

2 案例分析

2.1 数据说明

用于案例分析的数据我们采用了随机数生成函数来生成[1]，并且考虑一些现实的因素来使得生成的数据更加逼真，例如，**生师比** (中国的生师比大概为 14:1, 但是往往有超过一般的同学不会参与 $SRTP$, 因此我们进行模拟的生师比低于该值)，**每个老师拥有的研究数量** (以浙江大学教师个人主页为例，一般为 3-4 个，该生产器生成函数为 4 个)，**热门老师和冷门老师**的出现的情况、由于精力和资源分配导致的老师具有**指导学生数量的上限以及下限**等等，详细的生成函数代码 $dataGenerator.ipynb$。

如图 Figure 2 所示，展示了所有的学生对于研究方向和科研导师的喜爱程度的分布情况，其中 $top1$ 表示该研究方向是该第一中意的研究方向，$tutor1$ 表示该导师是该学生第一中意的导师（其余数字标注以此类推）。科研看到，这样的分布是不那么均匀的，也比较符合现实生活中的情况。

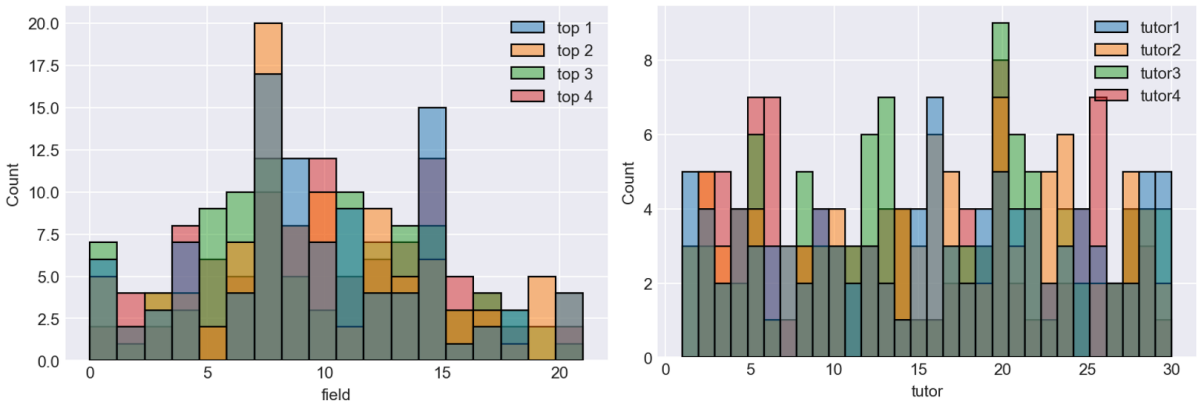

Figure 2: field preference and tutor

[1]之后所使用的数据可以见于附件中的数据集，其文件组织在附录中有所说明，当然，你也可以使用我们提供的随机生成函数来生成一组新的数据

2.2 MILP I

如果不引入松弛变量[2]，其求解代码可以见于附件 $MILP-I-without-slack.ipynb$ 或附录，而采用我们上述模型的求解代码可以见于附件 $MILP-I$ 或附录。其求解结果的分布分别可见于图 Figure 3 以及图 Figure 4。

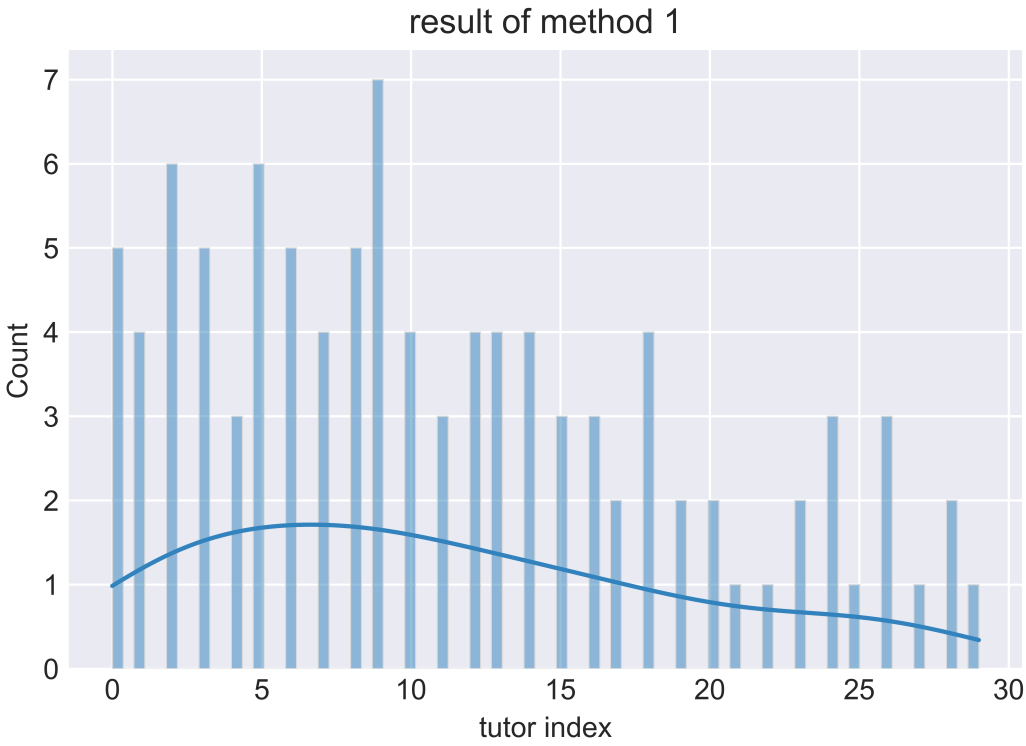

Figure 3: out for MILP-I-without-slack

[2]不引入松弛变量，即只满足第一个混合整数线性规划的约束方程的前两个方程，并且额外满足一个只能选一个导师和项目的方程，详细可见于 1.4 中

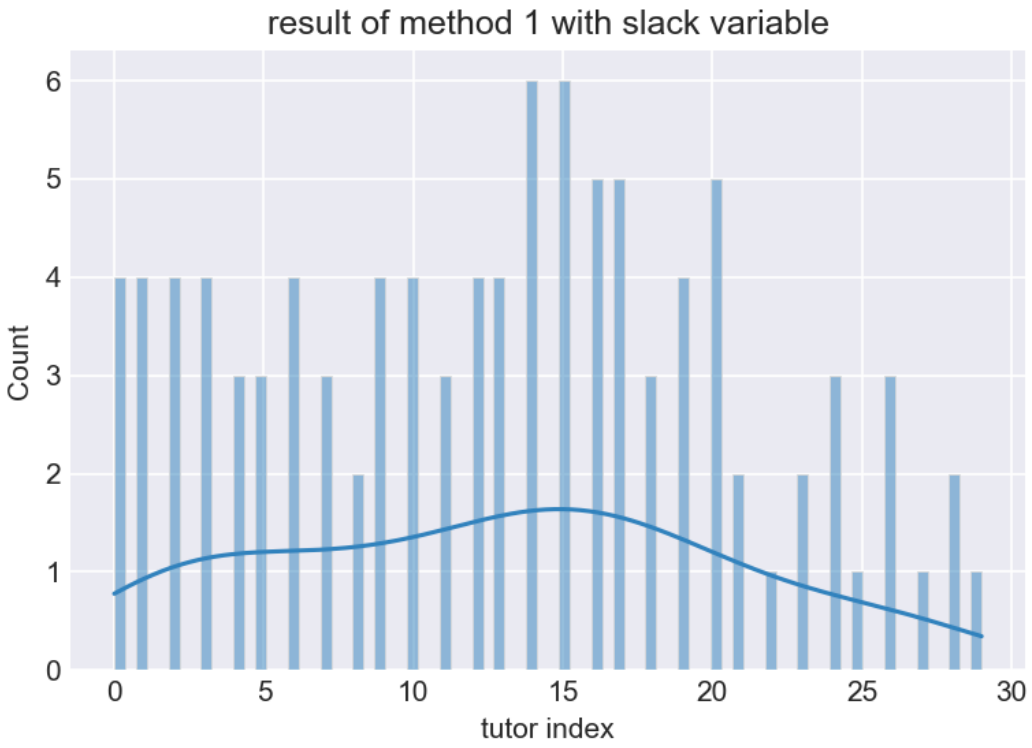

Figure 4: out for MILP-I

之后，我们对最终对学生学生选到的 $SRTP$ 的导师为该生的低级志愿进行了统计，统计结果可见于 Figure 5 以及 Figure 6

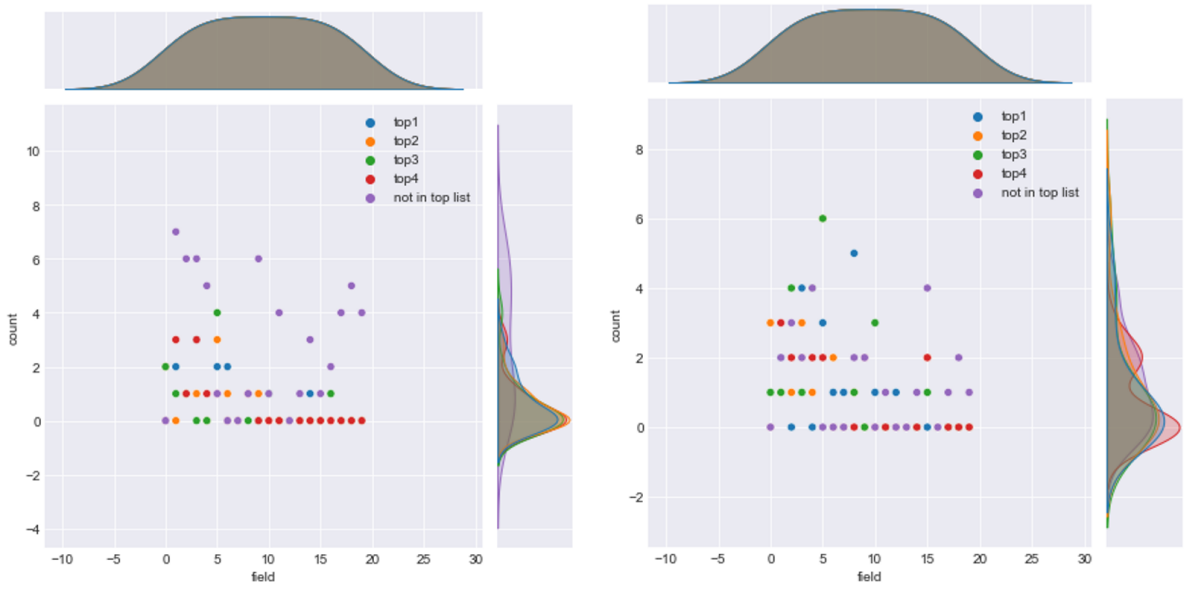

Figure 5: Scatter plot(左侧为不引入松弛变量，右侧为引入松弛变量)

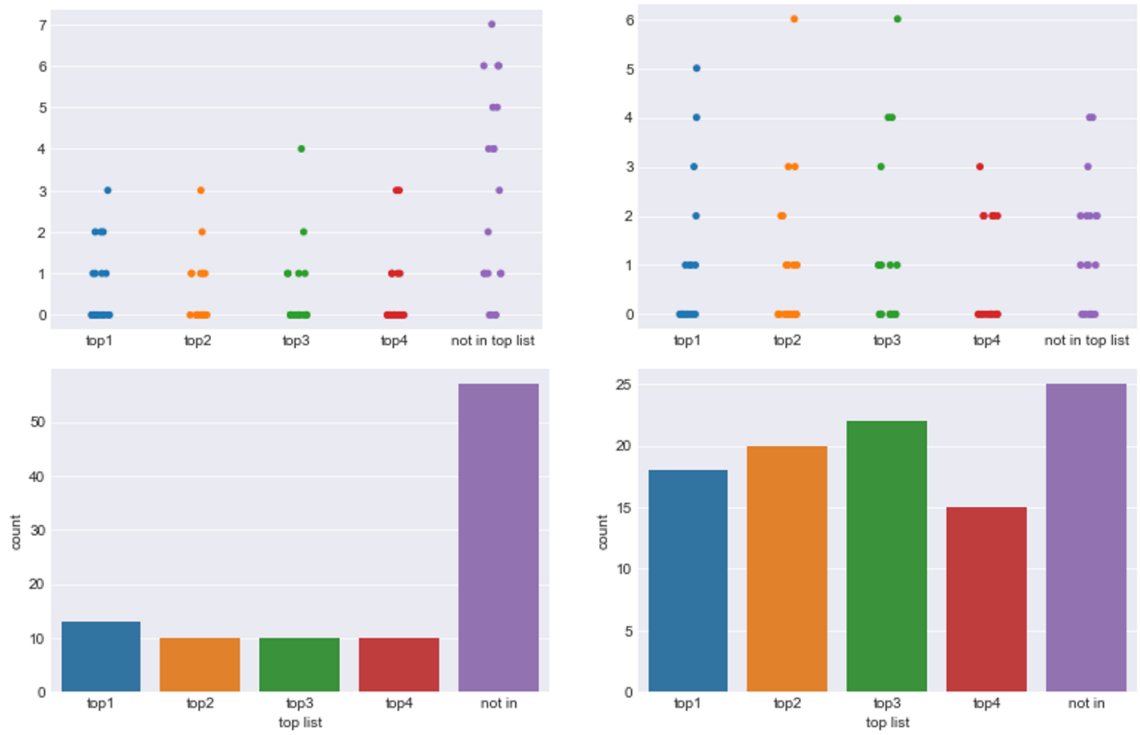

Figure 6: Choose the result of the voluntary order(左侧为不引入松弛变量，右侧为引入松弛变量)

对比两个统计结果，我们可以清楚得发现，如果引入了松弛变量，即使用我们所提出的**第二个混合整数线性规划模型**可以有效进一步优化学生的总体的满意度，即在使得更多的学生能够匹配到中意的 $SRTP$ 项目的前提下（如 MILP 所求的情况下），进一步使得学生的总体满意度优化，如上图 Figure 9 所示 not in list 的学生数量进一步减少，能够选到第一志愿和第二志愿的学生数量也大大增加。

2.3 MILP II

我们在 MILP I 中求解得到了最优的总体满意度 $\widehat{U} = \max \sum_{i \in S} u_i$, 争对上述数据的求解结果为 $\widehat{U} = \max \sum_{i \in S} u_i = 100$, 因此将这个引入约束方程，并且按照 1.7 中所给出的混合整数线性规划那样，我们修改了目标函数。其求解代码可以见于附件 $MILP - II.ipynb$ 或附录中，求解结果的大致分布如下图 Figure 7 所示。

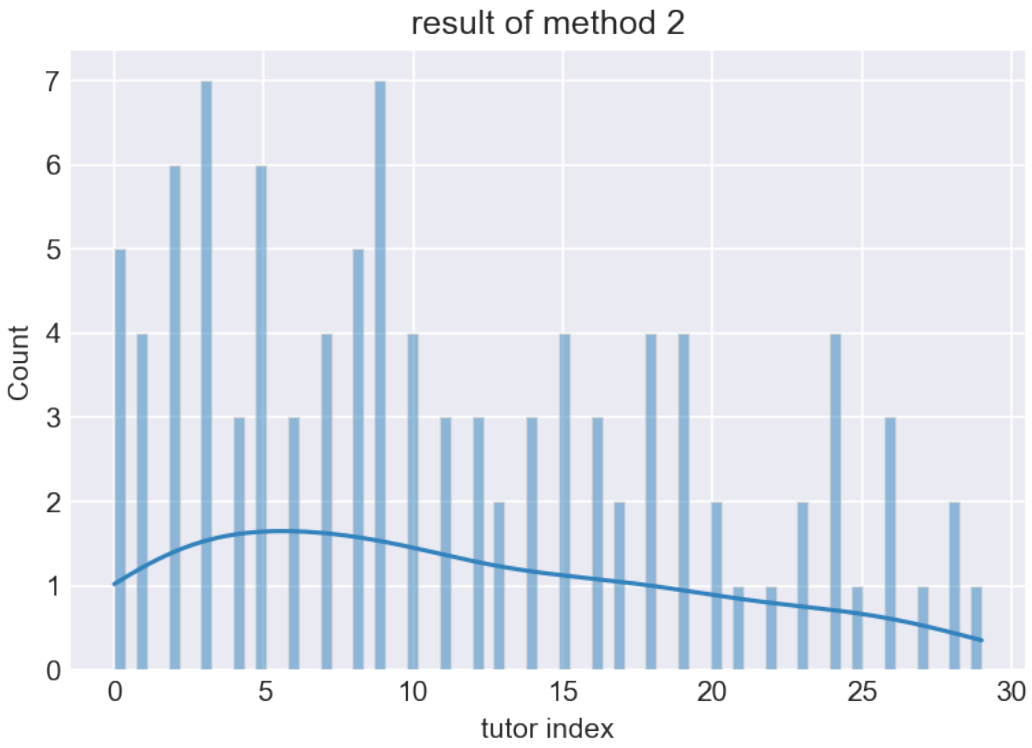

Figure 7: out for MILP-I-without-slack

之后，同 MILP I 中的那样，我们对最终对学生学生选到的 $SRTP$ 的导师为该生的低级志愿进行了统计，统计结果可见于 Figure 8 以及 Figure 9

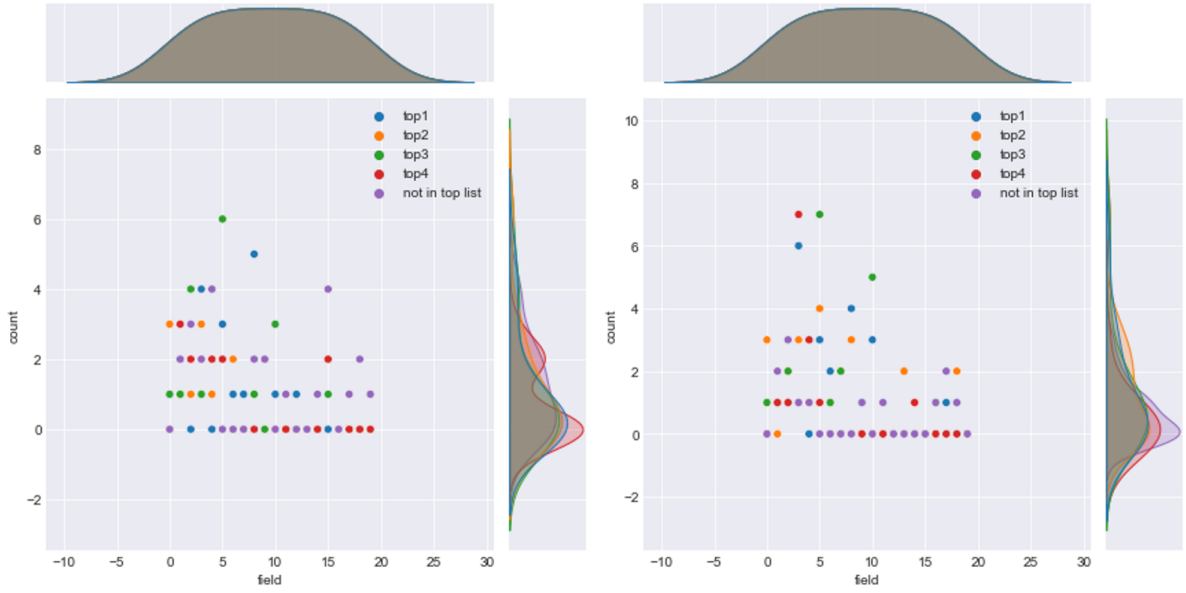

Figure 8: Scatter plot(左侧为 MILP I 求解结果，右侧 MILP II 求解结果)

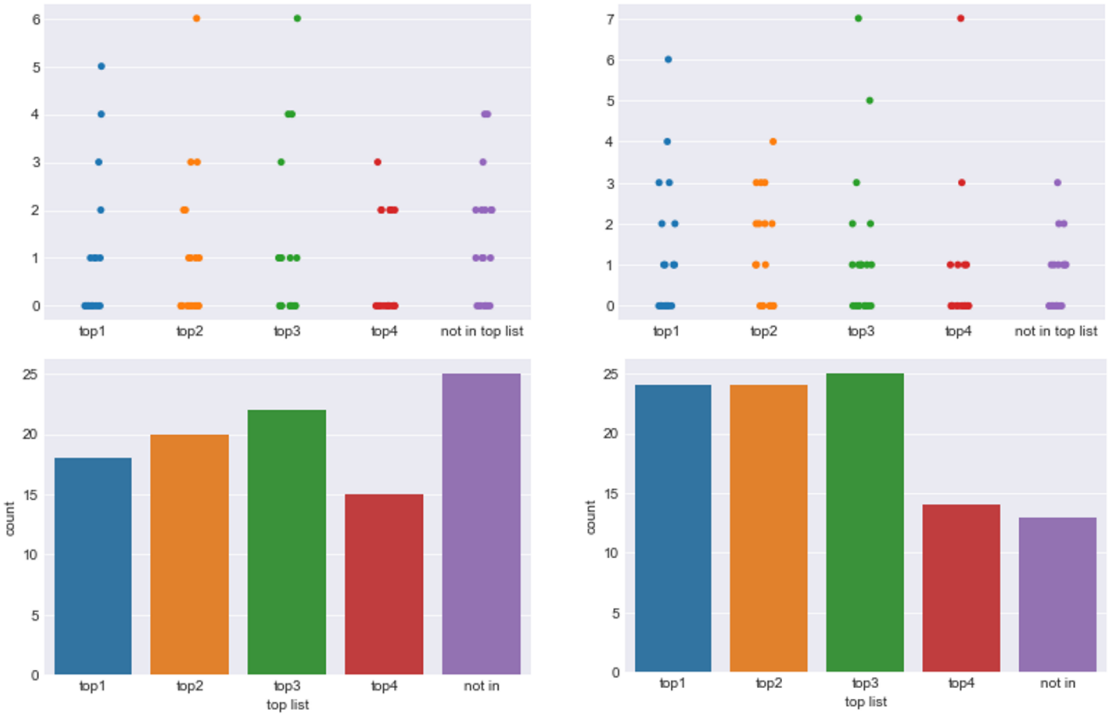

Figure 9: Choose the result of the voluntary order((左侧为 MILP I 求解结果，右侧 MILP II 求解结果)

对比两个统计结果，我们可以清楚得发现，从 MILP I 到 MILP II，即使用我们所提出的**第一个混合整数线性规划模型**可以有效提高学生的总体的满意度，使得更多的学生能够匹配到中意的 $SRTP$ 项目（如上图 Figure 6 所示 not in list 的学生数量大大减少，这意味着学生能选到调查时所填写的中意导师而不是被随机分配一个导师的的概率大大提高，而且特别是选到前 1，2，3 志愿的概率也有显著提升）。

3 模型评价

3.1 模型对比

先到先得

先到先得实质上是一种随机模型，显然，他的总满意度是一个随机值，由于往往最优值往往只有一个，至少是远少于 A_m^n(假设学生数量有 n, 所有老师能带的总学生数量为 n), 因此其总满意度一般来说都不是最优的 (甚至说不是局部最优的，比如，考虑两个同学，一个同学 S_1 所选择的研究方向 P_1 他不中意，但是对另一个同学 S_2 目前的研究方向 P_2 中意，而 S_2 对 P_1 和 P_2 都中意，显然可以通过交换这两个同学的研究方向使得匹配更优)。可以通过局部调整的方法找到更优解。（实际上，先到先得的规则类似于不加入松弛变量的方法，详细的对比可见于 2.2）

此外，分配过程中的意外变化可能会迫使所有学生再次重新分配。除了这些缺点之外，导师受欢迎程度的差异意味着一些学生到最后没有一个 $SRTP$ 项目可以参加。

而本研究所给出的模型仅仅需要事先搜集学生意向，并且进行一次求解，就可以提高总体的学生满意度，进而提高办学满意度。

成绩优先

成绩优先的匹配方法即 GPA 得分最高的学生将获得他第一志愿导师。然后一个人会简单地沿着 GPA 排名名单向下看，根据第一个选择进行分配，直到一个学生选择了一位名额已满的导师, 然后，该

学生将获得他/她的第二选择。但是，当接近底部的学生会有很大可能会被分配给中意程度排名非常靠后的教室。当然，这是一个非常容易使用的匹配算法，它确保了高分的学生得到一个非常满意的分配，但是往往对于成绩排名靠后的同学非常不利，并且会很大程度上引起他们的不满。

此外，成绩优先的方法所能应用的场景非常少，在某些没有成绩的、或者是难以衡量成绩的情况成绩优先的方法往往会失效, 比如硕士转博士的选拔，如何衡量一个硕士的“成绩”

如果引入广义的“成绩”，即不仅仅考虑 GPA 这一因素，而考虑更多的诸如志愿服务、科研情况、竞赛情况等更多的因素，将成绩的自变量多因子化，那么就需要引入一个多元函数，这样就过于复杂，不够简洁 (并且考如何设计这个多元函数，以及怎么样的多元函数科学都是非常大的研究问题)。

推荐录取

在现实的情况中，往往研究生的录取（尤其是博士生）需要获得推荐才能成功被录取（没有推荐的，往往录取率较低）。在现代信用理论体系下，推荐录取往往是以推荐人的信用作为担保的，它可以让导师能够选择到比较合适的学生开展项目，其作用体现在教师的反向选择上，而不在我们在本片文章中所研究的，如何能够使得尽量多的学生能够选到中意的导师和项目（或者说效益最大化）。尽

3.2 模型优势

该模型有一个显著的特点就是求解了**两个**混合整数线性规划（即使用了**二次优化**）而不是传统的一次性求解[3]，这样的求解有以下的显著优点：

(1) 相比于一次性求出结果，我们的模型能够先使得尽可能多的学生能够匹配到中意的 $SRTP$ 导师或者研究方向中的至少一个，以更好的保证公平性，让尽可能多的学生能够满意（如果采用一次求解，由于热门老师的出现等原因，可能会存在大量学生没有选到他所想选的 $SRTP$ 导师或者研究方向）。

(2) 我们的模型在保证普惠性的条件下，进一步按照学生的志愿顺序进行了优化，使得更多的学生能够选到他们想选择的 $SRTP$ 导师中更加满意程度靠前的。

(3) 该模型较为简单，并且能够适应大规模的数据求解。（在 intel core i7-11800H 上，使用 $GLPK\_MI$ 求解器，花费时间仅为 6.897e-01 seconds。而我们采用的测试数据集为 100 参与 $SRTP$ 的学生 ×30$SRTP$ 导师，也比较贴合一个高校中一个学院的实际情况）

3.3 优化空间

尽管该算法非常简洁，并且考虑了导师和研究方向两个维度，但是不同于现实中的情况，我们的研究割裂了导师和研究方向之间的有机联系——事实上，某个研究方向上，导师的学术水平有着非常清晰的排名（比如 H 指数、影响因子、年度引用数、职称等等）。而我们的研究只是停留在尽量使得学生选到满意的导师和研究方向，而且研究方向和导师之间的关系较为割裂，将这个方面完善是我们未来一个很好的改进的地方。

4 附录

4.1 数据集

生成的学生信息见附件中的 $studentdata.csv$

生成的教师信息见附件中的 $tutordata.csv$

[3]一次求解的方法即直接对学生的每个志愿赋权，混合整数线性规划的目标函数为最大化加权后的总满意度，详细代码可见附件或者附录中的 $.ipynb$

4.2 案例代码

Listing 1: dataGenerator.ipynb

```
import cvxpy as cp
from mip_cvxpy import PYTHON_MIP

import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

# generate student data
student = pd.read_csv("数据/tutor.csv")
# 项目
for i in range(0,student.shape[0]):
    #fieldlist = student.iloc[i,1:5]
    data = int(np.random.normal(10,5,size=1))
    if data >21 or data < 0:
        data = np.random.randint(1,11,size=1)
    student.iloc[i,1] = data
    for j in range(2,5):
        while 1:
            data = int(np.random.normal(10,5,size=1))
            if data >21 or data < 0:
                data = np.random.randint(1,11,size=1)
            student.iloc[i,j] = data
            for w in range(1,j):
                if student.iloc[i,j] == student.iloc[i,w]:
                    break
            if w==j-1 and student.iloc[i,j] != student.iloc[i,w]:
                break
# 导师
for i in range(0,student.shape[0]):
    data = int(np.random.normal(1,5,size=1))
    if data >21 or data < 0:
        data = np.random.randint(1,11,size=1)

    student.iloc[i,5] = np.random.randint(1,31,size=1)
    for j in range(6,9):
        while 1:
            student.iloc[i,j] = np.random.randint(1,31,size=1)
            for w in range(5,j):
                if student.iloc[i,j] == student.iloc[i,w]:
                    break
            if w==j-1 and student.iloc[i,j] != student.iloc[i,w]:
```

```
                break
student.head(5)

# generate superviosr data
tutorlist = pd.read_csv("数据/supervisor.csv")
tutorlist['lower']=''
tutorlist['upper']=''
for i in range(0,tutorlist.shape[0]):
    #fieldlist = student.iloc[i,1:5]
    tutorlist.loc[i,'lower']=int(np.random.randint(1,student.shape[0]/tutorlist.shape[0]+1,size=1))
    tutorlist.loc[i,'upper']=int(np.random.randint(1,4,size=1)+tutorlist.loc[i,'lower']+1)

    tutorlist.iloc[i,1] = np.random.randint(1,20,size=1)
    for j in range(2,5):
        while 1:
            tutorlist.iloc[i,j] = np.random.randint(1,7,size=1)
            for w in range(1,j):
                if tutorlist.iloc[i,j] == tutorlist.iloc[i,w]:
                    break
            if w==j-1 and tutorlist.iloc[i,j] != tutorlist.iloc[i,w]:
                break
tutorlist.head(2)
tutorlist.to_csv("tutordata.csv",index=False)
student.to_csv("studentdata.csv",index=False)

part = student.iloc[0:,1:5]
part.rename(columns ={'field1':"top 1","field2":"top 2","field3":"top 3","field4":"top
    4"},inplace=True)
plt.figure(dpi=150)
plt.style.use('seaborn-darkgrid') #'seaborn-bright'
figer = sns.histplot(part,bins=18)
figer.set_xlabel("field")
plt.savefig("fieldPreference.svg",dpi=1000)

part = student.iloc[0:,5:10]
plt.figure(dpi=150)
figer = sns.histplot(part,bins=30)
figer.set_xlabel("tutor")
plt.savefig("tutorPreference.svg",dpi=1000)
```

Listing 2: MILP-I-without-slack.ipynb.ipynb

```
import cvxpy as cp
from mip_cvxpy import PYTHON_MIP

import numpy as np
import pandas as pd
```

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

student = pd.read_csv("studentdata.csv")
tutorslist = pd.read_csv("tutordata.csv")

############################################################
# 变量声明
x = cp.Variable(shape=(student.shape[0],tutorslist.shape[0]),boolean=True)

constraints=[]
############################################################
# 加入约束1
constraints.append(x@np.ones((tutorslist.shape[0],1)) <= np.ones((student.shape[0],1)))
constraints.append(x@np.ones((tutorslist.shape[0],1)) >= np.ones((student.shape[0],1)))
############################################################
# 加入约束2
for i in range(0,tutorslist.shape[0]):
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] <= tutorslist["upper"][i])
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] >= tutorslist["lower"][i])

sum = 0
for i in range(0,student.shape[0]):
    t = np.zeros((tutorslist.shape[0],1))
    for j in range(1,5):
        t[student.iloc[i,j]] = 1
    for j in range(5,9):
        for k in range(0,tutorslist.shape[0]):
            # categories for a tutor
            if(student.iloc[i,j] in tutorslist.iloc[k,1:5].values.tolist()):
                t[k] = 1
    sum += x[i]@t


obj = cp.Minimize(-sum)
prob = cp.Problem(obj,constraints=constraints)
prob.solve(solver='GLPK_MI',verbose=True)

print("\nThe optimal value is", prob.value)
print("A solution x is")
output = []
for item in x.value:
    for i in range(0,tutorslist.shape[0]):
        if item[i]==1:
            output.append(i)
```

```
print(x.shape)
print(output)

plt.figure(dpi=1000)
plt.style.use('seaborn-darkgrid') #'seaborn-bright'
palette = plt.get_cmap('tab20c')#'Pastel2') # 'Set1'
figer = sns.histplot(output,kde = True,bins=80,shrink = 1, color = palette.colors[0], edgecolor =
    palette.colors[-1])#"none")#, element="step")# element = "poly") # cumulative = True
figer. set_title ("result of method 1")
figer.set_xlabel("tutor index")
plt.savefig("m1_1.svg",dpi=2000)

top1 = np.zeros((20,6))
# 对于所有学生
for i in range(0,student.shape[0]):
    # 对于它可选的导师
    for j in range(0,tutorslist.shape[0]):
        # 选中某个导师
        if x.value[i][j]==1:
            # 学生的志愿排序
            flag = 0
            for w in range(1,5):
                if student.iloc[i,w] in tutorslist.iloc[j,1:4].values.tolist():
                    top1[student.iloc[i,w]-1][w] +=1
                    flag = 1
                    break
            if flag==0:
                top1[tutorslist.iloc[j,1]][5]+=1
for i in range(0,20):
    top1[i][0] = i+1

plt.figure(dpi=1000)
figer = sns.jointplot(top.iloc[0:,1:])
figer.set_axis_labels(xlabel="field",ylabel="count")
plt.savefig("fieldWithindexM2.svg",dpi=1000)

figer = sns.stripplot(top.iloc[0:,1:])
plt.savefig("timesOfcatergoryM2.svg",dpi =1000)

times =[["top1","top2","top3","top4","not in"],
        [0,0,0,0,0]
        ]
for i in range(1,6):
    times[1][i-1]=np.sum(top.iloc[0:,i])
figer = sns.barplot(y=times[1],x=times[0],errorbar=None)
figer.set_ylabel("count")
figer.set_xlabel("top list")
```

```
plt.savefig("m11result.svg",dpi = 1000)
```

Listing 3: MILP-I.ipynb

```
import cvxpy as cp
from mip_cvxpy import PYTHON_MIP

import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

student = pd.read_csv("studentdata.csv")
tutorslist = pd.read_csv("tutordata.csv")

############################################################
# 变量声明
x = cp.Variable(shape=(student.shape[0],tutorslist.shape[0]),boolean=True)
y = cp.Variable(shape=(student.shape[0],1),boolean=True)

s1 = cp.Variable(shape=(student.shape[0],1),boolean=True)
s2 = cp.Variable(shape=(student.shape[0],1),boolean=True)

############################################################
# 加入约束7-9
constraints = [#s1>=np.zeros((student.shape[0],1)),
               #s2>=np.zeros((student.shape[0],1)),
               y >= np.ones((student.shape[0],1))-s1,
               y >= np.ones((student.shape[0],1))-s2,
               y <= 2*np.ones((student.shape[0],1))-s1-s2
              ]
############################################################
# 加入约束1
constraints.append(x@np.ones((tutorslist.shape[0],1)) <= np.ones((student.shape[0],1)))
constraints.append(x@np.ones((tutorslist.shape[0],1)) >= np.ones((student.shape[0],1)))
############################################################
# 加入约束2
for i in range(0,tutorslist.shape[0]):
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] <= tutorslist["upper"][i])
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] >= tutorslist["lower"][i])

# 加入约束4

for i in range(0,student.shape[0]):
    Tji = np.zeros((tutorslist.shape[0],1))
```

```
    for j in range(1,5):
        Tji[student.iloc[i,j]]=1
    constraints.append(x[i]@Tji+s2[i]==1)

############################################################
# 加入约束5
# w students' field
for w in range(1,5):
    # row number of students list
    for i in range(0,student.shape[0]):
        indexlist = np.zeros((tutorslist.shape[0],1))
        # find tutor
        for j in range(0,tutorslist.shape[0]):
            # categories for a tutor
            if(student.iloc[i,w] in tutorslist.iloc[j,1:5].values.tolist()):
                indexlist[j] = 1
        constraints.append(s1[i]<= 1 - x[i]@indexlist)

############################################################
# 加入约束6

for i in range(0,student.shape[0]):
    sum=0
    for w in range(1,5):
        indexlist = np.zeros((tutorslist.shape[0],1))
        # find tutor
        for j in range(0,tutorslist.shape[0]):
            # categories for a tutor
            if(student.iloc[i,w] in tutorslist.iloc[j,1:5].values.tolist()):
                indexlist[j] = 1
        sum +=indexlist
    constraints.append(s1[i]>= 1 - x[i]@sum)
############################################################

obj = cp.Minimize((-np.ones(((1,student.shape[0])))@y)
prob = cp.Problem(obj,constraints=constraints)
prob.solve(solver='GLPK_MI',verbose=True)

print("\nThe optimal value is", prob.value)
print("A solution x is")
output = []
for item in x.value:
    for i in range(0,tutorslist.shape[0]):
        if item[i]==1:
            output.append(i)
print(x.shape)
print(output)
```

```
plt.figure(dpi=150)
plt.style.use('seaborn-darkgrid') #'seaborn-bright'
palette = plt.get_cmap('tab20c')#'Pastel2') # 'Set1'
figer = sns.histplot(output,kde = True,bins=80,shrink = 1, color = palette.colors[0], edgecolor =
    palette.colors[-1])#"none")#, element="step")# element = "poly") # cumulative = True
figer. set_title ("result of method 1 with slack variable")
figer.set_xlabel("tutor index")
plt.savefig("m1_2.svg",dpi=2000)

top1 = np.zeros((20,6))
# 对于所有学生
for i in range(0,student.shape[0]):
    # 对于它可选的导师
    for j in range(0,tutorslist.shape[0]):
        # 选中某个导师
        if x.value[i][j]==1:
            # 学生的志愿排序
            flag = 0
            for w in range(1,5):
                if student.iloc[i,w] in tutorslist.iloc[j,1:4].values.tolist():
                    top1[student.iloc[i,w]-1][w] +=1
                    flag = 1
                    break
            if flag==0:
                top1[tutorslist.iloc[j,1]][5]+=1
for i in range(0,20):
    top1[i][0] = i+1

top = pd.DataFrame(top1,columns=["index","top1","top2","top3","top4","not in top list"])

plt.figure(dpi=1000)
figer = sns.jointplot(top.iloc[0:,1:])
figer.set_axis_labels(xlabel="field",ylabel="count")
plt.savefig("fieldWithindexM2.svg",dpi=1000)

figer = sns.stripplot(top.iloc[0:,1:])
plt.savefig("timesOfcatergoryM2.svg",dpi =1000)

times =[["top1","top2","top3","top4","not in"],
        [0,0,0,0,0]
        ]
for i in range(1,6):
    times[1][i-1]=np.sum(top.iloc[0:,i])
figer = sns.barplot(y=times[1],x=times[0],errorbar=None)
figer.set_ylabel("count")
figer.set_xlabel("top list")
plt.savefig("m12result.svg",dpi = 1000)
```

Listing 4: MILP-II.ipynb

```
import cvxpy as cp
from mip_cvxpy import PYTHON_MIP

import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

student = pd.read_csv("studentdata.csv")
tutorslist = pd.read_csv("tutordata.csv")
############################################################
# 变量声明
x = cp.Variable(shape=(student.shape[0],tutorslist.shape[0]),boolean=True)
y = cp.Variable(shape=(student.shape[0],1),boolean=True)

s1 = cp.Variable(shape=(student.shape[0],1),boolean=True)
s2 = cp.Variable(shape=(student.shape[0],1),boolean=True)

############################################################
# 加入约束7-9
constraints = [#s1>=np.zeros((student.shape[0],1)),
               #s2>=np.zeros((student.shape[0],1)),
               y >= np.ones((student.shape[0],1))-s1,
               y >= np.ones((student.shape[0],1))-s2,
               y <= 2*np.ones((student.shape[0],1))-s1-s2
              ]
############################################################
# 加入约束1
constraints.append(x@np.ones((tutorslist.shape[0],1)) <= np.ones((student.shape[0],1)))
constraints.append(x@np.ones((tutorslist.shape[0],1)) >= np.ones((student.shape[0],1)))
############################################################
# 加入约束2
for i in range(0,tutorslist.shape[0]):
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] <= tutorslist["upper"][i])
    constraints.append((np.ones((1,student.shape[0]))@x)[0][i] >= tutorslist["lower"][i])

# 加入约束4

for i in range(0,student.shape[0]):
    Tji = np.zeros((tutorslist.shape[0],1))
    for j in range(1,5):
        Tji[student.iloc[i,j]]=1
    constraints.append(x[i]@Tji+s2[i]==1)
```

```
############################################################
# 加入约束5
# w students' field
for w in range(1,5):
    # row number of students list
    for i in range(0,student.shape[0]):
        indexlist  = np.zeros((tutorslist.shape[0],1))
        # find tutor
        for j in range(0,tutorslist.shape[0]):
            # categories for a tutor
            if(student.iloc[i,w] in tutorslist.iloc[j,1:5].values.tolist()):
                indexlist[j] = 1
        constraints.append(s1[i]<= 1 - x[i]@indexlist)

############################################################
# 加入约束6

for i in range(0,student.shape[0]):
    sum=0
    for w in range(1,5):
        indexlist  = np.zeros((tutorslist.shape[0],1))
        # find tutor
        for j in range(0,tutorslist.shape[0]):
            # categories for a tutor
            if(student.iloc[i,w] in tutorslist.iloc[j,1:5].values.tolist()):
                indexlist[j] = 1
        sum +=indexlist
    constraints.append(s1[i]>= 1 - x[i]@sum)
############################################################

# add constrain 11
    constraints.append(np.ones(((1,student.shape[0])))@y==100)

################################################################
#redfine object function

sum = 0
for i in range(0,student.shape[0]):
    t = np.zeros((tutorslist.shape[0],1))
    # field sequence
    for j in range(1,5):
        t[student.iloc[i,j]] = 4 + 1 - j
    for j in range(5,9):
        for k in range(0,tutorslist.shape[0]):
            # categories for a tutor
            # if(student.iloc[i,j] in tutorslist.iloc[k,1:5].values.tolist()):
                t[k] += (4 + 5 -j)
```

```
    sum += x[i]@t


obj = cp.Minimize(sum)
#obj = cp.Minimize((-np.ones(((1,student.shape[0]))))@y)
prob = cp.Problem(obj,constraints=constraints)
prob.solve(solver='GLPK_MI',verbose=True)

print("\nThe optimal value is", prob.value)
print("A solution x is")
output = []
for item in x.value:
    for i in range(0,tutorslist.shape[0]):
        if item[i]==1:
            output.append(i)
print(x.shape)
print(output)
plt.figure(dpi=150)
plt.style.use('seaborn-darkgrid') #'seaborn-bright'
palette = plt.get_cmap('tab20c')#'Pastel2') # 'Set1'
figer = sns.histplot(output,kde = True,bins=80,shrink = 1, color = palette.colors[0], edgecolor =
    palette.colors[-1])#"none")#, element="step")# element = "poly") # cumulative = True
figer.set_title("result of method 2")
figer.set_xlabel("tutor index")
plt.savefig("m2.svg",dpi=2000)

top1 = np.zeros((20,6))
# 对于所有学生
for i in range(0,student.shape[0]):
    # 对于它可选的导师
    for j in range(0,tutorslist.shape[0]):
        # 选中某个导师
        if x.value[i][j]==1:
            # 学生的志愿排序
            flag = 0
            for w in range(1,5):
                if student.iloc[i,w] in tutorslist.iloc[j,1:4].values.tolist():
                    top1[student.iloc[i,w]-1][w] +=1
                    flag = 1
                    break
            if flag==0:
                top1[tutorslist.iloc[j,1]][5]+=1
for i in range(0,20):
    top1[i][0] = i+1

top = pd.DataFrame(top1,columns=["index","top1","top2","top3","top4","not in top list"])
```

```
plt.figure(dpi=1000)
figer = sns.jointplot(top.iloc[0:,1:])
figer.set_axis_labels(xlabel="field",ylabel="count")
plt.savefig("fieldWithindexM2.svg",dpi=1000)

figer = sns.stripplot(top.iloc[0:,1:])
plt.savefig("timesOfcatergoryM2.svg",dpi =1000)

times =[["top1","top2","top3","top4","not in"],
        [0,0,0,0,0]
        ]
for i in range(1,6):
    times[1][i-1]=np.sum(top.iloc[0:,i])
figer = sns.barplot(y=times[1],x=times[0],errorbar=None)
figer.set_ylabel("count")
figer.set_xlabel("top list")
plt.savefig("m2result.svg",dpi = 1000)
```