**黄山徒步路线体力消耗优化模型研究**

**摘要**

本文针对黄山旅游进行规划，利用数学建模的方法优化行程安排。本模型主要考虑了体力消耗因素，建立了相应的模型，并通过求解模型得出了待选徒步线路中最优的计划。结果显示，所提出的方案能够有效地节省体力，实现最佳的旅游徒步体验。

**一、问题描述**

本模型希望确定最合适的一条黄山徒步游览线路。在规划此行程时，主要考虑体力消耗因素，即人走此段线路消耗的平均能量。

**二、假设**

1. 游客的体力和耐力在正常范围内。

2. 所选数据源和公式绝对精确。

3. 天气条件良好，不影响游览。

4. 游客总质量70 kg。

**三、源数据选取**

本研究选择了AllTrails网站[[[1]](#endnote-2)]上用户自发分享的“排名”前10的黄山纯徒步路线，下载了GPX格式的路径文件以分析最佳路线。

**四、数学模型**

本研究中的数学模型以体力消耗为核心，结合距离、速度和坡度等因素，计算游客在不同路线上的能量消耗，算出能量消耗速率作为路线的评分。

对于一个确定的路线，有以下变量和常量：

**变量**

* ：采样点纬度变化量。数据源经少许处理可得。
* ：采样点经度变化量。数据源经少许处理可得。
* ：两个采样点之间的时间，单位 。数据源经少许处理可得。
* ：两个采样点之间的海拔变化，单位 数据源经少许处理可得。
* ：两个采样点之间的距离，单位 。使用 Haversine公式计算得出。
* ：两个采样点之间的游客平均速度，单位 。受地形和体能状况影响。使用已知量计算得出。
* ：两个采样点之间的游客消耗能量，单位 。使用已知量计算得出。
* ：路线最终得分，单位 。使用已知量计算得出。

**常量**

* ：地球半径，Haversine公式中的常数，取 。
* ：游客的体重，单位 。对能量消耗有显著影响。
* ：路线总时间，单位 。直接选取网站上的数据。

**五、计算方法**

**采样点间距离基于以下方法计算：**

首先，两个经纬度点之间的水平距离基于以下Haversine半正矢公式的变形式计算：

其中， 的表达式为：

注：该数学公式由AI辅助使用。

接着，使用勾股定理计算实际距离 ：

**采样点间游客平均速度基于以下公式计算：**

**采样点间能量消耗基于以下公式计算：**

公式中的200是一个经验系数。这个系数是基于生理学的观察，即平均每消耗1升氧气，身体大约会消耗5千卡的能量[[[2]](#endnote-3)]。把原单位带入，，即直接除以200 即可，整个公式把氧气消耗量转换成能量消耗。

其中，是每分钟的耗氧量，使用ACSM 公式计算[[[3]](#endnote-4)]：

* 跑步（速度 > 2.5 m/s）时：
* 步行（速度 ≤ 2.5 m/s）时：

**路线最终得分基于以下公式计算：**

**六、数据分析**

主代码 (./math\_analyze.py) 片段首先通过Haversine公式计算两个经纬度点之间的水平距离，并结合海拔差异来计算实际距离；然后使用ACSM（American College of Sports Medicine）的公式估算能量消耗。以下是各个步骤的详细求解过程：

### 1. Haversine公式计算水平距离并结合海拔差计算实际距离

**def** haversine(lat1, lon1, lat2, lon2, ele1, ele2):

...

# 计算水平距离

horizontal\_distance = EARTH\_RADIUS \* c

# 计算海拔差

elevation\_diff = ele2 - ele1

# 使用勾股定理计算实际距离

distance = math.sqrt(horizontal\_distance \*\* 2 + elevation\_diff \*\* 2)

**return** distance

此函数功能：

* 首先通过Haversine公式计算两个经纬度点之间的水平距离。Haversine公式通过球面三角函数计算两点间的弧长，再乘以地球半径得出两点间的地面距离。
* 然后，计算两个点的海拔差异 elevation\_diff。
* 最后，使用勾股定理计算实际的三维距离 distance，即水平方向的距离和垂直方向的海拔差的合成距离。

### 2. 使用ACSM公式计算能量消耗

**def** acsm\_energy\_expenditure(distance, elevation\_change, speed, weight):

...

**return** energy

此函数功能：

* 根据ACSM的能量消耗公式，能量消耗与运动速度、海拔变化、距离以及个人体重相关。
* 最终计算的能量消耗 energy是两个采样点间运动过程中消耗的能量。

### 3. 计算路线中的每段距离、海拔变化、速度和能量消耗

**def** calculate\_route\_energy(route\_points, weight):

...

**for** i **in** range(1, len(route\_points)):

...

energy = acsm\_energy\_expenditure(dist, ele\_change, speed, weight)

...

**return** pd.DataFrame({

'distance': distances,

'elevation\_change': elevations,

'speed': speeds,

'energy': energies

})

此函数功能：

* 遍历 route\_points列表，逐段计算每一段的距离、海拔变化、速度以及能量消耗。
* 通过 haversine函数计算两点之间的距离，并记录海拔变化。
* 使用 acsm\_energy\_expenditure函数计算每段路程中的能量消耗。
* 最终将所有计算结果整理为一个Pandas DataFrame，以便进一步分析和保存。

### 4. 从GPX文件中提取路线点数据

**def** extract\_route\_points(gpx\_file\_path):

...

**return** route\_points

此函数功能：

* 使用 gpxpy库解析单个GPX文件，从中提取每个路点的经纬度和海拔信息。
* 这些信息将用于后续（前面几个函数）的距离和能量消耗计算。

### 5. 处理全部GPX文件

**def** process\_all\_gpx\_files(directory, weight):

...

**for** filename **in** os.listdir(directory):

...

# 保存每条trail的结果为单独的CSV文件

output\_file = os.path.join(directory, f"{trail\_name}\_energy\_data.csv")

...

**return** all\_trail\_data

此函数功能：

* 遍历指定目录中的所有GPX文件，传入前面的函数提取路线点，并计算对应的能量消耗。
* 将每条路径的能量消耗结果保存为独立的CSV文件，同时将所有结果汇总，便于进一步分析。

### 6. 激活函数，进行操作

directory = './gpxfiles'

weight = 70 # 设定用户体重为70kg

trail\_data = process\_all\_gpx\_files(directory, weight)

函数定义完毕后，开始运行。此处代码指定GPX文件所在的目录和用户的体重，调用 process\_all\_gpx\_files函数处理所有文件并计算能量消耗。计算结果将存储在每个路径对应的CSV文件中，同时返回所有路径的数据供后续使用。

distance,elevation\_change,speed,energy

3.880008759856245,0.5099999999999909,0.6546633114442726,0.12862228677658272

6.920244108499101,1.1600000000000819,0.5838169711040282,0.25822470572995865

...

**至此，**我们已经得到了每个GPX文件对应的CSV数据，包括 、、、。接下来，数据总结代码 (./final\_calculation.py) 将对这些数据进行最后处理得到 。

**def** analyze\_energy\_data(directory):

...

**for** filename **in** os.listdir(directory):

if filename.endswith("\_matched\_energy\_data.csv"):

...

df = pd.read\_csv(csv\_file)

total\_energy = df['energy'].sum()

if trail\_name **in** estimated\_times:

...

energy\_per\_minute = total\_energy / estimated\_time

output\_lines.append(f"{trail\_name}: Total Energy = {total\_energy:.1f}, Energy per Minute = {energy\_per\_minute:.2f}")

else:

...

**至此，**得到总结文件：./RESULT/trail\_energy\_analysis.txt

Celestial\_Peak: Total Energy = 342.6, Energy per Minute = 1.25

Flying\_Rock\_Loop: Total Energy = 104.4, Energy per Minute = 1.24

Heavenly\_Peak\_and\_Lotus\_Pavillion: Total Energy = 185.1, Energy per Minute = 1.25

Huangshan\_Trail: Total Energy = 569.0, Energy per Minute = 1.23

Huangshan\_West\_Path\_to\_Greeting\_Pine: Total Energy = 214.8, Energy per Minute = 1.14

Huangshan\_West\_Sea\_Gate\_to\_West\_Scenic\_Area: Total Energy = 186.0, Energy per Minute = 1.18

Purple\_Cloud\_Peak: Total Energy = 82.4, Energy per Minute = 1.25

Shisun\_Gang\_: Total Energy = 37.4, Energy per Minute = 1.25

West\_Sea\_Canyon\_Trail: Total Energy = 76.8, Energy per Minute = 1.13

Wu\_Shan\_Yellow\_Mountains: Total Energy = 390.6, Energy per Minute = 1.20

**综上所述，**模型得出的最“省体力”路线为 West Sea Canyon Trail。

**七、模型检验**

介于本人没有实际登上黄山，本模型的检验只能依靠 AllTrails 网站上的路线难度评价进行。AllTrails 网站同样使用了一个数学模型估计路线的难度[[[4]](#endnote-5)]，但是不能确定具体的模型信息。下面是把路线难度评价和本模型输出结果对比的结果：

**Celestial\_Peak**: Hard, Total Energy = 342.6, Energy per Minute = 1.25

**Flying\_Rock\_Loop**: Moderate, Total Energy = 104.4, Energy per Minute = 1.24

**Heavenly\_Peak\_and\_Lotus\_Pavillion**: Hard, Total Energy = 185.1, Energy per Minute = 1.25

**Huangshan\_Trail**: Hard, Total Energy = 569.0, Energy per Minute = 1.23

**Huangshan\_West\_Path\_to\_Greeting\_Pine**: Easy, Total Energy = 214.8, Energy per Minute = 1.14

**Huangshan\_West\_Sea\_Gate\_to\_West\_Scenic\_Area**: Easy, Total Energy = 186.0, Energy per Minute = 1.18

**Purple\_Cloud\_Peak**: Moderate, Total Energy = 82.4, Energy per Minute = 1.25

**Shisun\_Gang\_**: Moderate, Total Energy = 37.4, Energy per Minute = 1.25

**West\_Sea\_Canyon\_Trail**: Easy, Total Energy = 76.8, Energy per Minute = 1.13

**Wu\_Shan\_Yellow\_Mountains**: Hard, Total Energy = 390.6, Energy per Minute = 1.20

可以看出，本论文所述模型基本能反映出路线的难度，例如 在 以下的路线难度评级均为Easy。如果和路线总消耗能量一起作为判断“省体力”的依据，结果将更加准确。

**八、结论**

通过对黄山旅游路线的分析和优化，我们得出了多条能节省体力消耗的最优路线。这些路线能够为不同体能水平的游客提供合理的选择，使他们能够在尽量少而平缓的体力消耗下，享受到黄山的美景。该模型和方案也可以推广应用于其他类似的旅游行程规划，为旅游者提供科学的行程建议。

**附录**

在本研究的计算与分析过程中，使用了多个目录和文件，以确保数据的处理和结果的生成能够顺利进行。具体说明如下：

1. **源文件存放：**
   * ./gpxroutes\_originally\_gpxfiles: 存放从AllTrails直接下载的GPX路线文件（包含时间数据，高度数据不精确）。
   * ./gpxtracks: 存放从AllTrails直接下载的GPX轨迹文件（包含精确的高度数据，但不含时间数据）。
   * ./merged\_gpx: 存放通过mergedata.py脚本合并后的GPX文件，这些文件结合了准确的高度数据和时间数据。
   * ./gpxfiles: 存放完好的待处理的GPX文件和主程序输出结果。
   * ./gpsfix: 包含几条GPX轨迹文件有一些小问题的路线数据，这些数据不能使用合并后的文件进行处理，只能使用GPX路线文件数据进行处理，处理工具为math\_analyze\_bugfix。
2. **数据处理与分析：**
   * math\_analyze.py: 主程序，用于分析./gpxfiles目录下的GPX文件，并生成相应的CSV文件。
   * final\_calculation.py: 用于计算最终的结果，生成trail\_energy\_analysis.txt文件，存放于./RESULT目录中。
   * mergedata.py: 用于合并GPX文件，生成包含精确高度和时间数据的文件，存放于./merged\_gpx目录中。
3. **其他说明：**
   * ./output\_history: 包含一些历史版本模型分析的结果数据。
   * ./RESULT: 最终输出数据存放目录。
   * ./time\_adjustment: 调整GPX文件的错误时间的文件。
   * debug.py: 用于调试程序的错误输出，暂未用于正式分析。

**参考文献及资料**

1. [] AllTrails. Huangshan National Forest Park[EB/OL]. [2024-08-18]. https://www.alltrails.com/parks/china/anhui/huangshan-national-forest-park. [↑](#endnote-ref-2)
2. [] Thompson DL. What is Oxygen Consumption? [J]. ACSM's Health & Fitness Journal, 2009, 14(1): 4. [↑](#endnote-ref-3)
3. [] Hall, C., Figueroa, A., Fernhall, B., & Kanaley, J. A. Energy Expenditure of Walking and Running: Comparison with Prediction Equations[J]. Medicine & Science in Sports & Exercise, 2004, 36(12): 2128-2134. [↑](#endnote-ref-4)
4. [] AllTrails Support. How are the difficulty ratings on AllTrails determined? [EB/OL]. [2024-08-18]. https://support.alltrails.com/hc/en-us/articles/16596491196436-How-are-the-difficulty-ratings-on-AllTrails-determined. [↑](#endnote-ref-5)