浅谈“调整”思想在信息学竞赛中的应用

浙江省绍兴市第一中学 唐文斌

摘要

当前信息学竞赛中的题目难度越来越大、数据关系越来越复杂，往往很难找到一种直接求得最优解的方法。退而求其次，先任意找到一个可行解，再对这个可行解通过一系列调整、变换，不断地对方案进行改进，最终符合我们的要求，这便是一种“调整”的思想。这种思想在信息学竞赛中有着相当广泛的应用，在一些非最优化的开放性问题中更有着杰出的表现，但是它在各类问题中的表现形式又是多种多样的。

本文将选取几个具有代表性的例题，说明“调整”思想在各类问题中的应用，并提炼出它们的共同点。

关键字

信息学竞赛

调整思想　　随机化　　调整/改进/变换

非最优化问题

引入

“调整”这个词语，大家应该都不会陌生，因为在日常的生活工作中经常能听到。例如我们平时洗澡的时候，如果水太烫，那么我们就把水温调低；如果水太凉，就把水温调高，这就是一种“调整”。

“调整”的本意为“改变原有的情况，使之更适应客观环境和要求”[1]，例如“产业结构调整”、“军事战略调整”等等都是通过对结构、战略的调整改良，使之更加优秀，从而赢得更大的利益。

这种思想，在计算机科学中自然并不少见。例如解决线性规划问题的经典算法——“单纯形算法”，以及目前很流行的“模拟退火算法”，都用到了这一思想。那么，让我们来看看这一思想在信息学竞赛解题中的精彩表现把！

“调整思想”在一类构造问题中的应用

[例题一]远程通信（Baltic 2001）

波罗的海上有两个小岛：Bornholm 和 Gotland。在每个小岛上都有一些神奇的远程通信端口，每个通信端口可以运行在两种模式下——发送模式和接收模式。Bornholm 和 Gotland 分别有 n 和 m 个这样的端口，每个端口都连接着另一个小岛某个端口，称为“目标端口”。

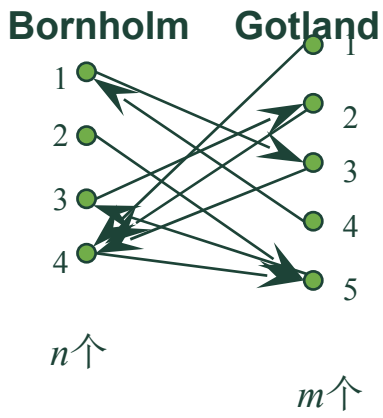
请设置这 n+m 个端口的模式，使得所有端口都处于工作状态，即：

- 对于处于接收模式的端口 A，另一个小岛上至少有一个以 A 为目标端口的端口被设置成发送模式。
- 对于处于发送模式的端口 B，它的目标端口必须处于接收状态。

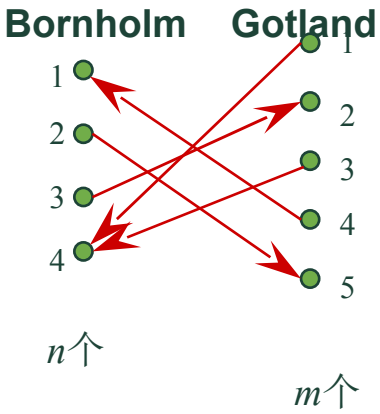
其中 $1 \le n, m \le 50000$。

如下图（每个点指向的点表示它的目标端口）：

[1] 摘自《现代汉语词典》



那么它的一种设置方案为：

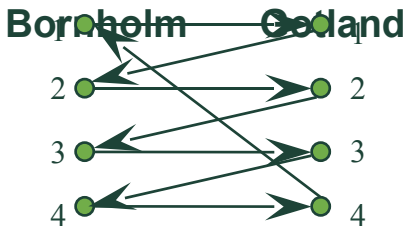


即 Bornholm 岛上 1 号、4 号端口与 Gotland 岛上 2 号、5 号端口被设置为接收状态，其他端口被设置为发送状态。

[分析]

我们先来观察一下样例，也许能带给我们一些比较有用的信息。可以发现，Gotland 上的 1 号、4 号端口，没有其他端口以它们为目标端口。因为所有端口都必须处于工作状态，所以这两个端口必须被设置为发送状态。

由于 Gotland 上的 1 号、4 号端口被设置成发送状态，它们的目标端口（即 Bornholm 上的 1 号端口）必须为设置为接收模式。因为 Bornhome 上 1 号端口被设置为接收模式，从而导致了 Gotland 上的 3 号端口必须被设置为发送模式……以此类推，我们就可以得出答案。然而这个简单的事实并不总是能帮我们找到方案，如下图：



上图中在每个岛中各有 4 个端口，并且对于每个端口都有其他端口以它为目标端口。事实上上图存在两个方案：Bornholm 上的端口都设置为发送模式且 Gotland 上的端口都设置为接收模式，或者反一下，Bornholm 上的端口都设置为接收模式而 Gotland 上的端口都设置为发送模式。也就是说，对于上图，没有哪个端口的模式可以被直接确定，那么我们先前提到的事实就不能帮助我们求得方案了。

虽然上面的事实看起来很有用，但是我们无法直接利用它得到方案。现在我们放弃这条思路，来看一种更简单的方法：

我们设所有 Bornholm 上的端口都处于发送状态，所有 Gotland 上的端口都处于接收状态。显然这样设置并不一定满足条件，因为有些 Gotland 上处于接收状态的端口可能是无用的。那么，我们将通过一种“调整”的方法，改进方案使之满足要求。

我们用伪代码来描述这个“调整”算法：

```
1:  设置 Bornholm 上所有端口为发送状态
2:  设置 Gotland 上所有端口为接收状态
3:  while Gotland 上存在一个无效的接收端口 x  do
4:      改变端口 x 的状态，设置为发送状态
5:      设置端口 x 的目标端口的状态为接收状态
```

第 4 行与第 5 行两部分就是执行我们所谓的“调整”过程。很显然，经过一次调整，Gotland 上的接收端口数目减少一，所以这个算法肯定是会结束的。那么，算法执行得到的方案一定可行么？我们来证明一下：

[证明]

- 对于 Gotland 上的接收端口，必然每一个都是有效的，不然算法不会结束
- 对于 Gotland 上的发送端口，我们在第 5 行设置它的目标端口为接收状态并且其状态不会被改变，所以 Gotland 上的发送端口也处于工作状态。
- 对于 Bornholm 上的接收端口，它会被设置为接收状态的原因在 Gotland 上有一个以它为目标端口的端口被设置为发送状态（第 4、5 行），所以它也处于工作状态。
- 对于 Bornholm 上的发送端口，它的目标端口一开始就被设置成接收状态了。

上述**[证明]**不仅证明了“调整”算法的正确性，同时也说明了任意输入都是有解的。

那么实现就很简单了，以此检查 Gotland 上每一个接收端口 x，如果它的入度为 0（即无用接收端口），则置它为发送状态，如果 x 的目标端口处于发送状态，则设置 x 的目标端口 y 为接收状态，修改 y 的目标端口 z 的入度。而这可能使得 z 变成一个无用的接收端口，

则对 z 再进行类似的调整。

由于每一个端口最多被调整一次，所以“调整”操作的平摊复杂度为 *O(1)*，所以总时间复杂度为$O(n+m)$，这已经是本问题的时间下界了。

[例题二]混合图的欧拉回路问题（经典问题）

给出一个 N 个点和 M 条边的混合图（即有些边是无向边，有的边是有向边）。试求出它的一条欧拉回路，如果没有，输出无解信息。

[分析]

由于无向边只能经过一次，所以本题中不能把一条无向边拆成两条方向相反的有向边，因而原来的“套圈” 算法变得不可行，我们只能把每条无向边定向，再求定向后的有向图的欧拉回路。

我们回想一下有向图存在欧拉回路的充要条件为：基图[2]连通，且每个节点的入度等于出度。

同样的，混合图存在欧拉回路的一个必要条件仍然是基图连通，在此前提下，如果它存在欧拉回路，即我们可以找到一种将无向边定向的方法，使得每个点的入度等于出度。所以如果存在下述情况，问题无解：

- 基图不连通
- 设点 u 在基图中的度数为$Deg[u]$。如果$Deg[u]$为奇数，则无解。如果从 u 发出的有向边个数或者进入到 u 点的有向边个数超过了$\frac{Deg[u]}{2}$，也无解。

而不满足上述情况的混合图，下面我们将知道它是必然有解的。

由上可知，我们的目标是“入度等于出度”，这个与网络流的流量平衡条件十分类似，所以本题可以用网络流来做。但是，该方法较为复杂且容易出错。下面我们来介绍一种应用“调整”思想的**“无向边任意定向算法”**：

顾名思义，首先我们把所有无向边任意定向得到一个“方案”。但事实上这个“方案”是不满足要求的，所以我们加了引号。那么现在的问题就在于，能否通过某些操作，使得原来不满足要求的“方案”成为真正的方案呢？

这正是我们的“调整”操作！

设点 u 在当前“方案”中的入度为$InDeg[u]$，出度为$OuDeg[u]$。显然存在$\sum_{u\in G} InDeg[u]=\sum_{u\in G} OuDeg[u]$。

每一次，我们寻找一个出度大于入度的点 a，即满足$OuDeg[a]>InDeg[a]$。如果找不到这样的点，当前方案已满足要求，因为$\prod_{u\in G} InDeg[u]=\prod_{u\in G} OuDeg[u]$。从点 a 开始，在沿

[2] 基图就是把所有的有向边变成无向边之后得到的图

着被定向的无向边，找到一条通往点 b 的路，满足点 b 的出度小于入度，即 $OuDeg[b] < InDeg[b]$。这样，我们把这条路上所有边的方向都反向，就使得点 a 出度减小、入度增加，点 b 出度增加、入度减小，而对于路上的中间点，入度出度都没有改变。这便是我们找寻的“调整”操作。点 a、点 b 出入度的变化意味着，我们现在得到的“方案”，比原有的“方案”出入度更“平衡”。所以只要不断找这样的点 a，找路径调整，即可。

现在只剩下一个问题，如果存在出度大于入度的点 a，是否每次都能沿着被定向的无向边找到一条通往满足出度小于入度的点 b 的路呢？

答案是肯定可以。其实这个也是很显然的，更正式一些，下面我们来给出证明：

[证明]

我们从一个出度大于入度的点 a 开始，目标是通过被定向的无向边走向一个入度大于出度的点 b。

假设我们当前到达了点 u，如果 u 的入度大于出度，那么已经达到目标。否则，u 的出度大于等于入度。由于从 u 发出的有向边个数或者进入到 u 点的有向边个数都不超过 $\frac{Deg[u]}{2}$（否则就无解），这说明我们肯定通过被定向的无向边走到另一个点 v。又由于总入度等于总出度，所以我们也不可能一直在出度大于等于入度的点上绕圈。

那么，从当前点 u 必然能往下走到 v，并且总能走到一个入度大于出度的点 b。

这个证明也表明了不满足上文中提到的几个条件的混合图是必然有解的。

每一次调整的时间复杂度为 $O(n+m)$，而调整的次数不会超过 $\frac{\hat{o}m\ddot{i}}{\square\sqrt{}\square}$。所以总时间复杂度为 $O(m*(n+m))$。

[小结]

对于上述两个例题，有一些线索启发着我们，但是我们却找不到好的方法利用线索直接得到答案。这个时候，我们先得到一个随意的“方案”，而这个“方案”可能是不符合要求的，经过逐步的“调整”，使得方案更加趋向于满足条件，最终得到符合要求的方案。调“不可行”为“可行”，从无到有，正是“调整”思想在这类构造问题中的应用。

“调整”思想在非最优化的开放性问题中的应用

[例题三] 零件装配（提交答案）

话说 HURRICANE 小组在实习中切割好了全部所需的 $n\hat{o}m$ 个不同的零件之后，又需要把他们装配在一起成为可以使用的产品。现已知每个产品均分为 m 个部分，每个部分恰

好为之前完成的一个零件。在装配产品的一个部分时，可以在为该部分设计出来的 *n* 个零件之间任选一个进行安装（但每个零件只能安装一次）。但由于结构上的差异，不可以选择为其他部分所设计的零件进行安装。

工厂内恰好有 *n* 条装配线可以用作产品的装配，所以可以同时开始 *n* 个产品的装配过程加快装配的速度。但不同的零件可能有不同的装配时间，所以我们在装配产品的时候可以通过选择适当的装配方案来减少最晚完成产品的装配时间。

你的任务就是给出一个装配方案，使得最晚完成产品的装配线所用时间尽可能少。

注：本题是一个提交答案题，一共给出 10 个输入数据。在最大数据中，*n* 和 *m* 高达 500。

[分析]

本题实际上就是给定一个矩阵，矩阵中同一列的数可以互相调换。要求输出一个经过若干次调换后的矩阵，使得矩阵中每行所有数的和中最大的和最小。

这个问题实际上是背包问题的一个扩展，因而是 NP 完全的。所幸题目并不是要我们求最优解，而是求一个尽可能优的解。

对于本题，动态规划、搜索显然不能胜任。在本题中动态规划的状态数目就是指数级的；而搜索，对于 N=500 这种规模也是望尘莫及的。

那么我们可以尝试一下贪心算法：

要求最大和最小，等价于要求所有和尽量平均。因为如果一个偏小了，必然有另一个偏大了。所以一个很直观的贪心想法就是把最大的和最小的凑在一起。

我们一列一列考虑，当我们处理第 i 列时，前 i-1 列已经安排好，不妨设第 p 行中已经安排好的前 i-1 个数的和为 S[p]。对于 i 列中的 n 个数，把最大的分配给 S[x]最小的第 x 行，把次大的分割 S[y]次小的第 y 行，以此类推。直到安排完所有数。

经测试可以发现，这个贪心算法得到的解并不是很优，因为贪心算法只关心眼前状态，缺乏全局观，而局部的最优，可能导致全局的不优。

那是否有其他更好的方法呢？

这正是“调整”思想大显身手的时候！

我们的目标是使得最大的和最小，也就是让所有的和尽量平均。假设有方案 A，设 $S[i]=\sum_{j=1..M} A[i][j]$，其中 i=1..N。如果存在两个数 A[r1][c] 和 A[r2][c]，使得交换这两个数后，S[r1]与 S[r2]将更加平均，即满足如下不等式：

$$|S[r1]-S[r2]|<|(S[r1]-A[r1][c]+A[r2][c])-(S[r2]-A[r2][c]+A[r1][c])|$$

那么我们交换这两个数，会使得 S 更加平均，也就是更加趋向于较优解，所以交换这两个数会使得方案更优。如果找不到这样两个数，我们不妨称这样的方案是“极优”的，也就是说不能同过交换两个数得到更优的方案。很显然，最优的方案必然是“极优”的。

所以我们的调整算法就是：

Step 1: 寻找两个元素 A[r1][c] 和 A[r2][c]，满足将其交换后 S[r1]与 S[r2]更加接近，如果找得到，转到 Step 2；否则转到 Step 3。

Step 2: 交换 A[r1][c] 和 A[r2][c]，转到 Step 1。

Step 3: 得到“极优”方案 A。

由于寻找满足条件 A[r1][c] 和 A[r2][c]是任意的，所以不同的初始方案 A，可能会得到不同的“极优”方案。那么我们可以多次随机不同的初始状态，经过上述调整之后，求得若干“极优”的方案，取一个最优的。事实证明，这个效果非常好。

[例题四]皇帝的困惑

有 N（N ≤ 10000）箱黄金，第 i 箱黄金的价值为 A[i]（0≤A[i]≤1000）。皇帝要将这 N 箱黄金赏给 M（M≤1000）位将军，每个人可以获得任意箱，但是一箱黄金不能分开发，只能发一位将军。请找出一种分黄金的方案，使得获得最多黄金的将军与获得最少黄金的将军两人得到的黄金数相差不超过 K。题目保证有解。

[分析]

稍加分析可知，这题也是 NPC 的，并且 N 和 M 还很大，所以也不能搜索。黄源河同学曾对本题做了深入的研究，使用了三种不同策略的贪心方法解决了该题（详见参考文献[A]）。这里简单介绍一下黄源河同学提出的这三种贪心方法：

[贪心一：发牌式贪心]

把这 N 箱黄金从大到小依次分发，每次发给当前拥有黄金最少的将军。时间复杂度为 $O(N\lg N+N\lg M)$。

[贪心二：针对于 K≤1 的发牌式贪心]

由于[贪心一]失败的数据大部分都是 K≤1，所以这里我们提出针对于 K≤1 的贪心二。

设 $\overline{A}=\left\lfloor \frac{\sum_{i=1..N} A[i]}{M} \right\rfloor$。如果 K=0，那么每个人都应该拿到 $\overline{A}$ 数量的黄金。如果 K=1，则应该有 $\sum_{1..N} A[i] \bmod M$ 的人得到 $\overline{A}$ +1 数量的黄金，而剩下的人得到 $\overline{A}$ 数量的黄金。这样，我们给每个人设置一个指标——即他应该拿多少黄金。然后使用发牌式的贪心，从大到小分配每一箱黄金，每个人拿的黄金数量不能超过他的指标。时间复杂度为 $O(N*M)$。

[贪心三：贪心二的加强版]

试想有两种方案，一种是选择最大的和最小的，另一种是选择第二大和第三大的。直观上应该是后者较好，因为我们可以把最小的留下来，去给其他人凑指标。而贪心二则会选择前一个方案。这里我们换一种分法，依次考虑每一个人。对于每一个人，给他找一个凑足指标的方法，并且他拿到的最小数最大。这个过程，实际上就是一个 01 背包问题，可以使用动态规划解决。总时间复杂度为 $O(N*\sum_{i=1..N} A[i])$。

虽说一个题目用三种算法有点不太光彩，但是这个做法巧妙地利用了数据间的关系。［贪心一］可以解决绝大多数 K 较大的数据。而一般随机生成的数据最优解往往是 K=0 或者 K=1，于是针对 K=0 或者 K=1 的情况增加了[贪心二]与[贪心三]。

巧妙利用数据的特点，设计三种算法解决问题，这正是一种“开放性”思想的体现。所以这种做法还是相当值得借鉴的。

那么，难道真的只能用如此多而繁琐的三种贪心么？有其他简捷明快的办法么？

“调整”思想让我们挺起胸膛，坚定地说“Yes！”。

我们来看一个合法方案，设分给第 i 个将军的黄金集合为 $S[i]$，设他得到的黄金数量为 $Sum[i]=\sum_{x\in S[i]}A[x]$。我们将每一个将军得到的黄金集合分为两个不相交的部分，即 $S[i]=SA[i]\cup SB[i]$ 且有 $SA[i]\cap SB[i]=\varnothing$。相应的，设 $SumA[i]=\sum_{x\in SA[i]}A[x]$，$SumB[i]=\sum_{x\in SB[i]}A[x]$。那么，可以用下面这个二分图来表示本方案。

现在，我们忽略原有的分配的信息，而把上图中每一个部分都看成一个独立的个体，每个将军要从上下两部分中各取一份。由贪心思想可知，我们让上面部分最大的一份与下面部分中最小的一份搭配，以此类推，可以使得总和更加平均（至少不会更差）！

那么这便是本题“调整”的过程。

可以发现，这个调整过程是相当简洁的。那么现在的问题是，如何把每一个将军分成两个部分呢？我们可以如下方法：

我们要把一个将军得到的黄金集合 $S[i]$ 分成两个不相交部分 $SA[i]$ 与 $SB[i]$。我们给集合 SA 设置集合一个指标 C。按照任意顺序，不断地把 $S[i]$ 中的黄金放到 $SA[i]$ 中，当 $SumA[i]$ 即 $SA[i]$ 中黄金数目之和达到 C 的时候停止，剩下的部分都放到 $SB[i]$ 中。而这个 C，可以随机指定。

可以知道，每一次调整的时间复杂度为 $O(N+M\lg M)$。而事实告诉我们，使用这个调整方法，经过很少几次调整就得到了满足条件的解答，运行效果非常好。

[小结]

在**[例题三]**中，我们是对方案中的单个元素进行调整，即“局部调整”。而在**[例题四]**中，我们是把整个方案进行一次“大手术”，即“整体调整”。两种调整法虽然在调整的形式和规模上有所不同，但是都得到了非常好的效果□

其实，**[例题三]**也可以用“整体调整”法，我们随机的选择一列，这一列中每一个元素作为二分图集合 A 中一个点。剩下的 M-1 列，每一行的集合作为二分图的另一个集合 B 中的一个点，使用类似于**[例题四]**中的最大-最小、次大-次小匹配。得到一个新方案。

同样，**[例题四]**也可以用“局部调整”，即每次尝试交换两个元素，只是这里要多加一个调整方法，即把一个元素从一个集合移动到另一个集合，因为每一个将军可以得到任意多箱黄金。

所以，这里解法并不是唯一的，我们只是选择了两个比较简单但是具有一定代表性的例子，希望能起到抛砖引玉的作用。□

总结

让我们先来回顾一下本文中提到的几个例子。

在**[例题一]**与**[例题二]**中，目标是求得符合某些条件的方案。而面对这样一个构造性的问题，虽然有一些初步的想法，却并不能帮助我们求得解答，让我们感觉有些无从下手，这个时候，先任意得到一个不满足条件的“方案”，再对其进行一系列的“调整”操作，最终符合要求。这正是“调整”思想在一类构造性问题中的作用：调“不可行”为“可行”。

在**[例题三]**与**[例题四]**中，目标是寻找一种较优的方案。而问题复杂、规模巨大，使常规方法有些束手无策。这个时候，我们从一个并不太优的方案开始，对其进行一系列的“调整”操作，最后得到较优甚至最优的方案。这便是“调整”思想在一类非最优化的开放性问题中的作用：调“可行”为“更优”。

调“不可行”为“可行”，这是一个“从无到有”的过程。

调“可行”为“更优”，这是一个“从有到优”的过程。

从无到有，从有到优，这正是“调整”思想的精髓所在。

感谢

感谢上海交通大学辛韬同学对于**[例题四]**的帮助

感谢长沙市雅礼中学雷涛同学提供**[例题一]**

感谢刘汝佳教练的一些提示

Acm.timus.ru

Acm.zju.edu.cn

参考文献

[A] 黄源河同学 2004 年的国家集训队作业《The Emperor's Riddle（URAL 1144）解题报告》

[B] Baltic Olympiad In Informatics 2001 Problem && Solution

[C]《算法艺术与信息学竞赛》 刘汝佳、黄亮

附录

[附录 A] 模拟退火算法简介

模拟退火算法来源于固体退火原理。将固体加温至温度充分高，再让其徐徐冷却。加温时，固体内部粒子随温升变为无序状，内能增大；而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态,内能减为最小。

根据 Metropolis 准则,粒子在温度 T 时趋于平衡的概率为 $e-\frac{\Delta E(\text{内能改变量})}{k(Boltzmann\text{常数})*T}$。

模拟退火算法的流程为

(1)初始化: 初始温度 T(足够大),初始解(S),L(每个温度的迭代次数)

(2)For k = 1 □ L Do

(3) 产生新解 S’

(4) 计算增量 dt’ = C(S’) – C(S)

(5) 如果 d’t < 0 接受新解 S’作为当前解

否则以概率 exp(- dt’ / T)接受 S’

(6) 如果满足终止条件则终止

(7)温度 T 减小(但保证 T>0),回到第(2)步

可以发现，(3)到(5)步这个过程，实际上就是在往寻找一个更优的解前进，这正是一种“调整”的思想。

[附录 B] 例题源程序：

[例题一]《远程通信》源程序：

Teleports.cc

[例题三]《零件装配》源程序：

Assembly.cc