可拼单型车辆订单分配模型

高世杰

摘要: 针对传统车辆路径规划问题并未考虑到可拼车条件的影响，为适应当下拼车出行越发普遍化的影响，本文将拼车作为新的订单模式纳入路径规划问题的考虑范围，并建立了对应的模型。本文首先建立了简易的图模型用来模拟现实世界的情况，然后基于用户效用考量规定了可拼单条件，并通过该模型获得相应的数据，最后根据该数据在满足所有用户需求的前提下极小化全体司机的总成本，建立优化模型。[本论文仅考虑 2 人一组的打包单]

Keywords: 路径规划、订单分配、拼单、接单条件

1. 引言

在最近几年的移动互联网发展中，出现了许多为用户解决即时分配问题的大型平台，特别是在打车和外卖领域。平台要解决的一个核心问题就是是订单分配，这种需求通常是随机的，并且有一定聚集性。如何快速地把这些订单分给周围的司机，既要满足最多的用户需求，又要使司机的服务成本最低，这便是平台需要思考的问题。随着新的车辆不断投入运行，高峰时段的路面变得更加得拥赌，用户能够接受得等待时间越来越短，需求日益增加，为了能够尽快得乘车，拼单的接受率不断增加，因此建立可拼单型车辆订单分配模型是改善当下运行状况的有效方案。

2. 图模型及有效单条件建立

2.1 图模型模拟城市网络

为了建立订单分配模型，最小化全体司机的服务成本，需要如 A 司机接订单 B 所花费的费用等相关数据的支持，因此需要对现实的数据进行采集。将城市视作网格建立图模型能够简单的模拟真实的情况。

模型对象　车、人 (起点/请求)、终点

模型假设　1. 一个格子表示一个点，所有点均能存在对象；
2. 任意相邻点间有弧连接，不相邻点间无弧连接
3. 任意相邻点间距离相等
4. 单位距离，单位时间，单位花费三者等价，因此均不带单位
5. 司机未接单前等待时间不算入乘客等车时间
6. 司机先接一单，再接一单视作两单，而非打包单
7. 一个点能同时存在车，人，终点，但每个数量上限为 1

输入　输入车辆位置，乘客位置及其对应终点

输出　输出 (车辆, 可负责订单，花费) 一组关系

* 订单可分为两种: 每一个请求为一种订单; 任意两个请求合并为一种订单 (打包单)

2.2 有效单条件

考虑到实际生活中一个司机并非适合去接所有打车的用户，不同用户的请求也并非均适合打包为一个订单，因此我们站在乘客耗时角度添加有效单条件，去除不合适的订单。

增设输入　1. 在司机接单后，被负责乘客的最长等待时间 W
2. 用户因打包单而花费的最长额外时间 T

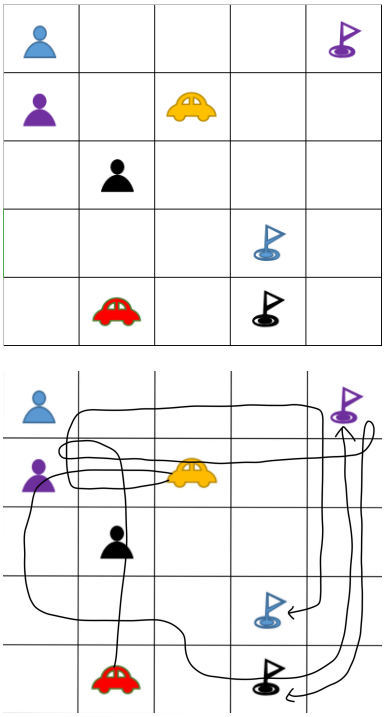
* 额外时间 = 真实时间-理想时间，其中真实时间为从接单开始到用户到达终点的耗时；理想时间为在该司机仅接这名用户的情况下从接单开始到用户到达终点的耗时

有效单条件　1. 用户等待时长 $w_i \leq$ W
2. 打包单中任意一名用户额外花费时间 $t_i \leq$ T

推论　打包单可行的必要条件为在该司机仅负责其中一名用户时满足有效单条件 1

2.3 案例分析

相关信息　1. 尺寸: 5×5(x=0–4,y=0–4)
2. 蓝色人:1　紫色人:2　黑色人:3
3. 黄车:1　红车:2
4. W=4　D=7



情况 *1*　首先假设 $D=\infty$，其他不变，可得输出 [(1, 1, 9), (1, 2, 7), (1, 3, 6), (1, (1, 2), 11), (1, (2, 3), 13), (2, 2, 9), (2, 3, 6), (2, (2, 3), 14)]。根据输出做简要分析: 1. 由于 2 号车距 1 号用户为 5，超过最长等待时间，因此不为可行单。2. 根据计算可得订单 (1, (1, 2), 11) 的额外时间由 1 号用户决定，t=2; (1, (2, 3),13) 的额外时间由 2 号用户决定，t=6; (2, (2, 3), 14) 的额外时间由 3 号用户决定，t=8。(* 打包单路径如上图显示)

情况 *2*　现令 D=7，可得输出 [(1, 1, 9), (1, 2, 7), (1, 3, 6), (1, (1, 2), 11), (1, (2, 3), 13), (2, 2, 9), (2, 3, 6)]，可以看到对于订单 (2,(2,3),14)，3 号用户的额外时间 t=8>7，所以该订单不为可行单。

2.4 总结

通过建立图模型以及有效单条件的约束，实现了 (司机，订单，花费) 三者的关联，并只保留了符合实际情况的可行单。为之后的订单分配问题提供了对应的数据，且保证了后续模型所讨论的订单均为本模型中的可行单，降低了后续模型的复杂度。

3. 订单分配模型

以实现所有司机总盈利最大化为目标，进行订单分配模型的建立。在固定时刻考虑总收益 R=M-O，其中 M 为乘客所付的车费，O 为油费；当车辆未接单时，可视作未行动不产生花费；同时由于起步价内单位收益最高，考虑盈利最大化的目标可以转化为极小化订单花费/路程，以达到单位时间收益最高的目的。因此最终的目标函数为极小化所有司机的总花费。

3.1 假设

由于绝大多数情况下司机的人数超过用户，且一个用户如不能被至少一个司机服务，那么就不将其纳入本轮订单分配的考虑中. 所以假设对于任意的用户请求至少有一个司机能够服务到。

3.2 模型设计

集合　V：表示所有司机的集合
R：表示所有请求的集合
T：表示所有订单的集合
T_i：表示司机 i 的可行单　$\forall i \in V$

参数　c_{ij}：表示 i 司机负责 j 订单的费用　$i \in V,\ j \in T_i$
d_{kj}：表示 k 请求是否在 j 订单中　$k \in R, j \in T$

决策变量　x_{ij}：表示 i 司机是否负责 j 订单　$i \in V,\ j \in T_i$

模型

$$min \quad \sum_{i \in V} \sum_{j \in T_i} c_{ij} x_{ij} \tag{1}$$

$$s.t. \quad \sum_{j \in T_i} x_{ij} \leq 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i \in V} \sum_{j \in T_i} d_{kj} x_{ij} = 1 \quad \forall k \in R \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in V, \forall j \in T_i \tag{4}$$

3.3 案例

tiny case　司机: rider1,rider2,rider3
乘客: request,request2,request3
订单: trip1,trip2,trip3,trip4,trip5

不同订单所包含的乘客，司机能负责的订单及其费用在以下表格中给出:

	trip1	trip2	trip3	trip4	trip5
request1	T	F	F	T	T
request2	F	T	F	T	F
request3	F	F	T	F	T

	trip1	trip2	trip3	trip4	trip5
rider1	10	5	4	F	11
rider2	9	F	9	F	F
rid3	5	2	F	10	F

最终方案　总花费为:13,司机与订单的负责关系为:(rider1,trip5),(rider3,trip2)。

large case　对测试案例中的 Task1 进行符合模型要求的数据转换，最终求解可得总花费：619.97；司机与订单的负责关系为:(rider18,trip53),(rider58,trip23),(rider159,trip77),(rider234,trip22) (rider20,trip9),(rider74,trip4),(rider193,trip39),(rider239,trip71) (rider21,trip58),(rider86,trip6),(rider198,trip1),(rider246,trip16),(rider22,trip45),(rider107,trip66),(rider208,trip47),(rider332,trip14) (rider35,trip57),(rider121,trip69),(rider216,trip28), (rider44,trip37),(rider125,trip29),(rider218,trip62), (rider50,trip42),(rider130,trip26),(rider219,trip67), 其中每个 trip 具体对应的乘客可由转换过程中的数据直接给出。(* 订单与乘客关系表在附录中给出)

4. 总结

本文旨在将新增拼单式订单纳入车辆订单分配问题进行分析建模。首先简单模拟了实际的运行情况，并基于用户的效用考量设置了有效单条件，其中有效单条件 1 为普通订单所需满足的条件，而有效单条件 2 则是针对打包单新增的约束条件。通过上述模拟实现了 (司机，订单，花费) 三者的关联，获得了所有可行方案的相关数据，这也与测试案例中所给出的数据格式相吻合。最后根据获得的数据信息，在满足所有客户均被服务的前提下，极小化总体的花费建立订单分配模型。在附件中用 python 实现了第一部分的模型，并用 ampl 实现了订单分配模型。对于相关的车辆订单分配问题，可以考虑用第一部分的模型输出订单分配模型所需的数据格式并用其求解，或自行给出订单分配模型所需的数据格式用其求解即可。

5. 展望

本文只针对 2 人一组的打包单展开了讨论，在实际生活中可能会产生更多人的打包单，但一般不会超过 3 人一组的打包单，因此可以基于现有的 2 人打包单订单分配模型实现允许 3 人的打包单模型。另外本文并未实现对 python，excel，ampl/线上 solver, 三者的数据联系，需要分步求解, 可在此方面设计一个一步式的程序方便问题的求解。

6. 参考文献

1. Alonso-Mora J , Samaranayake S , Wallar A , et al. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment[J]. Proceedings of the National Academy of Sciences, 2017, 114(3):462-467.

7. 附录

trip[1]:	request1		trip[42]:	request16	request34
trip[2]:	request1	request29	trip[43]:	request16	request35
trip[3]:	request2		trip[44]:	request17	
trip[4]:	request2	request12	trip[45]:	request17	request26
trip[5]:	request3		trip[46]:	request18	
trip[6]:	request3	request24	trip[47]:	request18	request33
trip[7]:	request4		trip[48]:	request19	
trip[8]:	request4	request6	trip[49]:	request19	request37
trip[9]:	request4	request13	trip[50]:	request20	
trip[10]:	request4	request20	trip[51]:	request20	request25
trip[11]:	request5		trip[52]:	request20	request28
trip[12]:	request5	request7	trip[53]:	request20	request38
trip[13]:	request5	request18	trip[54]:	request21	
trip[14]:	request5	request23	trip[55]:	request21	request36
trip[15]:	request5	request33	trip[56]:	request21	request38
trip[16]:	request6		trip[57]:	request21	request39
trip[17]:	request6	request13	trip[58]:	request22	
trip[18]:	request6	request20	trip[59]:	request22	request38
trip[19]:	request6	request38	trip[60]:	request23	
trip[20]:	request7		trip[61]:	request24	
trip[21]:	request7	request23	trip[62]:	request25	
trip[22]:	request7	request32	trip[63]:	request26	
trip[23]:	request8		trip[64]:	request26	request31
trip[24]:	request8	request30	trip[65]:	request27	
trip[25]:	request9		trip[66]:	request27	request30
trip[26]:	request9	request35	trip[67]:	request28	
trip[27]:	request10		trip[68]:	request28	request38
trip[28]:	request10	request37	trip[69]:	request29	
trip[29]:	request11		trip[70]:	request30	
trip[30]:	request11	request21	trip[71]:	request31	
trip[31]:	request12		trip[72]:	request32	
trip[32]:	request13		trip[73]:	request33	
trip[33]:	request13	request20	trip[74]:	request34	
trip[34]:	request13	request22	trip[75]:	request34	request35
trip[35]:	request13	request38	trip[76]:	request35	
trip[36]:	request14		trip[77]:	request36	
trip[37]:	request14	request19	trip[78]:	request36	request39
trip[38]:	request14	request37	trip[79]:	request37	
trip[39]:	request15		trip[80]:	request38	
trip[40]:	request15	request19	trip[81]:	request39	
trip[41]:	request16				