

数值最优化课程 project

车辆路径规划中的分配问题

晏浩洋 徐荻

2021 年 1 月 3 日

1 前言

本篇 report 附件 *code.zip* 中含有 10 个文件

vrp_task.txt 为源数据文件, *data.mat* 为预处理后的 mat 数据文件, *taxi.m*, *solver1.m*, *solver2.m* 为 matlab 代码文件, *solution1.mat*, *solutionh5.mat*, *solutionh7.mat*, *solutionh9.mat*, *solutionh11.mat* 为求解结果文件

taxi.m 可以将源数据的 txt 文件预处理为 0-1 变量储存为 mat 形式, 方便后续求解, 输出结果即为 *data.mat*

solver1.m 是利用 matlab 自带整数型线性规划求解函数 *intlinprog* 构建的求解器

solver2.m 是只利用 matlab 线性规划求解函数 *linprog* 和 *branch & bound*(分支定界法) 思路构建的求解器, 为本篇 report 的主要介绍部分, 其中分支层数 h 可以作为参数根据实际情况调整, h 越大, 求解结果越精确, 求解速度越慢

solution1.mat 为在 *data.mat* 基础上 *solver1.m* 的求解结果, 其中变量 *opt* 为 120×1 向量, 即对应 120 个 task 求得的最低成本, 变量 *x* 为 120×1 元胞, 每个元胞内为长度为 Scores 的仅含 0,1 的向量, 即对应 task 的最优方案, 1 即代表该行被采纳到最优方案中

solutionh5.mat, *solutionh7.mat*, *solutionh9.mat*, *solutionh11.mat* 为在 *data.mat* 基础上 *solver2.m* 取分支层数 $h = 5, 7, 9, 11$ 时对应的求解结果, 变量 *opt*, 变量 *x* 含义同上

2 数据预处理

本部分对应代码文件 *taxi.m*

对于源数据中总共 120 个 task, 每一个 task 创建对应 $Scores \times 3$ 的元胞, 3 列信息分别代表订单 ID, 司机 ID, 分配成本, 依次命名为 O, V, c 。根据 Orders 和 Vehicles 数量分别构建 $Orders \times 1$ 和 $Vehicles \times 1$ 的全零向量, 赋值给每一行的 O, V

利用 *fget1* 函数逐行读取源数据, 将此行 O, V 的第订单 ID, 司机 ID 个元素修改为 1, 即表示该行对应的订单和司机

Task=1 Orders=39 Vehicles=333 Scores=1536

Task	Order	Vehicle	Score
1	1	6	56.82
1	1	18	15.72
1	1	27	13.63
1	1	37	36.11
1	1	40	41.41
1	1	51	48.15
1	1	55	41.63
1	1	60	16.18
1	1	63	52.37
1	1	66	21.94
1	1	78	54.73
1	1	93	52.25

图 1: 源数据文件 *vrp_task.txt* 格式

	1	2	3	4
1	39x1 double	333x1 double	56.8200	
2	39x1 double	333x1 double	15.7200	
3	39x1 double	333x1 double	13.6300	
4	39x1 double	333x1 double	36.1100	
5	39x1 double	333x1 double	41.4100	
6	39x1 double	333x1 double	48.1500	
7	39x1 double	333x1 double	41.6300	
8	39x1 double	333x1 double	16.1800	
9	39x1 double	333x1 double	52.3700	
10	39x1 double	333x1 double	21.9400	
11	39x1 double	333x1 double	54.7300	
12	39x1 double	333x1 double	52.2500	
13	39x1 double	333x1 double	32.1100	
14	39x1 double	333x1 double	26.8600	

图 2: 预处理后的数据格式

3 建模

原问题即可表述为

$$\begin{aligned}
 \min \quad & c^T x \\
 \text{s.t.} \quad & o^T x = 1 \quad \forall o \in O \\
 & v^T x \leq 1 \quad \forall v \in V \\
 & x \in \{0, 1\} \quad \forall x
 \end{aligned} \tag{1}$$

4 整数型线性规划求解器

本部分对应代码文件 *solver1.m*

利用 matlab 自带整数型线性规划求解函数 *intlinprog*, 可直接对原问题求解

求解总耗时 5.074s, 调用 *intlinprog* 函数 120 次, 平均每个 task 耗时 0.042s, 除 task109 无解之外, 其余均求得最优解

函数名称	调用次数	总时间	自用时间
solver1	1	5.074 s	0.793 s
intlinprog	120	4.281 s	0.015 s
_hAndCut>intlinprogBranchAndCut.run	120	3.998 s	0.045 s
sbiClient	120	3.603 s	0.078 s
optim/private/sbiMex (MEX-file)	120	2.287 s	2.287 s
sbiClient>changeConstraintForm	120	0.584 s	0.561 s
sbiClient>getOptions	120	0.479 s	0.038 s
_edLinear.performExitflagAndMsgActions	120	0.350 s	0.018 s
sbiClient>getFixedOptions	120	0.324 s	0.010 s
sbiClient>prepareTempFileName	360	0.314 s	0.034 s
tempname	360	0.241 s	0.057 s
_dLinear>GeneralizedLinear.checkRun	120	0.193 s	0.005 s
intlinprogBranchAndCut.checkProblem	120	0.186 s	0.020 s

图 3: 整数型线性规划求解器耗时

	1	2	3
1	619.9700		
2	394.3800		
3	498.0900		
4	784.9600		
5	419.2100		
6	809.9100		
7	792.1300		
8	436.2300		
9	672.0400		
10	472.5100		
11	618.4000		
12	885.8900		
13	582.6000		
14	445.0700		

图 4: 整数型线性规划求解器输出结果

5 分支定界求解

本部分对应代码文件 *solver2.m*

由于本次 project 要求不能使用现有整数型线性规划求解器，因此尝试采用分支定界法结合已有的线性规划函数 *linprog* 自行编写求解器

本求解器总体思路为：先将原问题中的整数约束 $x \in \{0,1\}$ 松弛为 $0 \leq x \leq 1$ ，求解该线性规划问题，对于解中非整数的 x_i ，分别讨论其为 0 和 1 的情况，继续求解线性规划，如果该子问题无解或解超出已有最优值，则放弃该支；如果有解但依然无整数解，则继续分支讨论；如果出现整数解且该解对应成本小于已有最优值，则更新最优值

该求解器伪代码如下，具体代码请参考代码文件 *solver2.m*

```
1  对于每一个task
2      松弛掉原问题的整数约束，求解线性规划问题
3      判定，如果无解，则终止
4      对于解与0或1相差小于 $10^{-5}$ 的，近似为0,1
5      找到解中的非整数的位置及个数，命名为undef和noun
6      如果noun=0，即为整数解，最优解即为该解
7      否则，将undef的第一位放进def，将最优值设为正无穷
8      创建5棵高度为h的空的二叉树，即长度为 $2^h-1$ 向量，对应编号从上到下，从左到右排列，命名及含义分别为
9          issol，是否有解树，0表示无整数解，1表示有整数解，2表示无解，3表示解超过最优值上界
10         def，已确定订单编号树(仅在有整数解节点赋值)
11         xdef，已确定订单取0还是1树(在所有子节点赋值)
12         opti，最优值树(仅在有整数解节点赋值)
13         xxi，解树(在有解节点赋值)
14  对于每一个不是根节点的节点k
15      判定其父节点编号
16      如果issol(父节点)=0
17          如果孩子节点为其父节点的左儿子,则不采纳其父节点的undef
18          其xdef在父节点基础上增加元素0
19          删掉O，V，c对应的def(父节点)列
20          对于约束右边的1，减去O或V的第(def(父节点)*xdef(k))
21          求解该线性规划子问题
22          判定，如果无解，则终止此支
23          对于解与0或1相差小于 $10^{-5}$ 的，近似为0,1
24          找到解中的非整数的位置及个数，命名为undef和noun
25          将已确定的def插入回本次整数解,得到xxi(k)
26          该方案成本opti(k)=c*xxi(k)
27          如果该方案成本大于已有最优值，则终止此支
28          如果该方案为整数解
29              如果该方案成本小于已有最优值
30                  更新已有最优值，最优解
31              终止此支
29          如果该方案不为整数解
32              将undef第一位放进def(注意该处undef坐标是在已删除def列之后的，要换回原来坐标)
33              继续分支
34          如果孩子节点为其父节点的右儿子,则采纳其父节点的undef
35          其xdef在父节点基础上增加元素1
36          其余同上
37
```

取分支层数 $h = 5, 7, 9, 11$ 时分别测试该求解器，结果如下

函数名称	调用次数	总耗时	自用时间	总时间/调用 (深色条带 = 自用时间)
solver2	1	69.215 s	9.663 s	
linprog	882	59.552 s	0.935 s	
optim/private/lpsol	882	58.373 s	12.105 s	
optim/private/lpsol-ratiotest	28464	14.947 s	14.947 s	
optim/private/lpsol-direction	28464	14.659 s	12.108 s	
optim/private/lpsol-getpu	15118	5.942 s	3.233 s	
spdlags	29354	5.702 s	5.702 s	
spones	46418	4.283 s	4.283 s	
optim/private/lpsol-feasibility	15118	2.488 s	1.272 s	
optim/private/lpsol-reciprocal	42708	1.612 s	1.612 s	
optim/private/lpsol-complementy	15282	1.577 s	1.577 s	
colamd	882	0.764 s	0.183 s	
cnarkm/private/cnarkmconv	882	0.552 s	0.511 s	

图 5: 分支定界法, $h=5$

函数名称	调用次数	总耗时	自用时间	总时间/调用 (深色条带 = 自用时间)
solver2	1	116.254 s	16.972 s	
linprog	1392	99.282 s	1.445 s	
optim/private/lpsol	1392	97.462 s	19.544 s	
optim/private/lpsol-direction	45034	26.363 s	22.407 s	
optim/private/lpsol-ratiotest	45034	25.147 s	25.147 s	
optim/private/lpsol-getpu	23913	9.678 s	5.442 s	
spdlags	46434	8.962 s	8.962 s	
spones	73437	6.539 s	6.539 s	
optim/private/lpsol-feasibility	23913	3.928 s	2.114 s	
optim/private/lpsol-reciprocal	67563	2.581 s	2.581 s	
optim/private/lpsol-complementy	24308	2.311 s	2.311 s	
colamd	1392	1.224 s	0.291 s	
cnarkm/private/cnarkmconv	1392	0.889 s	0.828 s	

图 6: 分支定界法, $h=7$

函数名称	调用次数	总耗时	自用时间	总时间/调用 (深色条带 = 自用时间)
solver2	1	194.259 s	30.416 s	
linprog	2130	163.843 s	2.210 s	
optim/private/lpsol	2130	161.065 s	30.264 s	
optim/private/lpsol-direction	67512	49.888 s	43.822 s	
optim/private/lpsol-ratiotest	67512	41.038 s	41.038 s	
optim/private/lpsol-getpu	35890	14.557 s	8.405 s	
spdlags	69650	13.069 s	13.069 s	
spones	110252	9.977 s	9.977 s	
optim/private/lpsol-feasibility	35890	6.005 s	3.244 s	
optim/private/lpsol-reciprocal	101280	3.895 s	3.895 s	
optim/private/lpsol-complementy	37087	3.531 s	3.531 s	
colamd	2130	1.916 s	0.432 s	
cnarkm/private/cnarkmconv	2130	1.415 s	1.323 s	

图 7: 分支定界法, $h=9$

函数名称	调用次数	总耗时	自用时间	总时间/调用 (深色条带 = 自用时间)
solver2	1	341.445 s	60.899 s	
linprog	3836	280.545 s	3.995 s	
optim/private/lpsol	3836	275.564 s	48.126 s	
optim/private/lpsol-direction	112324	107.330 s	99.838 s	
optim/private/lpsol-ratiotest	112324	65.584 s	65.584 s	
optim/private/lpsol-getpu	60002	19.562 s	11.878 s	
spdlags	116168	16.214 s	16.214 s	
spones	184336	12.420 s	12.420 s	
optim/private/lpsol-feasibility	60002	8.533 s	5.019 s	
optim/private/lpsol-reciprocal	168498	5.304 s	5.304 s	
optim/private/lpsol-complementy	63560	4.876 s	4.876 s	
colamd	3836	3.134 s	0.605 s	
cnarkm/private/cnarkmconv	3836	2.416 s	2.259 s	

图 8: 分支定界法, $h=11$

将分支定界法求解结果与整数型线性规划求解器求解结果对比

$h=5$ 时, 求解总耗时 69.215s, 调用 *linprog* 函数 882 次, 平均每个 task 耗时 0.577s, 120 个 task 中 3 个无解, 其余 117 个 task 中 99 个求得最优解, 平均每个 task 成本高出最优值 0.323 元

$h=7$ 时, 求解总耗时 116.254s, 调用 *linprog* 函数 1392 次, 平均每个 task 耗时 0.969s, 120 个 task 中 1 个无解 (与整数型线性规划求解器结果相同), 其余 119 个 task 中 108 个求得最优解, 平均每个 task 成本高出最优值 0.259 元

$h=9$ 时, 求解总耗时 194.259s, 调用 *linprog* 函数 2130 次, 平均每个 task 耗时 1.619s, 120 个 task 中 1 个无解 (与整数型线性规划求解器结果相同), 其余 119 个 task 中 115 个求得最优解, 平均每个 task 成本高出最优值 0.040 元

$h=11$ 时, 求解总耗时 341.445s, 调用 *linprog* 函数 3836 次, 平均每个 task 耗时 2.845s, 120 个 task 中 1 个无解 (与整数型线性规划求解器结果相同), 其余 119 个 task 中 117 个求得最优解, 平均每个 task 成本高出最优值 0.023 元