**2019重庆邮电大学数学建模竞赛编号专用页**

**1.参赛队选择的题号信息与编号**

选择的题号：B题

阅卷编号：

注：阅卷编号由阅卷组老师在阅卷前填写。

**2. 参赛队员信息**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 队员1 | 队员2 | 队员3 |
| 姓名 | 陈恒宇 | 邓韦 | 殷寰宇 |
| 学号 | 2018210427 | 2018210379 | 2018210226 |
| 学院 | 通信学院 | 通信学院 | 通信学院 |
| 专业 | 通信工程 | 信息工程 | 电子信息工程 |
| 签名 |  |  |  |

注：学院填写学校规定统一的各个简称（如:通信学院、理学院、自动化学院等）。年级为入学年级（如2017级等），队员签名（**签名一定要手写**）表示遵守下面的承诺书。

**承 诺 书**

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。我们知道，抄袭别人的成果是违反竞赛章程和参赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平。

**手写承诺书如下:**

“两江游”轮船调度问题

摘要

本文先针对“两江游”轮船调度问题，利用整数规划和动态规划求得最优航程安排方案，并给出航行时间表。再通过旅游辐射场强模型，结合新航线码头周边景点相关数据确定码头客流量并给出新“两江游”航程安排，最后通过灰色GM(1,1)模型预测未来发展态势,并分析新航线对原有航线的影响。

针对问题一，考虑到轮船启航时刻以10分钟为基本单位，且载客量要求在90人到150人之间，确定了等候时间的几种可能取值。因17点前后游客到达码头的速率不同，在求解时分为17点时游轮在码头和不在码头两种情况。假设影响航程最优性的三个因素的重要性从高到低分别为：运营收入最大、运营次数尽量少和每次运载人数较为均衡，将收益设定为目标函数，同时根据载客量要求和运行时间等限制条件建立整数规划模型，编程求得最优解为白天每班等候50分钟，晚上每班等候30分钟，一天出航7次且全部满载，总载客量为1050人。

针对问题二，首先以游客的角度观察码头有无游轮，将多艘等效为一艘，通过游轮的有效等待时间和等效航行时间之间的关系，得出递推关系式。其次结合其他变量列出状态转移方程，然后根据载客量与时间的限制条件建立动态规划模型，编程求得最佳航行时间表，最后推出最少需要的游轮数为三艘，每天出航17次，其中15次满载，1次载120人，1次载90人，总载客量为2460人。

针对问题三，因为游客愿意花最多10分钟等待游轮返回，故9：50开始有游客到达码头。当返航游轮的有效航行时间不为零时，认为码头已有等候了10分钟的游客，因此游轮的有效等待时间减少10分钟，类比问题一、问题二建立数学模型，编程求得只有一艘游轮时的最优航程为一天出航8次，其中7次满载，1次载120人，总载客量为1170人；允许多艘游轮时，最少使用三艘可使得利润最大，一天出航17次，其中16次满载，1次载90人，总载客量为2490人。

针对问题四，首先通过查阅地图和分析周边景点密集程度的方式设定了新航线上的五个码头：磁器口码头、洪崖洞旅游客运码头、朝天门码头、海棠溪码头和相国寺码头，并给出了每个码头的环游路线。其次依据旅游辐射场强模型求出每个码头周边景点对该码头的辐射总场强，并根据2014年至2018年的“两江游”资料数据建立灰色G(1,1)模型，求出未来三年内“两江游”的客流量。最后结合2018年各月的“两江游”游客与航班次数数据、各码头的辐射总场强比值以及2019年“两江游”总载客量给出新航线2019年的航程安排，并分别评估分析其对原来“两江游”的正面和负面影响。

最后，我们对模型进行了优缺点评价，并给出了改进方案和推广方案。

**关键词：轮船调度 整数规划 动态规划 旅游辐射场强模型 灰色GM(1,1)模型 最优航程**

1 问题重述

1.1 问题背景

重庆“两江游”是指乘坐游轮欣赏长江与嘉陵江的景色，由朝天门码头出发，行至黄花园大桥，经两江交汇处重返朝天门码头，全程约20公里。为了使得游轮公司利益最大化，需要在保证运营收入最大的同时，尽量减少运营次数，同时使得每次运载人数较为均衡。因此，建立合理的航行时刻表，并结合现状给出新的航线安排显得尤为重要。

1.2 题目信息

目前提供“两江游”服务的游轮每艘满载量为150人，启航时载客量需达到满载人数的60%，每次游览时间约1小时，票价为118元/人。上午10点到下午17点，平均每分钟3人到达码头乘坐游轮；下午17点至晚上21点，平均每分钟5人到达码头乘坐游轮；其他时间段内几乎没有乘客。

1.3 题目要求

本文通过建立数学模型，着重解决以下问题：

（1）如果轮船公司只有1艘游轮，应当如何安排航程？一天的总载客量是多少?

（2）如果轮船公司有多艘游轮，应最少使用几艘？分别如何安排航程？每艘船载客量是多少？

（3）在游客愿意在港口花最多10分钟等待游轮返回的情况下，重新求解问题1和问题2。

（4）建立一条从磁器口沿嘉陵江到朝天门,经朝天门码头沿长江到九滨路(鹅公岩大桥)的新航线，并在沿途设置若干个码头。查询目前“两江游”游客的相关资料，建立数学模型，进行新航线的码头设置和航程安排，分析新航线的发展态势并对其对原来航线的影响进行评估。

2 模型假设

（1）游轮航行时间固定为1小时，不会提早或延迟回到码头 。

（2）有游轮停靠时游客愿意一直等待直到游轮起航。

（3）设定最优航程时三个考虑因素的重要性从高到低分别为：运营收入最大、运营次数尽量少和每次运载人数较为均衡。

（4）问题1、2中早上10点前和晚上21点后几乎没有游客达到码头。

（5）问题3中早上9：50开始有游客到达码头，且所有游客都愿意花10分钟等待游轮返航。

（6）不考虑游客上下船时间。

（7）轮船启航时刻以10分钟为基本单位。

（8）如果同时有多艘游轮停靠，后一艘需在前一艘离开后才允许上客，但返回的船下客不受影响。

（9）轮船公司的运营成本分为每次航行的运营成本与每天固定运营成本。

（10）“两江游”总人数不随新航线的开行而改变。

3 问题分析

3.1 问题一的分析及对策

问题一要求在轮船公司只有1艘游轮的情况下，最合理地安排航程并求出一天的总载客量。考虑到轮船启航时刻以10分钟为基本单位，因此游轮每次到达码头距下一次启航的时间，即每次等待时间为10分钟的整数倍。因载客量要求在90人到150人之间，通过计算可知白天的每次等待时间只能为30分钟、40分钟或50分钟，晚上的每次等待时间只能为20分钟或30分钟。考虑到下午17点前后游客到达码头的速率不同，在求解时分为下午17点游轮在码头和不在码头两种情况。假定设定最优航程时三个考虑因素的重要性从高到低分别为：运营收入最大、运营次数尽量少和每次运载人数较为均衡，将收益设定为目标函数，相同时再比较运载人数的均衡性。同时根据载客量要求和运行时间等限制条件建立整数规划模型，求出最优解。

3.2 问题二的分析及对策

问题二要求在轮船公司有多艘游轮的情况下，最合理地安排航程并求出一天的总载客量。采用将多艘游轮等效为一艘的方法，观察码头有无游轮，若有则表示游轮在等待载客量达到标准，没有则表示游轮在航行过程。当所有到达码头的乘客都参与“两江游”时游轮公司收益最大，中因此需要尽可能保证码头时刻有船。当时，显然无法达到这一要求，因此将游轮数量分为和两种情况分别讨论。根据变量间的相互关系建立动态规划模型，求出最佳航行时间表，并根据班次安排推出船的数量。

3.3 问题三的分析及对策

问题三要求在游客愿意花最多10分钟等待游轮返回的情况下，重新给出只有1艘游轮和有多艘游轮时的最优航程安排。考虑9:50就有乘客到达码头开始等待，针对只有1艘游轮的情况，每次游轮到达码头后的等待时间分别减少10分钟，即17点前只需额外等20分钟、30分钟或40分钟，17点后只需额外等10分钟或20分钟，建立整数规划模型求出最优解；针对多艘游轮，类比问题二，建立动态规划模型，并分别求出游轮数和两种情况下的最佳航行时间表，并根据排班情况推出最少需要的船数。

3.4 问题四的分析及对策

问题四要求建立一条“两江游”新航线，进行新航线的码头设置和航程安排，分析新航线的发展态势并评估其对原来航线的影响。首先通过查阅地图和分析周边景点密集程度的方式设定了沿线包括朝天门码头在内的五个码头，其次依据旅游辐射场强模型求出每个码头周边景点对该码头的辐射总场强，并将其作为安排新航线航程的重要依据。然后根据2014年至2018年的“两江游”资料数据建立灰色G(1,1)模型，预测未来“两江游”的发展态势。最后结合2018年各月的“两江游”数据、辐射总场强以及航线未来发展情况给出新航线的航程安排，并评估分析其对原来“两江游”的影响。

4 符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
|  | 总利润 |
|  | 载客量 |
|  | 航行总次数 |
|  | 等待时间 |
|  | 等待时间为的次数 |
|  | 船数 |
|  | 每次航行的运营成本 |
|  | 每天运行的固定成本 |
| m | 出航次数 |
|  | 第次出航的等效航行时间 |
|  | 第次出航的有效等待时间 |
|  | 第次返航后的总时间 |
|  | 游客到达码头的速率 |
|  | 景点和码头之间的最短时间距离 |
|  | i对j的辐射场强 |
|  | 景点质量 |

5 建模与求解

5.1 问题一

5.1.1 问题一的分析

问题一要求在轮船公司只有1艘游轮的情况下，最合理地安排航程并求出一天的总载客量。考虑到轮船启航时刻以10分钟为基本单位，因此游轮每次到达码头距下一次启航的时间，即每次等待时间为10分钟的整数倍。因载客量要求在90人到150人之间，通过计算可知白天的每次等待时间只能为30分钟、40分钟或50分钟，晚上的每次等待时间只能为20分钟或30分钟。考虑到下午17点前后游客到达码头的速率不同，在求解时分为下午17点游轮在码头和不在码头两种情况。假定设定最优航程时三个考虑因素的重要性从高到低分别为：运营收入最大、运营次数尽量少和每次运载人数较为均衡，将收益设定为目标函数，相同时再比较运载人数的均衡性。同时根据载客量要求和运行时间等限制条件建立整数规划模型，求出最优解。

5.1.2 模型的建立与求解

令 ；分别表示对应等待时间为出现的次数，为总载客量，为航行总次数。

情况一：17点时游轮在航行的过程中

航程最优要求运营收入最大、运营次数尽量少且每次运载人数较为均衡，故建立目标函数如下：

其中，

因17点游轮在航行过程中，且白天的最后一次出发时间在17点之前，故白天的总时间应大于分钟且小于；同时考虑全天的最后一班游轮在21点前发出，故全天总时间应小于分钟。因此建立约束条件如下：

编程枚举出符合整数规划的所有解（表格1）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 |  |  |  |  |  | 航行次数 | 总载客量 | 总利润 |
| 1 | 0 | 0 | 4 | 0 | 2 | 6 | 900 | 106200-6 |
| 2 | 0 | 0 | 4 | 0 | 3 | 7 | 1050 | 123900-7 |
| 3 | 0 | 0 | 4 | 1 | 1 | 6 | 850 | 100300-6 |
| 4 | 0 | 0 | 4 | 1 | 2 | 7 | 1000 | 118000-7 |
| 5 | 0 | 0 | 4 | 2 | 1 | 7 | 950 | 112100-7 |
| 6 | 0 | 0 | 4 | 3 | 0 | 7 | 900 | 106200-7 |
| 7 | 0 | 1 | 3 | 0 | 2 | 6 | 870 | 102660-6 |
| 8 | 0 | 1 | 3 | 0 | 3 | 7 | 1020 | 120360-7 |
| 9 | 0 | 1 | 3 | 1 | 2 | 7 | 970 | 114460-7 |
| 10 | 0 | 1 | 3 | 2 | 1 | 7 | 920 | 108560-7 |
| 11 | 0 | 1 | 3 | 3 | 0 | 7 | 870 | 102660-7 |
| 12 | 0 | 2 | 2 | 0 | 3 | 7 | 990 | 116820-7 |
| 13 | 0 | 2 | 2 | 1 | 2 | 7 | 940 | 110920-7 |
| 14 | 0 | 2 | 2 | 2 | 1 | 7 | 890 | 105020-7 |
| 15 | 0 | 2 | 2 | 3 | 0 | 7 | 840 | 99120-7 |
| 16 | 1 | 0 | 3 | 0 | 3 | 7 | 990 | 116820-7 |
| 17 | 1 | 0 | 3 | 1 | 2 | 7 | 940 | 110920-7 |
| 18 | 1 | 0 | 3 | 2 | 1 | 7 | 890 | 105020-7 |
| 19 | 1 | 0 | 3 | 3 | 0 | 7 | 840 | 99120-7 |
| 20 | 2 | 3 | 0 | 0 | 2 | 7 | 840 | 99120-7 |
| 21 | 2 | 3 | 0 | 1 | 1 | 7 | 790 | 93220-7 |
| 22 | 2 | 3 | 0 | 2 | 0 | 7 | 740 | 87320-7 |
| 23 | 2 | 3 | 0 | 3 | 0 | 8 | 840 | 99120-8 |
| 24 | 3 | 1 | 1 | 0 | 2 | 7 | 840 | 99120-7 |
| 25 | 3 | 1 | 1 | 1 | 1 | 7 | 790 | 93220-7 |
| 26 | 3 | 1 | 1 | 2 | 0 | 7 | 740 | 87320-7 |
| 27 | 3 | 1 | 1 | 3 | 0 | 8 | 840 | 99120-8 |
| 28 | 3 | 2 | 0 | 0 | 2 | 7 | 810 | 95580-7 |
| 29 | 3 | 2 | 0 | 1 | 1 | 7 | 760 | 89680-7 |
| 30 | 3 | 2 | 0 | 2 | 0 | 7 | 710 | 83780-7 |
| 31 | 3 | 2 | 0 | 2 | 1 | 8 | 860 | 101480-8 |
| 32 | 3 | 2 | 0 | 3 | 0 | 8 | 810 | 95580-8 |
| 33 | 4 | 0 | 1 | 0 | 2 | 7 | 810 | 95580-7 |
| 34 | 4 | 0 | 1 | 1 | 1 | 7 | 760 | 89680-7 |
| 35 | 4 | 0 | 1 | 2 | 0 | 7 | 710 | 83780-7 |
| 36 | 4 | 0 | 1 | 2 | 1 | 8 | 860 | 101480-8 |
| 37 | 4 | 0 | 1 | 3 | 0 | 8 | 810 | 95580-8 |
| 38 | 4 | 1 | 0 | 0 | 2 | 7 | 780 | 92040-7 |
| 39 | 4 | 1 | 0 | 1 | 1 | 7 | 730 | 86140-7 |
| 40 | 4 | 1 | 0 | 1 | 2 | 8 | 880 | 103840-8 |
| 41 | 4 | 1 | 0 | 2 | 0 | 7 | 680 | 80240-7 |
| 42 | 4 | 1 | 0 | 2 | 1 | 8 | 830 | 97940-8 |
| 43 | 4 | 1 | 0 | 3 | 0 | 8 | 780 | 92040-8 |
| 44 | 5 | 0 | 0 | 0 | 2 | 7 | 750 | 88500-7 |
| 45 | 5 | 0 | 0 | 0 | 3 | 8 | 900 | 106200-8 |
| 46 | 5 | 0 | 0 | 1 | 1 | 7 | 700 | 82600-7 |
| 47 | 5 | 0 | 0 | 1 | 2 | 8 | 850 | 100300-8 |
| 48 | 5 | 0 | 0 | 2 | 0 | 7 | 650 | 76700-7 |
| 49 | 5 | 0 | 0 | 2 | 1 | 8 | 800 | 94400-8 |
| 50 | 5 | 0 | 0 | 3 | 0 | 8 | 750 | 88500-8 |

表格 1

情况二：17点时游轮在码头

令，分别表示17点前后的等待时间，则

航程最优要求运营收入最大、运营次数尽量少且每次运载人数较为均衡，故建立目标函数如下：

其中，

因17点时游轮在码头，因此等待过程中经历了游客到达速度为3人/分钟和5人/分钟两种情况，等待时间分别为，且游客总数在90人到150人之间。考虑到17点之前到达码头，且到达后的等待时间小于50分钟，故白天的总时间应大于分钟且小于；同时考虑全天的最后一班游轮在21点前发出，故全天总时间应小于分钟。因此建立约束条件如下：

编程枚举出符合整数规划的所有解（表格2）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 |  |  |  |  |  | 航行次数 | 总载客量 | 总利润 |
| 1 | 2 | 2 | 0 | 0 | 2 | 7 | 840 | 99120-7A |
| 2 | 2 | 2 | 0 | 1 | 1 | 7 | 790 | 93220-7A |
| 3 | 2 | 2 | 0 | 2 | 0 | 7 | 740 | 87320-7A |
| 4 | 2 | 2 | 0 | 3 | 0 | 8 | 840 | 99120-8A |
| 5 | 3 | 0 | 1 | 0 | 2 | 7 | 840 | 99120-7A |
| 6 | 3 | 0 | 1 | 1 | 1 | 7 | 790 | 93220-7A |
| 7 | 3 | 0 | 1 | 2 | 0 | 7 | 740 | 87320-7A |
| 8 | 3 | 0 | 1 | 3 | 0 | 8 | 840 | 99120-8A |

表格 2

5.1.3 结果分析

由表格1和表格2可知，每天的运营次数为6次、7次和8次时，最大利润分别为106200-6A、123900-7A和106200-8A元，考虑到实际情况下游轮每次航行成本不可能大于载客量为90人时的收入10620元，而运营次数为6次和7次时的收入差为17700元，所以每天运营7次时的利润最大，且每次载客150人，满足载客量均衡的要求。航行时刻表如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 班次 | 启航时刻 | 回程时刻 | 载客量 |
| 1 | 10:50 | 11:50 | 150 |
| 2 | 12:40 | 13:40 | 150 |
| 3 | 14:30 | 15:30 | 150 |
| 4 | 16:20 | 17:20 | 150 |
| 5 | 17:50 | 18:50 | 150 |
| 6 | 19:20 | 20:20 | 150 |
| 7 | 20:50 | 21:50 | 150 |

表格 3

5.2 问题二

5.2.1 问题二的分析

问题二要求在轮船公司有多艘游轮的情况下，最合理地安排航程并求出一天的总载客量。采用将多艘游轮等效为一艘的方法，观察码头有无游轮，若有则表示游轮在等待载客量达到标准，没有则表示游轮在航行过程中。当所有到达码头的乘客都参与“两江游”时游轮公司收益最大，因此需要尽可能保证码头时刻有船。当时，显然无法达到这一要求，因此将游轮数量分为和两种情况分别讨论。根据变量间的相互关系建立动态规划模型[6]，求出最佳航行时间表，并根据班次安排推出船的数量。

5.2.2 模型的建立与求解

设阶段变量表示出航次数，状态变量表示第次出航的等效航行时间，决策变量表示第次出航前的等效等待时间，表示第次返航后的总时间，表示第次返航后的总载客量,表示游客达到码头的速率。

情况一:

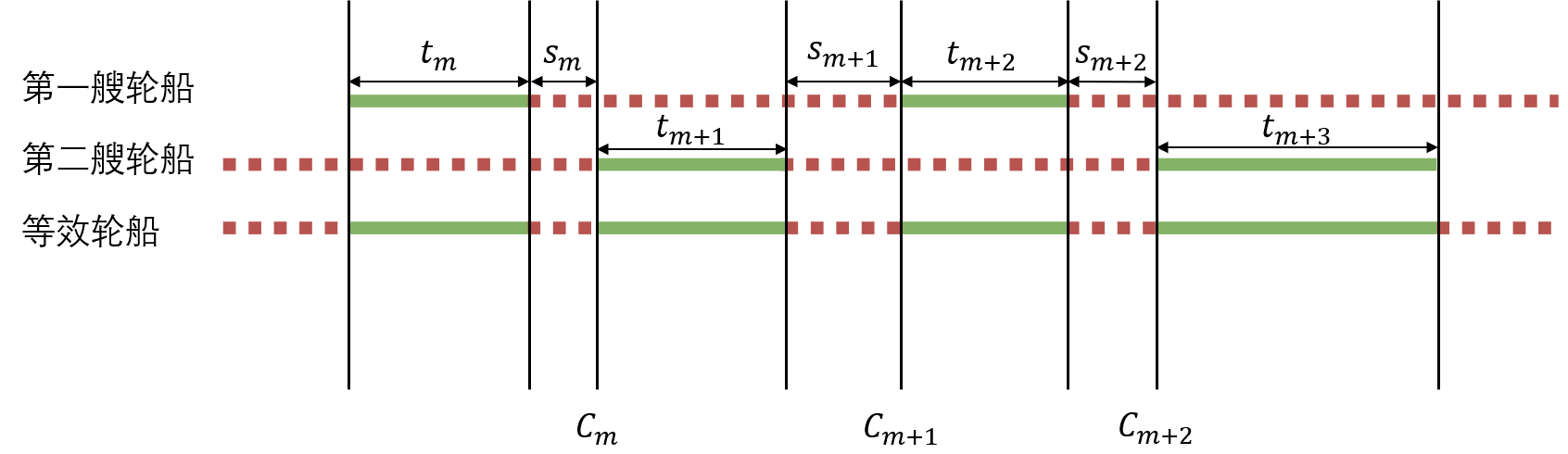


图 1

以岸上游客的角度观察码头的轮船状态，从而将多艘游轮等效为一艘。图1中，前两行分别表示两艘船的航行情况，最后一行表示等效轮船的航行和等待情况。线段表示在码头等待的时间，虚线表示航行的时间，因此可以看出与和的关系，由此得到递推关系式：

阶段M的允许决策集合为：

第M次返航后的总时间为：

第M次返航后的总载客量为：

游客达到码头的速率

最优指标表示第次出航到最后一次出航总载客量的最大值。根据最优化原理可以得到状态转移方程：

,

通过编程求得最优解见表格4。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 出航次数 | 有效等待时间 | 等效航行时间 | 总时间 | 单次载客量 | 总载客量 |
| 1 | 30 | 0 | 30 | 90 | 90 |
| 2 | 50 | 10 | 90 | 150 | 240 |
| 3 | 50 | 0 | 140 | 150 | 390 |
| 4 | 50 | 10 | 200 | 150 | 540 |
| 5 | 50 | 0 | 250 | 150 | 690 |
| 6 | 50 | 10 | 310 | 150 | 840 |
| 7 | 50 | 0 | 360 | 150 | 990 |
| 8 | 30 | 30 | 420 | 90 | 1080 |
| 9 | 30 | 0 | 450 | 150 | 1230 |
| 10 | 30 | 30 | 510 | 150 | 1380 |
| 11 | 30 | 0 | 540 | 150 | 1530 |
| 12 | 30 | 30 | 600 | 150 | 1680 |
| 13 | 30 | 0 | 630 | 150 | 1830 |
| 14 | 30 | 10 | 720 | 150 | 1980 |

表格 4

最大利润

情况二：

在所有到达码头的乘客都参与“两江游”时游轮公司收益最大，因此需要保证码头时刻有船，即。此时建立的动态规划模型如下：

阶段M的允许决策集合为：

第M次返航后的总时间为：

第M次返航后的总载客量为：

游客达到码头的速率

最优指标表示第次出航到最后一次出航总载客量的最大值。根据最优化原理可以得到递推关系式：

,/分钟

通过编程求得最优解见表格5。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 出航次数 | 有效等待时间 | 总时间 | 单次载客量 | 总载客量 |
| 1 | 50 | 50 | 150 | 150 |
| 2 | 50 | 100 | 150 | 300 |
| 3 | 40 | 140 | 120 | 420 |
| 4 | 30 | 170 | 90 | 510 |
| 5 | 50 | 220 | 150 | 660 |
| 6 | 50 | 270 | 150 | 810 |
| 7 | 50 | 320 | 150 | 960 |
| 8 | 50 | 370 | 150 | 1110 |
| 9 | 50 | 420 | 150 | 1260 |
| 10 | 30 | 450 | 150 | 1410 |
| 11 | 30 | 480 | 150 | 1560 |
| 12 | 30 | 510 | 150 | 1710 |
| 13 | 30 | 540 | 150 | 1860 |
| 14 | 30 | 570 | 150 | 2010 |
| 15 | 30 | 600 | 150 | 2160 |
| 16 | 30 | 630 | 150 | 2310 |
| 17 | 30 | 720 | 150 | 2460 |

表格 5

最大利润

5.2.3 结果分析

和时的最大利润之差为元，因为每次航行成本小于10620元，故选择作为最优解。根据表格5绘出轮船航行图如下：

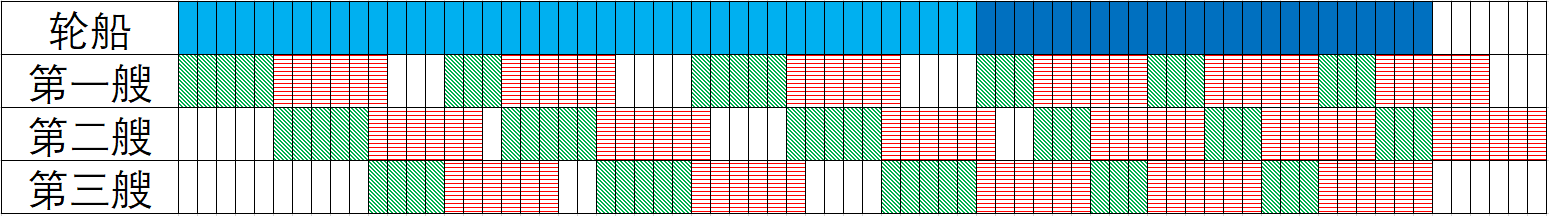


图 2

图中每一格表示10分钟，绿色斜条纹表示等待时间，红色横条纹表示航行时间，空白表示后一艘船到达码头时前一艘还未启航。由图可以看出，在只有3艘船时已经能够使得航程最优，同时为了降低运营成本，不考虑拥有更多游轮。航行时刻表如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 班次 | 启航时刻 | 回程时刻 | 载客量（人） |
| 1 | 10:50 | 11:50 | 150 |
| 2 | 11:40 | 12:40 | 150 |
| 3 | 12:20 | 13:20 | 120 |
| 4 | 12:50 | 13:50 | 90 |
| 5 | 13:40 | 14:40 | 150 |
| 6 | 14:30 | 15:30 | 150 |
| 7 | 15:20 | 16:20 | 150 |
| 8 | 16:10 | 17:10 | 150 |
| 9 | 17:00 | 18:00 | 150 |
| 10 | 17:30 | 18:30 | 150 |
| 11 | 18:00 | 19:00 | 150 |
| 12 | 18:30 | 19:30 | 150 |
| 13 | 19:00 | 20:00 | 150 |
| 14 | 19:30 | 20:30 | 150 |
| 15 | 20:00 | 21:00 | 150 |
| 16 | 20:30 | 21:30 | 150 |
| 17 | 21:00 | 22:00 | 150 |

表格 6

5.3 问题三

5.3.1 问题三的分析

问题三要求在游客愿意花最多10分钟等待游轮返回的情况下，重新给出只有1艘游轮和有多艘游轮时的最优航程安排。考虑9:50就有乘客到达码头开始等待，针对只有1艘游轮的情况，每次游轮到达码头后的等待时间分别减少10分钟，即17点前只需额外等20分钟、30分钟或40分钟，17点后只需额外等10分钟或20分钟，建立整数规划模型求出最优解；针对多艘游轮，类比问题二，建立动态规划模型，并分别求出游轮数和两种情况下的最佳航行时间表，并根据排班情况推出最少需要的船数。

5.3.2 模型的建立与求解

情况一：只有一艘游轮

令 ；分别表示对应等待时间为出现的次数，N为总载客量，M为航行总次数。航程最优要求运营收入最大、运营次数尽量少且每次运载人数较为均衡，故建立目标函数如下：

其中，

因白天的最后一次出发时间应在16：20之后，17点之前，故白天的总时间应大于分钟且小于；同时考虑全天的最后一班游轮在21点前发出，故全天总时间应小于分钟。因此建立约束条件如下：

编程枚举出符合整数规划的所有解（附录1 表格1），利润最大情况如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 |  |  |  |  |  | 航行次数 | 总载客量 | 总利润 |
| 1 | 0 | 1 | 4 | 0 | 3 | 8 | 1170 | 138060-8 |

表格 7

情况二：有多艘游轮

在问题二的基础上，考虑乘客愿意等待的情况。若后一艘游轮到达码头时前一艘还未启航，则有效等待时间不会改变；若码头没有游轮，则后一艘游轮到达时会因乘客愿意等待游轮返航而使得有效等待时间减少10分钟。

递推关系式为：

阶段M的允许决策集合为：

其中，

第M次返航后的总时间为：

第M次返航后的总载客量为：

游客达到码头的速率

最优指标表示第次出航到最后一次出航总载客量的最大值。根据最优化原理可以得到状态转移方程：

通过编程分别求出游轮数和两种情况下的最优解。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 出航次数 | 有效等待时间 | 等效航行时间 | 总时间 | 单次载客量 | 总载客量 |
| 1 | 50 | 0 | 50 | 150 | 150 |
| 2 | 40 | 20 | 110 | 150 | 300 |
| 3 | 40 | 10 | 160 | 150 | 450 |
| 4 | 40 | 10 | 210 | 150 | 600 |
| 5 | 40 | 10 | 260 | 150 | 750 |
| 6 | 30 | 10 | 300 | 120 | 870 |
| 7 | 40 | 10 | 350 | 150 | 1020 |
| 8 | 50 | 0 | 400 | 150 | 1170 |
| 9 | 30 | 10 | 440 | 120 | 1290 |
| 10 | 30 | 0 | 470 | 150 | 1440 |
| 11 | 20 | 30 | 520 | 150 | 1590 |
| 12 | 30 | 0 | 550 | 150 | 1740 |
| 13 | 20 | 20 | 590 | 150 | 1890 |
| 14 | 30 | 0 | 620 | 150 | 2040 |
| 15 | 20 | 40 | 730 | 150 | 2190 |

表格 8 时的最优解

最大利润

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 出航次数 | 有效等待时间 | 等效航行时间 | 总时间 | 单次载客量 | 总载客量 |
| 1 | 50 | 0 | 50 | 150 | 150 |
| 2 | 50 | 0 | 100 | 150 | 300 |
| 3 | 50 | 0 | 150 | 150 | 450 |
| 4 | 50 | 0 | 200 | 150 | 600 |
| 5 | 30 | 0 | 230 | 90 | 690 |
| 6 | 50 | 0 | 280 | 150 | 840 |
| 7 | 50 | 0 | 330 | 150 | 990 |
| 8 | 50 | 0 | 380 | 150 | 1140 |
| 9 | 50 | 0 | 430 | 150 | 1290 |
| 10 | 30 | 0 | 460 | 150 | 1440 |
| 11 | 30 | 0 | 490 | 150 | 1590 |
| 12 | 30 | 0 | 520 | 150 | 1740 |
| 13 | 30 | 0 | 550 | 150 | 1890 |
| 14 | 30 | 0 | 580 | 150 | 2040 |
| 15 | 30 | 0 | 610 | 150 | 2190 |
| 16 | 30 | 0 | 640 | 150 | 2340 |
| 17 | 30 | 0 | 730 | 150 | 2490 |

表格 9 时的最优解

最大利润

5.3.3 结果分析

在只有一艘游轮时，为了使得每次运载人数较为均衡，选择表格7中编号1的方案。考虑到现实中游客在中午等待较长时间满意度会低于其他时刻，因此给出如下航行时刻表：

|  |  |  |  |
| --- | --- | --- | --- |
| 班次 | 启航时刻 |  | 载客量 |
| 1 | 10:30 | 11:30 | 150 |
| 2 | 12:10 | 13:10 | 150 |
| 3 | 13:50 | 14:50 | 150 |
| 4 | 15:30 | 16:30 | 150 |
| 5 | 17:00 | 18:00 | 120 |
| 6 | 18:20 | 19:20 | 150 |
| 7 | 19:40 | 20:40 | 150 |
| 8 | 21:00 | 22:00 | 150 |

表格 10

和时的最大利润之差为元，因为每次航行成本小于10620元，故选择作为最优解。根据表格9绘出轮船航行图如下：

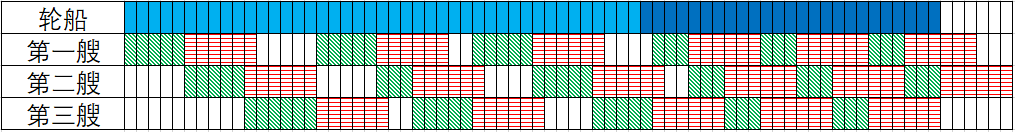


图 3

图中每一格表示10分钟，绿色斜条纹表示等待时间，红色横条纹表示航行时间，空白表示后一艘船到达码头时前一艘还未启航。由图可以看出，在只有3艘船时已经能够使得航程最优，同时为了降低运营成本，不考虑拥有更多游轮。航行时刻表如下所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 班次 | 启航时刻 | 回程时刻 | 载客量 |
| 1 | 10:40 | 11:40 | 150 |
| 2 | 11:30 | 12:30 | 150 |
| 3 | 12:20 | 13:20 | 150 |
| 4 | 13:10 | 14:10 | 150 |
| 5 | 13:40 | 14:40 | 90 |
| 6 | 14:30 | 15:30 | 150 |
| 7 | 15:20 | 16:20 | 150 |
| 8 | 16:10 | 17:10 | 150 |
| 9 | 17:00 | 18:00 | 150 |
| 10 | 17:30 | 18:30 | 150 |
| 11 | 18:00 | 19:00 | 150 |
| 12 | 18:30 | 19:30 | 150 |
| 13 | 19:00 | 20:00 | 150 |
| 14 | 19:30 | 20:30 | 150 |
| 15 | 20:00 | 21:00 | 150 |
| 16 | 20:30 | 21:30 | 150 |
| 17 | 21:00 | 22:00 | 150 |

表格 11

5.4 问题四

5.4.1 问题四的分析

问题四要求建立一条“两江游”新航线，进行新航线的码头设置和航程安排，分析新航线的发展态势并评估其对原来航线的影响。首先通过查阅地图和分析周边景点密集程度的方式设定了沿线包括朝天门码头在内的五个码头，其次依据旅游辐射场强模型[4] [5]求出每个码头周边景点对该码头的辐射总场强，并将其作为安排新航线航程的重要依据。然后根据2014年至2018年的“两江游”资料数据建立灰色G(1,1)模型，预测未来“两江游”的发展态势。最后结合2018年各月的“两江游”数据、辐射总场强以及航线未来发展情况给出新航线的航程安排，并评估分析其对原来“两江游”的影响。

5.4.2 新航线码头设置



图 4

通过搜索沿线各已有码头，分析其周边景点、商圈的密集程度及影响力，最终选定了包括朝天门码头在内的五个码头和五条环游线路（表格12）。

|  |  |  |
| --- | --- | --- |
| 编号 | 码头 | 路线 |
| 1 | 磁器口码头 | 磁器口码头→磁器口古镇→洪崖洞→朝天门→两江汇流→巴渝吊脚楼→长江大桥→重庆珊瑚公园→慈云寺  →重庆科技馆→重庆大剧院→磁器口码头 |
| 2 | 洪崖洞旅游客运码头 | 洪崖洞旅游客运码头→洪崖洞→朝天门→两江汇流→巴渝吊脚楼→长江大桥→重庆珊瑚公园→慈云寺→重庆科技馆→重庆大剧院→磁器口古镇→洪崖洞旅游客运码头 |
| 3 | 朝天门码头 | 朝天门码头→两江汇流→巴渝吊脚楼→长江大桥→  重庆珊瑚公园→慈云寺→重庆科技馆→重庆大剧院→  磁器口古镇→洪崖洞→朝天门→朝天门码头 |
| 4 | 海棠溪码头 | 海棠溪码头→慈云寺→重庆科技馆→大剧院→  磁器口古镇→洪崖洞→朝天门→两江汇流→巴渝吊脚楼→长江大桥→重庆珊瑚公园→海棠溪码头 |
| 5 | 相国寺码头 | 相国寺码头→磁器口古镇→洪崖洞→朝天门→两江汇流→巴渝吊脚楼→长江大桥→重庆珊瑚公园→慈云寺→  重庆科技馆→大剧院→相国寺码头 |

表格 12

为了得到周边景点对码头辐射的情况，以码头为受力点，周边景点为辐射源，常见交通方式为辐射媒介，建立旅游辐射场强模型如下：

其中，表示辐射源（景点），表示受力点（码头），表示景点的旅游人数，表示景点和码头之间的最短时间距离，为一常系数，表示辐射源对受力点的辐射场强，表示受力点处的总场强，因影响大小可以直接叠加，故在此忽略场强的矢量性。通过查阅资料得到各辐射源到受力点之间的最短时间距离及各辐射点处的旅游人数（表格13），带入公式求得各景点在码头处的场强及各码头的合场强（表格14）：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 码头 | /  分钟 | 景点 | 2018国庆（万） | | 2018五一（万） | 2019五一（万） | 分场强 | 合场强 |
| 洪崖洞旅游客运码头 | 2 | 洪崖洞 | 79.67 | 14.2 | | 41.7 | 1.88C | 2.30C |
| 6 | 长江索道 | 13.74 | 4.6 | | 7.99 | 0.36C |
| 10 | 重庆科技馆 | 16.8 | / | | 9 | 0.06C |
| 海棠溪码头 | 5 | 长江索道 | 13.74 | 4.6 | | 7.99 | 0.43C | 0.72C |
| 23 | 南山风景区 | 233.3 | 38.32 | | 184.18 | 0.27C |
| 22 | 南山植物园 | / | / | | 4.18 | 0.02C |
| 磁器口码头 | 14 | 磁器口古镇 | 62.69 | 19.57 | | 24.81 | 0.32C | 0.83C |
| 8 | 歌乐山烈士陵园 | 65.88 | 12.74 | | 14.8079 | 0.45C |
| 17 | 歌乐山森林公园 | / | 1.02 | | 1.88 | 0.06C |
| 朝天门码头 | 13 | 解放碑 | 454.8 | 200 | | 420.6 | 1.09C | 1.99C |
| 5 | 洪崖洞 | 79.67 | 14.2 | | 41.7 | 0.75C |
| 19 | 长江索道  （新华路） | 13.74 | 4.6 | | 7.99 | 0.11C |
| 16 | 重庆科技馆 | 16.8 | / | | 9 | 0.04C |
| 相国寺码头 | 4 | 观音桥 | 44.38 | 21.3 | | 33.4 | 1.15C | 1.15C |

表格 13

5.4.3 发展态势分析

由“两江游”2014年至2018年的资料数据（表格14）可以看出，游客数量基本呈指数型上升，故建立灰色GM(1,1)模型对未来三年的旅游情况进行预测。通过累加的方式，在时间轴上建立连续变化的微分方程，然后求解方程参数，进一步求解方程根实现预测。[1] [2]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 年份 | 2014年 | 2015年 | 2016年 | 2017年 | 2018年 |
| 游客量（万） | 55 | 98 | 142 | 207 | 235 |

表格 14

首先建立原始序列，表示数据的个数，并生成原始序列的一次累加序列，其中

其次生成序列的邻近等权数列,其中

对构建关于年份t的一阶微分方程：

其中，α为发展系数，μ为灰色作用量，需要在方程中求解。做矩阵、，

用最小二乘法求解灰色参数,表示的转置矩阵，表示的逆矩阵。

将求得的灰色参数带入方程中，解得方程的根为

将方程的解通过累减的方式逆向求出预测值，

用MATLAB做出预测值曲线见图5。

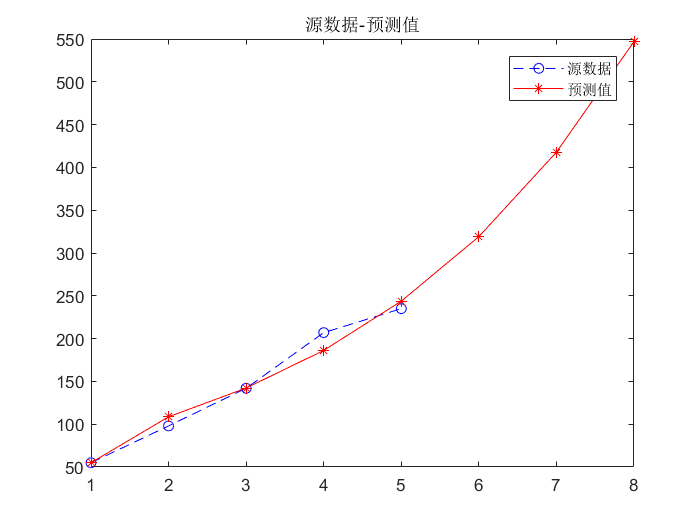


图 5

对模型进行后验差检验，设残差为，均值为，方差为,残差均值为，残差方差为，后残差比值为，小误差概率为，则：

其中，

计算求得=1,=0.0280。通过查找相关资料，得到模型的预测精度评定方法：

|  |  |  |
| --- | --- | --- |
| P | C | 预测精度等级 |
| >0.95 | <0.35 | 好 |
| >0.80 | <0.45 | 合格 |
| >0.70 | <0.50 | 勉强合格 |
| <=0.70 | >=0.65 | 不合格 |

表格 15

对照表格15中的模型预测精度等级，可以得出使用灰色GM(1,1)模型进行预测精度较高的结论，进而将未来五年的年份t带入方程中进行求解，得到未来三年的“两江游”航线游客数量（表格16）。

|  |  |  |  |
| --- | --- | --- | --- |
| 年份 | 2019年 | 2020年 | 2021年 |
| 游客量（万） | 319.13 | 417.79 | 546.94 |

表格 16

近年来，我国旅游业发展迅速，各景点游客数呈指数型增长。由图 5和表格16可知，目前重庆市“两江游”每年接待游客数已突破230万，新航线开行后，该数量将在三年内突破500万。我国是世界上旅游资源最丰富的国家，而重庆市又是我国旅游业最发达的城市之一，“两江游”项目作为重庆市旅游特色，其新航线的设立必然会使得该项目在未来拥有更多的客流量，进而获得更大的旅游收益，从而推动重庆市旅游业更快速稳健地发展。[3]

5.4.4 航行安排

新航线总长51.3千米，航行总时长约2.5小时，票价为235元。因新航线较原航线距离长，游览时间久且票价高，结合其码头分散的优点，做出如下假设：（1）设置新航线不影响参与“两江游”的乘客数量的变化趋势（2）游客量的增加比例与发班量的增加比例相同（3）该比例不随月份变化而变化。则表格16中的预测数据也可以表示新航线在未来三年内的总客流量，即2019年新航线将总计载客319.13万人，2018年各月的总载客量与总发班次数[7]及2019年的预测值如表格17所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 月份（2018） | 总发班次数 | 总载客量 | 月份(2019) | 总发班次数 | 总载客量 |
| 1 | 565 | 81000 | 1 | 767 | 109998 |
| 2 | 791 | 279000 | 2 | 1074. | 378882 |
| 3 | 650 | 172000 | 3 | 883 | 233576 |
| 4 | 701 | 217000 | 4 | 952 | 294686 |
| 5 | 670 | 217000 | 5 | 910 | 294686 |
| 6 | 672 | 194103 | 6 | 913 | 263592 |
| 7 | 458 | 148445 | 7 | 622 | 201588 |
| 8 | 670 | 285078 | 8 | 910 | 387136 |
| 9 | 565 | 199655 | 9 | 767 | 271132 |
| 10 | 755 | 288851 | 10 | 1025 | 392260 |
| 11 | 585 | 165282 | 11 | 794 | 224453 |
| 12 | 543 | 102586 | 12 | 737 | 139312 |
| 总计 | 7625 | 2350000 | 总计 | 10354 | 3191300 |

表格 17

假设新航线的五个码头之间客流量与航班数的比值即为各码头处合场强的比值，令朝天门码头为1.00，则场强的比值为。由此可以估算出未来一年内各码头每月的总载客量，并由此给出航程安排（表格18）。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 码头 | 月份 | 总发班次数 | 每天发班次数 | 码头 | 月份 | 总发班次数 | 每天发班次数 |
| 磁  器  口  码  头 | 1 | 253 | 8 | 洪  崖  洞  旅  游  客  运  码  头 | 1 | 79 | 3 |
| 2 | 355 | 13 | 2 | 110 | 4 |
| 3 | 292 | 9 | 3 | 91 | 3 |
| 4 | 315 | 10 | 4 | 98 | 3 |
| 5 | 301 | 10 | 5 | 93 | 3 |
| 6 | 302 | 10 | 6 | 94 | 3 |
| 7 | 206 | 7 | 7 | 64 | 2 |
| 8 | 301 | 10 | 8 | 93 | 3 |
| 9 | 253 | 8 | 9 | 79 | 3 |
| 10 | 339 | 11 | 10 | 105 | 3 |
| 11 | 262 | 9 | 11 | 81 | 3 |
| 12 | 244 | 8 | 12 | 76 | 2 |
| 总计 | 3421.835 | 9 | 总计 | 1061.949 | 3191300 |
| 朝  天  门  码  头 | 1 | 90 | 3 | 海  棠  溪  码  头 | 1 | 219 | 7 |
| 2 | 125 | 4 | 2 | 306 | 11 |
| 3 | 103 | 3 | 3 | 252 | 8 |
| 4 | 111 | 4 | 4 | 271 | 9 |
| 5 | 106 | 3 | 5 | 259 | 8 |
| 6 | 107 | 4 | 6 | 260 | 9 |
| 7 | 73 | 2 | 7 | 177 | 6 |
| 8 | 106 | 3 | 8 | 259 | 8 |
| 9 | 90 | 3 | 9 | 219 | 7 |
| 10 | 120 | 4 | 10 | 292 | 9 |
| 11 | 93 | 3 | 11 | 226 | 8 |
| 12 | 86 | 3 | 12 | 210 | 7 |
| 总计 | 1209 | 3 | 总计 | 2950 | 8 |
| 相  国  寺  码  头 | 1 | 127 | 4 | 相  国  寺  码  头 | 7 | 103 | 3 |
| 2 | 177 | 6 | 8 | 150 | 5 |
| 3 | 146 | 5 | 9 | 127 | 4 |
| 4 | 157 | 5 | 10 | 169 | 5 |
| 5 | 150 | 5 | 11 | 131 | 4 |
| 6 | 151 | 5 | 12 | 122 | 4 |
|  | 总计 | / |  |  | 总计 | 1711 | 5 |

表格 18

5.4.5 对原航线影响的评估分析

正面影响：

新航线包含了原来“两江游”航线沿途的所有景点，缓解了朝天门码头的客运压力。乘坐原航线的游客现可以就近选择靠近上一个游玩景点的码头欣赏更多的两江美景，故客流较原先更分散，这也提高了游客对原航线的满意度。同时，设立新航线必然会提升“两江游”服务的知名度，大批游客的参与将为原有航线带来新的生命力。

负面影响：

就朝天门码头而言，客流量的减少必然会使得经济收益减少，这不利于原有航线与码头的维护和整改。同时，游客量的分散使朝天门码头每天的发班量减少，即航班间的等候间隔变大，而过长的等待时间会降低游客的满意度，进而对原航线产生负面影响。

6 模型评价

6.1 模型优点

（1）问题一中运用整数规划模型得到航程安排和总载客量。该模型简单明了地求得规划范围内所有整数取值情况下的最优解，可以广泛应用与于实际问题中的生产、配置等问题。

（2）问题二、问题三中运用动态规划模型得到最佳航程安排。该模型将复杂的多阶段问题转化为若干单阶段子问题，在每阶段允许的决策范围内求出最优决策方案，从而使得整体的决策方案最优。

（3）问题四中运用灰色GM(1,1)模型对未来三年的“两江游”总客流量进行了预测。该模型适用于既含有已知信息又含有未知信息的灰色系统，通过弱化未知因素的影响，强化已知因素的方式，建立起在时间轴上连续变化的微分方程，在数据呈现指数变化趋势时具有很高的精度。

6.2 模型缺点

（1）通过枚举的方式求解整数规划模型在变量维数较高或规划范围较大时不具有可行性。

（2）灰色GM(1,1)模型主要反映了预测值的指数型增长或减少，在处理变化趋势不明显的数据时因缺少了线性因素而可能导致测量结果的不准确。

（3）动态规划模型在决策阶段数较多时难以快速求解出最优解。

6.3 模型改进

（1）求解高维整数规划问题时可以采用更具有可行性的舍入凑整法或分支定界法。

（2）由于灰色G(1,1)模型缺少了线性因素，因此考虑构建灰色线性回归组合模型，将灰色G(1,1)模型与线性回归模型进行有机组合，便可以同时弥补线性回归模型难以预测指数增长趋势的样本以及灰色G(1,1)模型缺少线性因素的不足。

1. 模型推广

（1）整数规划模型适用于决策变量为整数的情况，可以广泛应用于实际生活中的安排生产计划、存货布置、组件配套问题。

（2）动态规划模型将复杂的多阶段问题转化为若干单阶段子问题，降低了问题的复杂度，可以管饭应用于投资分配问题、背包问题、排序问题等。

（3）灰色线性回归组合模型可以对数据更为精准地进行预测，可用于车站客流量预测、GDP增长情况估计等各方面。

8 参考文献

[1]廖飞,赵宝江.基于灰色预测模型的黑龙江省入境旅游业发展趋势分析.时代金融，2015，4：270-272.

[2]廖飞.论灰色经济模型的构建[J].华章，2011(31):71.

[3]2018年中国旅游行业发展现状及发展趋势分析，

http://www.chyxx.com/industry/201804/634265.html，

2018-4-25。

[4]王利鑫,张元标.上海世博会对周边城市旅游辐射效应研究.地理与地理信息科学，2011，27（3）：105-108.

[5]王永明,马耀峰.上海入境旅游流对长江流域各省区空间场效应研究[J]. 经济地理,2010,30( 5):854-858.

[6]沈世云、杨春德，数学建模理论与方法，北京：清华大学出版社，2016.

[7] 数据来源：http://www.cse.net.cn

附录1

表格1 问题三情况一的所有情况

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 |  |  |  |  |  | 航行次数 | 总载客量 | 总利润 |
| 1 | 0 | 1 | 4 | 0 | 2 | 7 | 1020 | 120360-7A |
| 2 | 0 | 1 | 4 | 0 | 3 | 8 | 1170 | 138060-8A |
| 3 | 0 | 1 | 4 | 1 | 1 | 7 | 970 | 114460-7A |
| 4 | 0 | 1 | 4 | 1 | 2 | 8 | 1120 | 132160-8A |
| 5 | 0 | 1 | 4 | 2 | 0 | 7 | 920 | 108560-7A |
| 6 | 0 | 1 | 4 | 2 | 1 | 8 | 1070 | 126260-8A |
| 7 | 0 | 1 | 4 | 3 | 0 | 8 | 1020 | 120360-8A |
| 8 | 0 | 2 | 3 | 0 | 2 | 7 | 990 | 116820-7A |
| 9 | 0 | 2 | 3 | 0 | 3 | 8 | 1140 | 134520-8A |
| 10 | 0 | 2 | 3 | 1 | 1 | 7 | 940 | 110920-7A |
| 11 | 0 | 2 | 3 | 1 | 2 | 8 | 1090 | 128620-8A |
| 12 | 0 | 2 | 3 | 2 | 0 | 7 | 890 | 105020-7A |
| 13 | 0 | 2 | 3 | 2 | 1 | 8 | 1040 | 122720-8A |
| 14 | 0 | 2 | 3 | 3 | 0 | 8 | 990 | 116820-8A |
| 15 | 0 | 3 | 2 | 0 | 2 | 7 | 960 | 113280-7A |
| 16 | 0 | 3 | 2 | 0 | 3 | 8 | 1110 | 130980-8A |
| 17 | 0 | 3 | 2 | 1 | 1 | 7 | 910 | 107380-7A |
| 18 | 0 | 3 | 2 | 1 | 2 | 8 | 1060 | 125080-8A |
| 19 | 0 | 3 | 2 | 2 | 1 | 8 | 1010 | 119180-8A |
| 20 | 0 | 3 | 2 | 3 | 0 | 8 | 960 | 113280-8A |
| 21 | 0 | 4 | 1 | 0 | 2 | 7 | 930 | 109740-7A |
| 22 | 0 | 4 | 1 | 0 | 3 | 8 | 1080 | 127440-8A |
| 23 | 0 | 4 | 1 | 1 | 2 | 8 | 1030 | 121540-8A |
| 24 | 0 | 4 | 1 | 2 | 1 | 8 | 980 | 115640-8A |
| 25 | 0 | 4 | 1 | 3 | 0 | 8 | 930 | 109740-8A |
| 26 | 0 | 5 | 0 | 0 | 3 | 8 | 1050 | 123900-8A |
| 27 | 0 | 5 | 0 | 1 | 2 | 8 | 1000 | 118000-8A |
| 28 | 0 | 5 | 0 | 2 | 1 | 8 | 950 | 112100-8A |
| 29 | 0 | 5 | 0 | 3 | 0 | 8 | 900 | 106200-8A |
| 30 | 0 | 5 | 0 | 4 | 0 | 9 | 1000 | 118000-9A |
| 31 | 1 | 0 | 4 | 0 | 2 | 7 | 990 | 116820-7A |
| 32 | 1 | 0 | 4 | 0 | 3 | 8 | 1140 | 134520-8A |
| 33 | 1 | 0 | 4 | 1 | 1 | 7 | 940 | 110920-7A |
| 34 | 1 | 0 | 4 | 1 | 2 | 8 | 1090 | 128620-8A |
| 35 | 1 | 0 | 4 | 2 | 0 | 7 | 890 | 105020-7A |
| 36 | 1 | 0 | 4 | 2 | 1 | 8 | 1040 | 122720-8A |
| 37 | 1 | 0 | 4 | 3 | 0 | 8 | 990 | 116820-8A |
| 38 | 1 | 1 | 3 | 0 | 2 | 7 | 960 | 113280-7A |
| 39 | 1 | 1 | 3 | 0 | 3 | 8 | 1110 | 130980-8A |
| 40 | 1 | 1 | 3 | 1 | 1 | 7 | 910 | 107380-7A |
| 41 | 1 | 1 | 3 | 1 | 2 | 8 | 1060 | 125080-8A |
| 42 | 1 | 1 | 3 | 2 | 1 | 8 | 1010 | 119180-8A |
| 43 | 1 | 1 | 3 | 3 | 0 | 8 | 960 | 113280-8A |
| 44 | 1 | 2 | 2 | 0 | 2 | 7 | 930 | 109740-7A |
| 45 | 1 | 2 | 2 | 0 | 3 | 8 | 1080 | 127440-8A |
| 46 | 1 | 2 | 2 | 1 | 2 | 8 | 1030 | 121540-8A |
| 47 | 1 | 2 | 2 | 2 | 1 | 8 | 980 | 115640-8A |
| 48 | 1 | 2 | 2 | 3 | 0 | 8 | 930 | 109740-8A |
| 49 | 1 | 3 | 1 | 0 | 3 | 8 | 1050 | 123900-8A |
| 50 | 1 | 3 | 1 | 1 | 2 | 8 | 1000 | 118000-8A |
| 51 | 1 | 3 | 1 | 2 | 1 | 8 | 950 | 112100-8A |
| 52 | 1 | 3 | 1 | 3 | 0 | 8 | 900 | 106200-8A |
| 53 | 1 | 3 | 1 | 4 | 0 | 9 | 1000 | 118000-9A |
| 54 | 1 | 4 | 0 | 0 | 3 | 8 | 1020 | 120360-8A |
| 55 | 1 | 4 | 0 | 1 | 2 | 8 | 970 | 114460-8A |
| 56 | 1 | 4 | 0 | 2 | 1 | 8 | 920 | 108560-8A |
| 57 | 1 | 4 | 0 | 3 | 0 | 8 | 870 | 102660-8A |
| 58 | 1 | 4 | 0 | 3 | 1 | 9 | 1020 | 120360-9A |
| 59 | 1 | 4 | 0 | 4 | 0 | 9 | 970 | 114460-9A |
| 60 | 2 | 0 | 3 | 0 | 2 | 7 | 930 | 109740-7A |
| 61 | 2 | 0 | 3 | 0 | 3 | 8 | 1080 | 127440-8A |
| 62 | 2 | 0 | 3 | 1 | 2 | 8 | 1030 | 121540-8A |
| 63 | 2 | 0 | 3 | 2 | 1 | 8 | 980 | 115640-8A |
| 64 | 2 | 0 | 3 | 3 | 0 | 8 | 930 | 109740-8A |
| 65 | 2 | 1 | 2 | 0 | 3 | 8 | 1050 | 123900-8A |
| 66 | 2 | 1 | 2 | 1 | 2 | 8 | 1000 | 118000-8A |
| 67 | 2 | 1 | 2 | 2 | 1 | 8 | 950 | 112100-8A |
| 68 | 2 | 1 | 2 | 3 | 0 | 8 | 900 | 106200-8A |
| 69 | 2 | 1 | 2 | 4 | 0 | 9 | 1000 | 118000-9A |
| 70 | 2 | 2 | 1 | 0 | 3 | 8 | 1020 | 120360-8A |
| 71 | 2 | 2 | 1 | 1 | 2 | 8 | 970 | 114460-8A |
| 72 | 2 | 2 | 1 | 2 | 1 | 8 | 920 | 108560-8A |
| 73 | 2 | 2 | 1 | 3 | 0 | 8 | 870 | 102660-8A |
| 74 | 2 | 2 | 1 | 3 | 1 | 9 | 1020 | 120360-9A |
| 75 | 2 | 2 | 1 | 4 | 0 | 9 | 970 | 114460-9A |
| 76 | 2 | 3 | 0 | 0 | 3 | 8 | 990 | 116820-8A |
| 77 | 2 | 3 | 0 | 1 | 2 | 8 | 940 | 110920-8A |
| 78 | 2 | 3 | 0 | 2 | 1 | 8 | 890 | 105020-8A |
| 79 | 2 | 3 | 0 | 2 | 2 | 9 | 1040 | 122720-9A |
| 80 | 2 | 3 | 0 | 3 | 0 | 8 | 840 | 99120-8A |
| 81 | 2 | 3 | 0 | 3 | 1 | 9 | 990 | 116820-9A |
| 82 | 2 | 3 | 0 | 4 | 0 | 9 | 940 | 110920-9A |
| 83 | 3 | 0 | 2 | 0 | 3 | 8 | 1020 | 120360-8A |
| 84 | 3 | 0 | 2 | 1 | 2 | 8 | 970 | 114460-8A |
| 85 | 3 | 0 | 2 | 2 | 1 | 8 | 920 | 108560-8A |
| 86 | 3 | 0 | 2 | 3 | 0 | 8 | 870 | 102660-8A |
| 87 | 3 | 0 | 2 | 3 | 1 | 9 | 1020 | 120360-9A |
| 88 | 3 | 0 | 2 | 4 | 0 | 9 | 970 | 114460-9A |
| 89 | 3 | 1 | 1 | 0 | 3 | 8 | 990 | 116820-8A |
| 90 | 3 | 1 | 1 | 1 | 2 | 8 | 940 | 110920-8A |
| 91 | 3 | 1 | 1 | 2 | 1 | 8 | 890 | 105020-8A |
| 92 | 3 | 1 | 1 | 2 | 2 | 9 | 1040 | 122720-9A |
| 93 | 3 | 1 | 1 | 3 | 0 | 8 | 840 | 99120-8A |
| 94 | 3 | 1 | 1 | 3 | 1 | 9 | 990 | 116820-9A |
| 95 | 3 | 1 | 1 | 4 | 0 | 9 | 940 | 110920-9A |
| 96 | 3 | 2 | 0 | 0 | 2 | 8 | 890 | 105020-8A |
| 97 | 3 | 2 | 0 | 0 | 3 | 9 | 1040 | 122720-9A |
| 98 | 3 | 2 | 0 | 1 | 1 | 8 | 840 | 99120-8A |
| 99 | 3 | 2 | 0 | 1 | 2 | 9 | 990 | 116820-9A |
| 100 | 3 | 2 | 0 | 2 | 0 | 8 | 790 | 93220-8A |
| 101 | 3 | 2 | 0 | 2 | 1 | 9 | 940 | 110920-9A |
| 102 | 3 | 2 | 0 | 3 | 0 | 9 | 890 | 105020-9A |
| 103 | 4 | 0 | 1 | 0 | 2 | 8 | 890 | 105020-8A |
| 104 | 4 | 0 | 1 | 0 | 3 | 9 | 1040 | 122720-9A |
| 105 | 4 | 0 | 1 | 1 | 1 | 8 | 840 | 99120-8A |
| 106 | 4 | 0 | 1 | 1 | 2 | 9 | 990 | 116820-9A |
| 107 | 4 | 0 | 1 | 2 | 0 | 8 | 790 | 93220-8A |
| 108 | 4 | 0 | 1 | 2 | 1 | 9 | 940 | 110920-9A |
| 109 | 4 | 0 | 1 | 3 | 0 | 9 | 890 | 105020-9A |
| 110 | 5 | 1 | 0 | 0 | 2 | 8 | 870 | 102660-8A |
| 111 | 5 | 1 | 0 | 0 | 3 | 9 | 1020 | 120360-9A |
| 112 | 5 | 1 | 0 | 1 | 1 | 8 | 820 | 96760-8A |
| 113 | 5 | 1 | 0 | 1 | 2 | 9 | 970 | 114460-9A |
| 114 | 5 | 1 | 0 | 2 | 0 | 8 | 770 | 90860-8A |
| 115 | 5 | 1 | 0 | 2 | 1 | 9 | 920 | 108560-9A |
| 116 | 5 | 1 | 0 | 3 | 0 | 9 | 870 | 102660-9A |
| 117 | 6 | 0 | 0 | 0 | 2 | 8 | 840 | 99120-8A |
| 118 | 6 | 0 | 0 | 0 | 3 | 9 | 990 | 116820-9A |
| 119 | 6 | 0 | 0 | 1 | 1 | 8 | 790 | 93220-8A |
| 120 | 6 | 0 | 0 | 1 | 2 | 9 | 940 | 110920-9A |
| 121 | 6 | 0 | 0 | 2 | 0 | 8 | 740 | 87320-8A |
| 122 | 6 | 0 | 0 | 2 | 1 | 9 | 890 | 105020-9A |
| 123 | 6 | 0 | 0 | 3 | 0 | 9 | 840 | 99120-9A |

附录2

第一题 源代码

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x=0;

int i,j,k,a,b,z,m1,m2,t1,t2,c,d;

printf("%3c%3c%3c%3c%3c%3c%5c\n",'i','j','k','a','b','m','z');

for(i=0; i<=5; i++)

{

for(j=0; j<=4; j++)

{

m1=i+j;

if(i\*30+j\*40+m1\*60>480) break;

for(k=0; k<=4; k++)

{

m1=i+j+k;

if ((i\*30+j\*40+k\*50+m1\*60<=480)&&(i\*30+j\*40+k\*50+m1\*60>=370)) //保证白天最后一次航班在17:00前发出，16:10后返回

{

if(i\*30+j\*40+k\*50+m1\*60>=420)//假设白天航班返回时间在五点之后

{

for(a=0; a<=3; a++)

{

for(b=0; b<=3; b++)

{

m2=a+b;

if(i\*30+j\*40+k\*50+a\*20+b\*30+m1\*60+m2\*60>720) break;//最后一班船在21:00前发出

if(i\*30+j\*40+k\*50+a\*20+b\*30+m1\*60+m2\*60>600)

{

z=i\*30\*3+j\*40\*3+k\*50\*3+a\*20\*5+b\*30\*5;

x++;

printf("%3d%3d%3d%3d%3d%3d%5d%3d\n",i,j,k,a,b,m1+m2,z,x);

}

}

}

}

else

{

t1=420-(i\*30+j\*40+k\*50+m1\*60);

c=(150-t1\*3)/50;

d=(90-t1\*3)/50;

for(t2=1; t2<=d; t2++)

{

if(t1\*3+t2\*50>150) break;

for(a=0; a<=3; a++)

{

for(b=0; b<=3; b++)

{

m2=a+b+1;

if(420+a\*20+b\*30+m2\*60+t2\*10>720) break;//最后一班船在21:00前发出

if(420+a\*20+b\*30+m2\*60+t2\*10>600)

{

z=i\*30\*3+j\*40\*3+k\*50\*3+t1\*3+t2\*50+a\*20\*5+b\*30\*5;

x++;

printf("%3d%3d%3d%3d%3d%3d%5d%3d",i,j,k,a,b,m1+m2,z,x);

printf(" abc\n");

}

}

}

}

}

}

}

}

}

return 0;

}

第二题两艘船 源代码（c++）

/\*2-2 编译选项 g++ -std=c++14 -c \*/

#include <bits/stdc++.h>

using namespace std;

const int inf = 0x3f3f3f3f;

const int dayEnd = 420;

const int nightEnd = 660;

const int options[] = {20, 30, 40, 50};

const int maxCapacity = 150;

const int minCapacity = 90;

const int day = 3;

const int night = 5;

struct result

{

int people, times;

result(int p, int t): people(p), times(t) {}

result(): people(), times() {}

friend bool operator < (result r1, result r2)

{

if (r1.people == r2.people)

{

return r1.times > r2.times;

}

else

{

return r1.people < r2.people;

}

}

friend bool operator == (result r1, result r2)

{

return r1.times == r2.times && r1.people == r2.people;

}

result operator +(result r)

{

result R = result(r.times + times, r.people + people);

return R;

}

};

struct ship

{

int t, x, waitClock;

ship(int t, int x, int waitClock): t(t), x(x), waitClock(waitClock) {}

ship(): t(0), x(0), waitClock(0) {}

friend bool operator<(ship s1, ship s2)

{

if (s1.waitClock != s2.waitClock)

{

return s1.waitClock < s2.waitClock;

}

else

{

if (s1.t != s2.t)

{

return s1.t < s2.t;

}

else

{

return s1.x < s2.x;

}

}

}

};

inline int countPeople(int &time, int start)

{

int ret = 0;

int End = time + start;

if (start >= dayEnd) // 晚上

{

if (End > nightEnd)

{

time = nightEnd - start;

}

ret = night \* time;

}

else if (start < dayEnd && End > dayEnd)// 分别一半

{

ret = day \* (dayEnd - start) + night \* (End - dayEnd);

}

else //白天

{

ret = day \* time;

}

if (ret >= minCapacity && ret <= maxCapacity)

{

return ret;

}

else

{

return -inf;

}

}

map<ship, result> mp;

result dfs(ship s)

{

//printf("t:%4d x:%4d waitClock:%4d\n", s.t, s.x, s.waitClock);

if (s.waitClock >= nightEnd)

{

return result(0, 0);

}

else

{

if (mp.count(s) > 0)

{

return mp[s];

}

else

{

vector<result> vec;

vec.push\_back(dfs(ship(20, max(0, 60 - s.x - 20), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(30, max(0, 60 - s.x - 30), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(40, max(0, 60 - s.x - 40), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(50, max(0, 60 - s.x - 50), s.waitClock + s.t + s.x)));

result best = \*max\_element(vec.begin(), vec.end());

mp[s] = result(best.people + countPeople(s.t, s.waitClock), best.times + 1);

return mp[s];

}

}

}

struct answer

{

int id, wait, spend, start, load, cntLoad;

answer(int id, int wait, int spend, int start, int load, int cntLoad): id(id), wait(wait), spend(spend), start(start), load(load), cntLoad(cntLoad){}

friend bool operator<(answer a1, answer a2)

{

return a1.id < a2.id;

}

friend bool operator==(answer a1, answer a2)

{

return a2.id == a1.id &&

a2.wait == a1.wait &&

a2.spend == a1.spend &&

a2.start == a1.start &&

a2.load == a1.load &&

a2.cntLoad == a1.cntLoad;

}

};

ostream& operator<<(ostream &os, const answer &a)

{

os << a.id << "\t" << a.wait << "\t" << a.spend << "\t" << a.start << "\t" << a.load << endl;

return os;

}

int main()

{

vector<pair<result, int> > ans;

ans.push\_back(make\_pair(dfs(ship(30, 0, 0)), 30));

ans.push\_back(make\_pair(dfs(ship(40, 0, 0)), 40));

ans.push\_back(make\_pair(dfs(ship(50, 0, 0)), 50));

sort(ans.begin(), ans.end());

int p = ans[2].first.people, t = ans[2].first.times;

cout << "班次\t有效等待时长\t等效航行时间\t开始等待时刻\t装载人数" << endl;

vector<answer> Answer;

for (auto i: mp)

{

ship f = i.first;

result s = i.second;

Answer.push\_back(answer(t - s.times + 1, f.t, f.x, f.waitClock, countPeople(f.t, f.waitClock), s.people));

}

sort(Answer.begin(), Answer.end());

int cntId = 1, cntLoad = p;

answer tmp = answer(1, ans[2].second, 0, 0, countPeople(ans[2].second, 0), ans[2].first.people);

int fst = 1;

for (vector<answer>:: iterator iter = Answer.begin(); iter != Answer.end(); iter++)

{

answer item = \*iter;

if (cntId == item.id && cntLoad == item.cntLoad && cntId <= t && item.start == tmp.spend + tmp.start + tmp.wait || (fst && item == tmp))

{

fst = 0;

tmp = item;

cout << item;

cntId++;

cntLoad -= item.load;

}

}

return 0;

}

第二题三艘船 源代码（c++）

/\*2-3 编译选项 g++ -std=c++14 -c \*/

#include <bits/stdc++.h>

using namespace std;

const int inf = 0x3f3f3f3f;

const int dayEnd = 420;

const int nightEnd = 660;

const int options[] = {20, 30, 40, 50};

const int maxCapacity = 150;

const int minCapacity = 90;

const int day = 3;

const int night = 5;

struct result

{

int people, times;

result(int p, int t): people(p), times(t) {}

result(): people(), times() {}

friend operator < (result r1, result r2)

{

if (r1.people == r2.people)

{

return r1.times > r2.times;

}

else

{

return r1.people < r2.people;

}

}

friend operator == (result r1, result r2)

{

return r1.times == r2.times && r1.people == r2.people;

}

result operator + (result r)

{

result R = result(r.times + times, r.people + people);

return R;

}

};

struct ship

{

int t, x, waitClock;

ship(int t, int x, int waitClock): t(t), x(x), waitClock(waitClock) {}

ship(): t(0), x(0), waitClock(0) {}

friend bool operator<(ship s1, ship s2)

{

if (s1.waitClock != s2.waitClock)

{

return s1.waitClock < s2.waitClock;

}

else

{

if (s1.t != s2.t)

{

return s1.t < s2.t;

}

else

{

return s1.x < s2.x;

}

}

}

};

inline int countPeople(int &time, int start)

{

int ret = 0;

int End = time + start;

if (start >= dayEnd) // 晚上

{

if (End > nightEnd)

{

time = nightEnd - start;

}

ret = night \* time;

}

else if (start < dayEnd && End > dayEnd)// 分别一半

{

ret = day \* (dayEnd - start) + night \* (End - dayEnd);

}

else //白天

{

ret = day \* time;

}

if (ret >= minCapacity && ret <= maxCapacity)

{

return ret;

}

else

{

return -inf;

}

}

map<ship, result> mp;

result dfs(ship s)

{

//printf("t:%4d x:%4d waitClock:%4d\n", s.t, s.x, s.waitClock);

if (s.waitClock >= nightEnd)

{

return result(0, 0);

}

else

{

if (mp.count(s) > 0)

{

return mp[s];

}

else

{

vector<result> vec;

vec.push\_back(dfs(ship(20, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(30, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(40, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(50, 0, s.waitClock + s.t + s.x)));

result best = \*max\_element(vec.begin(), vec.end());

mp[s] = result(best.people + countPeople(s.t, s.waitClock), best.times + 1);

return mp[s];

}

}

}

struct answer

{

int id, wait, spend, start, load, cntLoad;

answer(int id, int wait, int spend, int start, int load, int cntLoad): id(id), wait(wait), spend(spend), start(start), load(load), cntLoad(cntLoad){}

friend bool operator<(answer a1, answer a2)

{

return a1.id < a2.id;

}

friend bool operator==(answer a1, answer a2)

{

return a2.id == a1.id &&

a2.wait == a1.wait &&

a2.spend == a1.spend &&

a2.start == a1.start &&

a2.load == a1.load &&

a2.cntLoad == a1.cntLoad;

}

};

ostream& operator<<(ostream &os, const answer &a)

{

os << a.id << "\t" << a.wait << "\t" << a.spend << "\t" << a.start << "\t" << a.load << endl;

return os;

}

int main()

{

vector<pair<result, int> > ans;

ans.push\_back(make\_pair(dfs(ship(30, 0, 0)), 30));

ans.push\_back(make\_pair(dfs(ship(40, 0, 0)), 40));

ans.push\_back(make\_pair(dfs(ship(50, 0, 0)), 50));

sort(ans.begin(), ans.end());

int p = ans[2].first.people, t = ans[2].first.times;

cout << "班次\t有效等待时长\t等效航行时间\t开始等待时刻\t装载人数" << endl;

vector<answer> Answer;

for (auto i: mp)

{

ship f = i.first;

result s = i.second;

Answer.push\_back(answer(t - s.times + 1, f.t, f.x, f.waitClock, countPeople(f.t, f.waitClock), s.people));

}

sort(Answer.begin(), Answer.end());

int cntId = 1, cntLoad = p;

answer tmp = answer(1, ans[2].second, 0, 0, countPeople(ans[2].second, 0), ans[2].first.people);

int fst = 1;

for (vector<answer>:: iterator iter = Answer.begin(); iter != Answer.end(); iter++)

{

answer item = \*iter;

if (cntId == item.id && cntLoad == item.cntLoad && cntId <= t && item.start == tmp.spend + tmp.start + tmp.wait || (fst && item == tmp))

{

fst = 0;

tmp = item;

cout << item;

cntId++;

cntLoad -= item.load;

}

}

return 0;

}

第三题重新考虑第一问 源代码

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x=0;

int i,j,k,a,b,z,m1,m2,t1,t2,c,d,e;

printf("%3c%3c%3c%3c%3c%3c%5c\n",'i','j','k','a','b','m','z');

for(i=0; i<=6; i++)

{

for(j=0; j<=5; j++)

{

m1=i+j;

if(i\*20+j\*30+m1\*60>490) break;

for(k=0; k<=4; k++)

{

m1=i+j+k;

if ((i\*20+j\*30+k\*40+m1\*60<=490)&&(i\*20+j\*30+k\*40+m1\*60>=390)) //保证白天最后一次航班在17:00前发出，16:10后返回

{

if(i\*20+j\*30+k\*40+m1\*60>=430)//假设白天航班返回时间在五点之后

{

for(a=0; a<=4; a++)

{

for(b=0; b<=4; b++)

{

m2=a+b;

if(i\*20+j\*30+k\*40+a\*10+b\*20+m1\*60+m2\*60>730) break;//最后一班船在21:00前发出

if(i\*20+j\*30+k\*40+a\*10+b\*20+m1\*60+m2\*60>610)

{

z=i\*30\*3+j\*40\*3+k\*50\*3+a\*20\*5+b\*30\*5;

x++;

e=118\*z;

printf("%3d%3d%3d%3d%3d%3d%8d-%dA%6d%4d\n",i,j,k,a,b,m1+m2,e,m1+m2,z,x);

}

}

}

}

else

{

t1=420-(i\*20+j\*30+k\*40+m1\*60);

c=(120-t1\*3)/50;

d=(60-t1\*3)/50;

for(t2=1; t2<=d; t2++)

{

if(t1\*3+t2\*50>150) break;

for(a=0; a<=4; a++)

{

for(b=0; b<=3; b++)

{

m2=a+b+1;

if(420+a\*10+b\*20+m2\*60+t2\*10>730) break;//最后一班船在21:00前发出

if(420+a\*10+b\*20+m2\*60+t2\*10>610)

{

z=i\*30\*3+j\*40\*3+k\*50\*3+(t1+10)\*3+t2\*50+a\*20\*5+b\*30\*5;

x++;

e=118\*z;

printf("%3d%3d%3d%3d%3d%3d%8d-%dA%6d%4d",i,j,k,a,b,m1+m2,e,m1+m2,z,x);

printf(" abc\n");

}

}

}

}

}

}

}

}

}

return 0;

}

第三题重新考虑第二问两艘船 源代码（c++）

/\*\*

3-2-2 编译选项 g++ -std=c++14 -c

\*/

#include <bits/stdc++.h>

using namespace std;

const int inf = 0x3f3f3f3f;

const int dayEnd = 420;

const int nightEnd = 660;

const int options[] = {20, 30, 40, 50};

const int maxCapacity = 150;

const int minCapacity = 90;

const int day = 3;

const int night = 5;

struct result

{

int people, times;

result(int p, int t): people(p), times(t) {}

result(): people(), times() {}

friend bool operator < (result r1, result r2)

{

if (r1.people == r2.people)

{

return r1.times > r2.times;

}

else

{

return r1.people < r2.people;

}

}

friend bool operator == (result r1, result r2)

{

return r1.times == r2.times && r1.people == r2.people;

}

result operator +(result r)

{

result R = result(r.times + times, r.people + people);

return R;

}

};

struct ship

{

int t, x, waitClock;

ship(int t, int x, int waitClock): t(t), x(x), waitClock(waitClock) {}

ship(): t(0), x(0), waitClock(0) {}

friend bool operator<(ship s1, ship s2)

{

if (s1.waitClock != s2.waitClock)

{

return s1.waitClock < s2.waitClock;

}

else

{

if (s1.t != s2.t)

{

return s1.t < s2.t;

}

else

{

return s1.x < s2.x;

}

}

}

};

inline int countPeople(int &time, int start, int is0) //new countPeople

{

is0 = (is0 == 0 ? 0 : 1);

int ret = 0;

int End = time + start;

if (start >= dayEnd) // 晚上

{

if (End > nightEnd)

{

time = nightEnd - start;

}

ret = night \* (time + 10 \* is0);

}

else if (start < dayEnd && End > dayEnd)// 分别一半

{

ret = day \* (dayEnd - start + 10 \* is0) + night \* (End - dayEnd);

}

else //白天

{

ret = day \* (time + 10 \* is0);

}

if (ret >= minCapacity && ret <= maxCapacity)

{

return ret;

}

else

{

return -inf;

}

}

map<ship, result> mp;

result dfs(ship s)

{

//printf("t:%4d x:%4d waitClock:%4d\n", s.t, s.x, s.waitClock);

if (s.waitClock >= nightEnd)

{

return result(0, 0);

}

else

{

if (mp.count(s) > 0)

{

return mp[s];

}

else

{

vector<result> vec;

vec.push\_back(dfs(ship(20, max(0, 60 - s.x - 20), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(30, max(0, 60 - s.x - 30), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(40, max(0, 60 - s.x - 40), s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(50, max(0, 60 - s.x - 50), s.waitClock + s.t + s.x)));

result best = \*max\_element(vec.begin(), vec.end());

mp[s] = result(best.people + countPeople(s.t, s.waitClock, s.x), best.times + 1);

return mp[s];

}

}

}

struct answer

{

int id, wait, spend, start, load, cntLoad;

answer(int id, int wait, int spend, int start, int load, int cntLoad): id(id), wait(wait), spend(spend), start(start), load(load), cntLoad(cntLoad){}

friend bool operator<(answer a1, answer a2)

{

return a1.id < a2.id;

}

friend bool operator==(answer a1, answer a2)

{

return a2.id == a1.id &&

a2.wait == a1.wait &&

a2.spend == a1.spend &&

a2.start == a1.start &&

a2.load == a1.load &&

a2.cntLoad == a1.cntLoad;

}

};

ostream& operator<<(ostream &os, const answer &a)

{

os << a.id << "\t" << a.wait << "\t" << a.spend << "\t" << a.start << "\t" << a.load << endl;

return os;

}

int main()

{

vector<pair<result, int> > ans;

ans.push\_back(make\_pair(dfs(ship(30, 0, -10)), 30));

ans.push\_back(make\_pair(dfs(ship(40, 0, -10)), 40));

ans.push\_back(make\_pair(dfs(ship(50, 0, -10)), 50));

sort(ans.begin(), ans.end());

int p = ans[2].first.people, t = ans[2].first.times;

cout << "班次\t有效等待时长\t等效航行时间\t开始等待时刻\t装载人数" << endl;

vector<answer> Answer;

for (auto i: mp)

{

ship f = i.first;

result s = i.second;

Answer.push\_back(answer(t - s.times + 1, f.t, f.x, f.waitClock, countPeople(f.t, f.waitClock, f.x), s.people));

}

sort(Answer.begin(), Answer.end());

int cntId = 1, cntLoad = p;

answer tmp = answer(1, ans[2].second, 0, -10, countPeople(ans[2].second, 0, 0), ans[2].first.people);

int fst = 1;

for (vector<answer>:: iterator iter = Answer.begin(); iter != Answer.end(); iter++)

{

answer item = \*iter;

if (cntId == item.id && cntLoad == item.cntLoad && cntId <= t && item.start == tmp.spend + tmp.start + tmp.wait || (fst && item == tmp))

{

fst = 0;

tmp = item;

cout << item;

cntId++;

cntLoad -= item.load;

}

}

return 0;

}

第三题重新考虑第二问两艘船 源代码（c++）

/\*\*

3-2-3 编译选项 g++ -std=c++14 -c

\*/

#include <bits/stdc++.h>

using namespace std;

const int inf = 0x3f3f3f3f;

const int dayEnd = 420;

const int nightEnd = 660;

const int options[] = {20, 30, 40, 50};

const int maxCapacity = 150;

const int minCapacity = 90;

const int day = 3;

const int night = 5;

struct result

{

int people, times;

result(int p, int t): people(p), times(t) {}

result(): people(), times() {}

friend operator < (result r1, result r2)

{

if (r1.people == r2.people)

{

return r1.times > r2.times;

}

else

{

return r1.people < r2.people;

}

}

friend operator == (result r1, result r2)

{

return r1.times == r2.times && r1.people == r2.people;

}

result operator + (result r)

{

result R = result(r.times + times, r.people + people);

return R;

}

};

struct ship

{

int t, x, waitClock;

ship(int t, int x, int waitClock): t(t), x(x), waitClock(waitClock) {}

ship(): t(0), x(0), waitClock(0) {}

friend bool operator<(ship s1, ship s2)

{

if (s1.waitClock != s2.waitClock)

{

return s1.waitClock < s2.waitClock;

}

else

{

if (s1.t != s2.t)

{

return s1.t < s2.t;

}

else

{

return s1.x < s2.x;

}

}

}

};

inline int countPeople(int &time, int start)

{

int ret = 0;

int End = time + start;

if (start >= dayEnd) // 晚上

{

if (End > nightEnd)

{

time = nightEnd - start;

}

ret = night \* time;

}

else if (start < dayEnd && End > dayEnd)// 分别一半

{

ret = day \* (dayEnd - start) + night \* (End - dayEnd);

}

else //白天

{

ret = day \* time;

}

if (ret >= minCapacity && ret <= maxCapacity)

{

return ret;

}

else

{

return -inf;

}

}

map<ship, result> mp;

result dfs(ship s)

{

//printf("t:%4d x:%4d waitClock:%4d\n", s.t, s.x, s.waitClock);

if (s.waitClock >= nightEnd)

{

return result(0, 0);

}

else

{

if (mp.count(s) > 0)

{

return mp[s];

}

else

{

vector<result> vec;

vec.push\_back(dfs(ship(20, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(30, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(40, 0, s.waitClock + s.t + s.x)));

vec.push\_back(dfs(ship(50, 0, s.waitClock + s.t + s.x)));

result best = \*max\_element(vec.begin(), vec.end());

mp[s] = result(best.people + countPeople(s.t, s.waitClock), best.times + 1);

return mp[s];

}

}

}

struct answer

{

int id, wait, spend, start, load, cntLoad;

answer(int id, int wait, int spend, int start, int load, int cntLoad): id(id), wait(wait), spend(spend), start(start), load(load), cntLoad(cntLoad){}

friend bool operator<(answer a1, answer a2)

{

return a1.id < a2.id;

}

friend bool operator==(answer a1, answer a2)

{

return a2.id == a1.id &&

a2.wait == a1.wait &&

a2.spend == a1.spend &&

a2.start == a1.start &&

a2.load == a1.load &&

a2.cntLoad == a1.cntLoad;

}

};

ostream& operator<<(ostream &os, const answer &a)

{

os << a.id << "\t" << a.wait << "\t" << a.spend << "\t" << a.start << "\t" << a.load << endl;

return os;

}

int main()

{

vector<pair<result, int> > ans;

ans.push\_back(make\_pair(dfs(ship(30, 0, -10)), 30));

ans.push\_back(make\_pair(dfs(ship(40, 0, -10)), 40));

ans.push\_back(make\_pair(dfs(ship(50, 0, -10)), 50));

sort(ans.begin(), ans.end());

int p = ans[2].first.people, t = ans[2].first.times;

cout << "班次\t有效等待时长\t等效航行时间\t开始等待时刻\t装载人数" << endl;

vector<answer> Answer;

for (auto i: mp)

{

ship f = i.first;

result s = i.second;

Answer.push\_back(answer(t - s.times + 1, f.t, f.x, f.waitClock, countPeople(f.t, f.waitClock), s.people));

}

sort(Answer.begin(), Answer.end());

int cntId = 1, cntLoad = p;

answer tmp = answer(1, ans[2].second, 0, -10, countPeople(ans[2].second, 0), ans[2].first.people);

int fst = 1;

for (vector<answer>:: iterator iter = Answer.begin(); iter != Answer.end(); iter++)

{

answer item = \*iter;

if (cntId == item.id && cntLoad == item.cntLoad && cntId <= t && item.start == tmp.spend + tmp.start + tmp.wait || (fst && item == tmp))

{

fst = 0;

tmp = item;

cout << item;

cntId++;

cntLoad -= item.load;

}

}

return 0;

}

灰色模型 源代码（matlab）

clear

syms a u;

c=[a,u]';%构成矩阵

[A] = xlsread('people.xlsx');

A = A';

Ago=cumsum(A);%原始数据一次累加,得到1-AGO序列xi(1)。

n=length(A);%原始数据个数

for k=1:(n-1)

Z(k)=(Ago(k)+Ago(k+1))/2; %Z(i)为xi(1)的紧邻均值生成序列

end

Yn =A;%Yn为常数项向量

Yn(1)=[]; %从第二个数开始，即x(2),x(3)...

Yn=Yn';

E=[-Z;ones(1,n-1)]';%累加生成数据做均值

c=(E'\*E)\(E'\*Yn);%利用公式求出a，u

c= c';

a=c(1);%得到a的值

u=c(2);%得到u的值

F=[];

F(1)=A(1);

for k=2:(n+3)

F(k)=(A(1)-u/a)/exp(a\*(k-1))+u/a;%求出GM(1,1)模型公式

end

G=[];

G(1)=A(1);

for k=2:(n+3)

G(k)=F(k)-F(k-1);%两者做差还原原序列，得到预测数据

end

t1=1:n;

t2=1:n+3;

plot(t1,A,'bo--');hold on;plot(t2,G,'r\*-');title('源数据-预测值');legend('源数据','预测值');

%后验差检验

e=A-G(1:n);

q=e/A;%相对误差

s1=var(A);

s2=var(e);

c=s2/s1;%方差比

len=length(e);

p=0; %小误差概率

for i=1:len

if(abs(e(i))<0.6745\*s1)

p=p+1;

end

end

p=p/len;

s22 = var(A);

s23 = var(G);