对于问题一，

算法时间复杂度为

任务是包含了洒水作业时间，路程时间

路程时间= 前去洒水点的时间+回补给点时间

为的是任务时间最小。

决策变量



若它是解，则为1，否则为0(我考虑了一下，不应该写成，因为这是决策变量，具体来说，是这个意思

如果以变量集合代表一种可行方案的话，那么就记



所以真正关心的不是变量X的值，而是这个变量X为1时候的具体方案，即ijk)。

因此我们即是要求解

对于满足约束条件，且使得最远时间最小（目标函数）的，所有，那么表示一组的可行方案。

强调这句话的意思是，关注的是方案，而不是X本身。

为了方便，洒水车的出发点跟编号大小有关系，1-10在D1

为了记号方便，记表示决策变量中出现的洒水车，也就是这个决策中待执行洒水任务的洒水车。*u*代表车型

那么初始时，停靠在两个站点的洒水车如下



使用获取车型



也就是说表示第

用



获取每辆洒水车的出发点

用



这个获取每辆洒水车的洒水作业时间 。

为了方便，也常记为

待优化的目标函数是 ，为使得最长洒水任务的那个洒水车的任务时间最小化，即



应该满足以下约束：

第一问的约束

每辆洒水车只能去喷洒点执行喷洒作业一次。



每个喷洒点只能被喷洒一次。



每个补给点不能补给超过8次



额外的约束，最长任务时间肯定超过每辆车本身大于等于自身执行任务的时间。当它是最长的执行任务时间时取等。



其中每辆洒水车的任务时间由下式组成：

任务时间=前去作业点的时间+喷洒作业的时间+返回补给点的时间

也就是



写成数学表达式就是：









其中表示，构建了以任意两点可达距离为权值（并非直接距离）的邻接矩阵，通过Floyd算法求解出的代价矩阵。

（需要插入Floyd算法吗？）

由于不同类型的车行驶速度不同，并且在支道和主道的行驶速度也不同。所以对于每一种车型都有其代价矩阵。其权值表示时间开销。

比如就代表类型为*u*的汽车，从它的停靠点（起点）到作业点的时间开销。

其中表示，在部分决策i确定的（即此时i为已知），而其他决策j,k的所有情况下的洒水时间。所以这是一个集合。

第一问的求解结果

任务完成用时：3.04 h

平均用时：2.46 h

编号：A-1用时：2.2ht路线：D1 -> F58(作业点) -> Z03(补水点)

编号：A-2用时：2.42ht路线：D1 -> F32(作业点) -> Z04(补水点)

编号：A-3用时：2.43ht路线：D1 -> F30(作业点) -> Z04(补水点)

编号：A-4用时：2.27ht路线：D1 -> F34(作业点) -> Z04(补水点)

编号：A-5用时：2.24ht路线：D1 -> F42(作业点) -> Z05(补水点)

编号：A-6用时：2.33ht路线：D1 -> F39(作业点) -> Z05(补水点)

编号：B-7用时：2.72ht路线：D1 -> F35(作业点) -> Z04(补水点)

编号：B-8用时：3.04ht路线：D1 -> F43(作业点) -> Z05(补水点)

编号：B-9用时：1.96ht路线：D1 -> F57(作业点) -> Z03(补水点)

编号：B-10用时：2.93ht路线：D1 -> F40(作业点) -> Z05(补水点)

编号：A-11用时：2.24ht路线：D2 -> F49(作业点) -> Z02(补水点)

编号：A-12用时：2.21ht路线：D2 -> F48(作业点) -> Z02(补水点)

编号：A-13用时：2.15ht路线：D2 -> F25(作业点) -> Z01(补水点)

编号：A-14用时：2.5ht路线：D2 -> F44(作业点) -> Z01(补水点)

编号：A-15用时：2.21ht路线：D2 -> F26(作业点) -> Z01(补水点)

编号：A-16用时：2.42ht路线：D2 -> F45(作业点) -> Z01(补水点)

编号：B-17用时：2.87ht路线：D2 -> F46(作业点) -> Z01(补水点)

编号：B-18用时：2.37ht路线：D2 -> F47(作业点) -> Z01(补水点)

编号：B-19用时：2.84ht路线：D2 -> F24(作业点) -> Z01(补水点)

编号：B-20用时：2.8ht路线：D2 -> F50(作业点) -> Z01(补水点)

问题二

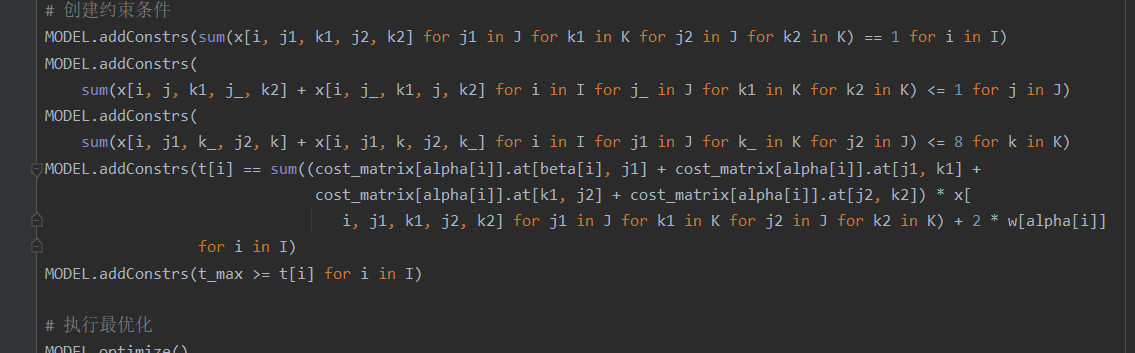
这是未改进的

如果以表示的最优解 ，以表示 ，这好比优化



那么原优化目标为





，从for的层数可以看出，算法的时间复杂度是



尽管线性规划有成熟的理论求解，但这显然不是一个好的优化目标。而且从题中可以看出，可选的喷洒作业点是很多的，是非常大的。那么以上算法的时间复杂度很高，第二次指派洒水任务时，未充分利用第一次已经执行了洒水任务的信息，而造成了对一些不必要的情况迭代，这也就是暴力求解（穷举）。显然，随着问题规模的增大，任务越来越复杂，暴力求解并不是个好的方法，以下可以避免如此高的复杂度。

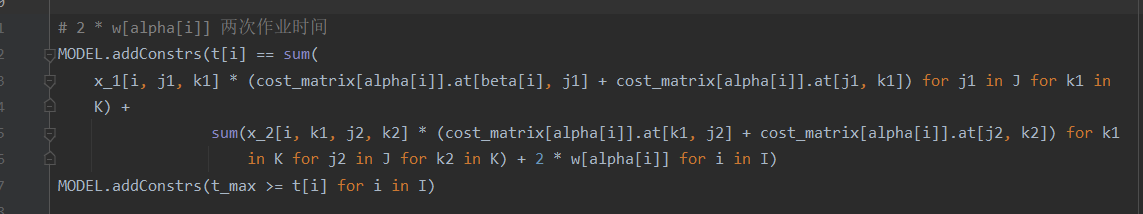
因此，考虑优化



其中表示在第一步求解的局部最优解的情况下，再去求解两部的全局最优解。而我们知道，每辆洒水车执行完作业后，都会回到补给点。那么第二次执行洒水任务，只依赖于第一次洒水任务执行完，每辆洒水车所在的补给点位置。那么可写出算法时间复杂度为



这就将算法复杂度下降一阶。



第二问的约束



引入两段决策变量。都以x记，但以下标区分。具体来说，第一段决策变量同问题1，，第二段决策变量

问题二有以下约束：

第一阶段每个洒水车只能喷洒一次



第二阶段每个洒水车只能喷洒一次



第二阶段的起点是第一次喷洒后前往的补给点。



对于每辆车，不能喷洒同一个作业点



（这个条件隐含了下两个条件）

第一阶段每个喷洒点至多被喷洒一次（见上）

第二阶段每个喷洒点至多被喷洒一次（见上）

在这两次作业中，补给点仍然只能最多补给8辆



对于每辆洒水车，其任务完成时间，为从出发到第二次到达补给点所经历的时间

这是第一问的任务时间



仿照上式，可写出第二问的任务时间一共会路径转移四次（因为一共会经过5个点，含出发点），所以会有四次代价+2\*作业时间构成。



列出规划：

目标：



约束：



















第二问求解的答案

Multi-objectives: solved in 3.26 seconds, solution count 7

任务完成用时：4.69 h

平均用时：4.46 h

编号：A-1t用时：4.56ht路线：D1 -> F20(作业点 1) -> Z06(补水点 1) -> F19(作业点 2) -> Z06(补水点 2)

编号：A-2t用时：4.41ht路线：D1 -> F38(作业点 1) -> Z05(补水点 1) -> F34(作业点 2) -> Z04(补水点 2)

编号：A-3t用时：4.68ht路线：D1 -> F36(作业点 1) -> Z06(补水点 1) -> F13(作业点 2) -> Z06(补水点 2)

编号：A-4t用时：4.62ht路线：D1 -> F30(作业点 1) -> Z04(补水点 1) -> F11(作业点 2) -> Z04(补水点 2)

编号：A-5t用时：4.5ht路线：D1 -> F37(作业点 1) -> Z06(补水点 1) -> F12(作业点 2) -> Z06(补水点 2)

编号：A-6t用时：4.65ht路线：D1 -> F39(作业点 1) -> Z05(补水点 1) -> F23(作业点 2) -> Z05(补水点 2)

编号：B-7t用时：4.63ht路线：D1 -> F43(作业点 1) -> Z05(补水点 1) -> F41(作业点 2) -> Z05(补水点 2)

编号：B-8t用时：4.38ht路线：D1 -> F42(作业点 1) -> Z05(补水点 1) -> F40(作业点 2) -> Z05(补水点 2)

编号：B-9t用时：4.49ht路线：D1 -> F57(作业点 1) -> Z03(补水点 1) -> F58(作业点 2) -> Z03(补水点 2)

编号：B-10t用时：4.06ht路线：D1 -> F35(作业点 1) -> Z04(补水点 1) -> F33(作业点 2) -> Z04(补水点 2)

编号：A-11t用时：4.64ht路线：D2 -> F49(作业点 1) -> Z02(补水点 1) -> F28(作业点 2) -> Z02(补水点 2)

编号：A-12t用时：4.65ht路线：D2 -> F27(作业点 1) -> Z02(补水点 1) -> F51(作业点 2) -> Z03(补水点 2)

编号：A-13t用时：4.16ht路线：D2 -> F45(作业点 1) -> Z01(补水点 1) -> F52(作业点 2) -> Z01(补水点 2)

编号：A-14t用时：4.57ht路线：D2 -> F01(作业点 1) -> Z06(补水点 1) -> F10(作业点 2) -> Z06(补水点 2)

编号：A-15t用时：4.49ht路线：D2 -> F25(作业点 1) -> Z02(补水点 1) -> F31(作业点 2) -> Z04(补水点 2)

编号：A-16t用时：4.42ht路线：D2 -> F44(作业点 1) -> Z01(补水点 1) -> F53(作业点 2) -> Z01(补水点 2)

编号：B-17t用时：4.16ht路线：D2 -> F24(作业点 1) -> Z01(补水点 1) -> F47(作业点 2) -> Z01(补水点 2)

编号：B-18t用时：4.37ht路线：D2 -> F26(作业点 1) -> Z02(补水点 1) -> F48(作业点 2) -> Z02(补水点 2)

编号：B-19t用时：4.69ht路线：D2 -> F29(作业点 1) -> Z04(补水点 1) -> F32(作业点 2) -> Z04(补水点 2)

编号：B-20t用时：4.12ht路线：D2 -> F46(作业点 1) -> Z01(补水点 1) -> F50(作业点 2) -> Z01(补水点 2)

第三问的约束

引入三端决策变量。都以x记，但以下标区分。具体来说，第一段决策变量同问题1，，第二段决策变量，第三段决策变量

问题二有以下约束：

第一阶段每个洒水车只能喷洒一次



第二阶段每个洒水车只能喷洒一次



第三阶段每个洒水车只能喷洒一次



第二阶段的起点是第一次喷洒后前往的补给点。



第三阶段的起点是第二次喷洒后前往的补给点。



对于每辆车，三次作业不能喷洒同一个作业点



（这个条件隐含了下两个条件）

第一/二/三阶段每个喷洒点至多被喷洒一次（见上）

第一/二/三阶段每个喷洒点至多被喷洒一次（见上）

在这三次作业中，补给点仍然只能最多补给8辆



对于每辆洒水车，其任务完成时间，为从出发到第二次到达补给点所经历的时间

列出规划：

目标：



约束：





















第四问的约束

那篇博客的聚类，大概意思是把“现有的6个补给点强行作为6个聚类中心，然后再选2个“这样写进规划的约束里。然后对每个点进行分类就行了。

引入决策变量，表示从（候选点集合，作业点集合）

（为了与前文保持一致，这里记号是k,i而不是k,j，为的就是保持（作业点）这个符号表示含义不变。

如果以表示8个聚类中心的话，那么，也就是



对于从J几几那些点选的新补给点，将他们与Z一起并入集合里的元素，那么它也得满足补给不超过8次

限制了必须有8个补给点，其中每个候选点（包含6个补给点）都有两种选择，要么可以成为新补给点，要么不是补给点



但是这并没有约束选出了多少个新补给点，那么需增加



每个作业点只能被喷洒一次



对于再高规模的算法，显然只能用启发式算法。比如遗传算法。

后面问题可能的分步决策



有理由说明。（降低问题规模），聚类的思想

