依据时差和飞行疲劳的国际会议选址模型

摘要

对于国际会议来说，时差和飞行疲劳对参会者的工作效率有显著的负面影响。会议举办方可以通过延长参会者在目标城市的休息时间以期待减少工作效率的损失，但这同时也增加了会议举办方的组织成本，因此如何平衡时差、旅行疲劳和到达后休息时间的关系对一个预算有限的会议举办方尤为重要。为了更好地协助举办方做好这个平衡，本文探索了时差和飞行疲劳对参会者到达目标城市时初始工作状态的影响，以及到达后的休息对时差相应的恢复效果。在本文模型计算出的推荐地点举办国际会议，会使参赛者在开会期间的工作效率达到最佳。此推荐将为会议举办方选择目标会议城市提供参考。

本文首先通过时差子模型计算出参会者跨越一定时区第一天到达目标城市时的工作效率，再通过飞行疲劳子模型算出参赛者第一天的疲劳程度，对工作效率进行向下矫正，得出参会者第一天的实际工作效率。之后，本文通过计算预算和消费得到参赛者可以休息的天数。在恢复子模型中，输入休息天数和到达目标城市第一天的效率可以得到所有参赛者会议期间的效率总和。本文将各个备选目标城市得到的所有参会者总效率进行打分、排名，得出最终城市备选名单。

在时差子模型中，本文参会者了双过程模型（The Two-process Model）对参会者在原城市的昼夜节律效率函数和稳态效率函数进行求解。在此模型的基础上，我们探索了在不同时差下双过程模型函数的变化，从而计算出时差对于参赛者效率的影响。双过程模型经过严谨的科学实验研究，在此基础上建立的模型预测能力良好。本模型能够准确求出时差带给参赛者工作状态相较原城市状态的比值，结果符合众多文献和经验描述。

在飞行疲劳子模型中，本文对文献中飞行疲劳数据进行拟合，求出飞行疲劳关于飞行时间的函数。模型通过地球标准椭球面上两点间距离算出飞行距离，将其带入疲劳函数得出参赛者的飞行疲劳指数。本文将时差子模型中的工作效率得数与飞行疲劳指数结合，得到该参会者在目标城市的初始效率值。飞行疲劳函数拟合效果良好。

在恢复子模型中，本文根据文献和经验描述，推导出工作效率关于休息时间的微分方程。本文采用的微分方程是自治的，即恢复情况只和工作效率自身值有关，而不和特定时间点有关。这符合时差和疲劳带给人影响的物理意义。微分方程的系数是根据经验描述“恢复时间为三分之二个跨越时区”确定的，我们将时差和相应经验性的恢复时间带入到微分方程中，得出不同时差下的系数，将其平均。这有效减少了极端情况的出现，使不同时差下恢复情况符合资料。本文假设举办方预算除去机票和开会期间的酒店费用全部用在提前将参会者送达目标城市的额外酒店钱中。本文采用的酒店价格为真实数据。

为方便举办方计算，模型允许用户自定义会议预算。用户在我们提供的图形界面输入参会者城市信息后，程序将根据模型计算出推荐城市排名并在地图上进行标注。程序内置了全球各地区、国家77个城市的信息，包括经纬度和酒店价格信息；这些真实数据增加了预测结果的可靠性。另外，本地数据库减少了联网查询的时间，增加了运算速度。

关键词 时差 会议选址 昼夜节律 稳态 工作效率 飞行疲劳 微分方程 图形界面

目录

一、问题重述

题目给出给定数量和地区的参会者名单，要求我们通过模型运算，确定会议地点，使参会者在会议期间工作效率的总和最大。

前人做过许多关于时差和睡眠质量的研究。在"Physiological Changes Underlying Jet Lag”[1]中，简述了时差给人们的影响及原因，并提出了向西飞行比向东飞行更容易导时差，适应当地的情况。“Taking the Lag out of Jet Lag through Model-Based Schedule Design”[2]，“Sleep Homeostasis And Models Of Sleep Regulation”[3]，"Optimal Phase-Tracking Of The Nonlinear Circadian Oscillator”[4]和“Interactive Mathematical Models of Subjective Alertness and Cognitive Throughput in Humans”[5]这五篇论文详述了数学模拟的睡眠模型。其中三过程模型(The Three-process Model）指出工作效率被稳态、昼夜节律、睡眠惯性三个因素影响。

针对时差带来的工作效率的损失，本文采用了此模型之前提出的双过程模型（The Two-process Model),该模型指出工作效率只被稳态和昼夜节律操控。针对时差的恢复，"Predicting Sleep Latency From The Three-Process Model Of Alertness Regulation”[6]中，作者详细描述了了当人倒时差时，昼夜节律逐渐变化为原城市状态的过程。在"Fatigue In Two-Pilot Operations: Implications For Flight And Duty Time Limitations”[7]中，作者对飞机上的乘务员们进行了飞行疲劳的调查，统计了在不同工作时长下乘务员的劳累程度。本文直接引用了该论文的数据，进行飞行疲劳函数的拟合。

本文讨论时差、旅行疲劳和会前休息带给参会者工作状态的影响，将多个参会者的情况进行整合，筛选出开会期间工作效率最大的城市。本文将会议举办方资金设成一个参数，除去机票和进行会议时三天的酒店费用外，全部用在将参会者提前运达参会地点，使其入住酒店进行调整。

由于不同国家的购买力水平不同，在飞机价格和酒店价格上将会有所差距。购买力低的国家，飞机和旅店价格相应较低，因此可以有更多的钱投入到休息恢复的旅店钱中；反之，价格水平高的国家，投入休息恢复的钱便会减少。为了权衡刚下飞机的工作效率的不同和各国家价格波动所带来的会前休息长短的不同，本文通过建模准确求出数据库中77个重点城市在开会期间的效率总值并根据得出的效率为每个候选城市进行打分，从而进行比较，最终在这77个城市中择优选定推荐开会地点，为IMMC组委会提供参考。

为了提升用户体验，模型允许用户自定义资金数量，在输入参会者地区和会议时间后，有直观的界面显示选定的城市。模型采用了网上真实的飞机和旅店数据，自制数据库，加快了程序运行的时间。

二、基本假设

1. 所有参会者都坐飞机前往会议所在的城市

理由：这是一个全球性的科技商业研讨会，参会者来自全球各地。由于参会者所在的城市大多相差很远，通过飞机到达目标城市是最有效地方式，因此本文假设所有人员都坐飞机前往目标城市。

2. 所有参会者每天晚24点睡觉，早上8点起床。起床后立刻投入工作，中途不考虑休息时间。

理由：成人每天需要8小时左右的睡眠。由于每天工作安排非常紧张，我们假设8AM至晚24点参会者一直处在工作状态中。

3.　　参会者在坐上飞机当天0点后开始按目标城市的时间表进行作息

理由：当到达一个新地方时，人们通常立刻遵循新城市的作息，快速适应新城市的时区。调整作息的时间每人会有所不同，这涉及到了航班时间、个人习惯等未知因素。因此，为了简化模型，我们假设参会者统一在坐上飞机当天0点开始调整到城市作息。

4.　　影响效率的因素只有时差，旅行疲劳和休息时间

理由：根据众多文献显示，长途旅行影响工作效率的主要因素为时差和旅行疲劳，到达目的地后主要影响因素为休息时间。数据库中的酒店价格为当地平均水平，因此在酒店环境质量上会有所差别。但由于酒店环境没有起到主要影响作用，我们对环境造成的效率影响不予考虑。

5.　　资金有限，且除去机票和进行会议时三天酒店的费用，其他所有预算将用于提前抵达城市调整时差的酒店费用中。

理由：由于本文假设休息时间会对工作效率产生影响，而休息效率由资金剩余决定，为了能够比较出所有因此本模型将会议举办方提供的资金设为一个定量，即除去机票和进行会议时三天酒店的费用，其他所有预算将用于提前抵达城市调整时差的酒店费用中。

6.　　所有参会者于同一天抵达目标城市

理由：在国际会议中，参会者通常在同一天抵达城市。

7.　　参会者不会采用药物、光疗等医学方法加快时差恢复的速度。

三、符号约定

t = 时间（h）

Δn = 时差（h）

P = 工作效率

P_0= 原城市工作效率

P_1= 时差影响后的到达目标城市第一天的初始效率

$P_{初}$= 时差和飞行疲劳同时影响后的到达目标城市第一天的初始效率

S = 稳态效率

S_a= 醒时稳态效率

S_s= 睡眠稳态效率

S_1= 参会者到达目标城市时的初始稳态效率

S_0= 原城市稳态效率

b = 原城市零点稳态效率数值

α_a= 原城市醒时稳态效率系数

α_s= 原城市睡眠稳态效率系数

β_a= 目标城市初始醒时稳态效率系数

β_s= 目标城市初始睡眠稳态效率系数

q = 原城市零点效率

r_{Hw}= 稳态的衰减速率（h^{-1}）

r_{Hs1}= 稳态恢复的速率（h^{-1}）

u_c= 昼夜节律的渐进值

u_h= 稳态回复的增进值

t_w= 上一次睡眠前清醒时长（h）

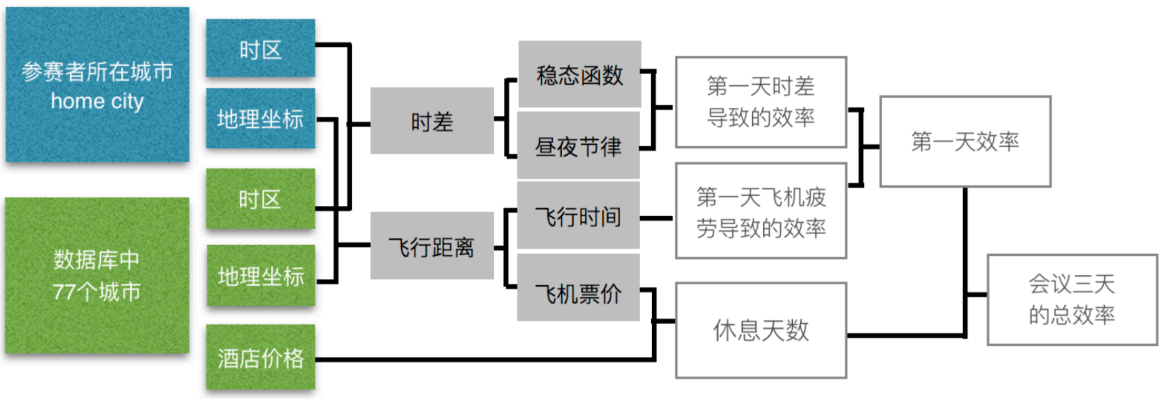
C = 昼夜节律效率

C_0= 原城市昼夜节律效率

C_1 = 参会者到达目标城市时的初始昼夜节律效率
$L_{飞}$ = 飞行距离
T = 飞行时间（h）
$T_{休}$ = 提前抵达目标城市的休息时间
F = 飞行疲劳
k = 飞行疲劳函数系数
b = 疲劳函数系数
b_0 = 疲劳模型中最优拟合线斜率
R = 时差和飞行疲劳基本恢复的百分比（%）
M = 资金预算($)
$M_{飞机}$ = 单人飞机价格($)
$M_{酒店}$ = 单人酒店价格($)

四、总模型概述

1. 模型思路



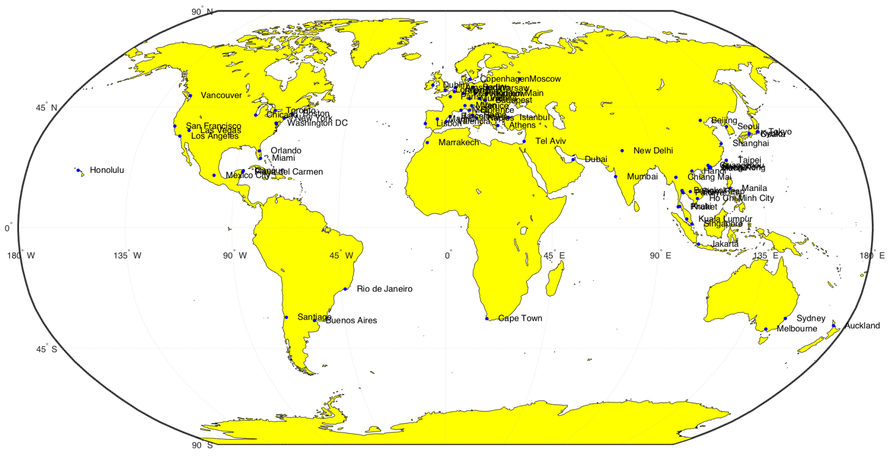
图一 总模型思路

用户输入参会者的人数、城市和月份的信息后，我们首先将在数据库中搜索这些城市的地理位置和时区。通过计算 77 个备选城市和参会者所在城市的时区差，可以得到参会者时差影响的到达目标城市第一天时差影响的初始效率 P_1。之后，通过计算 77 个备选城市和参会者所在城市间的距离，可以算出飞行距离、飞行时间、和飞机票价。通过飞机时间和效率的函数式，可以得到参会者的飞行疲劳系数。将飞行疲劳系数和时差决定的 P_1进行结合即可得到参会者第一天的效率 $P_{初}$。通过总支出和目标城市的机票、酒店价格可以算出休息天数。从而得到会议期间参会者的效率 P。分别将每个城市的效率和最高总效率的城市的效率求得一个百分比作为每个城市的分数，并将 77 个城市的出的分数进行排名，便可得到推荐的候选城市名单。

2. 数据库

我们选择了77个旅游人数最多的城市作为可供选择的目标城市。旅游人数的多少可以反映城市的受欢迎程度，即气候适宜、风景优美、名胜古迹较多等。这些条件都可以让参会者舒适愉快，工作效率也相对较高，因此适合召开科技商业研讨会。

我们得到了这80个城市的地理位置[8]、时区[9]、酒店单价[12]、飞机票价等信息，放到数据库中，方便之后模型的计算。



图二 数据库城市分布

如图为 77 个城市的分布图。可以看出城市主要集中于欧洲、东南亚和北美。

本文数据库只包括了 77 座城市。这 77 座城市主要集中在欧洲、东南亚、和北美地区，对于其他大陆涉及较少。为了减少联网查询数据的时间，本文采用本地数据库。由于时间和酒店价格资料有限，本文无法在短时间内构件强大的数据库。如果时间充裕，本文可以拓宽数据库容量，增加亚洲、非洲、南美的城市数量，为会议举办方提供更全面的参考。

五. 时差子模型

1. 问题分析

在过去的三十年中，双过程模型（The Two-process Model）一直作为人们衡量睡眠和光照对于人日常效率大小的方法。该模型提出人在一天特定时间的清醒程度分别由昼夜节律（Process C）和稳态（Process S）决定。

人身体内的昼夜节律（Circadian Rhythm）使人以24小时左右的周期进行活动。昼夜节律具有内源性和持续性。内源性指昼夜节律是人体体内长时间稳定的一个周期性变化过程，只有在遇到时差转变时，在同步器（Zeitgeber）的刺激下会进行相应的调整，并最终稳定在新时区的稳定状态。持续性指在时差调整后，昼夜节律会在24小时内保持原城市的初始状态。本模根据这一信息型假设到达新时区的第一天昼夜节律保持原时区的节律，在休息过程中逐渐调整到新时区的稳定状态。

睡眠时间和长短决定了体内机能调节的稳态（Homeostasis），从而决定了人们在该时刻的稳态效率。为了得到不同时差下到达目的地第一天在工作时间段（8AM－24PM）的时差影响的初始效率P1，我们假设该参会者在到达会议地点第一天的0点开始调整成当地作息，即睡眠时间和长短和当地作息保持一致。

我们从两个方面思考这个问题：

1）参会者在原城市工作时间段的效率（初始昼夜节律和稳态函数状态）

2）参会者在时差作用下工作效率的变化（时差带来的昼夜节律和稳态函数的变化）

本文首先根据双过程模型对参会者在原城市的工作效率进行求解。再在双过程模型的基础上根据时差对昼夜节律函数和稳态函数的变化进行求解并得出在新城市工作效率相对于原城市工作效率的比例。

2. 模型的求解

1）参会者在原城市工作时间段的效率

a. 稳态效率函数

稳效率态函数分为醒时稳态效率和睡眠稳态效率，双过程模型中的醒时稳态和睡眠稳态和其参数关系如下[5]：

（醒时稳态效率） $$\frac{dS_a}{dt} = -2t_w(r_{Hw})^2(S-u_c)$$

（睡眠稳态效率） $$\frac{dS_s}{dt} = r_{Hsl}(u_H - S)$$

其中参数为：

$$r_{Hw} = \tfrac{1}{32.0}h^{-1}$$
$$r_{Hsl} = \tfrac{1}{2.14}h^{-1}$$
$$u_H = 0.9896$$
$$u_c = 0.1503$$

S数值越大，稳态效率却大；S数值越小，稳态效率越小。通过方程可以看出，稳态效率最大值约为1。

解得微分方程：

$$S_a = 0.1503 + \alpha_a e^{\frac{-2\times16}{32\times32}t}$$
$$S_s = 0.9896 - \alpha_s e^{\frac{-1}{2.14}t}$$

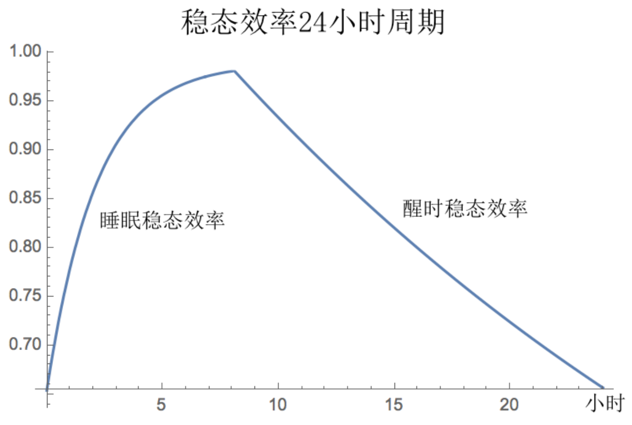
其中α_a，α_s为未知系数。

参会者在原城市的的昼夜节律形成稳定状态，即该参会者在任意时刻的效率值等于第二天该时刻的值，从而形成循环。因此可列出下列方程：

$$0.9896-(0.9896-q)e^{-\frac{8}{2.14}} = 0.1503+\alpha_a e^{\frac{8\times(-2)\times16}{32\times32}}$$
$$0.1503+\alpha_a e^{\frac{24\times(-2)\times16}{32\times3}} = 0.9896-(0.9896-q)e^{0}$$

解得：

$$\alpha_a = 0.335$$
$$\alpha_s = 1.067$$
$$S_0(0) = 0.655$$



图三 稳态效率24小时周期

图为参会者在原城市的稳态24小时变化图。在0点至早8点，参会者处于睡眠状态，稳态效率持续升高。睡眠状态下的稳态效率为参会者如果在睡眠状态下工作的效率，但不算在工作时间段的总效率当中。到达早8点清醒时刻，稳态效率持续下降，直至24点，到达0点水平。

对工作时间段的值进行积分，可得出参会者一天的工作稳态效率：

$$\int_{8}^{24} S_0 = 12.915$$

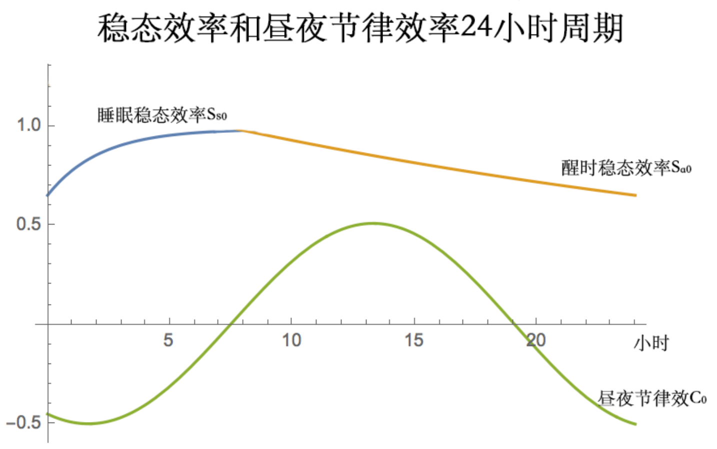
b. 昼夜节律函数

根据双过程模型，参会者在原城市的昼夜节律效率函数如下[3] [11]：

$$C = A\{0.97\sin[\omega(t-t_0)] + 0.22\sin[2\omega(t-t_0)] + 0.07\sin[3\omega(t-t_0)] + 0.03\sin[4\omega(t-t_0)] + 0.001\sin[5\omega(t-t_0)]\}$$

其中常数为：

$$\omega = \tfrac{2\pi}{\tau}$$
$$\tau = 24h$$
$$t_0 = 8.6h$$
$$A = 0.4$$



图四 稳态效率和昼夜节律效率24小时周期

上图为参会者的在原城市昼夜节律效率24小时变化图。稳态效率在大约19点日落后开始为负，在凌晨2点达到最低点，8点后上升至正数，在13点达到顶峰。

对工作时间段的昼夜节律效率函数进行积分，可得出参会者一天的工作昼夜节律效率：

$$\int_{8}^{24} C_0 = 4.105$$

因此，参会者在原城市的工作效率总和为：

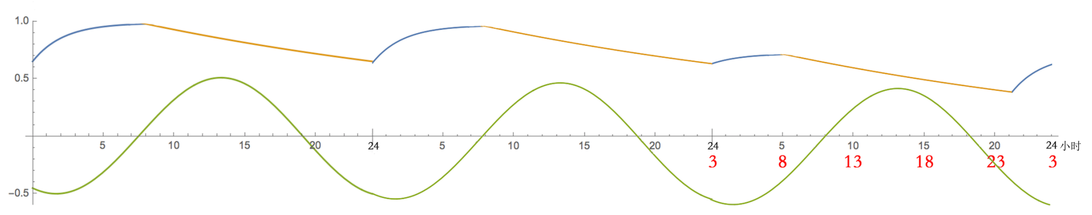
$$P_0 = \int_{8}^{24} S_0 + \int_{8}^{24} C_0 = 17.020$$

2） 参会者在时差作用下工作时间段的效率变化

假设参会者在第一天24点后开始调整成会议地点作息。时差为△n（△n以原城市作为初始值逆时针递增，0≤△n<24）。

当△n<8时，参会者在原城市24点时进入睡眠状态，将点（△n，0.655）带入S_s中，可求得β_a和 β_s；同理，当8≤△n<24时，参会者在24点进入清醒状态，将（△n，0.655）带入S_a中，可求出β_a和β_s。积分可得会议时区8AM至24PM的稳态效率总值。

稳态效率和昼夜节律效率72小时变化



前48小时为原城市效率，稳态效率和昼夜节律效率不变；
后24小时为调整作息后的效率，昼夜节律效率不变，稳态效率下降。工作时段（8AM—24AM）稳态效率和昼夜节律效率明显下降。

红色数字为目标城市时间

图五 稳态效率和昼夜节律效率72小时变化

如图前 48 小时，参会者在原城市，稳态效率和昼夜节律效率保持不变。后 24 小时，该名参会者经历了 3 个时差的跨越(△n=3)。在第二天 24 时，该名参会者进入到了睡眠状态，此时为目标城市的早上 3 点。经过 5 个小时睡眠，参会者清醒后，稳态效率逐渐下降。可观察到稳态效率整体的向下平移。

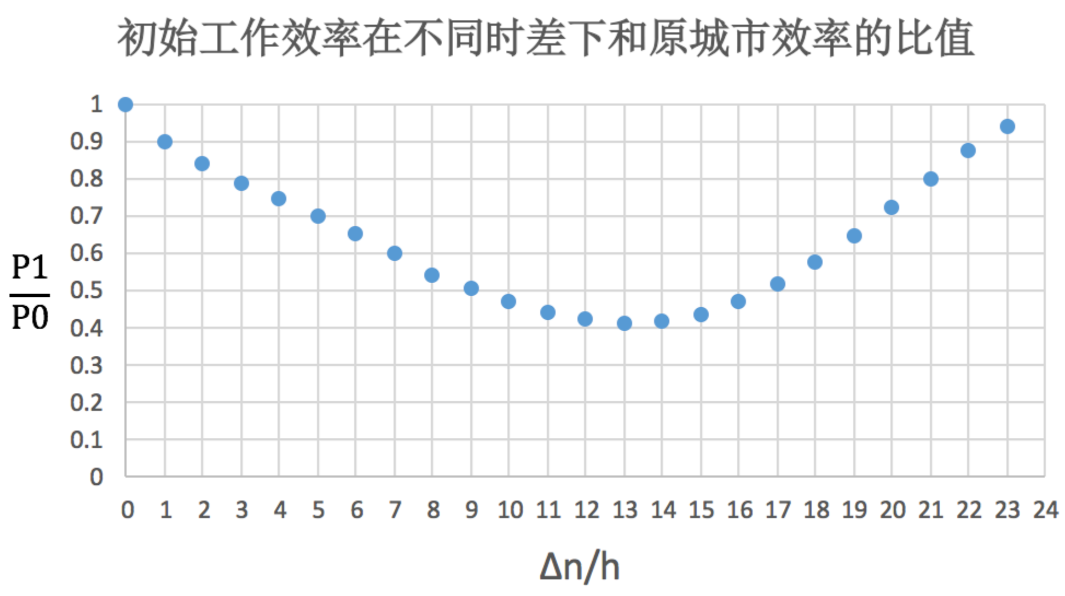
由于昼夜节律的持续性，在第二天的24小时中昼夜节律效率函数保持和原城市时间一致。积分求出目标城市8AM至24PM的昼夜节律效率总值，可得到参会者到达目标城市时的时差影响的初始效率：

$$P_1 = \int_{8}^{24} S_1 + \int_{8}^{24} C_1$$

将各时差P_1得数和原城市工作效率P_0进行比较，得出时差影响的初始效率P_1相对于原效率P_0的比值。

时差	0	1	2	3	4	5	6	7	8	9	10	11
效率	1	0.901	0.837	0.789	0.746	0.700	0.651	0.596	0.539	0.504	0.471	0.443
时差	12	13	14	15	16	17	18	19	20	21	22	23
效率	0.422	0.412	0.416	0.435	0.469	0.517	0.578	0.647	0.722	0.799	0.873	0.940

表一 不同时差下的效率P_0



图六 不同时差下的效率P_0

如图为不同时差下第一天初始工作效率的变化图像。效率最低点发生在14小时时差处。当△n远离14时，工作效率呈线性增长。观察到向西飞相同的时区和向东飞相同的时区带来的工作效率减少较少，与众多文献论述相符[2]。

3. 模型评价

1）本模型采用了双过程模型对稳态效率和昼夜节律效率进行精准的预测。在原双过程模型的基础下，我们对时差带来的效率变化做出分析，能预测出任意时差下的效率值。由于时差带来的影响数据大多由主观进行评价且数据量不足，无法对本文结果进行精准评估。但是本模型得出的结果显示向西飞行比向东飞行效率减少要小，这一点符合文献中的论述[2]。另外，本模型开篇引用的双过程模型建立在严谨的医学实验之下，对原城市下的稳态效率和昼夜节律效率评价科学可靠，在此基础上建立的模型预测能力期望值较好。

2）模型将时差给人带来的影响细化到24小时。可以推测出一天任意给定时刻的效率值。为参会者的休息调整策略提供了很大的指导作用。但由于本文假设参会者不会不会采用药物、光疗等医学方法加快时差恢复的速度，因此本模型不对调整策略进行深入分析。

3）模型求出各时差下工作效率相对于原城市工作效率的比值。这些数据将被带入到飞行疲劳子模型和恢复子模型中进一步计算旅途劳累和休息对效率产生的变化。

4. 改进方向

1）模型假设参会者在离开原城市当天0点开始调整其作息，此时完全按照目标城市作息生活。然而现实中，当时差相差很大时，人们不一定可以直接转变作息，按目标城市时间生活。因此，如果有更多时间，该模型可以包含人们无法成功导时差、以及下飞机后再开始倒时差的情况。

2）1999年提出的三过程模型（The Three-process Model）[5]中加入了对睡眠惯性（Sleep Inertia）的论述。该模型表示工作效率同时被稳态效率、昼夜节律效率和睡

眠惯性效率同时决定。睡眠惯性指在人刚刚清醒时机体没有调整到工作状态下的工作效率的降低。在该模型中给出的睡眠惯性w_0为定值，w_0＝－0.5346。本文没有考虑睡眠惯性问题是考虑到时差打乱了原有的睡眠节律，从而使睡眠惯性难以预测。另外，由于睡眠惯性为定值，因此对于不同时差下的效率比较没有产生影响，故不予以考虑。

六、飞行疲劳子模型

1. 问题分析

旅客乘坐飞机出行，尤其是国际远距离、长时间航班，会产生疲劳感。过往研究表明，旅行疲劳主要成因有：飞行途中有限的饮食带来的脱水；长时间处于狭小空间之中；飞行途中活动范围、机会受限；高空所带来的低压低氧状态；正常的昼夜作息规律被扰乱；精神高度紧张等[13]。并且，旅行疲劳会加剧时差所带来的旅行后负面生理反应。因此我们将疲劳程度引入对于与会者工作效率的考量中。我们通过城市的地理位置，算出了两座城市间的距离和飞机的时长，并分析了坐飞机时的劳累程度和飞机时长的关系，得到了每名与会者坐飞机时的疲劳程度。

2. 模型建立

我们首先确定了出发城市和目标城市经纬度的位置("Data Science Toolkit")。通过 MATLAB 的查询，得到了城市间的飞行距离 L。

由于飞机的平均速度为 900km/h，可以计算出飞行时间：

$$T = \frac{L}{900km/h}$$

在“Fatigue in Two-Pilot Operations: Implications for Flight and Duty Time Limitations” 的文献中，我们发现：疲劳程度和飞行时间呈正相关趋势，但飞行时间越长疲劳度增加值越小，且疲劳值和飞行时间成二次函数的关系[7]。在这篇文献之中，对于疲劳值的衡量采用了国际民航组织（ICAO）采用的萨-佩二氏量度（Samn-Perelli 7-pt Scale)，其取值的范围为1-7，并且在取值为7时是疲劳度最大时[7]。在对飞行疲劳的调查中，我们发现了以下关于飞行时长和疲劳程度的实验数据：

时长（小时）	0	3	4.5	6	7.5	9	10.5	12
疲劳程度	0	1.95	2.33	2.66	2.93	3.11	3.24	3.29

表二 飞行疲劳论文中的数据点

我们运用二次函数对时长和疲劳程度进行拟合，得到了函数：

$$F = b\sqrt{T}$$

$$b = 0.14$$

其中 F 值的大小衡量了人感受到疲劳对自身生理机能的影响，我们将每一项疲劳程度值通过和满分 7 分（影响最大）进行比较得出了影响工作效率的百分比。通过将 100%和 F 取差我们得到了最终衡量有飞行疲劳与无飞行疲劳的效率比的系数。我们将这个系数与理论上没有飞行疲劳、仅有时差带来的第一日效率值进行相乘，便可得到该参会者到达另外一个城市的初始效率值，公式如下：

$$P_{初}=P_1\times(1-F)$$

3. 灵敏度分析

疲劳度模型的一个关键参数，b，即上述函数中的 0.14，取决于我们对飞行时长与疲劳程度的平方进行线性拟合产生的最优拟合线的斜率 b_0。因此，我们进行了对于疲劳子模型对于 b_0 的灵敏度分析。

由上表可以得到 b_0 为 0.919，我们以此为中心构造了一个 95%的置信区间

$$b_0 \in (0.14 \pm t_{0.95}(d.f. = 6) \times s_{b_0})$$

得到了 b_0 的上下限分别为 0.742 与 1.095，因而得到函数：

$$F = (0.115, 0.165)\sqrt{T}$$

我们采用的出发城市为 Sapporo、Ulan Bator、Almaty、Gdansk、Strasburg、Burgundy、Brighton、Boston，以及假设会议举办方拥有$8,000 的预算，从而得到了如下的结果：

b 取值	第一选择	第二选择	第三选择	第四选择	第五选择
0.115	Krakow	Istanbul	Chiang Mai	Pattaya	Brussel
0.14	Krakow	Istanbul	Chiang Mai	Pattaya	Brussel
0.165	Krakow	Istanbul	Chiang Mai	Pattaya	Brussel

表三 b 值在$8,000 预算下的灵敏性分析

在将预算降低为$6,000 之后，运用相同的城市集，得到的结果如下：

b 取值	第一选择	第二选择	第三选择	第四选择	第五选择
0.115	Krakow	Brussel	Valencia	Istanbul	Berlin
0.14	Krakow	Brussel	Valencia	Istanbul	Berlin
0.165	Krakow	Brussel	Valencia	Istanbul	Berlin

表四 b 值在$6,000 预算下的灵敏性分析

由此可见，b_0 在 95%置信区间之中浮动时，随之产生的 b 的浮动对于最终结果输出的影响及其微乎其微，甚至在上述情况下不会影响到我们设计的算法对于前几位推荐城市的计算结果。

4. 模型评价

1） 模型运用权威论文中的定性分析并结合真实数据拟合出疲劳函数。数据真实可靠，对结果预测预期良好。拟合结果很好，拟合函数残差分析

$$R^2 = 0.9644$$

5. 后续改进方向

1）本模型忽略了不同时间乘坐飞机疲劳程度的不同。资料显示，夜晚坐飞机带来的疲劳程度大于白天坐飞机带来的疲劳程度。但由于无法确定参会者乘坐飞机时的准确时间，从而无法确定疲劳程度的改变。为了简化模型并对缺失信息进行响应，本模型中没有考虑航班时间对疲劳程度的影响。如果有时间，可以细化参会者上下飞机的时间段，计算不同时段飞机对疲劳度的影响。

2）本模型对于疲劳度的描述直接将萨-佩二氏量度通过百分比的方式转换为了对于效率影响的百分比，然而在实际中萨-佩二氏量度是一个偏主观、离散的量度，在后续模型改进中可以寻找一种更客观衡量疲劳对于效率减损的影响的度量方式。

3）本模型假设了飞行疲劳带来的效率减损和时差所带来的效率减损可通过相乘的方式进行叠加。在我们所接触到的文献之中并没有对于飞行疲劳以及时差综合征效果叠加的研究，因而我们做了一定合理的假设；若有机会可以更细致地查阅过往研究，并在模型改进中引入比简单相乘更细化的叠加方式。

七、恢复子模型

1. 问题分析

当人们到达一个新时区时，会因为时差、旅途劳累等原因感到不适应。随着时间的变化，人们的状态越来越好。在逐渐适应当地时差时，人们在会议城市的时间越长，工作效率的增长指越小，且呈指数分布。因此，我们用了指数函数来模拟人们恢复时差的过程。

2. 模型的建立

1）恢复场的设计

我们假设时差和旅途劳累带来的效率可以被量化为一个数值，且这个值随时间的变化趋势只和自身大小有关而不和特定时间和特定时差或疲劳程度有关。

$$\frac{dP}{dt}=k(1-P)$$

由于人们在会议城市的时间越长，效率逐渐增加的越慢，我们列了微分方程：

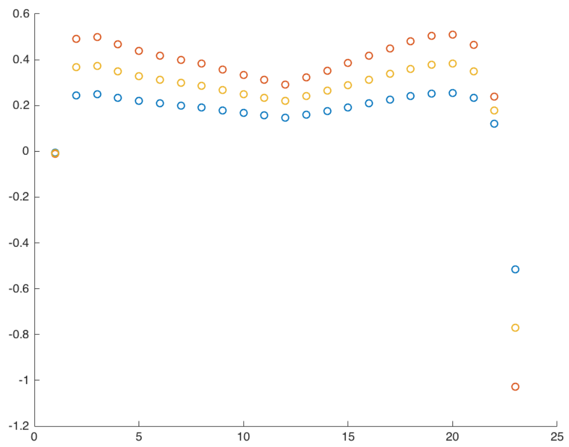
$$P=1-(1-P_{\text{初}})e^{-kt}$$

其中 P 初为到达目标城市时第一天的效率，t 表示在城市停留的天数。

在“Jet Lag: Trends And Coping Strategies”中，记录了平均人们要花费跨越时区乘以三分之二的天数[2]，来基本恢复时差带来的影响，即：

$$P(\tfrac{2}{3}\triangle n)\geq R$$

其中 R 为基本恢复时的效率。



图七 不同 R 值和时差下的 k 值

通过观察不同 R 下，k 值的变化。我们发现当 R＝90%时，k 的浮动较小，且 R 符合“基本恢复”这一物理意义。图中标注的蓝色圆点为 R＝90%时，k 的取值。当时差 Δn＝1，24 时，k 值为负。这是因为 0.9 大于 Δn＝1，24 时 $P_{初}$的取值，这表示 P 需要负增长来达到 R 值水平。因此，我们去除了这两个特殊点后对 k 值进行平均，得到 k 值大小。平均处理有效减少了极端恢复情况的出现。

在模型一中，我们得到了不同时差下第一天的效率。将 24 个 n 和其所对应的 P 带入方程：

$$P(\tfrac{2}{3}\Delta n) = R$$

可以得到 24 个不同的 k 值。我们发现：在 R 等于 0.9 时，24 个 k 值的方差最小，即 90%为最合适的彻底恢复时的效率。此时，解出 k 等于 0.304。

因此，可以得到效率关于停留天数的函数为：

$$P = 1-(1-P_0)e^{-0.304t}$$

2）具体恢复时间的计算

在假设中，我们提出由于资金有限，为了使参会者在举行会议时工作效率最高，最好的方法便是让他们提前几天到达城市进行休息，适应时差、并缓解旅行疲劳。因此，除去机票、和进行会议时三天酒店的费用，其他所有预算讲用于提前抵达城市、调整时差的酒店费用。那么参会者提前抵达酒店的天数为：

$$\mathrm{T}_{\text{休}} = \frac{M - \sum \mathrm{M}_{\text{飞机}} - 3\sum \mathrm{M}_{\text{酒店}}}{\sum \mathrm{M}_{\text{酒店}}}$$

得出的 $T_{休}$即为会议前参会者休息的天数。

飞机票价和飞行距离成正比关系。在“Airline fare analysis: comparing cost per mile”这篇文章中，记录了各大航空公司不同飞行距离下的飞机票价[10]("Airline Fare Analysis: Comparing Cost Per Mile - Rome2rio")：

$$\mathrm{M}_{\text{飞机}} = \$50 + (L \times \$0.11)$$

在实际运算中，对于某些城市我们有时会得到小于零的休息天数，这意味着组委会提供的预算不足以支付从发城市到当地的机票以及会议期间三天的酒店花费，也意味着应将此城市剔出候选城市范围。若所有城市的休息天数皆小于零，则意味着组委会的预算不足以举办此次会议，需要考虑增大预算数额以支付必要成本。

3. 灵敏性分析

依据上述的模型，恢复时间对于最终参会者工作效率的影响显著，而其中，上述 Waterhouse 提出的需要花费跨越时区数的三分之二的天数来恢复效率的经验结果的准确性又对我们模型得出结果尤为重要，因此我们将做如下的灵敏度分析。
在模型求解中我们采用了 2/3，下面我们分别采用 1/2，2/3，1 进行求解，并且令彻底恢复时效率依旧等于 90%，得到了如下的 k 值：

1/2	2/3	1
0.4053	0.3040	0.2027

表五 不同经验描述下的 k 值

将 k 分别带入恢复子模型以及最终使用总模型分析时，得出了如下的结果（出发城市分别为 Sapporo、Ulan Bator、Almaty、Gdansk、Strasburg、Burgundy、Brighton、Boston）：

恢复时间（天/时区）	k	第一选择	第二选择	第三选择	第四选择	第五选择
1/2	0.4053	Krakow	Brussels	Valencia	Istanbul	Berlin
2/3	0.3040	Krakow	Brussels	Valencia	Istanbul	Berlin
1	0.2027	Krakow	Brussels	Valencia	Istanbul	Berlin

表六 不同 k 值下的目标城市结果

可以看到即使经验结果在一个较大的范围内浮动（1/2～1），即使对于最终城市分数的计算有数值上的影响，但由于 k 的变化对所有城市分数数值带来的变化趋势相同，因而对于最终得出的城市结果的前几名依旧影响不显著。由此得出最终模型在跨越时区数以及需要恢复效率所花费天数的比例经验结果方面是稳健的。

4. 模型评价

1）本子模型根据过往研究中的实验资料和经验数据对恢复程度进行评估，能够合理模拟并预测恢复趋势。

2）在本模型中，时差和旅途劳累带来的效率减损可以被量化为一个数值，且这个值随时间的变化趋势只和自身大小有关而不和特定时间和特定时差或疲劳程度有关，此特点大大简化了本模型的运算时间。

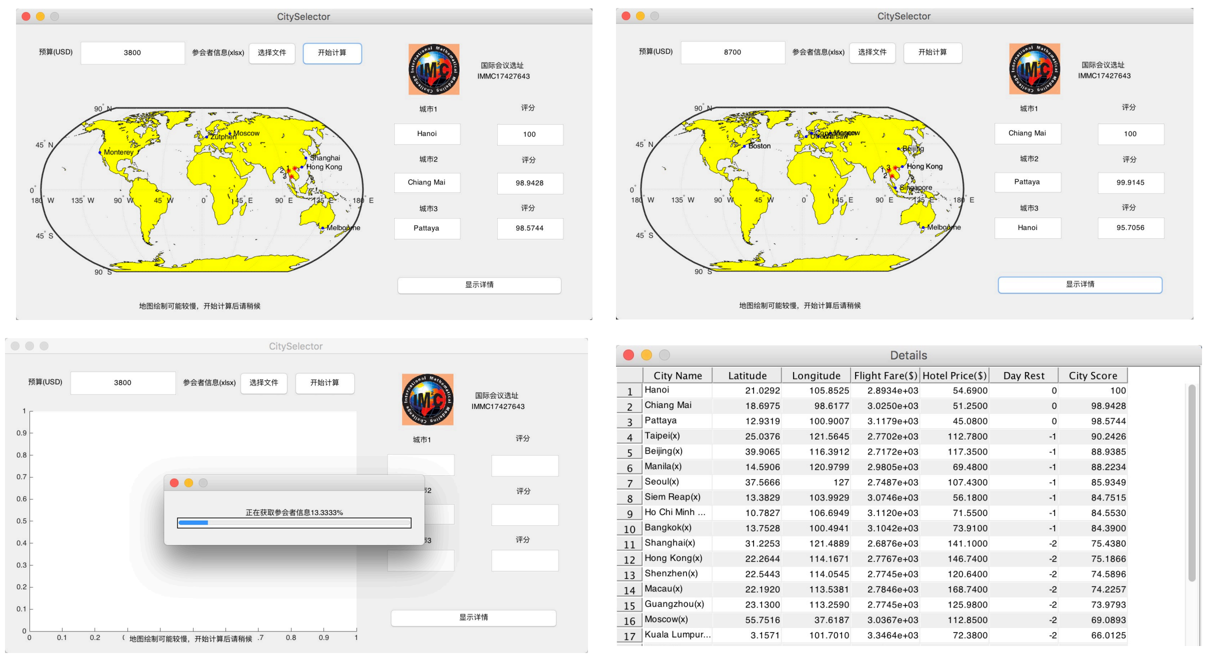
3）恢复场的 k 值是在剔除了浮动极端大的点后平均了所有时区的 k 值求得的，用该 k 值进行预测可减少恢复时间极端而带来的模型不准确的情况。

5. 后续改进方向

1）本文在求定恢复场的 k 值时是根据“三分之二时区”的恢复时长这一经验值进行求解。并且自定义了恢复的百分比（90%）。这一不精准评价降低了模型的精准预估能力。

八、结果分析

1.图形界面



预算(USD) 3800

城市1	评分
Hanoi	100
城市2	**评分**
Chiang Mai	98.9428
城市3	**评分**
Pattaya	98.5744

预算(USD) 8700

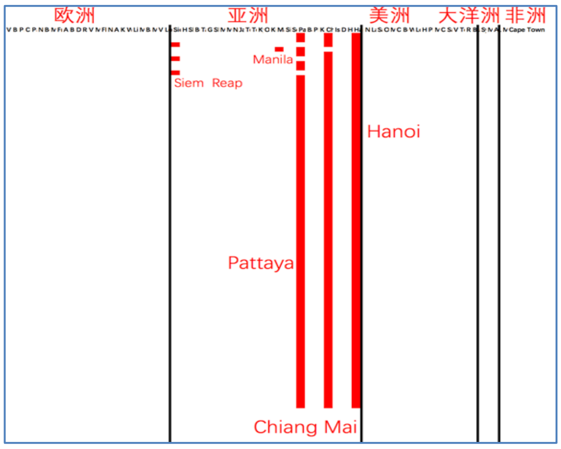
城市1	评分
Chiang Mai	100
城市2	**评分**
Pattaya	99.9145
城市3	**评分**
Hanoi	95.7056

	City Name	Latitude	Longitude	Flight Fare($)	Hotel Price($)	Day Rest	City Score
1	Hanoi	21.0292	105.8525	2.8934e+03	54.6900	0	100
2	Chiang Mai	18.6975	98.6177	3.0250e+03	51.2500	0	98.9428
3	Pattaya	12.9319	100.9007	3.1179e+03	45.0800	0	98.5744
4	Taipei(x)	25.0376	121.5645	2.7702e+03	112.7800	-1	90.2426
5	Beijing(x)	39.9065	116.3912	2.7172e+03	117.3500	-1	88.9385
6	Manila(x)	14.5906	120.9799	2.9805e+03	69.4800	-1	88.2234
7	Seoul(x)	37.5666	127	2.7487e+03	107.4300	-1	85.9349
8	Siem Reap(x)	13.3829	103.9929	3.0746e+03	56.1800	-1	84.7515
9	Ho Chi Minh ...	10.7827	106.6949	3.1120e+03	71.5500	-1	84.5530
10	Bangkok(x)	13.7528	100.4941	3.1042e+03	73.9100	-1	84.3900
11	Shanghai(x)	31.2253	121.4889	2.6876e+03	141.1000	-2	75.4380
12	Hong Kong(x)	22.2644	114.1671	2.7767e+03	146.7400	-2	75.1866
13	Shenzhen(x)	22.5443	114.0545	2.7745e+03	120.6400	-2	74.5896
14	Macau(x)	22.1920	113.5381	2.7846e+03	168.7400	-2	74.2257
15	Guangzhou(x)	23.1300	113.2590	2.7745e+03	125.9800	-2	73.9793
16	Moscow(x)	55.7516	37.6187	3.0367e+03	112.8500	-2	69.0893
17	Kuala Lumpur...	3.1571	101.7010	3.3464e+03	72.3800	-2	66.0125

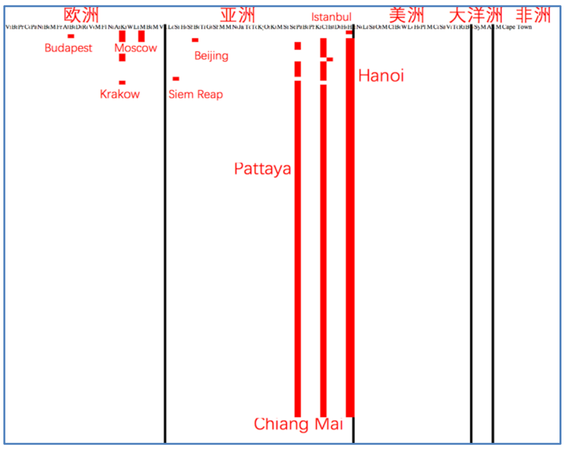
图八 图形界面

为了方便用户使用，我们在模型和算法的基础上设计了图形界面。用户只需要通过 xlsx 文件传入参会者城市信息和预算金额，即可输出模型计算出的推荐排名前三的城市和地图上相应的地理位置。全数据库的评分排名情况可以在“显示详情”按钮中查看。当预算金额为最小值时（即只有三个城市的费用小于总预算金额时），数据集 1 中推荐的城市是 Hanoi, Chiang Mai, Pattaya；数据集 2 中推荐的城市是 Chiang Mai, Pattaya, Hanoi。

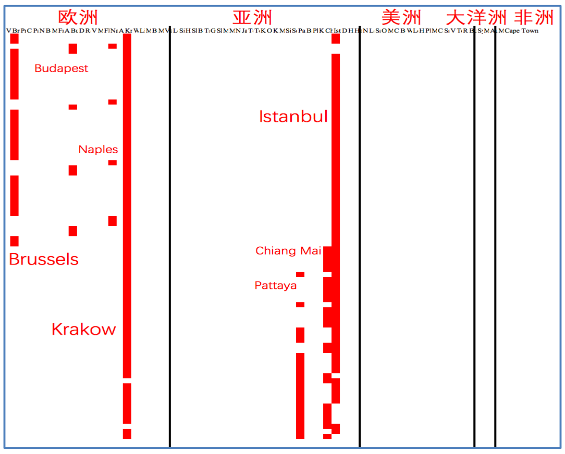
2.特定参会者列表在不同预算下的的模型输出分析



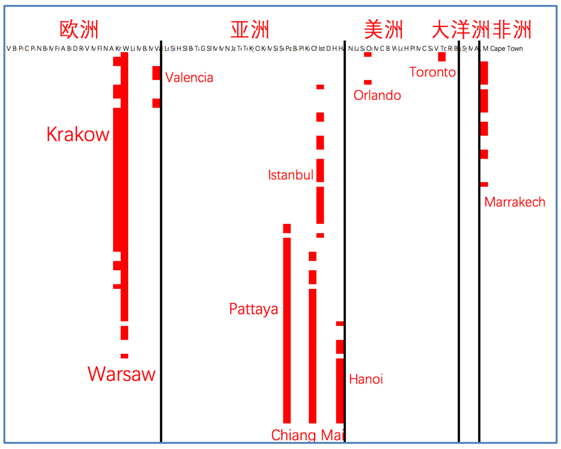
题中测试 1



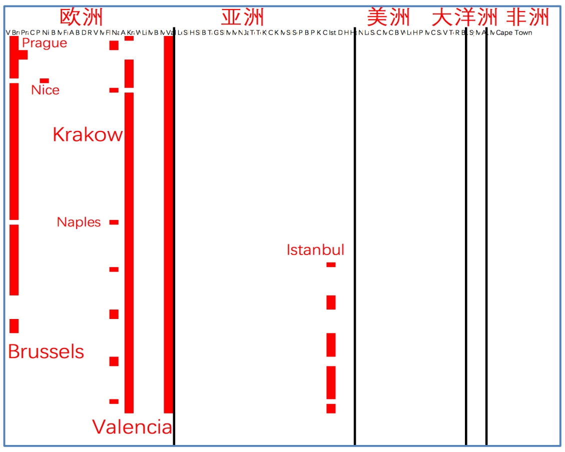
题中测试 2



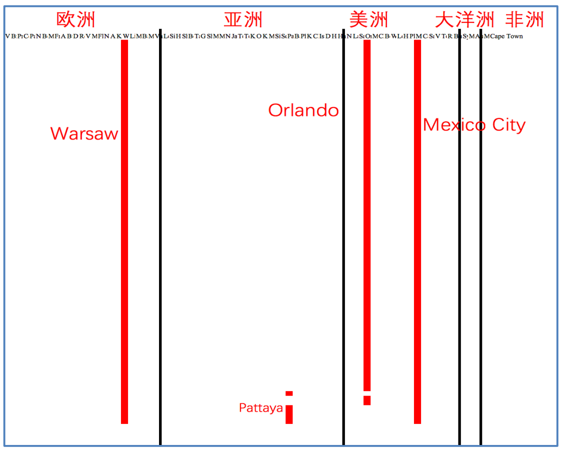
欧洲-俄罗斯-中国



美洲-非洲



欧洲



美洲

图九 特定参会者列表在不同预算下的的模型输出分析

上图为题目中包含的参会者数据集和我们自行生成的数据集（包括参会者分布在欧洲、美洲、中国-美国等其他五个情形）的结果测试。测试分析了同一参会者列表在不同预算下评分前三名的城市的分布与变化。每个图的横向为 77 个目标城市列表（按全球地区排序），纵向为预算金额（步伐间隔为 100 美元，顶端为最小预算，向下依次递

增），每个预算选取三个评分最高的城市标记在相应的城市下。每种参会者情形的具体结果与分析请参见附录。

由这些分析图可以得出如下十分符合直觉的结论：在预算较小时，所得出的推荐城市倾向于来自和所有参会者城市距离短、时差小的城市，因为机票在小预算中相比酒店会占据更大比例，而且时差带来的影响相比酒店的休息时间所恢复的效率影响更大；在预算逐渐增大时，所允许的酒店休息天数增大，使得酒店价格低廉的城市的恢复效果抵消了长途飞行带来的时差和疲劳。因此，在大预算时，我们可以看到推荐城市结果逐渐向东南亚的 Pattaya、Chiang Mai、 Hanoi 以及东欧的 Krakow 转移。

九、参考资料

[1] Arendt, J, and V Marks. "Physiological Changes Underlying Jet Lag.". BMJ 284.6310 (1982): 144-146. Web.

[2] Waterhouse, Jim et al. "Jet Lag: Trends And Coping Strategies". The Lancet 369.9567 (2007): 1117-1129. Web.

[3] Borbély, Alexander A., and Peter Achermann. "Sleep Homeostasis And Models Of Sleep Regulation". Institute of Pharmacology and Toxicology, University of Zürich Winterthurerstr, 150, CH-8057 Zürich, Switzerland (1999)

[4] Bagheri, Neda, Jorg Stelling, and Francis Doyle. "Optimal Phase-Tracking Of The Nonlinear Circadian Oscillator". American Control Conference June 8-10, 2005. Portland, OR, USA (2005)

[5] Jewett, Megan E., and Richard E. Kronauer. "Interactive Mathematical Models Of Subjective Alertness And Cognitive Throughput In Humans". Journal of Biological Rhythms 14.6 (1999): 588-597. Web.

[6] ÄKERSTEDT, TORBJÖRN, and SIMON FOLKARD. "Predicting Sleep Latency From The Three-Process Model Of Alertness Regulation". Psychophysiology 33.4 (1996): 385-389. Web.

[7] Powell, David et al. "Fatigue In Two-Pilot Operations: Implications For Flight And Duty Time Limitations". Aviation, Space, and Environmental Medicine 79.11 (2008): 1047-1050. Web.

[8] "Data Science Toolkit". *Datasciencetoolkit.org*. N.p., 2017. Web. 4 Apr. 2017.

[9] "Timezonedb". *TimeZoneDB*. N.p., 2017. Web. 4 Apr. 2017.

[10] "Airline Fare Analysis: Comparing Cost Per Mile - Rome2rio". *Rome2rio*. N.p., 2017. Web. 4 Apr. 2017.

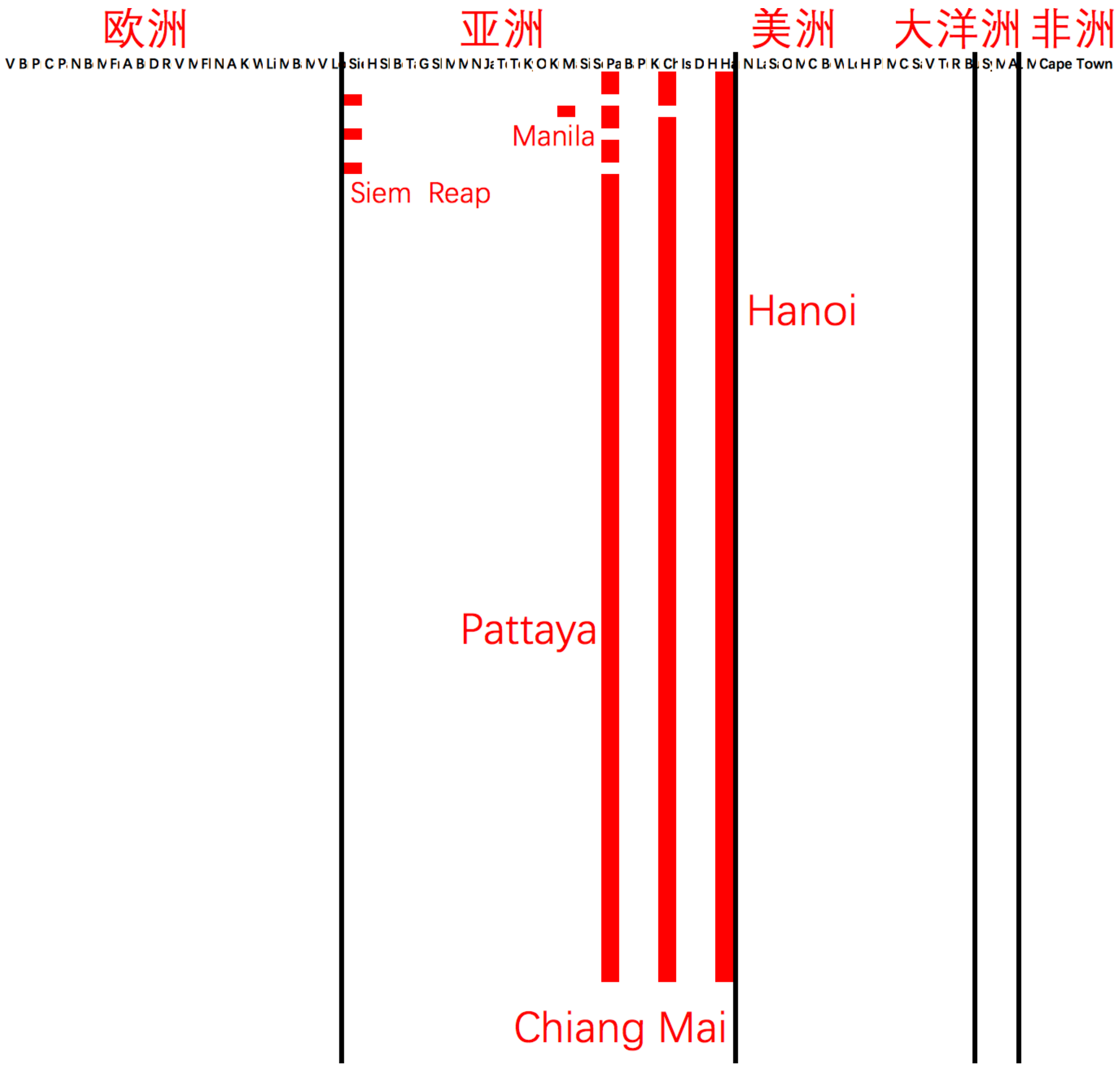
[11] Dean, Dennis A., Daniel B. Forger, and Elizabeth B. Klerman. "Taking The Lag Out Of Jet Lag Through Model-Based Schedule Design". PLoS Computational Biology 5.6 (2009): e1000418. Web.

[12]"Hotels.Com - Cheap Hotels With Secret Prices & Free* Nights". Hotels.com. N.p., 2017. Web. 4 Apr. 2017.

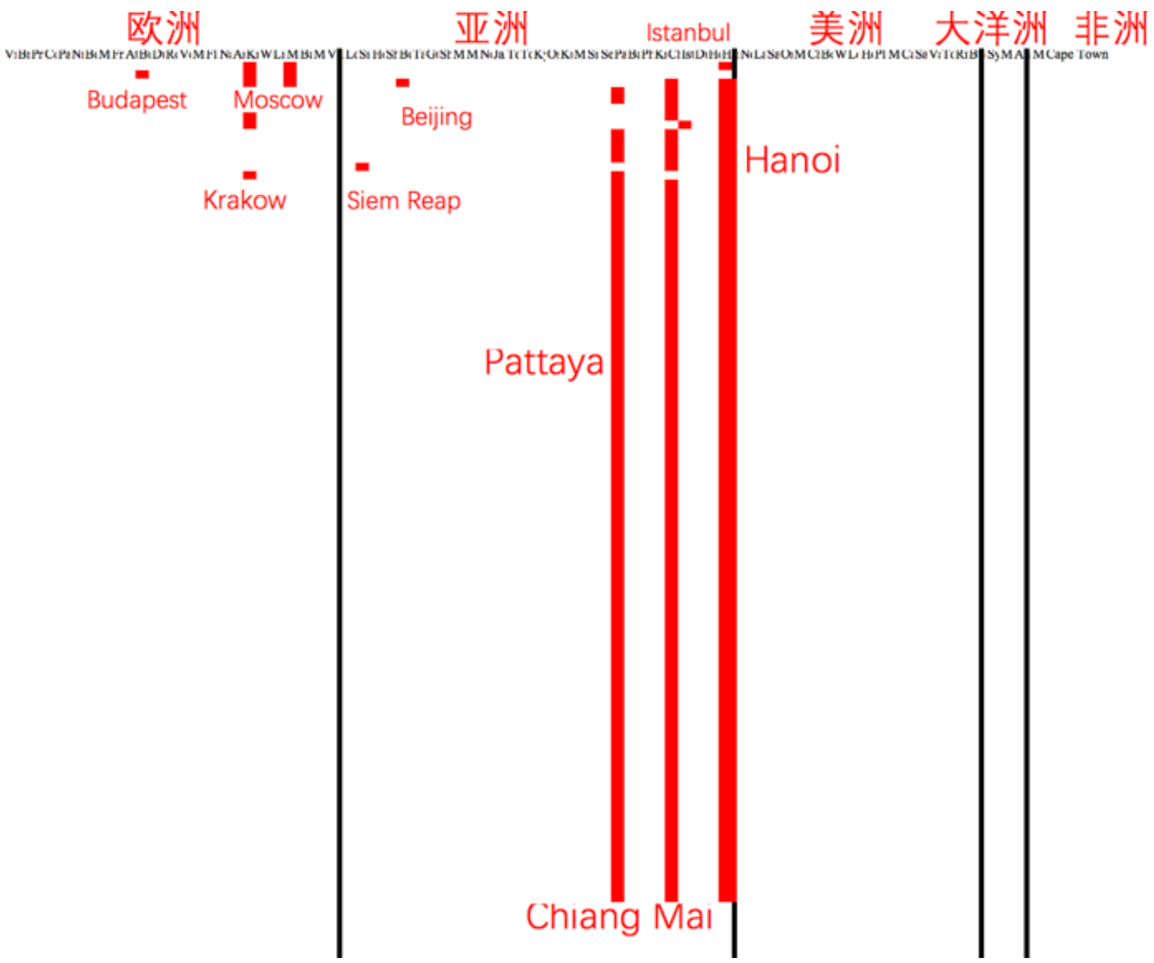
[13] "Defining, Measuring, And Predicting Fatigue". N.p., 2017. Web.

十、附录

1.结果分析—参会者信息相同时资金波动对城市选择影响

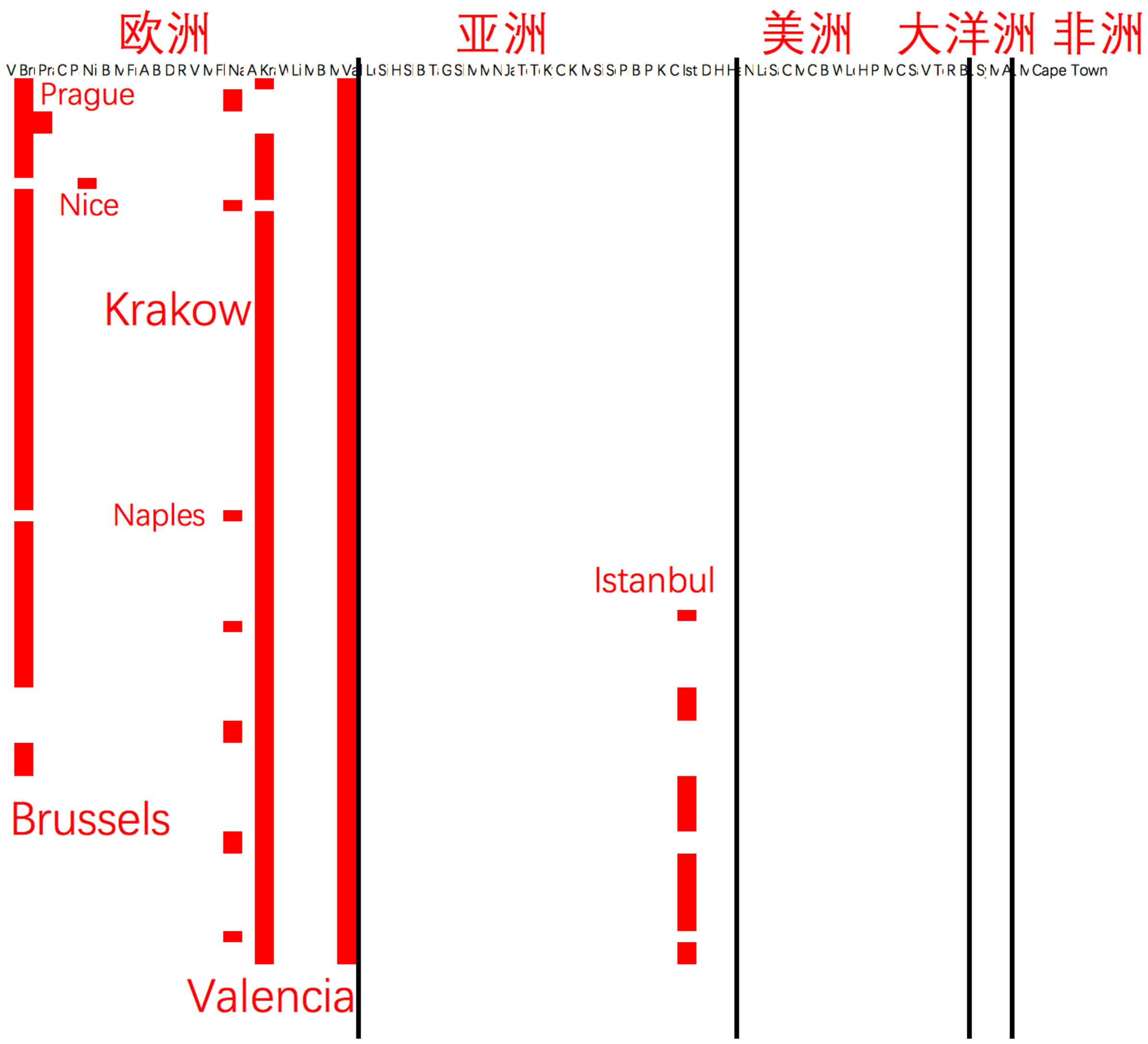


【题中测试数据1】International Small Meeting
六名参会成员分别来自Monterey, Zutphen, Melbourne, Shanghai, Hong Kong, Moscow这六个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为3800美元，低于3800美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为11700美元。



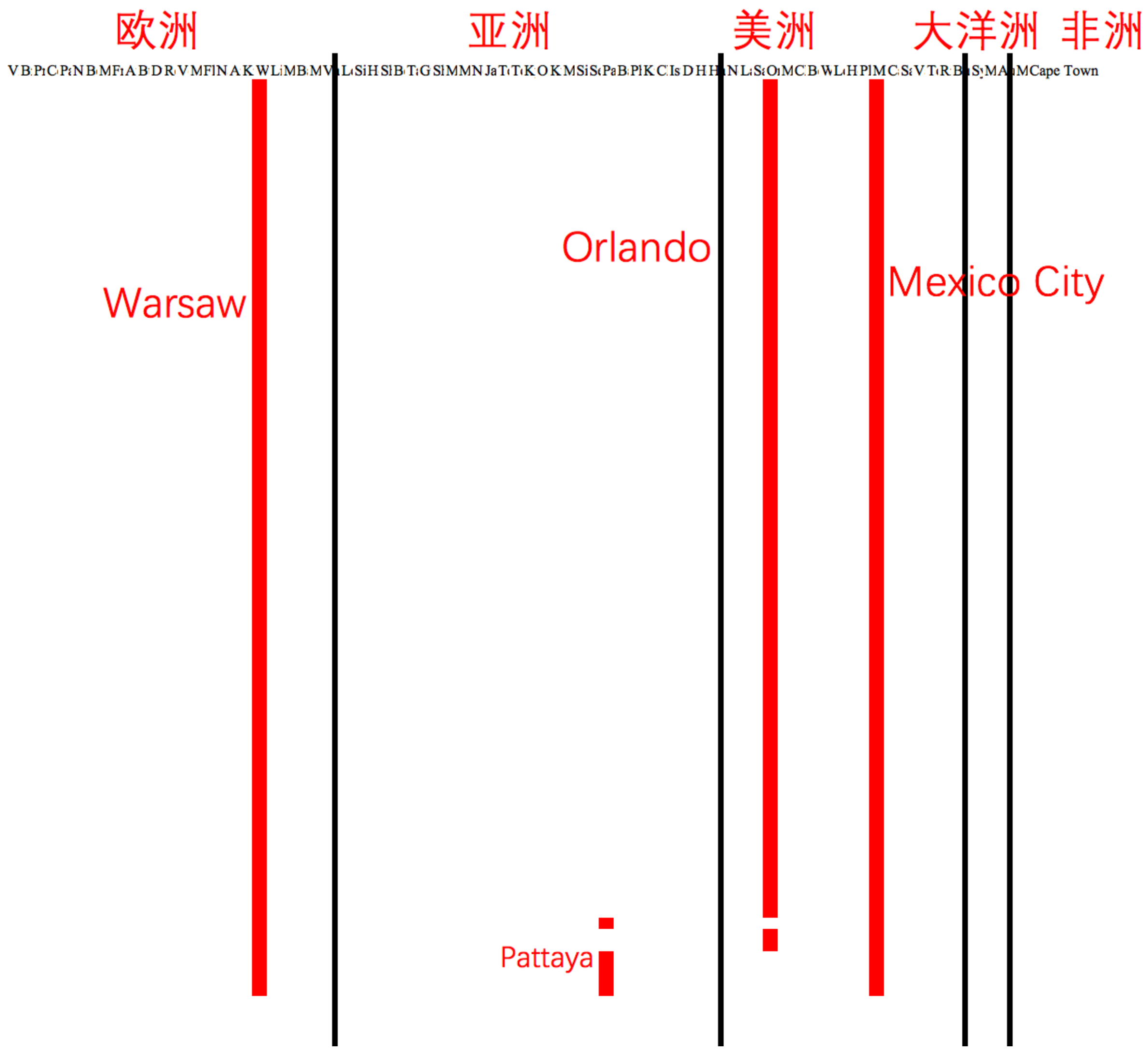
【题中测试数据2】International Big Meeting

十一名参会成员分别来自Boston(有两位成员), Singapore, Beijing, Hong Kong(有两位成员), Moscow, Utrecht, Warsaw, Copenhagen, Melbourne 这九个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为8700美元，低于8700美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为16100美元。

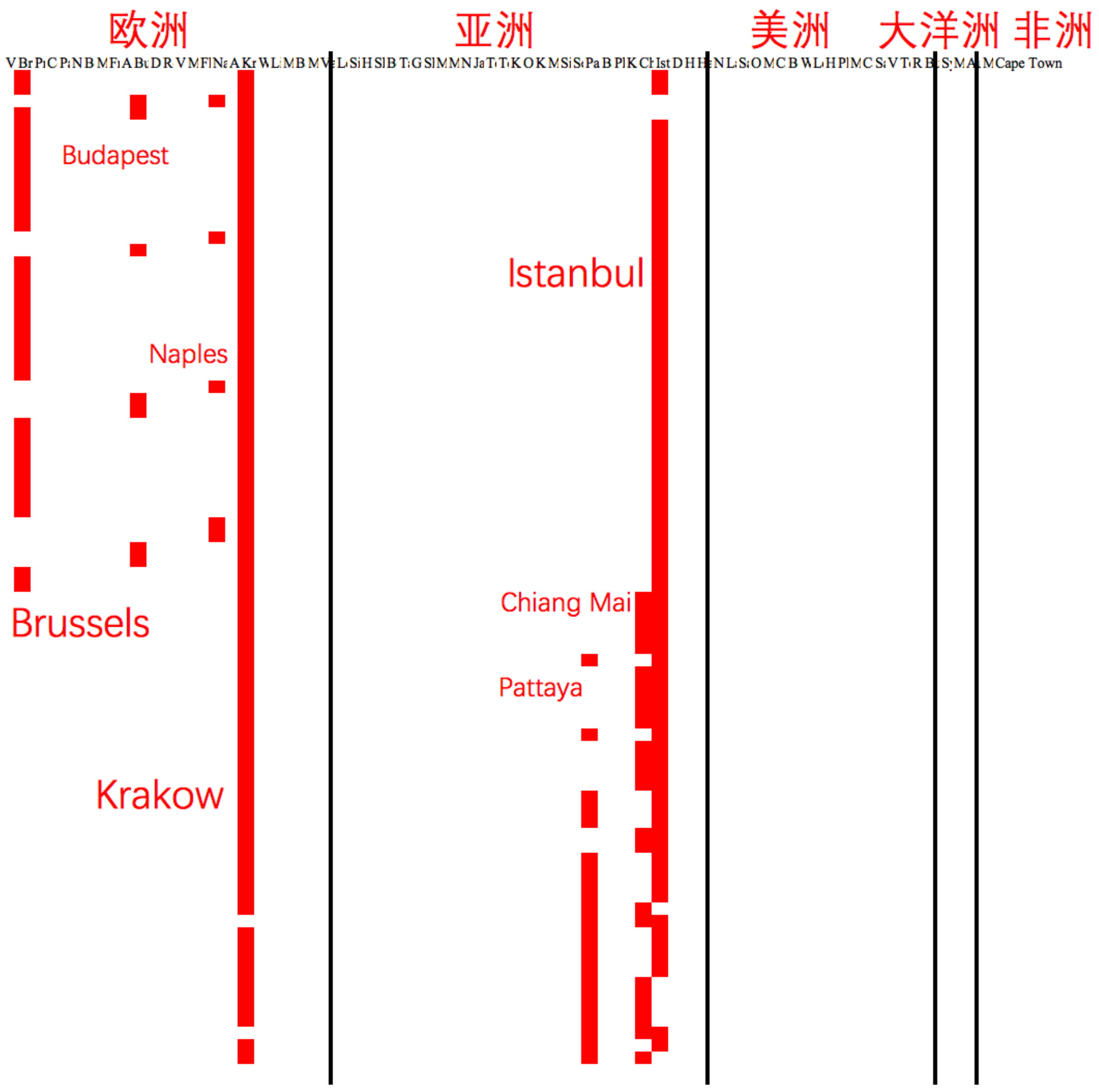


【欧洲测试数据】

十名参会成员分别来自Oslo, Helsinki, Strasburg, Cannes, Monaco, Burgundy, Bath, Glasgow, Rome, Ibiza这十个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为3700美元，低于3700美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为11600美元。

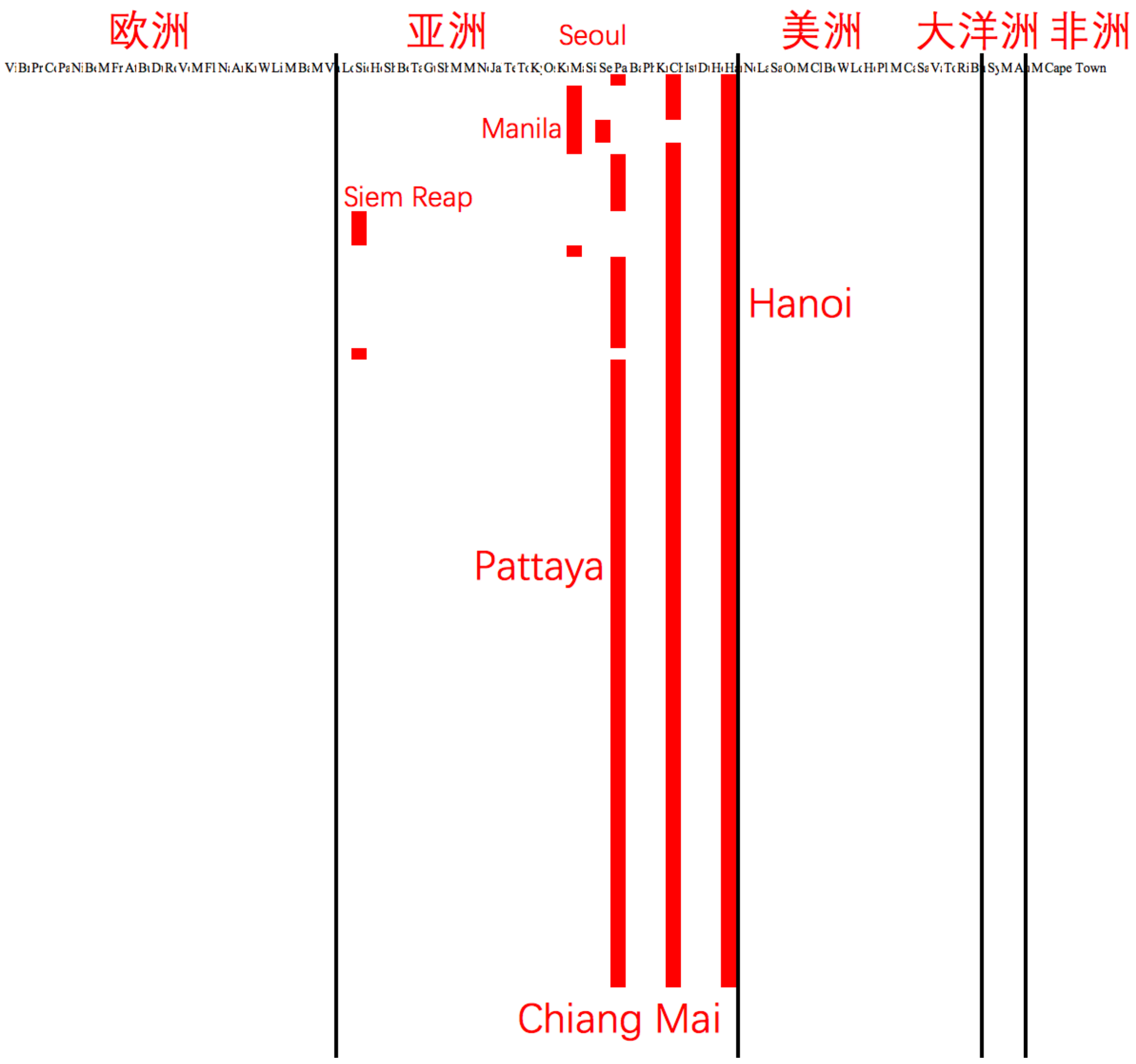


【美洲测试数据】
七名参会成员分别来自Bogota, Cancun, Buenos Aires, Belize, Panama City, Boston, Anchorage这七个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为4900美元，低于4900美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为13000美元。

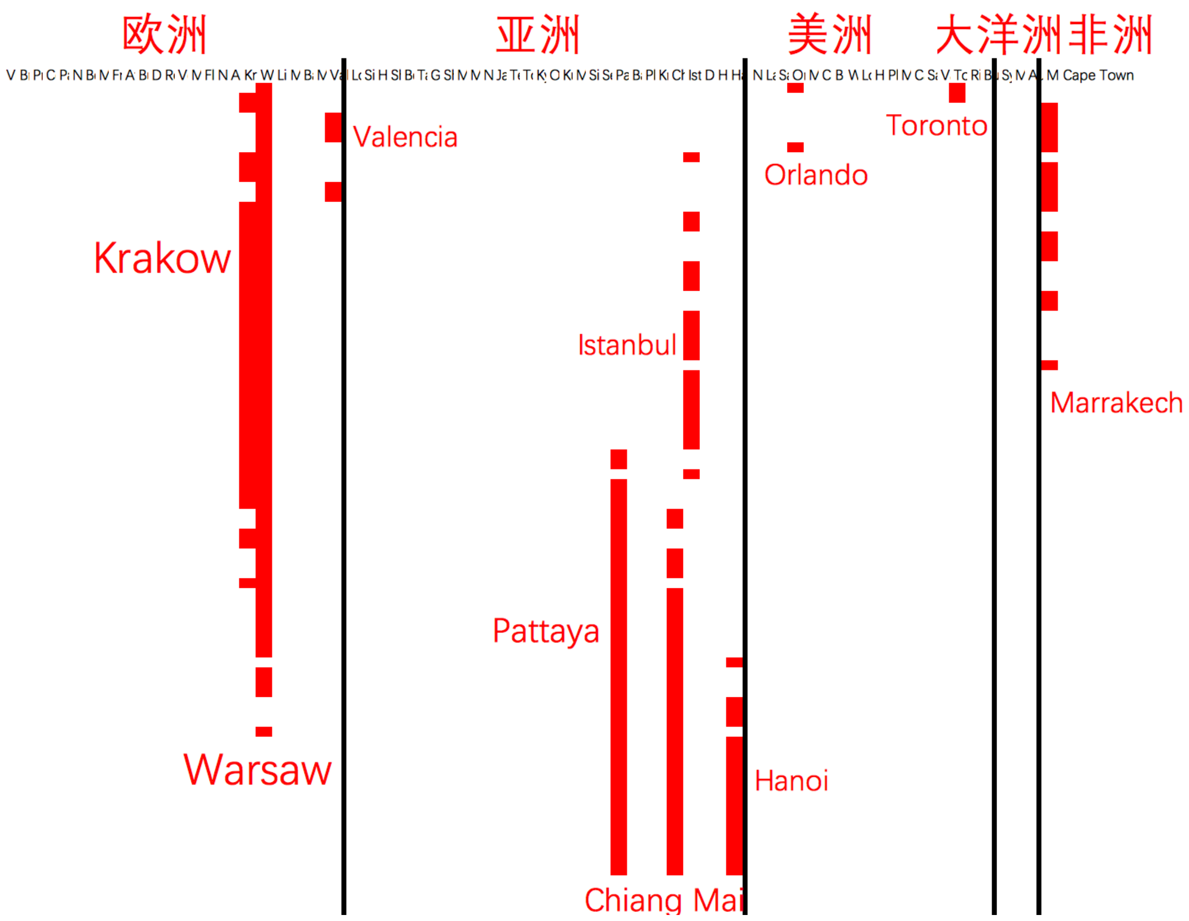


【欧洲-俄罗斯-中国测试数据】

十二名参会成员分别来自Vladivostok, Gdansk, Reims, Brighton, Versailles, Harbin, Malta, Nicosia, Novgorod, Athens, Shenyang, Ulan Bataar这十二个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为6000美元，低于6000美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为13900美元。



【中国-美国测试数据】
二十名参会成员分别来自Beijing, Chongqing, Shenzhen(有两位成员), Shanghai, Tianjin, Guangzhou, Hong Kong(有两位成员), Taipei, Wuxi, Boston, New York, New Haven, Dayton, Palo Alto, San Francisco, Chicago, Philadelphia, Detroit这十八个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为13700美元，低于13700美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为21600美元。



【美洲-非洲测试数据】

七名参会成员分别来自Addis Abeba, Kinsasha, Sao Tome, Johannesburg, Toronto, Washington DC, Talahasee这七个城市。横坐标为77个目标城市列表（按大陆排序），纵坐标为预算金额（单位为100美元），每个预算选取三个评分最高的城市画在图中。其中最小值预算为5600美元，低于5600美元将会有少于三个城市满足机票及酒店的最低资金需求。图中最大预算为13600美元。

2.城市信息

	Name	Time Zone	Airport ID	Lat	Lng	Hotel Price
1	Vienna	2	'VIE'	48.192309	16.371364	126.84
2	Brussels	2	'BRUS'	50.838534	4.375434	100.36
3	Prague	2	'PRG'	50.08744	14.421256	108.65
4	Copenhagen	2	'CPH'	55.686724	12.570069	173.24
5	Paris	2	'PARI'	48.856506	2.352133	156.19
6	Nice	2	'NCE'	43.700936	7.268391	136.68
7	Berlin	2	'BERL'	52.517037	13.38886	114.05
8	Munich	2	'MUC'	48.13641	11.57753	151.98

9	Frankfurt am Main	2	'FRAN'	50.110653	8.682093	136.23
10	Athens	3	'ATH'	37.984149	23.727984	117.93
11	Budapest	2	'BUD'	47.498382	19.040471	100.38
12	Dublin	1	'DUB'	53.349307	-6.261175	156.76
13	Rome	2	'ROME'	41.893344	12.483072	133.58
14	Venice	2	'VENI'	45.437191	12.33459	175.14
15	Milan	2	'MILA'	45.466797	9.190498	152.95
16	Florence	2	'FLOR'	43.769871	11.255576	145.98
17	Naples	2	'NAPL'	40.844116	14.2423	95.01
18	Amsterdam	2	'AMS'	52.371009	4.900112	162.9
19	Krakow	2	'KRK'	50.061947	19.936856	77.3
20	Warsaw	2	'WARS'	52.2333	21.0166	83.64
21	Lisbon	1	'LIS'	38.713057	-9.138006	120.51
22	Moscow	3	'MOSC'	55.751634	37.618704	112.85
23	Barcelona	2	'BCN'	41.38256	2.177135	154.1
24	Madrid	2	'MAD'	40.416705	-3.703583	121.45
25	Valencia	2	'VLC'	39.4666	0.375	94.18
26	London	0	'LOND'	51.507276	-0.12766	192.52
27	Siem Reap	7	'REP'	13.38285	103.992885	56.18
28	Hong Kong (SAR)	8	'LBP'	22.264412	114.167061	146.74
29	Shanghai	8	'CSHA'	31.225344	121.488892	141.1
30	Beijing	8	'BJSA'	39.90647	116.391195	117.35
31	Taipei	8	'TPET'	25.037638	121.564459	112.78
32	Guangzhou	8	'CAN'	23.130004	113.259001	125.98
33	Shenzhen	8	'SZX'	22.544267	114.054533	120.64
34	Macau (SAR)	8	'RUS'	22.191951	113.538122	168.74
35	Mumbai	5.5	'BOM'	18.95238	72.832711	143.57
36	New Delhi	5.5	'DEL'	28.613897	77.215956	104.72
37	Jakarta	7	'CGKI'	-6.181015	106.828335	87.04
38	Tel Aviv	3	'TELA'	32.080481	34.780527	182.63
39	Tokyo	9	'TYOA'	35.680071	139.768522	155.88
40	Kyoto	9	'ITM'	35.01858	135.763835	162.24
41	Osaka	9	'OSAA'	34.685293	135.514694	135.91
42	Kuala Lumpur	8	'KULM'	3.157098	101.700953	72.38
43	Manila	8	'MNL'	14.590607	120.979901	69.48
44	Singapore	8	'SIN'	1.290453	103.852038	152.32
45	Seoul	9	'SELA'	37.566601	127	107.43
46	Pattaya	7	'UTP'	12.931859	100.900691	45.08
47	Bangkok	7	'BKKT'	13.752753	100.494086	73.91
48	Phuket	7	'HKT'	7.715984	98.322126	84.24

49	Krabi	7	'KBV'	7.841388	98.987392	83.64
50	Chiang Mai	7	'CNX'	18.697495	98.617712	51.25
51	Istanbul	3	'ISTA'	41.017058	28.985568	75.74
52	Dubai	4	'DXBA'	25.268352	55.296196	171.95
53	Ho Chi Minh City	7	'SGN'	10.782745	106.694898	71.55
54	Hanoi	7	'HAN'	21.02921	105.85247	54.69
55	New York	-4	'NYCA'	40.713054	-74.007228	200.95
56	Las Vegas	-7	'LAS'	36.169202	-115.140597	133.71
57	San Francisco	-7	'SFO'	37.78008	-122.420168	213.04
58	Orlando	-4	'ORLB'	28.538331	-81.378879	120.25
59	Miami	-4	'MIAA'	25.775084	-80.194702	189.52
60	Chicago	-5	'CHIA'	41.883229	-87.632398	196.93
61	Boston	-4	'BOS'	42.358894	-71.056742	251.16
62	Washington DC	-4	'WASA'	38.892062	-77.019912	201.57
63	Los Angeles	-7	'LAX'	34.052238	-118.243344	177.02
64	Honolulu	-10	'HNL'	21.30992	-157.858158	231.11
65	Playa del Carmen	-5	'CUN'	20.628223	-87.075658	212.1
66	Mexico City	-5	'MEX'	19.43253	-99.13321	109.81
67	Cancun	-5	'CUN'	21.170891	-86.840317	222.54
68	Santiago	-3	'SCL'	-33.437913	-70.650456	113
69	Vancouver	-7	'YVRA'	49.263566	-123.138572	171.98
70	Toronto	-4	'YTOA'	43.651893	-79.381713	146.52
71	Rio de Janeiro	-3	'RIOA'	-22.911014	-43.209373	128.56
72	Buenos Aires	-3	'BUEA'	-34.612869	-58.445979	108.62
73	Sydney	10	'SYD'	-33.854816	151.216454	158.21
74	Melbourne	10	'MELA'	-37.814255	144.963146	125.19
75	Auckland	12	'AKLN'	-36.541281	174.550609	124.54
76	Marrakech	1	'RAK'	31.62599	-7.988608	93.9
77	Cape Town	2	'CPT'	-33.928905	18.417249	144.88

3.题目给定情况结果

3.1 情形 1

预算为 3800USD 时（即最低可能预算），数据库中 77 个城市的信息及分数

City Name	Lat	Lon	Flight Cost (USD)	Hotel Price (USD)	Day Rest	City Score
'Hanoi'	21.02921	105.85247	2893.365664	54.69	0	100

'Chiang Mai'	18.697495	98.617712	3024.971961	51.25	0	98.94280978
'Pattaya'	12.931859	100.900691	3117.939893	45.08	0	98.57440734
'Taipei'	25.037638	121.564459	2770.204382	112.78	-1	90.24256213
'Beijing'	39.90647	116.391195	2717.234871	117.35	-1	88.93852536
'Manila'	14.590607	120.979901	2980.513043	69.48	-1	88.22336644
'Seoul'	37.566601	127	2748.702748	107.43	-1	85.93493682
'Siem Reap'	13.38285	103.992885	3074.566789	56.18	-1	84.75147577
'Ho Chi Minh City'	10.782745	106.694898	3112.032541	71.55	-1	84.55303354
'Bangkok'	13.752753	100.494086	3104.172701	73.91	-1	84.38998306
'Shanghai'	31.225344	121.488892	2687.622395	141.1	-2	75.43795098
'Hong Kong'	22.264412	114.167061	2776.674262	146.74	-2	75.18659456
'Shenzhen'	22.544267	114.054533	2774.534202	120.64	-2	74.5895619
'Macau'	22.191951	113.538122	2784.60805	168.74	-2	74.22567843
'Guangzhou'	23.130004	113.259001	2774.467187	125.98	-2	73.97933808
'Moscow'	55.751634	37.618704	3036.673604	112.85	-2	69.08933391
'Kuala Lumpur'	3.157098	101.700953	3346.368512	72.38	-2	66.01251725
'Osaka'	34.685293	135.514694	2846.539942	135.91	-2	65.83214657
'Kyoto'	35.01858	135.763835	2848.190087	162.24	-2	65.76010515
'Tokyo'	35.680071	139.768522	2895.388504	155.88	-2	64.91963804
'Berlin'	52.517037	13.38886	3337.620662	114.05	-2	63.96415006
'Krabi'	7.841388	98.987392	3257.603974	83.64	-2	63.62775159
'Brussels'	50.838534	4.375434	3474.453003	100.36	-2	63.5836447
'Phuket'	7.715984	98.322126	3267.803522	84.24	-2	63.51234314
'Krakow'	50.061947	19.936856	3314.112804	77.3	-2	63.49520431
'Prague'	50.08744	14.421256	3372.291619	108.65	-2	63.4831541
'Istanbul'	41.017058	28.985568	3403.642588	75.74	-2	63.07399766
'Vienna'	48.192309	16.371364	3390.977885	126.84	-2	62.99625863
'Budapest'	47.498382	19.040471	3375.800191	100.38	-2	62.86406567
'New Delhi'	28.613897	77.215956	3125.061216	104.72	-2	59.96262088
'Mumbai'	18.95238	72.832711	3350.37474	143.57	-2	58.2931658
'Singapore'	1.290453	103.852038	3375.543951	152.32	-3	39.92600958
'Amsterdam'	52.371009	4.900112	3431.566343	162.9	-3	37.87609273
'Copenhagen'	55.686724	12.570069	3303.27679	173.24	-3	37.4646611
'Frankfurt am Main'	50.110653	8.682093	3433.558553	136.23	-3	36.98831513
'Munich'	48.13641	11.57753	3447.232882	151.98	-3	35.99337406
'Paris'	48.856506	2.352133	3558.701528	156.19	-3	35.1093784
'Venice'	45.437191	12.33459	3503.706552	175.14	-3	34.80193007
'Athens'	37.984149	23.727984	3536.983189	117.93	-3	34.67304889
'Milan'	45.466797	9.190498	3544.826311	152.95	-3	34.55346784
'Florence'	43.769871	11.255576	3559.947415	145.98	-3	33.96804243
'Jakarta'	-6.181015	106.828335	3549.430999	87.04	-3	33.84233895

'Valencia'	44.933228	4.892081	3621.915088	94.18	-3	33.6954623
'Nice'	43.700936	7.268391	3618.553334	136.68	-3	33.47927506
'Tel Aviv'	32.080481	34.780527	3535.779841	182.63	-3	33.46436675
'Rome'	41.893344	12.483072	3590.651642	133.58	-3	33.2680449
'Naples'	40.844116	14.2423	3592.803865	95.01	-3	32.99304202
'Barcelona'	41.38256	2.177135	3760.066628	154.1	-3	31.60932788
'Dubai'	25.268352	55.296196	3439.055224	171.95	-3	31.3675768
'Madrid'	40.416705	-3.703583	3885.091498	121.45	-3	30.05309974
'London'	51.507276	-0.12766	3538.656914	192.52	-3	29.14863611
'Dublin'	53.349307	-6.261175	3603.60511	156.76	-3	27.61420321
'Lisbon'	38.713057	-9.138006	4024.897212	120.51	-3	22.74438365
'San Francisco'	37.78008	-122.420168	3859.658238	213.04	-3	15.38382699
'Los Angeles'	34.052238	-118.243344	3997.646301	177.02	-3	13.44326836
'Las Vegas'	36.169202	-115.140597	3996.479435	133.71	-3	12.97329894
'Vancouver'	49.263566	-123.138572	3709.557643	171.98	-3	12.79700003
'Honolulu'	21.30992	-157.858158	3898.850427	231.11	-3	10.83817729
'Toronto'	43.651893	-79.381713	4228.345513	146.52	-3	6.812314147
'Boston'	42.358894	-71.056742	4312.285913	251.16	-3	6.250009475
'New York'	40.713054	-74.007228	4339.702005	200.95	-3	6.078862498
'Chicago'	41.883229	-87.632398	4207.153168	196.93	-3	5.982325239
'Washington DC'	38.892062	-77.019912	4368.075043	201.57	-3	5.90843336
'Melbourne'	-37.814255	144.963146	4332.969738	125.19	-4	-5.417770983
'Sydney'	-33.854816	151.216454	4330.237092	158.21	-4	-10.39450492
'Marrakech'	31.62599	-7.988608	4188.299888	93.9	-4	-21.16721499
'Warsaw'	37.958434	-76.755329	4394.393217	83.64	-4	-21.32516415
'Auckland'	-36.541281	174.550609	4792.331201	124.54	-4	-28.56589759
'Orlando'	28.538331	-81.378879	4604.72965	120.25	-4	-43.38724616
'Miami'	25.775084	-80.194702	4687.95176	189.52	-4	-44.17023893
'Mexico City'	19.43253	-99.13321	4645.027822	109.81	-4	-44.84390093
'Cancun'	21.170891	-86.840317	4745.570204	222.54	-4	-45.95555233
'Playa del Carmen'	20.628223	-87.075658	4757.380527	212.1	-4	-46.06319412
'Cape Town'	-33.928905	18.417249	5179.435691	144.88	-5	-91.05251784
'Rio de Janeiro'	-22.911014	-43.209373	5829.191957	128.56	-6	-213.9357596
'Santiago'	-33.437913	-70.650456	6072.992401	113	-6	-217.8606262
'Buenos Aires'	-34.612869	-58.445979	6109.330604	108.62	-7	-345.103913

3.2.情形 2

预算为 8700USD 时（即最低可能预算），数据库中 77 个城市的信息及分数

City Name	Lat	Lon	Flight Cost (USD)	Hotel Price (USD)	Day Rest	City Score
'Hanoi'	21.02921	105.85247	5831.840178	54.69	2	100
'Chiang Mai'	18.697495	98.617712	5974.207423	51.25	2	99.64492781
'Pattaya'	12.931859	100.900691	6195.999775	45.08	2	99.5331499
'Krakow'	50.061947	19.936856	5517.559875	77.3	1	94.24187864
'Istanbul'	41.017058	28.985568	5766.623321	75.74	1	93.11438461
'Siem Reap'	13.38285	103.992885	6161.4672	56.18	1	92.978345
'Beijing'	39.90647	116.391195	5423.986828	117.35	0	86.46392058
'Berlin'	52.517037	13.38886	5487.71339	114.05	0	86.13490396
'Moscow'	55.751634	37.618704	5265.821283	112.85	0	86.03973582
'Brussels'	50.838534	4.375434	5645.239087	100.36	0	85.95037316
'Prague'	50.08744	14.421256	5559.495776	108.65	0	85.78860985
'Budapest'	47.498382	19.040471	5612.299852	100.38	0	85.36602334
'Manila'	14.590607	120.979901	6242.175612	69.48	0	85.15711181
'Valencia'	44.933228	4.892081	5881.522216	94.18	0	84.95044263
'Kuala Lumpur'	3.157098	101.700953	6653.069966	72.38	0	84.52769091
'Naples'	40.844116	14.2423	5924.712795	95.01	0	84.49008548
'Ho Chi Minh City'	10.782745	106.694898	6279.930201	71.55	0	83.83876288
'Bangkok'	13.752753	100.494086	6162.546653	73.91	0	83.79294682
'Seoul'	37.566601	127	5655.792639	107.43	0	82.88653658
'Hong Kong'	22.264412	114.167061	5739.99478	146.74	-1	76.58896546
'Shenzhen'	22.544267	114.054533	5733.651344	120.64	-1	76.16904366
'Macau'	22.191951	113.538122	5745.522365	168.74	-1	75.98597549
'Guangzhou'	23.130004	113.259001	5719.552416	125.98	-1	75.78008223
'Copenhagen'	55.686724	12.570069	5406.415681	173.24	-1	75.51449839
'Amsterdam'	52.371009	4.900112	5581.114358	162.9	-1	74.8918327
'Taipei'	25.037638	121.564459	5830.104619	112.78	-1	74.32304244
'Frankfurt am Main'	50.110653	8.682093	5615.42515	136.23	-1	74.318756
'Shanghai'	31.225344	121.488892	5671.415165	141.1	-1	74.27207811
'Munich'	48.13641	11.57753	5659.382836	151.98	-1	73.87111216
'Vienna'	48.192309	16.371364	5610.280518	126.84	-1	73.81453471
'Paris'	48.856506	2.352133	5759.662095	156.19	-1	73.77367591
'Venice'	45.437191	12.33459	5758.419514	175.14	-1	73.31880261
'Milan'	45.466797	9.190498	5796.437991	152.95	-1	73.29941888
'Florence'	43.769871	11.255576	5840.323724	145.98	-1	72.98144414

'Nice'	43.700936	7.268391	5898.017558	136.68	-1	72.89809552
'Rome'	41.893344	12.483072	5903.080051	133.58	-1	72.63157089
'Barcelona'	41.38256	2.177135	6079.563996	154.1	-1	72.27624381
'Warsaw'	37.958434	-76.755329	6829.204384	83.64	-1	71.99178033
'Madrid'	40.416705	-3.703583	6226.699695	121.45	-1	71.83841335
'Athens'	37.984149	23.727984	5933.423485	117.93	-1	71.48114603
'Krabi'	7.841388	98.987392	6440.034066	83.64	-1	70.86787432
'Phuket'	7.715984	98.322126	6450.577235	84.24	-1	70.79867552
'Dublin'	53.349307	-6.261175	5768.825606	156.76	-1	70.75244804
'Osaka'	34.685293	135.514694	5907.439268	135.91	-1	69.3347902
'Kyoto'	35.01858	135.763835	5905.306615	162.24	-1	69.30865613
'Jakarta'	-6.181015	106.828335	7234.947036	87.04	-1	69.11207887
'Tokyo'	35.680071	139.768522	5982.397804	155.88	-1	68.8722515
'New Delhi'	28.613897	77.215956	5833.657884	104.72	-1	68.75746169
'Lisbon'	38.713057	-9.138006	6403.713495	120.51	-1	68.69901245
'Mumbai'	18.95238	72.832711	6213.804671	143.57	-1	67.80676656
'Marrakech'	31.62599	-7.988608	6698.06965	93.9	-1	67.71844912
'Singapore'	1.290453	103.852038	6737.787134	152.32	-2	56.84857865
'London'	51.507276	-0.12766	5708.455683	192.52	-2	54.36812437
'Tel Aviv'	32.080481	34.780527	6056.708508	182.63	-2	53.7167326
'Dubai'	25.268352	55.296196	6133.693221	171.95	-2	51.60495658
'Boston'	42.358894	-71.056742	6559.186142	251.16	-2	48.74985684
'New York'	40.713054	-74.007228	6669.291461	200.95	-2	46.93633355
'Washington DC'	38.892062	-77.019912	6786.983112	201.57	-2	46.22522951
'Toronto'	43.651893	-79.381713	6619.073899	146.52	-2	45.96647793
'Orlando'	28.538331	-81.378879	7484.049571	120.25	-2	42.68169742
'Miami'	25.775084	-80.194702	7656.263441	189.52	-2	42.07528577
'Chicago'	41.883229	-87.632398	6809.426561	196.93	-2	40.84738331
'Vancouver'	49.263566	-123.138572	6741.422229	171.98	-2	35.01406463
'Las Vegas'	36.169202	-115.140597	7290.221086	133.71	-2	34.19880407
'San Francisco'	37.78008	-122.420168	7223.342498	213.04	-2	34.0095964
'Los Angeles'	34.052238	-118.243344	7391.761564	177.02	-2	33.82316243
'Melbourne'	-37.814255	144.963146	9246.729632	125.19	-3	16.67424959
'Cape Town'	-33.928905	18.417249	9023.623028	144.88	-3	16.54288354
'Sydney'	-33.854816	151.216454	9169.054218	158.21	-3	15.01866374
'Cancun'	21.170891	-86.840317	7985.439328	222.54	-3	8.529330229
'Playa del Carmen'	20.628223	-87.075658	8020.234468	212.1	-3	8.367549127
'Mexico City'	19.43253	-99.13321	8117.020678	109.81	-3	7.035783952
'Honolulu'	21.30992	-157.858158	7680.557839	231.11	-3	2.738638023

'Auckland'	-36.541281	174.550609	9721.3986	124.54	-4	-35.14797338
'Rio de Janeiro'	-22.911014	-43.209373	9859.588921	128.56	-4	-35.89042104
'Buenos Aires'	-34.612869	-58.445979	10550.1875	108.62	-5	-94.87462627
'Santiago'	-33.437913	-70.650456	10646.79134	113	-5	-95.20489152

4.模型中系数结果求解

4.1 时差子模型－求解系数 Mathematica 代码

```
c =.;(*b*)
c1 =.;(*Alpha_a*)

c2 =.;(*Alpha_s*)
s = 8;(*睡眠时长*)
w = 24 - s;(*清醒时长*)
tw = 16;(*Prior Wakefulness*)
a = -2*tw/32/32;
b = 1/2.14;
c1 = Solve[{0.9896 - (0.9896 - c)*E^(-s*b) == 0.1503 + c1*E^(s*a)},
    c1][[1, 1, 2]];
c = Solve[{0.1503 + c1*E^(24*a) == 0.9896 - (0.9896 - c)*E^(0)},
    c][[1, 1, 2]];
c1
c2 = 0.9896  -  c
```

4.2 时差子模型－求解参会者到达目标城市时的时差影响的初始效率 P1 Mathematica 代码

```
d =.;(*稳态函数平移量(计算过程中将稳态函数 S₀平移以增加计算速度，误差较小)*)
n =.;(*时差 delta_n*)
n = 15(*0<=delta_n<24*)
d = If[0 <= n <= 8,
   Solve[{0.654525 == 0.9896 - 0.3350745629030397`*E^(-.467*n) + d},
    d], Solve[{0.654525 == (1.067445258710798`)*
        E^(-2*16/32/32*n) + .153 + d}, d]][[1, 1, 2]]

(Integrate[(1.067445258710798)*E^(-2*16/32/32*x) + .153 + d, {x, 8,
     24}] + If[0 <= n <= 8,
    Integrate[.4*(0.97*Sin[2*Pi/24*(x - 8.6 + Pi/2)] +
        0.22*Sin[2.2*Pi/24*(x - 8.6)] +
        0.07*Sin[2.3*Pi/24*(x - 8.6)] +
        0.03*Sin[2.4*Pi/24*(x - 8.6)] +
        0.001*Sin[2.5*Pi/24*(x - 8.6)]), {x, 8 - n, 24 - n}],
    Integrate[.4*(0.97*Sin[2*Pi/24*(x - 8.6 + Pi/2)] +
         0.22*Sin[2.2*Pi/24*(x - 8.6)] +
         0.07*Sin[2.3*Pi/24*(x - 8.6)] +
         0.03*Sin[2.4*Pi/24*(x - 8.6)] +
         0.001*Sin[2.5*Pi/24*(x - 8.6)]), {x, 0, 24 - n}] +
     Integrate[.4*(0.97*Sin[2*Pi/24*(x - 8.6 + Pi/2)] +
         0.22*Sin[2.2*Pi/24*(x - 8.6)] +
         0.07*Sin[2.3*Pi/24*(x - 8.6)] +
         0.03*Sin[2.4*Pi/24*(x - 8.6)] +
         0.001*Sin[2.5*Pi/24*(x - 8.6)]), {x, 32 - n,
       24}]])/(Integrate[(1.067445258710798)*
      E^(-2*16/32/32*x) + .153, {x, 8, 24}] +
   Integrate[.4*(0.97*Sin[2*Pi/24*(x - 8.6 + Pi/2)] +
       0.22*Sin[2.2*Pi/24*(x - 8.6)] +
       0.07*Sin[2.3*Pi/24*(x - 8.6)] +
       0.03*Sin[2.4*Pi/24*(x - 8.6)] +
       0.001*Sin[2.5*Pi/24*(x - 8.6)]), {x, 8, 24}])
```

4.3 恢复子模型－求解系数 k MATLAB 代码

```
%Output potential k(recovery rate constant)s and plot them.
function k = plotK(R)
T=1:24;
P=[0.9005608464302958,0.8367423804991914,0.7890108461156795,0.7456944
416744423,0.7004560629381782,0.6506367514096876,0.5961206794439693,0.
5385102640448087,0.5035437110135517,0.4710781234051797,0.442535068214
01933,0.4218361448959889,0.4122667403046987,0.4162373882345549,0.4351
1030863881023,0.4691039029658156,0.5172820940502202,0.577629007433226
8,0.6472030852209465,0.7223587922671959,0.7990190675955129,0.87297796
81897424,0.9402108220275376,0.997168804108945];
%R=0.9;
for i=1:23
   k(i)=log((1-P(i))/(1-R))/(2/3*(12-abs(12-T(i))));
end
hold on
scatter(1:23,k)
disp(var(k))
```

5.GUI 程序核心代码

5.1 参会者城市列表地理编码

```
%Output a location's geographic coordinate by entering the name of the
location
function [lat,lon] = getCoor(city)
%We have several alternative APIs:
%Google Geocoding API
%req =
strcat('https://maps.googleapis.com/maps/api/geocode/json?address=',c
ity,'&key=AIzaSyCU9ICU3_2GI1ClRYWgTivPhpIC3-I9E4c');
%Data Science Toolkit API
%req =
strcat('http://www.datasciencetoolkit.org/maps/api/geocode/json?senso
r=false&address=',city)
%MapQuest API
req =
strcat('http://www.mapquestapi.com/geocoding/v1/address?key=Cl4tM9Qbl
dsgs6SGWu9GJldpGY8VjKSu&location=',city)
res = webread(req);
%Google & DST
% lat = res.results.geometry.location.lat;
% lon = res.results.geometry.location.lng;
%MapQuest
lat=res.results.locations(1).latLng.lat;
lon=res.results.locations(1).latLng.lng;
```

4.2获取参会者城市列表时区

```
%Output the time zone of the location with given latitude and longitude.
function gmt = getTimeZone(lat,lon)
lat = num2str(lat);
lon = num2str(lon);
req =
strcat('http://api.timezonedb.com/v2/get-time-zone?key=J1J7DMMJJKWU&f
ormat=json&by=position&lat=',lat,'&lng=',lon)
res = webread(req);
gmt = res.gmtOffset/3600;
```

5.2 GUI 以及计算

```
function varargout = CitySelector(varargin)
% CITYSELECTOR MATLAB code for CitySelector.fig
%      CITYSELECTOR, by itself, creates a new CITYSELECTOR or raises the
existing
%      singleton*.
```

```
%
%      H = CITYSELECTOR returns the handle to a new CITYSELECTOR or the
handle to
%      the existing singleton*.
%
%      CITYSELECTOR('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in CITYSELECTOR.M with the given input
arguments.
%
%      CITYSELECTOR('Property','Value',...) creates a new CITYSELECTOR or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before CitySelector_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to CitySelector_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CitySelector

% Last Modified by GUIDE v2.5 04-Apr-2017 16:05:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CitySelector_OpeningFcn, ...
                   'gui_OutputFcn',  @CitySelector_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before CitySelector is made visible.
function CitySelector_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CitySelector (see VARARGIN)

% Choose default command line output for CitySelector
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
%Load IMMC logo
axes(handles.axes2);
imshow('IMMC.jpg');

% UIWAIT makes CitySelector wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = CitySelector_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%Note: matrices with name beginning with 'peop' are related to meeting
%participants' home cities, while matrices with name beginning with 'city'
%are related to pool cities (cities in database)(The two names, pool and
database, are interchangable here).

%Declare input variables
budget=str2double(get(handles.budget,'String'));
global file;
%Create waitbar
hwait=waitbar(0,'请等待>>>>>>>>');

%Load 77-city database
cityName={'Vienna';'Brussels';'Prague';'Copenhagen';'Paris';'Nice';'B
erlin';'Munich';'Frankfurt am
Main';'Athens';'Budapest';'Dublin';'Rome';'Venice';'Milan';'Florence'
;'Naples';'Amsterdam';'Krakow';'Warsaw';'Lisbon';'Moscow';'Barcelona'
;'Madrid';'Valencia';'London';'Siem Reap';'Hong
Kong';'Shanghai';'Beijing';'Taipei';'Guangzhou';'Shenzhen';'Macau';'M
umbai';'New Delhi';'Jakarta';'Tel Aviv';'Tokyo';'Kyoto';'Osaka';'Kuala
Lumpur';'Manila';'Singapore';'Seoul';'Pattaya';'Bangkok';'Phuket';'Kr
abi';'Chiang Mai';'Istanbul';'Dubai';'Ho Chi Minh City';'Hanoi';'New
York';'Las Vegas';'San
Francisco';'Orlando';'Miami';'Chicago';'Boston';'Washington DC';'Los
Angeles';'Honolulu';'Playa del Carmen';'Mexico
City';'Cancun';'Santiago';'Vancouver';'Toronto';'Rio de
Janeiro';'Buenos
Aires';'Sydney';'Melbourne';'Auckland';'Marrakech';'Cape Town'};
cityCoor=[48.1923090000000,16.3713640000000;50.8385340000000,4.375434
00000000;50.0874400000000,14.4212560000000;55.6867240000000,12.570069
0000000;48.8565060000000,2.35213300000000;43.7009360000000,7.26839100
000000;52.5170370000000,13.3888600000000;48.1364100000000,11.57753000
00000;50.1106530000000,8.68209300000000;37.9841490000000,23.727984000
0000;47.4983820000000,19.0404710000000;53.3493070000000,-6.2611750000
0000;41.8933440000000,12.4830720000000;45.4371910000000,12.3345900000
000;45.4667970000000,9.19049800000000;43.7698710000000,11.25557600000
00;40.8441160000000,14.2423000000000;52.3710090000000,4.9001120000000
0;50.0619470000000,19.9368560000000;52.2333000000000,21.0166000000000
;38.7130570000000,-9.13800600000000;55.7516340000000,37.6187040000000
;41.3825600000000,2.17713500000000;40.4167050000000,-3.70358300000000
;39.4666000000000,0.375000000000000;51.5072760000000,-0.1276600000000
00;13.3828500000000,103.992885000000;22.2644120000000,114.16706100000
0;31.2253440000000,121.488892000000;39.9064700000000,116.391195000000
;25.0376380000000,121.564459000000;23.1300040000000,113.259001000000;
22.5442670000000,114.054533000000;22.1919510000000,113.538122000000;1
```

```
8.9523800000000,72.8327110000000;28.6138970000000,77.2159560000000;-6
.18101500000000,106.828335000000;32.0804810000000,34.7805270000000;35
.6800710000000,139.768522000000;35.0185800000000,135.763835000000;34.
6852930000000,135.514694000000;3.15709800000000,101.700953000000;14.5
906070000000,120.979901000000;1.29045300000000,103.852038000000;37.56
66010000000,127;12.9318590000000,100.900691000000;13.7527530000000,10
0.494086000000;7.71598400000000,98.3221260000000;7.84138800000000,98.
9873920000000;18.6974950000000,98.6177120000000;41.0170580000000,28.9
855680000000;25.2683520000000,55.2961960000000;10.7827450000000,106.6
94898000000;21.0292100000000,105.852470000000;40.7130540000000,-74.00
72280000000;36.1692020000000,-115.140597000000;37.7800800000000,-122.
420168000000;28.5383310000000,-81.3788790000000;25.7750840000000,-80.
1947020000000;41.8832290000000,-87.6323980000000;42.3588940000000,-71
.0567420000000;38.8920620000000,-77.0199120000000;34.0522380000000,-1
18.243344000000;21.3099200000000,-157.858158000000;20.6282230000000,-
87.0756580000000;19.4325300000000,-99.1332100000000;21.1708910000000,
-86.8403170000000;-33.4379130000000,-70.6504560000000;49.263566000000
0,-123.138572000000;43.6518930000000,-79.3817130000000;-22.9110140000
000,-43.2093730000000;-34.6128690000000,-58.4459790000000;-33.8548160
000000,151.216454000000;-37.8142550000000,144.963146000000;-36.541281
0000000,174.550609000000;31.6259900000000,-7.98860800000000;-33.92890
50000000,18.4172490000000];
cityTimeZone=[2;2;2;2;2;2;2;2;2;3;2;1;2;2;2;2;2;2;2;2;1;3;2;2;2;1;7;8
;8;8;8;8;8;8;5.50000000000000;5.50000000000000;7;3;9;9;9;8;8;8;9;7;7;
7;7;7;3;4;7;7;-4;-7;-7;-4;-4;-5;-4;-4;-7;-10;-5;-5;-5;-3;-7;-4;-3;-3;
10;10;12;1;2];
cityHotel=[126.840000000000;100.360000000000;108.650000000000;173.240
000000000;156.190000000000;136.680000000000;114.050000000000;151.9800
00000000;136.230000000000;117.930000000000;100.380000000000;156.76000
0000000;133.580000000000;175.140000000000;152.950000000000;145.980000
000000;95.0100000000000;162.900000000000;77.3000000000000;83.64000000
00000;120.510000000000;112.850000000000;154.100000000000;121.45000000
0000;94.1800000000000;192.520000000000;56.1800000000000;146.740000000
000;141.100000000000;117.350000000000;112.780000000000;125.9800000000
00;120.640000000000;168.740000000000;143.570000000000;104.72000000000
0;87.0400000000000;182.630000000000;155.880000000000;162.240000000000
;135.910000000000;72.3800000000000;69.4800000000000;152.320000000000;
107.430000000000;45.0800000000000;73.9100000000000;84.2400000000000;8
3.6400000000000;51.2500000000000;75.7400000000000;171.950000000000;71
.5500000000000;54.6900000000000;200.950000000000;133.710000000000;213
.040000000000;120.250000000000;189.520000000000;196.930000000000;251.
160000000000;201.570000000000;177.020000000000;231.110000000000;212.1
00000000000;109.810000000000;222.540000000000;113;171.980000000000;14
```

```
6.520000000000;128.560000000000;108.620000000000;158.210000000000;125
.190000000000;124.540000000000;93.9000000000000;144.880000000000];

%Read meeting participants locations
[null,peopName]=xlsread(file);

l=length(peopName);

%Initialize matrices
peopCoor=zeros(l,2);
peopTimeZone=zeros(l,1);
%peopAirId=cell(l,1);

%Generate participants info table
for i=1:l
    str=['正在获取参会者信息',num2str(i/l*40),'%'];
    waitbar(i/l*0.4,hwait,str);
    %Compare participants' locations info with database info.
    result=strcmp(cityName,char(peopName(i)));
    %If info found, copy from the database.
    if sum(result)~=0
        %peopAirId(i)=cellstr(peopName(result));
        peopCoor(i,1)=cityCoor(result,1);
        peopCoor(i,2)=cityCoor(result,2);
        peopTimeZone(i)=cityTimeZone(result);
    %If not found, attempt to get info from the Internet databases: Data
Science
    %Toolkit and Time Zone Database.
    else
        try
            [x,y]=getCoor(char(peopName(i)));
            peopCoor(i,1)=x;
            peopCoor(i,2)=y;
            peopTimeZone(i)=getTimeZone(x,y);
            %peopAirId(i)=getNearestAirport();
        catch err
        end
    end
end

L=length(cityName);
%Load Day1 performance constants(without flight fatigue). For details of
%calculation please refer to the paper.
```

```
P=[0.9005608464302958,0.8367423804991914,0.7890108461156795,0.7456944
416744423,0.7004560629381782,0.6506367514096876,0.5961206794439693,0.
5385102640448087,0.5035437110135517,0.4710781234051797,0.442535068214
01933,0.4218361448959889,0.4122667403046987,0.4162373882345549,0.4351
1030863881023,0.4691039029658156,0.5172820940502202,0.577629007433226
8,0.6472030852209465,0.7223587922671959,0.7990190675955129,0.87297796
81897424,0.9402108220275376,0.997168804108945];
%Initialize matrices.
cityMark=zeros(L,1);
t_rest_m=zeros(L,1);
fare_m=zeros(L,1);
%Calculating mark for the j-th city in the 77-city pool.
for j=1:L
    %Update waitbar
    str=['正在筛选城市',num2str(j/L*20+40),'%'];
    waitbar(j/L*0.2+0.4,hwait,str);
    %Save info of the j-th city temporarily.
    lat=cityCoor(j,1);
    lon=cityCoor(j,2);
    hotel=cityHotel(j);
    %Calculate time zone difference for each meeting participant.
    delta_n=round(abs(peopTimeZone-cityTimeZone(j)));
    delta_n(delta_n==0)=24;
    %Calculate traveling distance for each meeting participant to j-th
city
    %in the pool.
    dis=zeros(1,l);
    for k=1:l

dis(k)=deg2km(distance(peopCoor(k,1),peopCoor(k,2),lat,lon)); %%%%%%%
    end
    %Calculate air fare. For details of the following codelines, please
refer to the paper.
    fare=sum(dis)/1.609*0.11+50*l; %convert from km to mile
    fare_m(j)=fare;
    %Calculate time for rest (day) under the current budget constraint.
    t_rest=(budget-fare)/(l*hotel)-3;
    t_rest=round(t_rest);
    %Mark '(x)' after name of cities in the pool cannot reached undr the
crrent budget.
    if t_rest<0
        cityName(j)=strcat(cityName(j),'(x)');
    end
    t_rest_m(j)=t_rest;
    %Calculate fatigue coefficient of each participant.
    t_flight=dis/900;
    fatigue=1-0.14*sqrt(t_flight);
    %Calculate Day1 performance of each person.
    for people=1:l;
        peopDay1(people)=P(delta_n(people))*fatigue(people); %%!
    end
    %Calculate total performance of each person during the three-day
    %meeting interval.
    Perf=@(t)1-(1-peopDay1)*exp(-0.304*t);
    peopPerf=Perf(t_rest+1)+Perf(t_rest+2)+Perf(t_rest+3); %%1.round
2.t+1?
    totPerf=sum(peopPerf);
    cityMark(j)=totPerf;
end
%Scale city marks
cityMark=cityMark/max(cityMark)*100;
%For debug&detail
disp(cityMark)
disp(t_rest_m)
disp(fare_m)

str=['正在生成结果','70','%'];
waitbar(0.7,hwait,str);

%Generate detailed result table
fare_mCell=num2cell(fare_m);
cityHotelCell=num2cell(cityHotel);
t_rest_mCell=num2cell(t_rest_m);
cityMarkCell=num2cell(cityMark);
cityCoorCell=num2cell(cityCoor);
result=[cityName,cityCoorCell,fare_mCell,cityHotelCell,t_rest_mCell,c
ityMarkCell];
result=sortrows(result,-7);
global Gresult; Gresult=result
%xlswrite('temp.xlsx',result,'A2:D78');

%Update top 3 recommendations to the GUI
set(handles.C1N,'String',char(result(1,1)));
set(handles.C2N,'String',char(result(2,1)));
set(handles.C3N,'String',char(result(3,1)));
set(handles.C1S,'String',num2str(result{1,7}));
set(handles.C2S,'String',num2str(result{2,7}));
set(handles.C3S,'String',num2str(result{3,7}));

str=['正在绘制地图','90','%'];
waitbar(0.9,hwait,str);
delete(hwait);

%Plot map
axes(handles.mapaxes);
h=worldmap('World');
geoshow('landareas.shp','FaceColor', 'yellow');
%map of home cities
for k=1:1
   plotm(peopCoor(k,1),peopCoor(k,2),'Marker',
'.','MarkerSize',20,'Color', 'blue');
end
textm(peopCoor(:,1),peopCoor(:,2)+5,peopName);
%map of top 3 recommendation cities
for k=1:3
   plotm(cell2mat(result(k,2)),cell2mat(result(k,3)),'Marker',
'*','Color', 'red');
end
textm(cell2mat(result(1:3,2)),cell2mat(result(1:3,3))-10,{'1','2','3'
});

function budget_Callback(hObject, eventdata, handles)
% hObject    handle to budget (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of budget as text
%        str2double(get(hObject,'String')) returns contents of budget as
a double


% --- Executes during object creation, after setting all properties.
function budget_CreateFcn(hObject, eventdata, handles)
% hObject    handle to budget (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Retrive result data
global Gresult;
%Display result details in a new window
open('Details.fig');
h=guihandles;
set(h.table,'ColumnName',{'City Name','Latitude','Longitude','Flight
Fare($)','Hotel Price($)','Day Rest','City Score'});
set(h.table,'ColumnWidth',{77,77,77,77,77,77,77});
set(h.table,'ColumnEditable',false(1,7));
set(h.table,'Data',Gresult);

function C1N_Callback(hObject, eventdata, handles)
% hObject    handle to C1N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C1N as text
%        str2double(get(hObject,'String')) returns contents of C1N as a
double


% --- Executes during object creation, after setting all properties.
function C1N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C1N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function C1S_Callback(hObject, eventdata, handles)
% hObject    handle to C1S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C1S as text
%        str2double(get(hObject,'String')) returns contents of C1S as a
double


% --- Executes during object creation, after setting all properties.
function C1S_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C1S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function C2N_Callback(hObject, eventdata, handles)
% hObject    handle to C2N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C2N as text
%        str2double(get(hObject,'String')) returns contents of C2N as a
double


% --- Executes during object creation, after setting all properties.
function C2N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C2N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function C2S_Callback(hObject, eventdata, handles)
% hObject    handle to C2S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C2S as text
%        str2double(get(hObject,'String')) returns contents of C2S as a
double


% --- Executes during object creation, after setting all properties.
function C2S_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C2S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function C3N_Callback(hObject, eventdata, handles)
% hObject    handle to C3N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C3N as text
%        str2double(get(hObject,'String')) returns contents of C3N as a
double


% --- Executes during object creation, after setting all properties.
```

```
function C3N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C3N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function C3S_Callback(hObject, eventdata, handles)
% hObject    handle to C3S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C3S as text
%        str2double(get(hObject,'String')) returns contents of C3S as a
double


% --- Executes during object creation, after setting all properties.
function C3S_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C3S (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%Get user-selected input file (info of participants) path
[FileName,PathName] = uigetfile('*.xlsx','Select the Data file');
global file;
file = fullfile(PathName,FileName);
```

5.3 导出结果为 CSV

```
function cell2csv(filename,cellArray,delimiter)
% Writes cell array content into a *.csv file.
%
% CELL2CSV(filename,cellArray,delimiter)
%
% filename     = Name of the file to save. [ i.e. 'text.csv' ]
% cellarray    = Name of the Cell Array where the data is in
% delimiter = seperating sign, normally:',' (default)
%
% by Sylvain Fiedler, KA, 2004
% modified by IMMC17427643, 2007
if nargin<3
    delimiter = ',';
end
datei = fopen(filename,'w');
for z=1:size(cellArray,1)
    for s=1:size(cellArray,2)
        var = eval(['cellArray{z,s}']);
        if size(var,1) == 0
            var = '';
        end
        if isnumeric(var) == 1
            var = num2str(var);
        end
        fprintf(datei,var);
        if s ~= size(cellArray,2)
            fprintf(datei,[delimiter]);
        end
    end
    fprintf(datei,'\n');
end
fclose(datei);
```

6.数据库信息生成

6.1 生成数据库城市列表坐标

```
%Generate a table containing the geographic coordinates of the cities in
%the database.
l=length(cityName);
cityCoor=zeros(l,2);
req =
'http://www.mapquestapi.com/geocoding/v1/batch?key=Cl4tM9Qbldsgs6SGWu
9GJldpGY8VjKSu';
for i=1:l
    req=strcat(req,'&location=',char(cityName(i)));
end
res=webread(req);
for i=1:l
    cityCoor(i,1)=res.results(i).locations(1).latLng.lat;
    cityCoor(i,2)=res.results(i).locations(1).latLng.lng;
end
disp(cityCoor)
```

6.2 绘制数据库城市分布地图

```
%Generate a map of cities in the database.
cityName={'Vienna';'Brussels';'Prague';'Copenhagen';'Paris';'Nice';'B
erlin';'Munich';'Frankfurt am
Main';'Athens';'Budapest';'Dublin';'Rome';'Venice';'Milan';'Florence'
;'Naples';'Amsterdam';'Krakow';'Warsaw';'Lisbon';'Moscow';'Barcelona'
;'Madrid';'Valencia';'London';'Siem Reap';'Hong
Kong';'Shanghai';'Beijing';'Taipei';'Guangzhou';'Shenzhen';'Macau';'M
umbai';'New Delhi';'Jakarta';'Tel Aviv';'Tokyo';'Kyoto';'Osaka';'Kuala
Lumpur';'Manila';'Singapore';'Seoul';'Pattaya';'Bangkok';'Phuket';'Kr
abi';'Chiang Mai';'Istanbul';'Dubai';'Ho Chi Minh City';'Hanoi';'New
York';'Las Vegas';'San
Francisco';'Orlando';'Miami';'Chicago';'Boston';'Washington DC';'Los
Angeles';'Honolulu';'Playa del Carmen';'Mexico
City';'Cancun';'Santiago';'Vancouver';'Toronto';'Rio de
Janeiro';'Buenos
Aires';'Sydney';'Melbourne';'Auckland';'Marrakech';'Cape Town'};
cityCoor=[48.1923090000000,16.3713640000000;50.8385340000000,4.375434
00000000;50.0874400000000,14.4212560000000;55.6867240000000,12.570069
0000000;48.8565060000000,2.35213300000000;43.7009360000000,7.26839100
000000;52.5170370000000,13.3888600000000;48.1364100000000,11.57753000
00000;50.1106530000000,8.68209300000000;37.9841490000000,23.727984000
0000;47.4983820000000,19.0404710000000;53.3493070000000,-6.2611750000
0000;41.8933440000000,12.4830720000000;45.4371910000000,12.3345900000
000;45.4667970000000,9.19049800000000;43.7698710000000,11.25557600000
00;40.8441160000000,14.2423000000000;52.3710090000000,4.9001120000000
0;50.0619470000000,19.9368560000000;52.2333000000000,21.0166000000000
;38.7130570000000,-9.13800600000000;55.7516340000000,37.6187040000000
;41.3825600000000,2.17713500000000;40.4167050000000,-3.70358300000000
;39.4666000000000,0.375000000000000;51.5072760000000,-0.1276600000000
00;13.3828500000000,103.992885000000;22.2644120000000,114.16706100000
0;31.2253440000000,121.488892000000;39.9064700000000,116.391195000000
;25.0376380000000,121.564459000000;23.1300040000000,113.259001000000;
22.5442670000000,114.054533000000;22.1919510000000,113.538122000000;1
8.9523800000000,72.8327110000000;28.6138970000000,77.2159560000000;-6
.18101500000000,106.828335000000;32.0804810000000,34.7805270000000;35
.6800710000000,139.768522000000;35.0185800000000,135.763835000000;34.
6852930000000,135.514694000000;3.15709800000000,101.700953000000;14.5
906070000000,120.979901000000;1.29045300000000,103.852038000000;37.56
66010000000,127;12.9318590000000,100.900691000000;13.7527530000000,10
0.494086000000;7.71598400000000,98.3221260000000;7.84138800000000,98.
9873920000000;18.6974950000000,98.6177120000000;41.0170580000000,28.9
855680000000;25.2683520000000,55.2961960000000;10.7827450000000,106.6
94898000000;21.0292100000000,105.852470000000;40.7130540000000,-74.00
72280000000;36.1692020000000,-115.140597000000;37.7800800000000,-122.
420168000000;28.5383310000000,-81.3788790000000;25.7750840000000,-80.
1947020000000;41.8832290000000,-87.6323980000000;42.3588940000000,-71
.0567420000000;38.8920620000000,-77.0199120000000;34.0522380000000,-1
18.243344000000;21.3099200000000,-157.858158000000;20.6282230000000,-
87.0756580000000;19.4325300000000,-99.1332100000000;21.1708910000000,
-86.8403170000000;-33.4379130000000,-70.6504560000000;49.263566000000
0,-123.138572000000;43.6518930000000,-79.3817130000000;-22.9110140000
000,-43.2093730000000;-34.6128690000000,-58.4459790000000;-33.8548160
000000,151.216454000000;-37.8142550000000,144.963146000000;-36.541281
0000000,174.550609000000;31.6259900000000,-7.98860800000000;-33.92890
50000000,18.4172490000000];
l=length(cityName);
h=worldmap('World');
geoshow('landareas.shp','FaceColor', 'yellow');
for k=1:l
    plotm(cityCoor(k,1),cityCoor(k,2),'Marker',
'.','MarkerSize',20,'Color', 'blue');
end
textm(cityCoor(:,1),cityCoor(:,2)+5,cityName);
```

6.3 生成数据库城市列表时区

```
%Generate a table of time zones of the cities in the database.
```

```
l=length(cityCoor);
cityTimeZone=zeros(l,1);
for i=55:l
    lat=cityCoor(i,1);
    lon=cityCoor(i,2);
    cityTimeZone(i,1)=getTimeZone(lat,lon);
end
disp(cityTimeZone)
```

6.4 测试数据集与预算趋势分析

```
%Generate a matrix:
%column: 77 cities from database arranged in regions.
%row: (downward) budget with increment of 100USD starting from the minimum
%budget in order to have at least three candidate cities.
%Matrix analysisTable contains zeros and ones representing whether this
%city at the current budget has been selected into first three or not.
for b=1:80;
    %Because it's the only purpose of analysis of this part of code, codes
    %with behavior similar to others are omitted here.
    %Please refer to CitySelector.m to see the full behavior.
    first3=result(1:3,1);
    for i=1:3
        searchResult(i,:)=strcmp(cityName,char(first3(i)));
    end
    analysisTable(b,:)=sum(searchResult);
end
```

6.5 航班价格分析

6.5.1 获取最近空港

```
%Output the IATA code and full info list of nearest airpots, given the
%latitude and longitude of the location of information requested.
function [code,o] = getNearestAirport(lat,lon)
lat = num2str(lat);
lon = num2str(lon);
req =
strcat('https://iatacodes.org/api/v6/nearby?api_key=3c5351ff-813d-4f9
1-aeeb-2f8ff3ac825f&lat=',lat,'&lng=',lon,'&distance=100')
res = webread(req);
o=res.response(1:length(res.response));
%Filter bus and rail stations and ports out.
for i=length(o):-1:1
```

```
    if
~isempty([strfind(o(i).name,'Railway'),strfind(o(i).name,'Station'),s
trfind(o(i).name,'Port')])
        o(i)=[];
    end
end
code = struct('code', {o(1:length(o)).code});
```

6.5.2 获取航班价格（通过 IATA 机场代码）

```
%Output the mean air fare (data by skyscanner) in IQR, given IATA codes
of
%departure and arrival and date of flight.
function p = getPrice(dep,arr,date) %Date input example '2017-04'
req =
strcat('http://partners.api.skyscanner.net/apiservices/browsequotes/v
1.0/US/USD/en-US/',dep,'-sky/',arr,'-sky/',date,'?apiKey=jo8549393494
685019018171259821 27')
res = webread(req);
MinPrice = [res.Quotes.MinPrice].';
Q1=prctile(MinPrice,25);
Q3=prctile(MinPrice,75);
MinPrice=MinPrice(find(MinPrice>Q1&MinPrice<Q3));
p = mean(MinPrice);
```

6.5.3 获取航班价格（通过地理名称）

```
%Output the potential air fare and corresponding departure airport,
%given name of the departure location and the IATA code of the arrival
%airport and the date of flight of format 'YYYY-MM'.
function [price,depList] = calculatePrice(dep,arr,date)
[x,y]=getCoor(dep);
[depAir,depList]=getNearestAirport(x,y);
for i=1:length(depAir)
    try
        price(i)=getPrice(depAir(i).code,arr,date);
        break;
    catch err
    end
end
```

6.5.4 获取、生成数据库任意两城市间航班价格信息总表

```
%Generate a price matrix containing air fare between any two cities in
the
%77-city database.
hwait=waitbar(0,'Waiting>>>>>>>>');
```

```
l=length(airId);
Price=zeros(l,l);
for r=1:l
    dep=char(airId(r));
    for c=1:l
        if r==c
            continue;
        end
        try
            progress=((r-1)*l+c)/(l*l);
            str=['Retriving Data...',num2str(progress*100),'%'];
            waitbar(progress,hwait,str);
            price=getPrice(dep,char(airId(c)),'2017-04');
            Price(r,c)=sum(price);
        catch err
        end

disp(strcat(num2str(r),',',num2str(c),',',num2str(sum(price))))
    end
end
delete(hwait);
```

6.6 生成数据库中任意两城市间距

```
%Generate a distance matrix containing distances between every two cities
%in the 77-city database.
l=length(airId);
Distance=zeros(l,l);
for r=1:l
    Rcoor=cityCoor(r,:)
    Rx=Rcoor(1);Ry=Rcoor(2);
    for c=r:l
        if r==c
            continue;
        end
        try
            Ccoor=cityCoor(c,:)
            Cx=Ccoor(1);Cy=Ccoor(2);
            distance=distance(Rx,Ry,Cx,Cy);
            Distance(r,c)=distance;
        catch err
        end
        disp(strcat(num2str(r),',',num2str(c),',',num2str(distance)))
    end
end
```

```
%Complete the matrix
Distance=Distance+Distance';
```