一类调整算法在信息学竞赛中的应用

中国人民大学附属中学邓明扬

摘 要

调整是一种非完美算法，在许多情形下难以证明复杂度但却有着优越的表现。本文致力于研究一类调整算法在信息学竞赛中的应用，包括求解部分提交答案题，以及在随机数据下求解一些传统图论问题。

1 引言

组合优化题目在信息学竞赛中占了很大比重。一般地，组合优化问题有如下形式：一个问题有一些合法解和不合法解。每个合法解有一个对应的权值。你需要在所有合法解中找出权值最大的一个。

一种显然的做法是：先任取一个合法解，然后对合法解进行微调使得权值变大。一直操作直到无法进行。这一算法看似简单，但在许多问题中有出色的表现。

2 算法描述

本文中的调整算法有如下形式：

对一个组合优化问题，记 S 为所有可能状态的集合。$\forall x \in S$, 定义 $F(x)$ 为 x 的权值。

$T \subset S$ 为合法状态的集合。你需要求出 $x \in T$，使得 $F(x)$ 取得最值。[1]

在实际问题中，集合 T 往往具有某种局部性质。一般地，对于 $x \in T$，往往能找到 $N(x) \subset T$，满足 $\forall y \in N(x)$，y 与 x 的差异较小。我们称 $N(x)$ 为 x 的一个“邻域”。

我们考虑一种显然的搜索算法。首先任取一个 $x \in T$，一直重复以下过程直到超时：

1. 如果 $N(x)$ 中存在 y，满足 $F(y) > F(x)$，我们将 x 修改为 y，并进入下一轮。

2. 如果不满足 1 中的条件，但存在一些 $y \in N(x)$，满足 $F(x) = F(y)$，我们从中随机选取一个 y，将 x 修改为 y 并进入下一轮。

我们称这一算法为调整法。容易发现，每一轮我们在当前解的局部邻域内寻找一个较优解并更新。最终我们将求出一个局部较优解。

[1]本文中，如无特殊说明，默认最值为最大值

这一简单的算法，在设计得当的情况下，有着优越的表现。我们将用许多具体的例子说明这一点。

3 匹配问题

匹配问题时常可在信息学竞赛中见到。对于经典的问题，例如一般图/二分图最大匹配，已经有了多项式时间的传统算法。然而调整法给出了一个在随机数据实践中表现优越的简洁解法。

除此之外，一些不常规的隐式匹配问题往往是 NPC 的，或是需要特殊性质下的特殊解法。而调整法在这些问题中仍然有突出的表现。

3.1 一般图最大匹配

一般图最大匹配是图论中的经典问题。利用带花树或 Tutte 矩阵，可以在 $O(n^{\omega})$ 或 $O(n^3)$ 复杂度内求解。下面我们介绍一种在许多数据下运行表现良好的调整算法。

3.1.1 问题描述

一般图最大匹配指这样一个问题：

给定图 G = (V, E), 其中 V 是顶点集，E 是边集。求出最大的子集 $S \subset E$，满足 $\forall v \in V$，S 中以 v 为端点的边至多只有一条。

3.1.2 调整算法

以下是一种调整算法：

对于图 G = (V, E)，我们维护一个子集 $R \subset E$，表示当前匹配的边。每次随机一个未匹配点 S。若 S 有未匹配邻点 V，将 S 与 V 匹配。否则随机一个 S 的邻点 T，将 S 与 T 匹配，并将 T 原先匹配断掉。

一直重复直到超时。

在这一过程中，我们一直在对 R 进行局部微调，其大小不减且有可能增大。这符合我们之前对调整算法的描述。

3.1.3 算法运行表现

这一算法十分简洁，但最坏复杂度不是多项式级的。然而，在针对这一算法的数据出现前，该算法能通过评测网站 uoj.ac 上一般图最大匹配的全部测试点。在随机图中，经过实验，我们发现该算法均在较少的轮次中就找到了最大匹配。

笔者测试了算法在 $|V| = 500$ 的随机图下的表现。具体随机方式为：先生成 $|E|$，然后在所有 $|V|$ 个点，$|E|$ 条边的图中等概率选取一个。

经过实验，当 $|E|$ 在 700 到 800 的范围内随机时所需的期望轮次较多。笔者在此范围内随机了 5000 张图，算法所需的最多轮次数为 1114612，平均轮次数约为 13367。标准差为 27627。

我们可以看出，该算法在随机数据下表现良好。

然而，对于一些特殊图，算法并不能在短时间内求解。例如，对于一张二分图 $\mathrm{V} = (A_i, B_i | 1 \le i \le n)$，$\mathrm{E} = ((A_i, B_j) | i + j \le n + 1)$，当 $n = 40$ 时算法需要很久才能求出完美匹配。

3.1.4 算法分析

算法在最坏数据下的期望运行时间是指数级的。

这里给出算法期望复杂度 $O(n^{n/2+4})$ 的证明。

考虑一个最大匹配 T. 设当前匹配为 S。不妨当前匹配不是最大匹配。

我们考虑新建一个图 $G_1 = (V, E_1)$，一条边在 G_1 中当且仅当这条边出现在 T 和 S 恰好之一中。（即：G_1 为当前匹配与最大匹配的对称差。）对于 G_1 中的一条边，如果它在 T 中则将该边染为蓝色，如果它在 S 中则将该边染为红色。

由于每个点在 T 中和 S 中的度数均不超过 1，每个点在 G_1 中的度数均不超过 2。因此，G_1 由一些环和链构成。由于任何相邻两条边不同色，G_1 中没有奇环；又因为蓝色边比红色边多，因此存在一条链首尾均是蓝色，链上红蓝交替（用到相邻两条边不同色）。

此时任取一条这种链，该链上除了首尾点外均已匹配。期望至多 $O(n)$ 轮后，调整算法随机取点会取到这条链的链首或链尾。此时如果匹配了该点连接的红边，这条链的长度将减小 2。否则链长增加至多 1。由于长度减少的概率至少为 $1/n$，在期望 $O(n^{n/2+1})$ 次选取首尾之后长度会有连续的 $n/2$ 次减少，此时匹配数增加了 1。由于最大匹配的大小是 $O(n)$ 的，因此期望 $O(n^{n/2+1} * n * n)$ 轮之后可以得到最大匹配。由于每轮花费 $O(n)$ 的时间，算法的期望复杂度不超过 $O(n^{n/2+4})$。

这一理论上界非常巨大，但算法在极限数据下的运行速度的确很慢。但该算法便于实现，且在许多随机数据及非精心构造的数据下表现优越，因此很适用于解决现实生活中遇到的问题，以及信息学竞赛中的提交答案题或在一些特定模型下、数据有特殊性质的传统题。

对于一般图最大匹配，该算法的正确性有保证。我们也需要说明，后文中的许多调整算法笔者并不能证明其正确性，但同样在现实生活/提交答案题中有出色的表现。

3.2 隐式匹配问题

在信息学竞赛中，另有一大类问题与匹配相关，但常常无法转化成常规的一般图最大匹配。我们称其为隐式匹配问题。对于这类问题，前文提及的调整算法常常奏效。尽管复杂度并没有严格的证明，但往往有出色的实际表现。后文将列举两道隐式匹配例题。

3.3 CF Round 562 E

3.3.1 问题描述

给定序列 x_i $(0 \le i < 2^n)$，请你构造两个 0 到 $2^n - 1$ 的排列 p_i, q_i $(0 \le i < 2^n)$，满足 $\forall 0 \le i < 2^n$，有 $p_i \oplus q_i = x_i$ 或输出无解。

$n \le 12$.

3.3.2 调整算法

容易发现，当 $\oplus_{0 \le i < 2^n} x_i$ 不为 0 时，原问题无解。下面考虑 $\oplus_{0 \le i < 2^n} x_i = 0$ 的情形。可以证明此时原问题均是有解的，证明参见参考文献 [2].

考察这样一种调整算法。

维护当前的 p 和 q，某些位置可能没有值。每次随机一个 p 中没有值的位置 p_i，如果存在一个值 v 使得 $v \oplus p_i$ 未在 q 中出现，且 v 未在 p 中出现，则将 p_i 设为 v，q_i 设为 $v \oplus a_i$。否则随机一个未出现在 p 中的 v，将 p_i 设为 v，q_i 设为 $v \oplus p_i$，并将 $v \oplus p_i$ 原先在 q 中出现的位置和其对应的 p 清空。

3.3.3 算法运行表现

在比赛中，该算法可以在 265ms（约 1/5 时限）内通过原题，从表现上有接近 $O(n^2)$ 的运行效率。碍于水平所限，笔者无法证明其正确性、复杂度的下界也无法构造数据使其答案错误或超时。笔者猜想该算法在这一数据范围内无法卡掉，希望有意愿的同学能在未来给出证明。

3.4 JOI2020 制作团子

3.4.1 问题描述

这是一道提交答案题。

你是制作团子的专家。你现在有若干个团子和竹签，团子被整体摆放在一个 R 行 C 列的格子里，每个格子恰好有一个团子。团子颜色为粉色 (P), 白色 (W), 绿色 (G)。

你每次会选择三个连续的团子，这三个团子必须沿着竖直方向 (从上往下)，水平方向 (从左往右) 或者对角线方向 (从左上至右下，或从右上至左下)。例如，如果你选择了竖直方向的三个团子，你会按照上-中-下的顺序依次将团子串到竹签上。一个团子只能被串在一根竹签上。

一串团子是漂亮的当且仅当竹签上串的团子的颜色依次为粉-白-绿或者绿-白-粉。

你想要制作尽量多的漂亮的团子。

其中最大的测试点大小约为 500 * 500。

3.4.2 调整算法

我们仍然考虑调整算法。每次我们随机考虑一个未匹配的 W 色点，并考虑以其为中心的竹签。如果存在一些这样的竹签上不包含匹配点，我们在这样的竹签中随机选择一个添加。否则枚举以它为中心、且只与一根已有竹签冲突的竹签，我们有一半概率用新竹签替换原来的。

3.4.3 算法运行表现

该算法可以在十分钟左右获得 95 分，在一小时左右得到 99 到 100 分。鉴于其非常容易实现，这样的得分效率可以说很高。

4 NP 问题

在许多 NP 问题中，调整算法能给出优秀的解。下文将介绍图染色和有向图哈密顿链的调整解法。

4.1 拉姆赛数

4.1.1 问题描述

请你对 137 阶完全图 5 染色，使得不存在同色三角形。

4.1.2 调整算法

考虑构造 $a_1, a_2, ..., a_{136}$，其中 $a_i \in \{0, 1, 2, 3, 4\}$，满足任意 i, j, a_i, a_j, a_{i+j} 不全相同。

之后我们将图中 i, j 点之间的颜色设置为 $a_{|i-j|}$ 即可。

构造 a 数组只需：最开始随机构造，每次随机选择 a 中的一位，固定其他位的值，在使得同色 a_i, a_j, a_{i+j} 对数最少的值中随机选取一个。可以通过尝试随机种子构造出 a。

4.2 图染色

4.2.1 问题描述

图染色问题是经典的 NP 问题。

给定一张图 $G = (V, E)$，让你构造一种方案将顶点用 k 种颜色染色，满足每条边的两端点不同色。

4.2.2 调整算法

我们需要最小化两端点同色的边的数目。当这一数目被减少到 0，我们便得到了合法的染色方案。

考察这一种调整算法：首先为每个点随机染一种颜色。之后每次随机一个点，考虑固定其余点的颜色不变，该点染不同颜色对同色边数量的贡献，在贡献最少的颜色中等概率选取一个作为该点的颜色。

4.2.3 算法运行表现

虽然得到染色方案很难，但一个染色是否合法是容易判定的，且生成一个可以 K 染色的数据并不困难。因此我们可以在随机生成数据下考虑调整算法的表现。

我们采取以下方式生成可三染色的图：首先固定 $|V|, |E|$。然后对于每个顶点，将其随机染成三种颜色之一，并随机生成 $|E|$ 条两端点不同色的边。

笔者生成了 120 张 |V| = 500，|E| 在 0 到 45000 内随机的三染色图。这些图中，算法在 118 张上得到了正确的结果，调整的最大轮次数为 98587，平均轮次数为 5466.48，标准差为 12163.5。

4.2.4 算法分析

该算法的正确性并没有保证：因为存在一个染色方案，使得局部无法调整，但是全局并非最优。例如一个四元环，四个点颜色分别为黑黑白白，每个环上黑点外接一个白点、每个环上白点外接一个黑点。此时该染色无法调整，但是原图存在合法的二染色。

然而算法在随机数据下表现出色，因此可以常常应用于提交答案题或实际问题中。

4.3 有向图哈密顿链

4.3.1 问题描述

有向图哈密顿链也是经典的 NPC 问题。

给定一张有向图 $G = (V, E)$，请你构造一条经过每个点恰好一次的路径（起点终点不给定）。

4.3.2 调整算法

维护边的一个尽量大的子集，满足只考虑这些边时每个点出入度都不超过 1，且不构成圈。

如果子集大小达到 $n - 1$，则找到了一条哈密顿路。

考虑调整维护子集。按随机顺序考虑边，如果加入后不构成圈，且加入之后所有点度数均仍合法，则加入这条边。

否则如果不构成圈，但有一个点度数不合法，则以一半概率加入并把该点相连的与新加入边矛盾的边断掉。

使用 LCT 判断是否成圈。

4.3.3 算法运行表现

笔者采用以下方法生成了大量的图进行试验：

首先加入边 $(i, i + 1)$，其中 $1 \le i \le n - 1$，以保证存在哈密顿链。然后随机加入若干有向边，并将顶点重标号。

笔者按以上方法随机生成了 20 张 |V| = 50000，|E| 在 |V| - 1 到 |V| + 200000 间等概率随机的图。其中有 1 张图该算法未能求出哈密顿链；对于其余的图，算法最多考虑的边数为 4170064，平均约为 1825990，标准差约为 1040240。换言之，在实验数据中百分之 95 的随机图均能在几秒内求解。

4.3.4 算法分析

该算法不一定是正确的。因为有可能加入了一些边后，局部无法调整，但是全局并非最优。例如：对于 5 个点的有向图，边集为 { (4, 1), (1, 3), (5, 2), (4, 2), (5, 1), (3, 4) }，(5, 1, 3, 4, 2) 为一个合法的哈密顿链。然而，当选择了边集 { (4, 1), (1, 3), (5, 2) } 后，无法进行任何调整。这也是实验部分一些图未能求解的原因。但此时改变随机顺序再次求解，往往能求出答案。

4.3.5 一些规约

事实上，给定起点终点的哈密顿路径问题及哈密顿圈问题均可归约到这一问题。

如果给定起点终点，只需新建两个点 U, V；然后 U 到起点连一条边，终点到 V 连一条边

如果要求哈密顿圈，可以枚举一条边，转化成给定起点终点情形。

如果是无向图，只需正反边各加一遍即可。

4.4 NP 问题的近似解

调整法也可以用于求解许多 NP 问题的近似解。一般地，许多 NP 问题可以归约到每个变量取值在 $\{0, 1\}$ 中的线性规划。对于这种线性规划问题，常常有如下的近似算法：

首先忽略变量取值在 $\{0, 1\}$ 中的条件，弱化为变量取值在 $[0, 1]$ 中。弱化版本是一个正常的线性规划问题，可以用弱多项式的内点法求解。此时得到了一个解，但每个变量的取值是分数。我们想将这组解转化成一组整数解，并保持原先尽量多的性质。

转化成整数解的方法通常有以下两种：一种是对于一个分数解中取值为 x 的变量，有 x 的概率将其转化为 1，$1 - x$ 的概率将其转化为 0. 这一算法保证了各线性组合的期望。另一种是调整法，固定一些限制不变的情况下，调整剩余变量的取值直到剩余变量中有一些变为 1 或 0. 这一算法能得到较优的近似比。算法中固定的限制往往是：在余下变量任意取的情形下，可能超额较多的限制。由于剩余变量越少，可能超额较多的限制也越少，通常可以选取近似比，使得固定的限制数小于变量数。此时调整总能进行。

这一类调整算法以及调整和概率方法的结合可以参见参考文献 [1].

5 算法前景

5.1 实际应用

调整算法具有广泛的适用性，在许多题目中均可应用，并且能用于解决许多困难的问题。信息学竞赛中，用调整法解提交答案题往往能在短时间内取得大量分数；另外，对于部分传统题目，调整法也能取得不错的效果。

5.2 一些可能的优化

本文中提到的调整方法较为朴素简洁、容易实现。在此基础上，有一些可能的优化。

5.2.1 局部多次调整

由于目前的随机化算法每次在全局随机取一个变量调整，有可能一个刚被调整过的变量很快又被调整回去，加大了时间开销。因此一种可能的优化方式是：在一个调整过后将其固定，并在其附近寻找其影响到的位置再进行调整，直到该轮调整次数达到某一阈值后，将所有固定的变量改回不固定并开始下一轮调整。

5.2.2 调整与模拟退火

模拟退火是一类基于概率的调整算法，与文中提及调整算法的区别在于：就算解变的更劣，模拟退火也以一定概率接受新解。这样的算法不保证调整的单调性，但是扩大了调整的状态空间，因此也是一个可能的优化方向。

参考文献

[1] Nikhil Bansal, On a generalization of iterated and randomized rounding. In *Proceedings of the 51stAnnual ACM SIGACT Symposium on Theory of Computing*, pages 1125–1135, 2019.

[2] CodeForces Round 562 Editorial