报名序号：002545

论文题目：B 题 大学生平衡膳食食谱的优化设计及评价

大学生平衡膳食食谱的优化设计及评价

摘　要

制订均衡的膳食食谱，形成良好的饮食习惯，是保证大学生身体健康、精神饱满的关键因素。我们通过建立**膳食食谱营养分析评价模型**和**基于“高精微异”遗传算法**和**帕累托前沿的优化模型**实现了对于食谱的评分和优化，为制订合理的日食谱和周食谱提供了参考和依据。

基于问题一，我们首先查询中国食物成分表，补充和完善各食物对应的成分数据。之后完成膳食食谱营养评价过程各个步骤的计算，使用**熵权法结合 TOPSIS** 的方法，计算各成分指标的权重，结合每一项成分指标的得分和其对应的权重，计算出男女大学生食谱的得分，分别为 49.63 和 61.71。第二问通过观察比较食谱中各项成分指标的具体得分进行调整，得到男女大学生改进后的食谱得分，分别为 54.22 和 64.63，较原食谱有所提高。

基于问题二，我们首先使用 **K-means++算法**对所有菜品进行聚类，从 141 种菜品中挑选出 35 种具有代表性的菜品，将数据降维。将优化设计方案中的各项指标要求范围设定为约束条件，使用**“高精微异”遗传算法**进行单目标优化问题的求解。分别以氨基酸评分最大和价格最低为目标，得到男女大学生优化设计后的日食谱，得到的最大氨基酸评分分别为 103.13，98.81，价格最低为 26.5 元和 25.5 元。第三问在兼顾前两问的优化目标时使用**帕累托前沿**得到最优解，得到的两份食谱总体评分，分别为 83.75 和 80.28。我们建立的优化模型可以针对其目标进行有效的优化，同时相较于原食谱显著提高了总体评分。

基于问题三，在周平衡膳食食谱的设计中，将问题二中得到的日食谱作为星期一的食谱，找出其中相对误差较大和较小的指标，分别缩小和放大其约束条件的范围，再次利用优化模型得到星期二的食谱，并以此类推得到七天的周平衡食谱。使用平衡膳食的基本准则对该食谱进行验证。

基于问题四，我们依托于所建模型和优化结果，从食物多样、种类丰富，合理安排餐次比，满足营养素定量要求三个方面，针对于男女生提供了大学生膳食平衡倡议书。

综上所述，我们的模型考虑了多种指标对食谱的影响，根据不同目标设计出最合理的食谱，并使各项指标尽可能接近参考值。可以将其推广至不同类型的人群，这对于今后的人体饮食结构研究具有重要的参考价值。

关键词：遗传算法；K-means++聚类算法；帕累托最优；熵权法；TOPSIS

1 问题重述

1.1 问题背景

大学生不仅要注重对学识的积累，更要养成健康的饮食习惯。充足且均衡的营养和能量，是迎接繁忙的生活与学业的刚需。然而，现如今许多大学生却饮食结构混乱，养成了许多不良习惯。有些大学生忽略早餐，频繁地依赖外卖或快餐；有些大学生通过节食而导致营养不良。在这个人生阶段的大学生应该重视对健康营养基本知识的掌握，养成良好的饮食习惯。它们不仅关乎身体的健康，更是促进学业和发展的基石。

《中国居民膳食指南》给出了中国居民平衡膳食宝塔以及中国居民平衡膳食餐盘图，并提出平衡膳食的基本准则，大学生每日能量摄入目标，主要营养素定量要求等多种衡量标准。参考这些标准制定合理健康的食谱，是养成良好饮食习惯最直接的方案。

1.2 问题重述

- 问题一：依据附件 4 中的“膳食食谱营养评价过程”制定合适的评价模型，对附件 1、附件 2 中的两份食谱做出评价，并依据附件 3 中的食堂菜品信息对食谱进行调整改进并全面评价。
- 问题二：分别以蛋白质氨基酸评分最大，用餐费用最经济，和以上二者兼顾作为目标建立优化模型，基于附件 3 食堂的菜品信息对男女生的日平衡膳食食谱进行优化设计和评价，之后完成对这些由不同优化目标得到的食谱的比较分析。
- 问题三：分别以问题二中的三种目标建立优化模型，加入周平衡膳食食物成分总数的约束条件，基于附件 3 食堂菜品信息设计男生和女生的周平衡膳食食谱，并进行评价及比较分析。
- 问题四：针对大学生饮食结构及习惯，结合以上三问的评价及优化结果写一份健康饮食、平衡膳食的倡议书。

2 问题分析

2.1 问题一的分析

我们需要按照附件 4，完成膳食食谱营养评价过程的 1-5 步。然后，记录单日食谱中总食物成分的实际值与参考值的差距并将其作归一化处理，结果作为食谱各项食物成分指标。使用熵权法结合 TOPSIS 的方法，确定指标正负性，计算各指标权重，结合每一项指标的得分和与其对应的权重分别计算出两份食谱的得分，完成食谱评价模型的建立。观察比较两份食谱每项食物成分指标的具体得分情况，损有余补不足，分别对两份食谱进行少量改进，利用评价模型计算改进后的食谱的得分并与原始食谱进行比较。

2.2 问题二的分析

根据题目所给的三种不同目标，我们首先使用遗传算法建立单目标膳食平衡优化模型，而对于有多个目标的协同优化问题，则考虑在遗传算法中使用帕累托前沿得到不同偏好的均衡解，筛选均衡解以引导方案迭代进化方向。由于附件 3 食堂菜单中的食物种类过多，我们使用 K-means++ 的聚类方法将所有菜品依据其营养素成分进行分类，选出具有代表性的菜品。在制定约束条件时，需参照附件 4 中“平衡膳食食谱优化设计原则”并考虑指标的具体规则，同时保留一定裕度以确保结果的完整性和有效性。结合代表菜品与平衡膳食约束条件，对若干组随机生成的方案使用遗传算法进行迭代优化，得到优化后的食谱。得到优化食谱后，我们再次采用问题一中的食谱评价模型对优化食谱做出评价，并与原食谱进行比较分析，验证膳食平衡优化模型的可行性。最后，我们对三种不同目标函数下得到的优化食谱进行对比分析并评价。

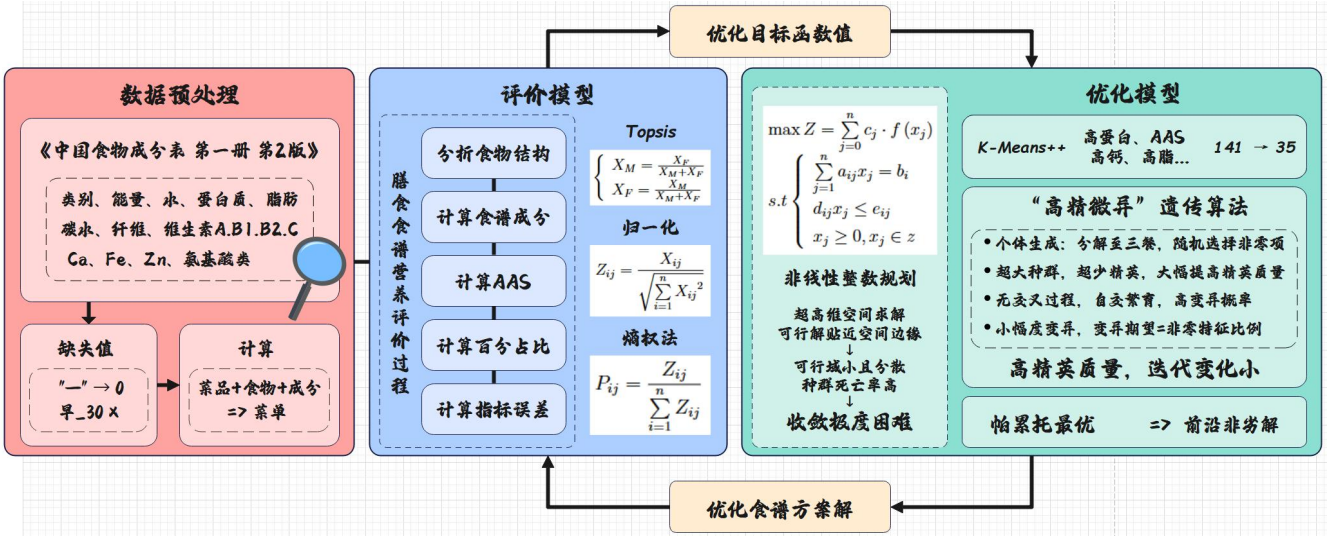
2.3 问题三的分析

问题三中的目标函数与问题二相同，但食谱的优化及评价范围从一日扩展到了一周，在满足日平衡膳食食谱约束条件的同时还要满足周平衡膳食食谱的食物种类大于 25 种，得到优化设计的周食谱后进行比较分析评价的过程与问题二相同。

2.4 问题四的分析

最后我们需要针对大学生饮食结构及习惯偏好，结合以上建立的评价模型和优化模型，写一份健康饮食、平衡膳食的倡议书。

2.5 流程图



3 模型假设

- 假设题目所给数据以及查询得到的数据均准确无误
  我们假设，在官方发布的《中国食物成分表》[1] 中查询到的各食物所含营养成分均准确无误，查询结果为“——”的数据统一认为是 0。

- 假设营养素成分在烹饪前后没有消耗
  我们假设，食物成分在制作过程中保持不变，即不同菜品对应的食物成分均相同且等于《中国食物成分表》所查数据。

- 假设菜品只含数据中所给的食物
  我们假设，菜品中仅含附件 3 食堂菜品信息中食物，忽略其他食物及对应成分。

4 定义与符号说明

符号	定义
X_{ij}	第 j 个营养素指标第 i 个样本与参考值之差的绝对值
Z_{ij}	标准化后第 j 个营养素指标第 i 个样本与参考值之差的绝对值
P_{ij}	第 j 个营养素指标第 i 个样本所占的权重
S_i	第 i 份食谱的评价得分
AAS_i	第 i 天的氨基酸评分
$Price_i$	第 i 天的消费总价格

表 1: 符号定义与说明

5 数据处理

5.1 查询各种食物的成分含量

附件 3 食堂菜品信息中一共出现了 61 种食物，我们查询《中国食物成分表》获取所有 61 种食物成分的具体数值，包括能量、水、蛋白质等产能营养素、维生素 A 等非产能营养素、异亮氨酸等氨基酸。依照附件 4“平衡膳食的基本准则”和《中国食物成分表》将食物成分分为五大类。以上内容形成食物成分数据表，见支撑材料 1。

按照中的五大类食物进行分类和编码，并补全所有食物主要成分的产能营养素，非产能营养素和八种必需氨基酸的数据，得到主要成分营养素数据表，见支撑材料 1。随后，对于该高校食堂提供的所有食物，我们将每种食物对应的主要成分所包含的营养素数据乘以可食部，得到了食物菜单营养素数据表, 见支撑材料 2。

5.2 缺失值填补和异常值处理

我们假设《中国食物成分表》所提供的数据都是准确无误的，其中结果为“——”的未测定项均视为 0。附件 3 食堂菜品信息中的早餐缺失第 30 项，我们直接将其删除并依次重命名了后续四项菜品的序号。

5.3 计算各个菜品的成分含量

由于制定食谱的基本单位是一份菜品，而非其中食物，对于附件 3 中菜单，我们查询食物成分数据表中各个菜品对应的食物，将其乘上实际可食部，统一菜品成分全部加和，可半份菜品统一按照半份计算，得到菜品成分数据表，见支撑材料 2。为方便分别计算，又将其按照早、午、晚餐分割为三个餐次食物成分菜单。

6 问题一：基于熵权法和 TOPSIS 的评价模型

6.1 膳食食谱营养分析评价模型建立

针对给定的日食谱, 我们按照膳食食谱营养评价过程，分析食物结构，计算食谱中的营养素含量并与参考值进行比对，后使用熵权法结合 TOPSIS 方法建立了定量的膳食食谱营养分析评价模型。

6.1.1 分析食物结构

我们按平衡膳食的基本准则中的五种类别将男大学生和女大学生食谱中食物进行归类排序，列出每种食物的数量如图所示：

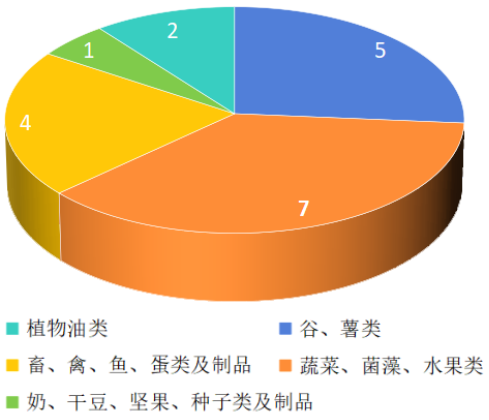


图 1: 男大学生一日食谱食物结构

图 2: 女大学生一日食谱食物结构

通过统计图表我们可以看出，男大学生和女大学生的一日食谱中均包含了五大类别的食物，且食物种类大于 12 种，食物多样，搭配合理，满足平衡膳食的基本准则。

两份食谱包含了较多种类的畜、禽、鱼、蛋类及制品，蔬菜、菌藻、水果类这两类食物，奶、干豆、坚果、种子类及制品，植物油类的食物种类较少，男大学生选择谷、薯类食物的种类明显大于女大学生。

6.1.2 计算评价食谱提供的能量、餐次比、能量来源及非产能主要营养素含量

通过我们在数据处理中得到的食物菜单营养素数据表，我们计算得到男大学生、女大学生食谱中的主要营养素含量及参考值如下表所示：

营养素	实际值	参考值	单位	营养素	实际值	参考值	单位
能量	2517.12	2400	kcal	维生素 A	254.75	800	ug
水	592.904	/	g	维生素 B1	1.0505	1.4	mg
蛋白质	77.63	75	g	维生素 B2	0.965	1.4	mg
脂肪	122.683	66.67	g	维生素 C	50.8	100	mg
碳水化合物	274.338	345	g	钙	557.51	800	mg
膳食纤维	13.805	/	g	铁	26.348	12	mg
				锌	14.7123	12.5	mg

表 2: 男大学生食谱营养素含量及参考值

营养素	实际值	参考值	单位	营养素	实际值	参考值	单位
能量	1318	1900	kcal	维生素 A	188.7	700	ug
水	445.3	/	g	维生素 B1	0.817	1.2	mg
蛋白质	52.34	59.375	g	维生素 B2	0.571	1.2	mg
脂肪	50.95	52.78	g	维生素 C	83.4	100	mg
碳水化合物	162.695	273.125	g	钙	282	800	mg
膳食纤维	8.145	/	g	铁	12.495	20	mg
				锌	39.76	7.5	mg

表 3: 女大学生食谱营养素含量及参考值

通过与参考值的比较，我们对两份食谱做出以下评价：

- 男大学生: 在能量摄入上达到了摄入目标，但在产能营养素的定量要求中，脂肪摄入过多，碳水化合物摄入不足。在非产能营养素中，维生素和钙的摄入过少而铁的摄入量过多。
- 女大学生: 在能量摄入上远远未达到摄入目标，在产能营养素的定量要求中，各种营养素均较少，碳水化合物摄入缺少得最为严重。而在非产能营养素中，除了锌元素外均距摄入标准有很大的差距。

除了以上能量摄入以及非产能主要营养素的摄入指标之外，我们还计算了两份食谱中早，中，晚饭的餐次比并与标准值比较如下表所示：

餐次	男大学生（%）	女大学生（%）	参考值（%）
早餐	30.64	24.25	30
午餐	35.32	45.01	30-40
晚餐	34.03	30.74	30-40

表 4: 男，女大学生食谱餐次比及参考值

可以发现，男大学生食谱的餐次比满足参考值要求，三餐摄入量搭配合理均衡。而女大学生食谱中，早餐摄入过少，午餐摄入过多，三餐餐次比较不合理。

6.1.3 评价每餐的蛋白质氨基酸评分

由附件 4，我们采用氨基酸评分（AAS）的方法对两份食谱中混合食物进行计算。用被测食物蛋白质的必需氨基酸与参考蛋白质的氨基酸模式进行比较，计算出比值，比值最低者为第一限制氨基酸。依照混合食物蛋白质氨基酸评分小于 60 为不合理，60—80 为不够合理，80—90 为比较合理，大于 90 为合理的规则，我们得到如下评价图：

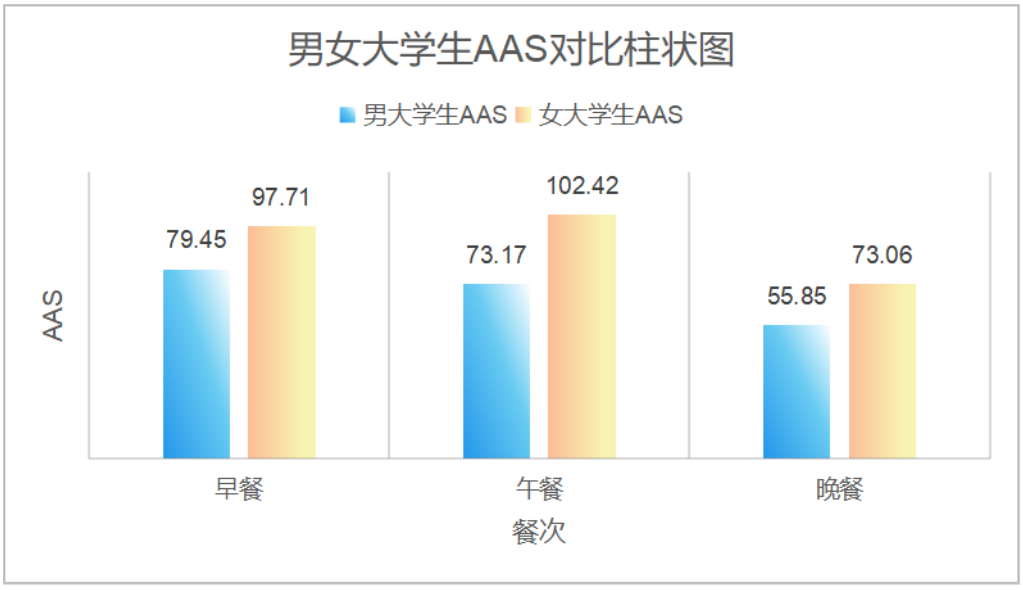


图 3: 男女大学生 AAS 对比柱状图

通过上图我们发现，男大学生早餐，午餐的氨基酸评分不够合理，而晚餐的氨基酸评分不合理。女大则在早餐和午餐表现合理，晚餐却不够合理。综合比较二者三餐的 AAS，女大学生整体的氨基酸摄入情况比男大学生合理。

6.1.4 整体评价模型建立

分别计算出男大学生、女大学生每日摄入能量、三餐能量分配占总能量的百分比、宏量营养素供能占比、非产能营养素摄入量与其相对应的标准值之间的差值，并进一步计算出差值与标准值之比，即相对差值，作为评价模型的指标输入。

我们接下来将以上步骤中得到的各个指标进行整合，使用熵权法结合 TOPSIS 建立膳食食谱营养分析评价模型。

熵权法是一种赋权方法。根据各指标数值变化对整体的影响，计算指标的熵值，进而确定权重。TOPSIS（Technique for Order Preference by Similarity to an Ideal Solution）法是 C.L.Hwang 和 K.Yoon 于 1981 年首次提出，TOPSIS 法根据有限个评价对象与理想化目标的接近程度进行排序的方法，是在现有的对象中进行相对优劣的评价 [2]。通过以上方法的结合，我们确定了各个指标在评分体系中的权重，并针对两份食谱进行了综合的评价打分。具体过程如下：

首先，我们确定各个指标的正负性。我们发现，指标中除了氨基酸评分（ASS），其余皆为负向化指标，即实际值与参考值的绝对值之差越小代表指标越优秀。因此我们需要将这些指标进行正向化的操作，公式如下：

$$\begin{cases} X_M = \frac{X_F}{X_M + X_F} \\ X_F = \frac{X_M}{X_M + X_F} \end{cases}$$

其中，X_M 代表男大学生食谱的指标，X_F 代表女大学生食谱的指标。由此，我们得到了全部指标的正向化矩阵：

$$X = \begin{bmatrix} X_{11} & \dots & X_{1m} \\ X_{21} & \dots & X_{2m} \end{bmatrix}_{2\times m}$$

之后，我们对以上矩阵进行标准化处理，公式及标准化矩阵如下：

$$Z_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{n} {X_{ij}}^2}}$$

$$Z = \begin{bmatrix} Z_{11} & \dots & Z_{1m} \\ Z_{21} & \dots & Z_{2m} \end{bmatrix}_{2\times m}$$

我们使用如下公式计算第 j 项指标下第 i 个样本所占的权重，并将其作为相对熵计算中的概率。

$$P_{ij} = \frac{Z_{ij}}{\sum_{i=1}^{n} Z_{ij}}$$

由以上步骤，我们得到了各个指标对应的权重 P_{ij}, 将其与各个指标标准化后的 Z_{ij} 相乘并求和，便可得到最终一份食谱的得分 S_i：

$$S_i = \sum_{j=1}^{n} P_{ij} \cdot Z_{ij}$$

计算得到男大学生食谱的评分为 0.496，女大学生食谱的评分为 0.617，所有指标的加权值及得分见支撑材料 3。女大学生的食谱较男大学生得分更高，这是由于女大学生食谱在 AAS，产能营养素等方面比男大学生食谱表现更好，与之前逐项的分析结果一致。但二者的食谱评分均不高，仍可以对许多指标进行改进优化，以形成更加均衡合理的膳食食谱。

6.2 基于评价模型对食谱的改进

基于以上分析，我们对两份食谱提出针对性的改进意见。若某一指标得分较低同时加权较高，意味着对该指标的改善可以较大程度上影响食谱的总体评分。

- 男大学生食谱中选择的食物含铁量过大，氨基酸评分低，这两项为需要调整的主要指标。因此我们将男大学生食谱早餐中的两份油条换成两份烤地瓜，午餐中的拌木耳换成香菇炒油菜，晚餐中的炸鸡块变成炒牛肉。
- 女大学生食谱中评分低，加权高的指标为餐次比，其中早餐、晚餐占比过低，因此建议女大学生在早餐中加一份油条，晚餐中加一份米饭。

男大学生			女大学生		
指标	原食谱评分	改进后评分	指标	原食谱评分	改进后评分
铁	23.9	30.9	早餐餐次比	10.0	54.0
AAS	79.4	111.5	晚餐餐次比	18.5	43.7
总体评分	49.63	54.22	总体评分	61.71	64.63

表 5: 男，女大学生食谱改进指标对比值

将以上改进方案输入评价模型进行计算，得到最终的评分：男大学生食谱为 0.542，女大学生食谱为 0.646，具体各指标评分见附件 3。与初始方法相比，食谱的总评分有了一定程度的提高，这说明我们的改进方案具有改善膳食平衡的实际意义，同时也证明了评价模型的可行性。同时，我们注意到只依照某几项指标进行人为调整会导致其他指标出现不同程度的波动，不利于总体评分的提升，这也是调整后评分仍不高的主要原因。因此，我们需要提出更合理的优化方法，以更有效地提升整体食谱质量。

7 问题二：基于“高精微异”遗传算法和帕累托最优的优化模型

7.1 基于 K-means++ 的菜品聚类

在进行优化算法的设计时，我们发现附件 3 食堂菜单中早午晚餐的菜品的种类共有 141 种。显然，在 141 维空间内寻找最优解并不是一个明智的选择。进一步观察发现，菜单中包含有大量对于食谱优化意义和影响不大的菜品，严重影响了优化效果。因此，我们使用 K-means++ 算法，对所有菜品进行聚类，挑选出不同食物成分特点的代表性菜品作为优化算法的输入。

K-means 是机器学习和数据分析中广泛使用的聚类算法。这种无监督学习算法将数据集分成预先确定的 K 个簇，每个数据点属于其最近的质心所在的簇。该算法首先通过随机初始化 K 个质心，然后迭代地将数据点分配到最近的簇，并更新质心，直到收敛。K-means 旨在最小化簇内平方和，表示每个簇内数据点的方差。

在标准的 K-means 算法中，初始质心是从数据点中随机选择的，这可能导致次优解和收敛速度较慢。K-means++ 通过使用更复杂的初始化过程来解决这个问题，我们可以根据结果选择 K。以下是 K-means++ 的简要概述 [3]：

K-means++ 的基本步骤如下：

- 第一步：随机从数据集中选择一个样本作为初始簇中心 c_1。
- 第二步：计算每个样本与当前簇中心的最短距离（即到最近现有簇中心的距离），记为 D(x)。然后，计算每个样本被选为下一个簇中心的概率 P(x)，使用根据这些概率轮盘赌选择方法选择下一个簇中心。簇中心选择概率公式如下：

$$P(x) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}$$

- 第三步：重复第二步，直到簇中心个数为 K。
- 第四步：根据每个点与质心之间的距离对点进行聚类，并通过取每个簇中点的平均值来更新每个簇的质心值。
- 第五步：继续更新质心位置并重新分配点到簇，直到簇中心的位置不再改变。

我们分别将附件 3 食堂菜单中早、午、晚餐的菜品各自依照其食物成分的特定分成五类，通过绘制箱线图的方式直观反映各类别在其主要聚类影响因素上的表现。以早餐为例，其聚类箱线图如下图所示：

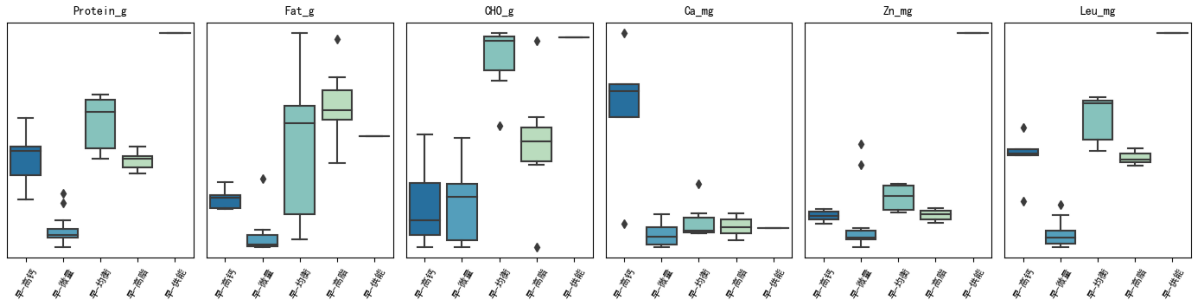


图 4: 早餐聚类箱线图

对于早、午、晚餐菜品聚类得到的五个类别中，我们挑选出几种最具代表性的菜品。这样，我们便将优化算法的变量数从 141 个降维为 35 个，有利于算法的后续实现。相关的聚类结果如下表：

早餐		午餐		晚餐	
类型	代表食物	类型	代表食物	类型	代表食物
高钙	拌豆腐	高蛋白	炒肉扁豆	AAS	干炸黄花鱼
高脂	馅饼	高钙	白菜炖豆腐	均衡	土豆丝饼
均衡	馄饨	AAS	鸡蛋饼	高蛋白	炒肉扁豆
供能	鸡排面	供能	砂锅面	供能	水饺
微量	拌土豆丝	微量	炒豆芽粉条	微量	炒豆芽粉条

表 6: 食物聚类结果表

7.2 以蛋白质氨基酸评分最大为目标的平衡膳食优化模型

7.2.1 传统遗传算法

对于食谱蛋白质氨基酸评分最大化的优化问题，我们将蛋白质氨基酸评分设定为目标函数，将各个营养素的摄入范围、餐次比范围等指标设定为约束条件，建立平衡膳食优化模型，对满足约束条件的目标函数进行规划求解，求取其最大值。由于最终得到的菜品以份数呈现，自变量 x_i 的取值应为整数。因此，模型的建立便转化为求解一种纯整数的规划问题，这种问题通常具有如下的普遍形式：

$$\max Z = \sum_{j=0}^{n} c_j \cdot f(x_j)$$

$$s.t \begin{cases} \sum_{j=1}^{n} a_{ij} x_j = b_i \\ d_{ij} x_j \leq e_{ij} \\ x_j \geq 0, x_j \in z \end{cases}$$

为求解上述方程，我们使用遗传算法优化模型参数。遗传算法是模拟自然界生物进化过程与机制求解极值问题的一类自组织、自适应人工智能技术, 其基本思想是模拟自然界遗传机制和生物进化论而形成的一种过程搜索最优解的算法, 具有坚实的生物学基础 [4]。

在传统遗传算法中，随机生成有若干个体的种群，对该种群每个个体求取对应的目标函数，目标函数最大的若干个体被定义为种群精英，用于种群繁衍进化。下一代种群由上一代精英和他们的子代组成，子代由精英通过交叉和变异产生。交叉即子代个体的特征来自上一代精英特征的随机组合，变异即子代个体的部分特征以一定概率发生突变。由此产生的次代种群最大程度保留了原代种群中能使目标函数最大的特征，并以此为基础进行延伸，使得次代种群在目标函数最大这一种群进化目标上显著优于原代种群，实现优化迭代。

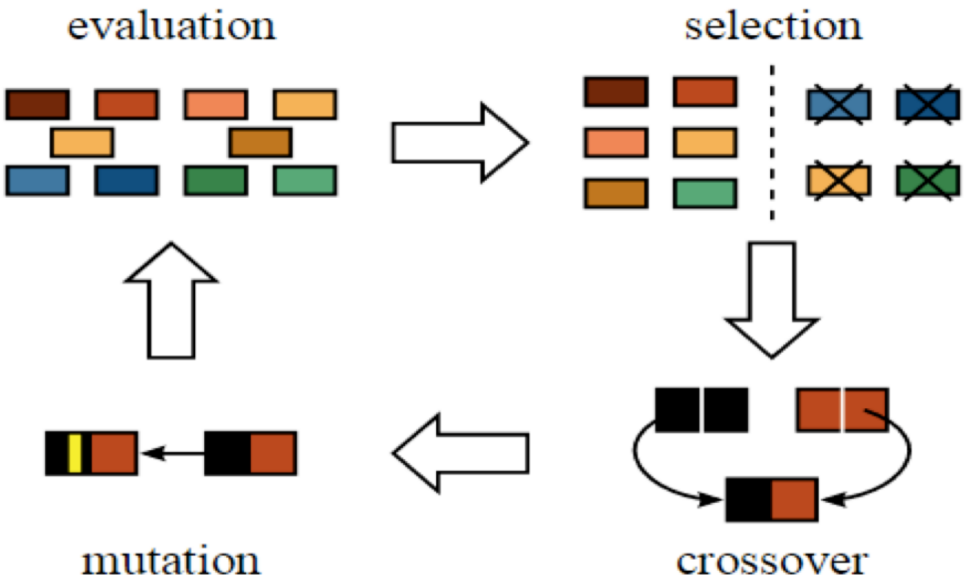


图 5: 遗传算法原理图

7.2.2 “高精微异”遗传算法

在平衡膳食食谱规划这一场景中，即便前期已然通过聚类筛选出 35 个代表性菜品，将变量维度由 141 维降至 35 维，在 35 维空间中寻得可行解仍然如大海捞针。由实际生活常识可知，平衡膳食食谱不可能出现十余个菜品的情形，也不应出现单餐同一菜品选择多次的情形，因此可行解应满足多数变量为零、少数变量非零且较小，在 35 维空间中贴近空间边缘。

显然传统遗传算法将做大量的无用功，极难收敛至可行解，因此需要设计专用遗传算法。我们设计的“高精微异”遗传算法具有以下特点：

- 对于种群中个体的生成，考虑将菜品重新分解为三餐，并随机选择 3 至 6 个非零项，非零项仅在 1 至 4 中取值，如此生成的种群个体更容易接近变量空间边缘。

- 设置超大种群规模，超少精英数量，大幅提高精英质量，使得模型能够以较少的迭代次数确定可行解位置，同时以较少的试错次数排除无效空间。一旦确定可行解位置，后续种群将紧密围绕可行解进行延伸，最大程度弥补了在高维空间不容易得到可行解、个体无效率高的不足。

- 种群延伸无交叉过程，即自交繁育。由于可行解分布于空间边缘，不同可行解之间特征差异巨大，不具有交叉价值。对于高维变量空间中两个可行域，其间各取一个可行解交叉所得到的子代极大概率不落在空间边缘，该子代极大概率对进化贡献为 0。因此，种群进化选择自交繁育，跳过交叉过程。

- 种群变异概率高。由于种群延伸无交叉过程，子代大概率经历变异过程，否则子代将与亲代完全相同，失去繁衍进化意义。

- 种群变异幅度小，种群变异期望与非零特征比例相等。由于精英个体的特征仅有少数非零项，大部分特征均为零，因此精英个体的每个非零特征都是珍贵的，微小的特征变动都将导致子代的空间位置与亲代有较大偏差，致使子代死亡。因此，子代变异过程不能出现大幅度特征变化，特征变异在期望上应与非零特征比例匹配。

上述遗传算法主要有着精英质量高、迭代变化小的特点，能更好地适应可行域小且分散、收敛困难的情况，对平衡膳食食谱规划效果更好、效率更高。

7.2.3 模型设定与优化结果

在本问中，由于种群进化的约束条件均为日平衡膳食约束，因此种群以日食谱为个体，我们将 $maxAAS_{day}$ 确定为种群进化目标，AAS_{day} 称日氨基酸评分，为餐氨基酸评分加权和，定义如下：

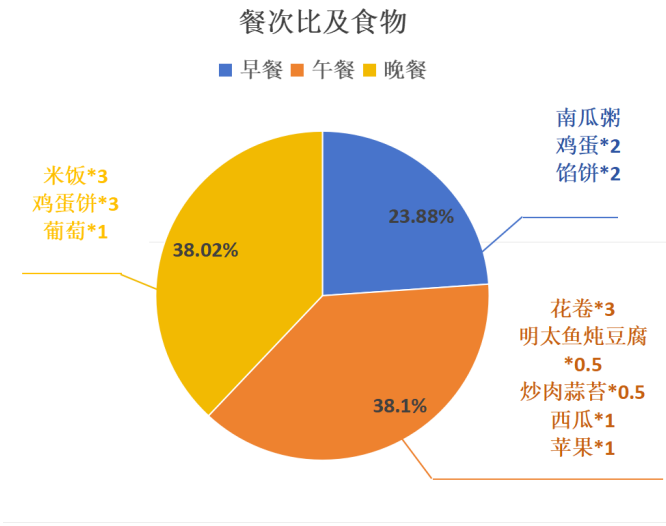
$$AAS_{day} = \sum w_i \cdot AAS_i$$

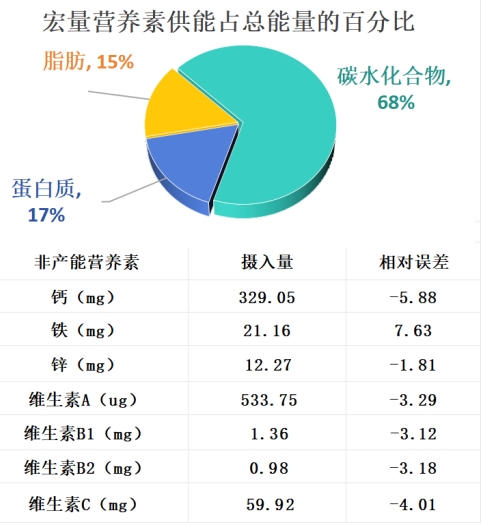
日氨基酸评分定义式中，w_i 为权重，即每餐摄入的蛋白质占日摄入蛋白质的比值，ASS_i 为每餐氨基酸评分，基于附件 4 中的平衡膳食食谱优化设计原则，我们确定以下指标的数值或范围为优化模型的约束条件：

指标		约束	指标		约束
能量摄入		±10%	非产能营养素	钙	±30%
产能营养素	蛋白质	10% - 15%		铁	±30%
	脂肪	20% - 30%		锌	±30%
	碳水化合物	50% - 65%		维生素 A	±30%
餐次比	早餐	10% - 15%		硫胺素	±30%
	午餐	20% - 30%		核黄素	±30%
	晚餐	50% - 65%		维生素 C	±30%

表 7: 以 AAS 最大为目标的优化模型的约束条件

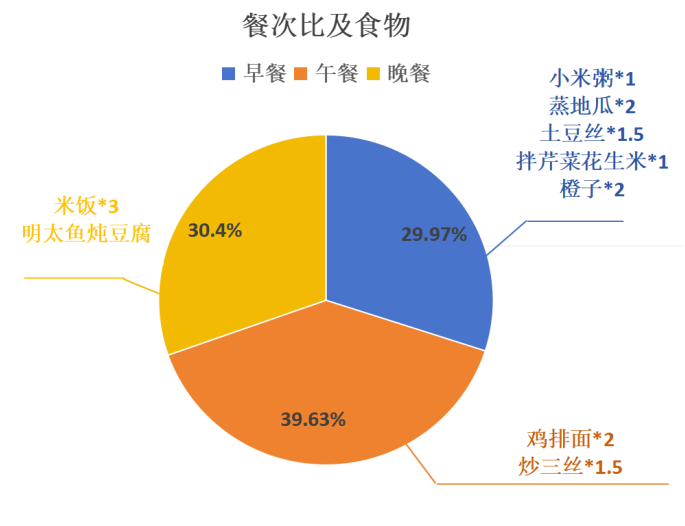
结合代表性菜品和上述约束条件，使用“高精微异”遗传算法进行模型优化求解，种群大小 10000，迭代次数 100 代，得到如下优化后的男女大学生的两份食谱：

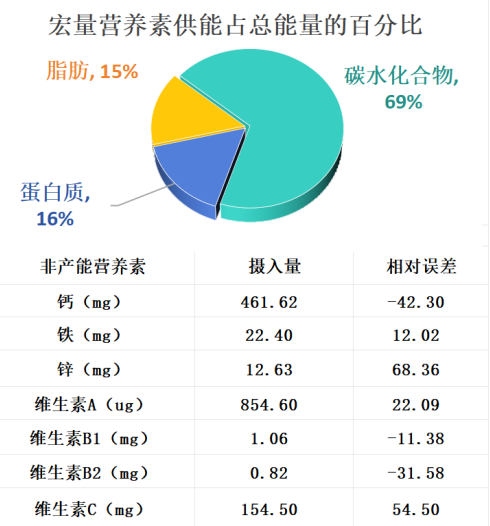




非产能营养素	摄入量	相对误差
钙（mg）	329.05	-5.88
铁（mg）	21.16	7.63
锌（mg）	12.27	-1.81
维生素A（ug）	533.75	-3.29
维生素B1（mg）	1.36	-3.12
维生素B2（mg）	0.98	-3.18
维生素C（mg）	59.92	-4.01

图 6: 男大学生 AAS 优化食谱





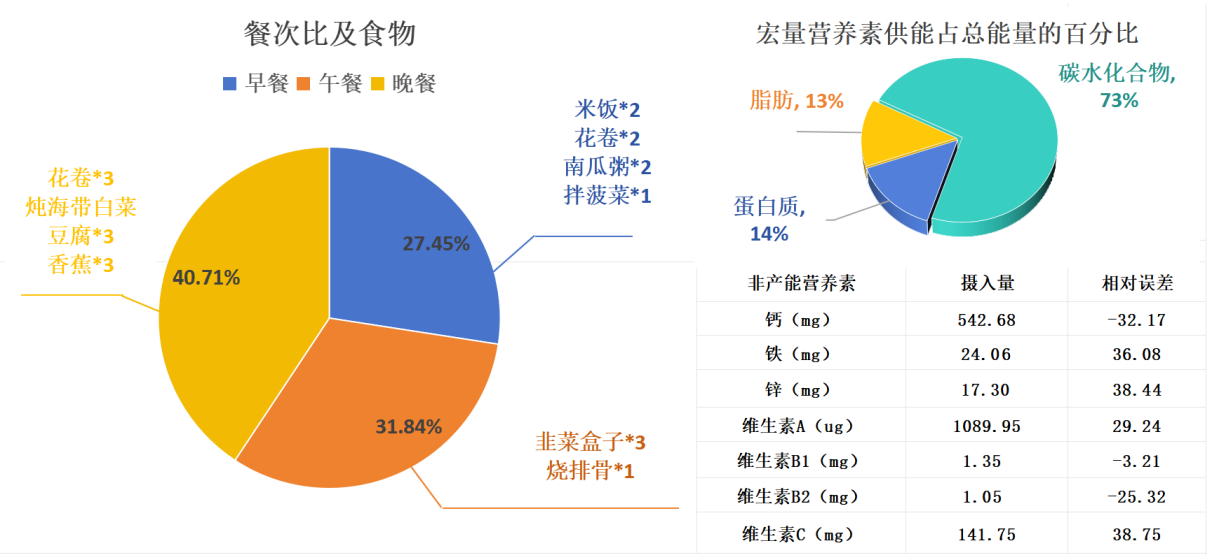
非产能营养素	摄入量	相对误差
钙（mg）	461.62	-42.30
铁（mg）	22.40	12.02
锌（mg）	12.63	68.36
维生素A（ug）	854.60	22.09
维生素B1（mg）	1.06	-11.38
维生素B2（mg）	0.82	-31.58
维生素C（mg）	154.50	54.50

图 7: 女大学生 AAS 优化食谱

使用问题一中所建立的膳食食谱营养分析评价模型对改进后的食谱进行评价，得到以下结果：在以上两份食谱中，男大学生优化食谱 AAS 的评分为 103.13，总评分为 85.63；女大学生优化食谱 AAS 的评分为 98.81，总评分为 83.15。相较于原食谱都有了很大的提高。这是由于我们将 AAS 评分最高设定为唯一的目标函数，并适当放宽了其余指标的约束条件。可以发现，两份食谱的总评分相较于问题一中手动调整的食谱评分都有了显著提高，验证了我们使用遗传算法得到的优化模型是合理的。然而，优化目标的不同和约束条件的范围仍会影响食谱的总体评分，在接下来的分析中我们通过调整这些参数来不断探索评分更高的优秀食谱。

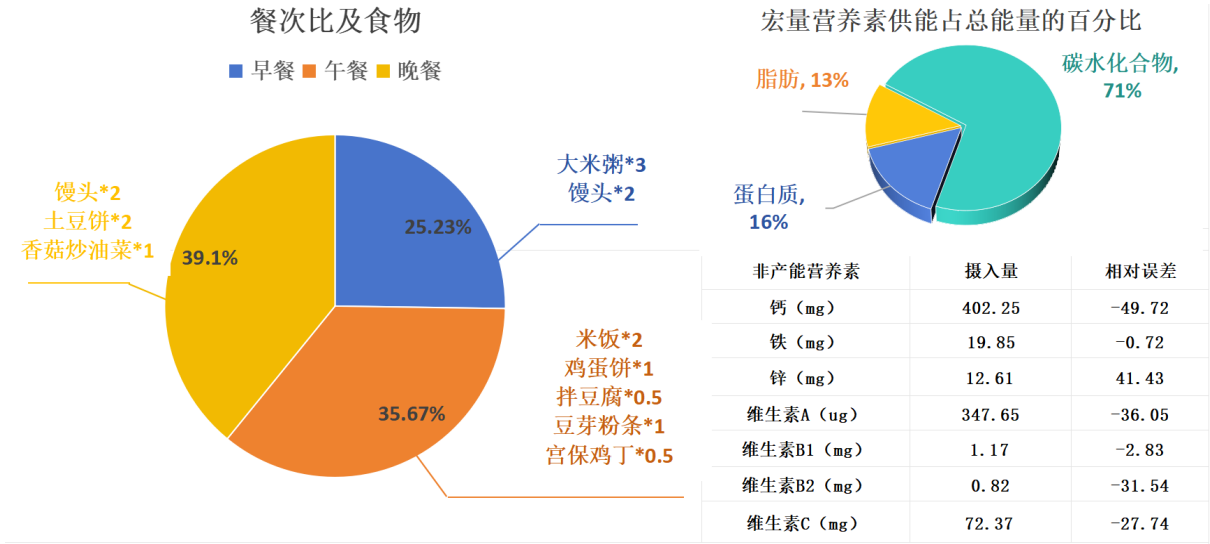
7.3 以用餐费用最经济为目标的优化模型

这一问的求解方法与上一问相似，不同的是将目标函数改变为 $min\sum Price_i$，得到如下优化后的两份食谱：



非产能营养素	摄入量	相对误差
钙（mg）	542.68	-32.17
铁（mg）	24.06	36.08
锌（mg）	17.30	38.44
维生素A（ug）	1089.95	29.24
维生素B1（mg）	1.35	-3.21
维生素B2（mg）	1.05	-25.32
维生素C（mg）	141.75	38.75

图 8: 男大学生价格最低优化食谱



非产能营养素	摄入量	相对误差
钙（mg）	402.25	-49.72
铁（mg）	19.85	-0.72
锌（mg）	12.61	41.43
维生素A（ug）	347.65	-36.05
维生素B1（mg）	1.17	-2.83
维生素B2（mg）	0.82	-31.54
维生素C（mg）	72.37	-27.74

图 9: 女大学生价格最低优化食谱

在这两份食谱中，男大学生的一日消费为 26.5 元，总评分为 81.52；女大学生则为 26.5 元，总评分为 78.47，在满足前一问的约束条件同时保证价格较低。我们发现，这样的设计虽然可以保证目标函数，即价格较低的实现，但食谱的总评分相较于以 AAS 为目标的优化大幅降低。这是由于本问并没有使用对总体食谱评分有影响的指标作为目标函数。因此，我们进一步探究了在保证食谱评分的同时兼顾经济性的可能优化模型。

7.4 兼顾蛋白质氨基酸评分及经济性的优化模型

对于多目标的优化问题，我们选择使用帕累托最优进行求解。帕累托最优（Pareto Optimality），又称帕累托效率（Pareto Efficiency），是经济学和多目标优化中的一个核心概念，描述了一种资源分配方式，其中无法通过重新分配资源来使一个个体的状况变得更好而不使另一个个体的状况变得更糟。在多目标优化问题中，帕累托最优解（Pareto Optimal Solution）是这样的一种解，即不存在另一个解可以在某些目标上取得更好的结果，而在其他目标上不变或更好 [5]。具体地说，对于一个多目标优化问题：

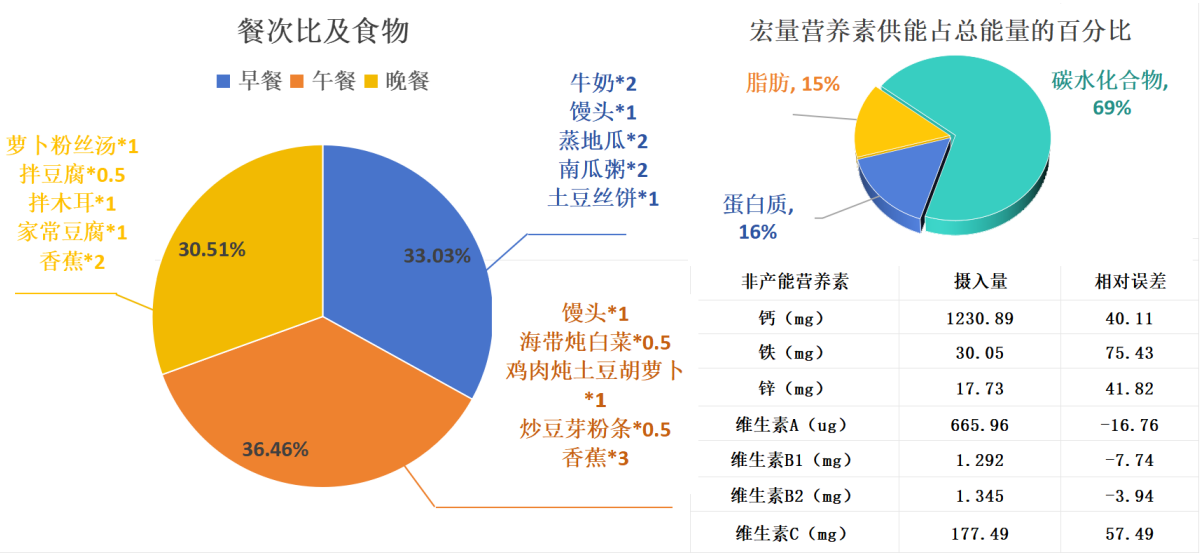
$$\max f(x) = (f_1(x), f_2(x), ..., f_m(x))$$

其中 $f(x)$ 是目标函数向量，x 是决策变量向量。如果 x^* 是帕累托最优的，那么不存在另一个解 x' 满足：

- 对于所有的 i 有 $f_i(x') \geq f_i(x^*)$；
- 并且至少存在一个 j 使得 $f_j(x') > f_j(x^*)$。

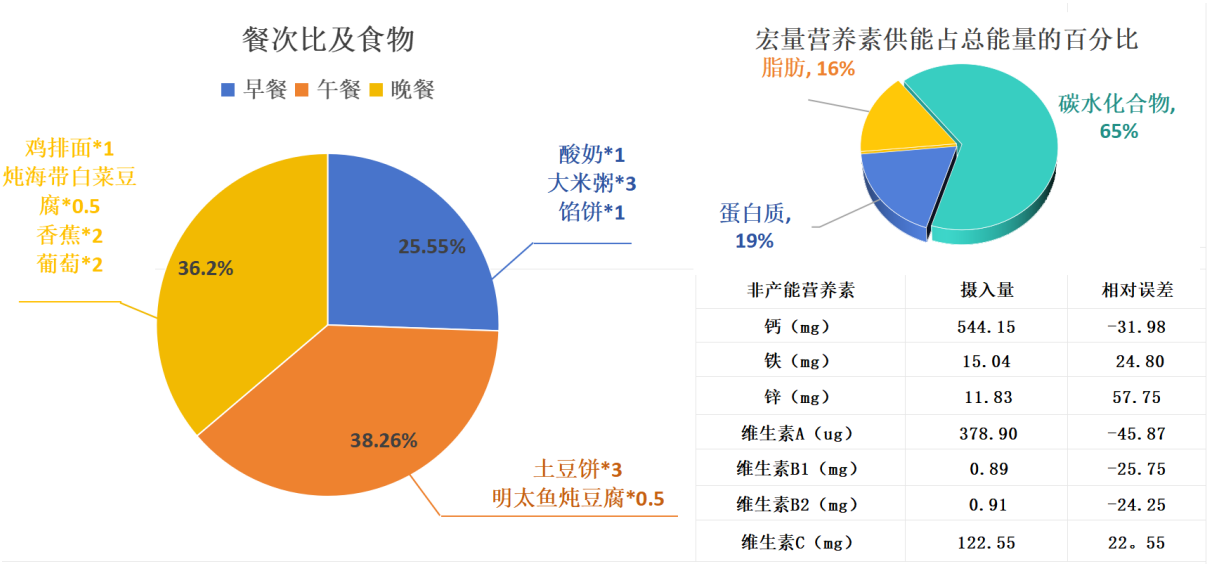
帕累托最优解的集合称为帕累托前沿（Pareto Front）。在帕累托前沿上的所有解都是不可被其他解完全支配的。这意味着在多目标空间中，帕累托前沿展示了所有非劣解的集合。

将帕累托最优应用于“高精微异”遗传算法，所有非劣解均被认为是种群精英加入下一代种群参与进化。种群大小为 5000，迭代次数 50 代，最后一代所得所有精英代表了不同非劣解的最优情况，取分价比最高值为所求情况。



非产能营养素	摄入量	相对误差
钙（mg）	1230.89	40.11
铁（mg）	30.05	75.43
锌（mg）	17.73	41.82
维生素A（ug）	665.96	-16.76
维生素B1（mg）	1.292	-7.74
维生素B2（mg）	1.345	-3.94
维生素C（mg）	177.49	57.49

图 10: 帕累托最优的男大学生优化食谱



非产能营养素	摄入量	相对误差
钙（mg）	544.15	-31.98
铁（mg）	15.04	24.80
锌（mg）	11.83	57.75
维生素A（ug）	378.90	-45.87
维生素B1（mg）	0.89	-25.75
维生素B2（mg）	0.91	-24.25
维生素C（mg）	122.55	22。55

图 11: 帕累托最优的女大学生优化食谱

以上两份多目标优化的食谱中，男大学生的总体评分为 83.75 分，女大学生为 80.28 分，且 AAS 评分和价格评分均位于前两小问所得的结果之间。这意味着我们得到了可以兼顾两个目标的最优解，即在保证一定的 AAS 和价格评分的同时维持总体评分较高。这样得到的优化食谱更加贴近生活，具有显著的实际意义。

7.5 不同目标优化模型的分析比较

优化目标		氨基酸评分（AAS）	价格	总分
男大学生	AAS 最优	103.13	39	85.63
	价格最低	92.56	26.5	81.52
	AAS 与价格兼顾	98.74	29	83.75
女大学生	AAS 最优	98.81	36	83.15
	价格最低	85.62	26.5	78.47
	AAS 与价格兼顾	97.10	25.5	82.28

表 8: 男、女大学生在不同目标优化模型的食谱分析比较表

通过上表可以看出，在以 AAS 最优为优化目标时，两份食谱的 AAS 评分和总分均为三项中最高，但与之相对地，食谱价格也为最高。在以价格最低为目标时，优化模型保证了食谱的经济性，但总体评分较低。而在兼顾 AAS 评分和价格这两个因素时，AAS 评分，价格和食谱的总体评分均介于两种单目标优化模型得到的食谱之间。这样的结果符合优化模型的逻辑，即为了满足单一目标，需牺牲其余某些指标的优越性。而以多目标进行优化则可以保证这些目标同时处于较为优越的状态。

采用遗传算法设计的优化模型可以很好地适应不同的优化目标和约束条件，而依托于帕累托最优，我们也能找到兼顾两个目标的优化食谱结果。

8 问题三：周平衡膳食食谱的优化设计

在周平衡膳食食谱的设计中，我们首先将问题二得到的食谱作为星期一的食谱，找出其中相对误差较大和较小的指标，分别缩小和放大其约束条件的范围，利用问题二的模型进行优化设计得到星期二的食谱，以此类推最终得到周平衡食谱，然后进行验证是否满足平衡膳食的基本准则中，每周摄入食物种类大于 25 种的要求。

由于本问中三种目标函数与问题二中相同，我们重复运行优化算法，并使用评价模型进行打分，得到男大学生，女大学生的周食谱及得分结果如下：

	日期	餐次	菜品		日期	餐次	菜品
男大学生	星期一	早餐	牛奶*1 蒸地瓜*3 馅饼*1 鸡蛋饼*1	女大学生	星期一	早餐	大米粥*3 馒头*2
		午餐	馒头*3 菠菜汤*1 拌豆腐*1 卷心菜抄木耳*1			午餐	大米饭*2 鸡蛋饼*1 拌豆腐*0.5 拌豆芽粉条*1 宫保鸡丁*0.5
		晚餐	花卷*3 韭菜盒子*2 香菇炒油菜*1			晚餐	馒头*2 土豆丝饼*2 香菇炒油菜*1
	星期二	早餐	南瓜粥*3 馅饼*2		星期二	早餐	牛奶*2 小米粥*3 拌海带丝*1.5
		午餐	花卷*3 明太鱼炖豆腐*1 炒肉蒜苔*1 西瓜*1 苹果*1			午餐	豆浆*3 土豆丝饼*2 海带炖白菜*1 西瓜*2
		晚餐	大米饭*2 鸡蛋饼*3 葡萄*1			晚餐	花卷*1 炒肉扁豆*0.5 香蕉*2
	星期三	早餐	牛奶*2 蒸地瓜*3 拌土豆丝*1		星期三	早餐	大米饭*2 南瓜粥*3 土豆丝饼*2 拌海带丝*1
		午餐	南瓜粥*3 韭菜盒子*1 香菇炒油菜*1 木须柿子*1			午餐	菠菜汤*3 拌豆腐*1 炒豆芽粉条*0.5 苹果*3 葡萄*2
		晚餐	花卷*3 土豆丝饼*1			晚餐	大米饭*2 馒头*2 西瓜*2
	星期四	早餐	牛奶*2 大米粥*2 南瓜粥*3 土豆丝饼*1 拌干豆腐*1		星期四	早餐	花卷*3 拌干豆腐*1 拌土豆丝*1
		午餐	大米饭*3 包子*3			午餐	花卷*3 炒豆芽粉条*0.5
		晚餐	馒头*2 拌木耳*1 炖海带白菜豆腐*1			晚餐	小米粥*1 鸡蛋饼*2 萝卜粉丝汤*4 卷心菜炒木耳*1.5
	星期五	早餐	馒头*3 拌干豆腐*3		星期五	早餐	酸奶*3 拌菠菜*1 拌土豆丝*1.5
		午餐	包子*1 韭菜盒子*2 宫保鸡丁*1 葡萄*3			午餐	南瓜粥*2 家常豆腐*1 香蕉*2 葡萄*4
		晚餐	大米饭*3 拌干豆腐*1 炒三丝*1.5 炒豆芽粉条*0.5 柚子*1			晚餐	大米饭*4 小米粥*2 炒肉青椒*1
	星期六	早餐	大米饭*2 花卷*2 拌菠菜*1		星期六	早餐	豆浆*1 馒头*3
		午餐	韭菜盒子*3 烧排骨*1			午餐	蒸饺*1 卷心菜炒木耳*0.5 烧排骨*0.5 香蕉*2
		晚餐	花卷*3 炖海带白菜豆腐*1 香蕉*3			晚餐	大米饭*1 土豆丝饼*3 拌豆腐*0.5
	星期日	早餐	蒸地瓜*2 南瓜粥*1 鸡排面*1 拌豆腐*2 苹果*1		星期日	早餐	南瓜粥*2 馄饨*1 拌干豆腐*1 拌芹菜花生米*1.5
		午餐	馒头*3 鸡肉炖土豆胡萝卜*1			午餐	鸡蛋饼*1 炒三丝*1.5 炒肉青椒*1 锅包肉*0.5
		晚餐	馒头*3 炒三丝*1.5			晚餐	花卷*4
得分							
AAS		93.23		AAS		94.09	
价格		204.5		价格		189.5	
总分		81.37		总分		80.28	

图 12: 价格优化的周食谱

	日期	餐次	菜品		日期	餐次	菜品
男大学生	星期一	早餐	南瓜粥*1 煮鸡蛋*2 馅饼*2	女大学生	星期一	早餐	酸奶*1 大米粥*3 馅饼*1
		午餐	花卷*3 明太鱼炖豆腐*0.5 炒肉蒜苔*0.5 西瓜*1 苹果*1			午餐	土豆饼*3 明太鱼炖豆腐*0.5
		晚餐	米饭*3 鸡蛋饼*3 葡萄*1			晚餐	鸡排面*1 香蕉*2 葡萄*2 炖海带白菜豆腐*0.5
	星期二	早餐	米饭*2 花卷*2 南瓜粥*2 拌菠菜*1		星期二	早餐	小米粥*1 蒸地瓜*2 橙于*2 拌土豆丝*1 拌芹菜花生米*1
		午餐	韭菜盒于*3 烧排骨*1			午餐	鸡排面*2 炒三丝*1.5
		晚餐	花卷*3 炖海带白菜豆腐*1 香蕉*3			晚餐	米饭*3 明太鱼炖豆腐*1.5
	星期三	早餐	牛奶*2 馒头*1 蒸地瓜*2 南瓜粥*2 土豆丝饼*1		星期三	早餐	煮鸡蛋*3 蒸地瓜*3 南瓜粥*2 拌木耳*1 橙于*2
		午餐	馒头*1 海带炖白菜*0.5 香蕉*3 苹果*3 鸡肉炖土豆胡萝卜*1 炒豆芽粉条*0.5			午餐	炒豆芽粉条*0.5 木须柿于*0.5 葡萄*3 宫保鸡丁*1 蜜瓜*1炒肉酸菜粉*1
		晚餐	萝卜粉丝汤*1 拌豆腐*0.5 拌木耳*1 家常豆腐*1.5 香蕉*2			晚餐	馒头*3 拌菠菜*1 拌木耳*1 葡萄*2
	星期四	早餐	牛奶*3 大米粥*3 蒸地瓜*3		星期四	早餐	煮鸡蛋*3 蒸地瓜*3 大米粥*2 拌木耳*1 苹果*1
		午餐	蒸饺*2 鸡蛋柿于汤*1 炒芹菜粉*1 炒豆芽粉条*1			午餐	南瓜粥*1 炒豆芽粉条*0.5 木须柿于*0.5 地三鲜*0.5 炒肉酸菜粉*1 宫保鸡丁*1
		晚餐	米饭*4 韭菜盒于*2 明太鱼炖豆腐*0.5 卷心菜炒木耳*1 香蕉*2			晚餐	馒头*3 拌菠菜*0.5 拌木耳*1.5 香蕉*1
	星期五	早餐	牛奶*1 豆浆*1 大米粥*3 馒头*1 煮鸡蛋*1		星期五	早餐	南瓜粥*2 鸡蛋饼*2 拌菠菜*0.5 拌海带丝*0.5
		午餐	鸡排面*1 萝卜粉丝汤*2 鸡肉炖土豆胡萝卜*1			午餐	南瓜粥*4 拌干豆腐*0.5 炒芹菜粉*0.5 炒肉酸菜粉*0.5 香蕉*3
		晚餐	花卷*3 宫保鸡丁*0.5 柚于*1 葡萄*3			晚餐	小米粥*1 拌豆腐*1 炒芹菜粉*0.5 炒三丝*1.5 苹果*3
	星期六	早餐	牛奶*3 花卷*2 拌干豆腐*0.5 拌土豆丝*1		星期六	早餐	牛奶*2 鸡排面*1 拌芹菜花生米*0.5
		午餐	南瓜粥*2 鸡排面*2 卷心菜炒木耳*1 炒肉蒜苔*1			午餐	地三鲜*1.5 西瓜*4 苹果*1
		晚餐	包于*1 萝卜粉丝汤*3 香蕉*4 炖海带白菜豆腐*0.5 炒芹菜粉*1			晚餐	小米粥*4 鸡蛋饼*2 西瓜*3
	星期日	早餐	牛奶*3 大米粥*3 花卷*2 蒸地瓜*2		星期日	早餐	牛奶*1 南瓜粥*2 鸡蛋饼*2 拌海带丝*0.5
		午餐	蒸饺*3 萝卜粉丝汤*1			午餐	馒头*2 南瓜粥*4 拌干豆腐*0.5 炒肉酸菜粉*0.5
		晚餐	花卷*1 水饺*1 拌干豆腐*1.5 拌木耳*0.5			晚餐	小米粥*1 明太鱼炖豆腐*0.5 炒芹菜粉*0.5 炒三丝*1.5 苹果*1
得分							
AAS		100.36		AAS		97.88	
价格		257		价格		226	
总分		86.72		总分		84.57	

图 13: AAS 优化的周食谱

	日期	餐次	菜品		日期	餐次	菜品
男大学生	星期一	早餐	牛奶*2 馒头*1 蒸地瓜*2 大米粥*2 馅饼*1	女大学生	星期一	早餐	小米粥*1 蒸地瓜*2 橙子*2 拌土豆丝*1 拌芹菜花生米*1
		午餐	韭菜盒子*3 烧排骨*1			午餐	
		晚餐	米饭*4 韭菜盒子*2 明太鱼炖豆腐*0.5 卷心菜炒木耳*1 香蕉*2			晚餐	小米粥*1 鸡蛋饼*2 萝卜粉丝汤*4 卷心菜炒木耳*1.5
	星期二	早餐	牛奶*1 豆浆*1 大米粥*3 馒头*1 煮鸡蛋*1		星期二	早餐	酸奶*3 拌菠菜*1 拌土豆丝*1.5
		午餐	馒头*1 海带炖白菜*0.5 香蕉*3 苹果*3 鸡肉炖土豆胡萝卜*1 炒豆芽粉条*0.5			午餐	鸡排面*2 炒三丝*1.5
		晚餐	花卷*3 炖海带白菜豆腐*1 香蕉*3			晚餐	馒头*3 拌菠菜*1 拌木耳*1 葡萄*2
	星期三	早餐	牛奶*3 大米粥*3 花卷*2 蒸地瓜*2		星期三	早餐	南瓜粥*2 鸡蛋饼*2 拌菠菜*0.5 拌海带丝*0.5
		午餐	花卷*3 明太鱼炖豆腐*0.5 炒肉蒜苔*0.5 西瓜*1 苹果*1			午餐	鸡蛋饼*1 炒三丝*1.5 炒肉青椒*1 锅包肉*0.5
		晚餐	馒头*2 拌木耳*1 炖海带白菜豆腐*1			晚餐	大米饭*2 馒头*2 西瓜*2
	星期四	早餐	蒸地瓜*2 南瓜粥*1 鸡排面*1 拌豆腐*2 苹果*1		星期四	早餐	酸奶*1 大米粥*3 馅饼*1
		午餐	馒头*3 鸡肉炖土豆胡萝卜*1			午餐	南瓜粥*4 拌干豆腐*0.5 炒芹菜粉*0.5 炒肉酸菜粉*0.5 香蕉*3
		晚餐	花卷*3 韭菜盒子*2 香菇炒油菜*1			晚餐	大米饭*4 小米粥*2 炒肉青椒*1
	星期五	早餐	馒头*3 拌干豆腐*3		星期五	早餐	大米粥*3 馒头*2
		午餐	蒸饺*2 鸡蛋柿子汤*1 炒芹菜粉*1 炒豆芽粉条*1			午餐	菠菜汤*3 拌豆腐*1 炒豆芽粉条*0.5 苹果*3 葡萄*2
		晚餐	花卷*3 炖海带白菜豆腐*1 香蕉*3			晚餐	小米粥*4 鸡蛋饼*2 西瓜*3
	星期六	早餐	牛奶*2 蒸地瓜*3 拌土豆丝*1		星期六	早餐	南瓜粥*2 馄饨*1 拌干豆腐*1 拌芹菜花生米*1.5
		午餐	南瓜粥*3 韭菜盒子*1 香菇炒油菜*1 木须柿子*1			午餐	馒头*2 南瓜粥*4 拌干豆腐*0.5 炒肉酸菜粉*0.5
		晚餐	米饭*3 鸡蛋饼*3 葡萄*1			晚餐	米饭*3 明太鱼炖豆腐*1.5
	星期日	早餐	牛奶*1 蒸地瓜*3 馅饼*1 鸡蛋饼*1		星期日	早餐	花卷*3 拌干豆腐*1 拌土豆丝*1
		午餐	馒头*3 菠菜汤*1 拌豆腐*1 卷心菜抄木耳*1			午餐	豆浆*3 土豆丝饼*2 海带炖白菜*1 西瓜*2
		晚餐	大米饭*3 拌干豆腐*1 炒三丝*1.5 炒豆芽粉条*0.5 柚子*1			晚餐	馒头*2 土豆丝饼*2 香菇炒油菜*1
得分							
AAS		96.73		AAS		95.69	
价格		234.5		价格		200.5	
总分		84.68		总分		82.36	

图 14: 兼顾 AAS 和价格优化的周食谱

通过以上以不同优化目标设计的周食谱，其总体评分均较高，满足平衡膳食的原则。遵循以上食谱的餐次比和搭配规则，男女生在一周中的每一天均可获得足够的营养摄入，产能营养素和非产能营养素。然而，在由算法优化得到的食谱难免会存在不符合实际生活情况之处，例如菜品中包含除食物主要成分之外的水等其他成分，会使菜谱中的每个菜品的摄入小于真实情况。这便需要使用该食谱者结合自身身体状况进行调整，以在保证膳食平衡的同时满足个人口味需求以及饥饿程度。

9　问题四：大学生膳食平衡倡议书

亲爱的大学生朋友们：

大学生活是我们人生中的重要阶段，也是我们身体发育和健康管理的关键时期。良好的饮食习惯不仅有助于我们的学习和生活，更是我们长期健康的基础。在此，我们倡议大家一起关注健康饮食，践行平衡膳食的原则。

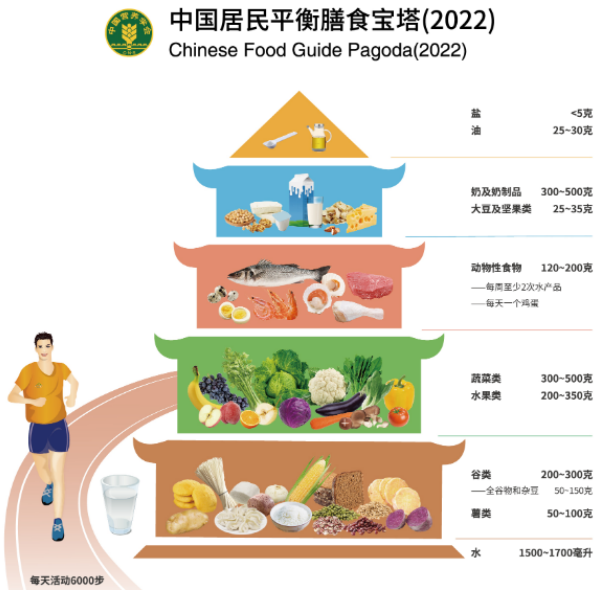


图 15: 中国居民平衡膳食宝塔

1. 食物多样，种类丰富：在我们的饮食中，应尽可能地选择不同种类的食物。我们每天摄入的食物要包括五大类别，分别为：谷、薯类；蔬菜、菌藻、水果类；畜、禽、鱼、蛋类及制品；奶、干豆、坚果、种子类及制品和植物油类。我们每天摄入的食物种类应该多于 12 种，每周更应达到 25 种以上。这样的食物多样性可以确保我们获得各种必需的营养素，促进身体的全面健康发展。

2. 合理安排餐次比：

- 早餐：占总能量的 30%

  早餐是一天中最重要的一餐，它为我们提供了一天所需的能量和营养。合理的早餐应包括谷物类（如全麦面包、燕麦）、蛋白质（如鸡蛋、牛奶）和水果。这样的搭配能够提供持久的能量，帮助我们集中注意力和保持活力。

- 午餐：占总能量的 30-40%

  午餐应该是一天中的第二大餐，提供足够的能量和营养来支撑下午的学习和工作。午餐应包括丰富的蔬菜、适量的蛋白质（如鱼、禽肉、豆制品）和碳水化合物（如米饭、面食、杂粮）。通过合理搭配，我们能够获得多种营养素，保持饱腹感和精力充沛的状态。

- 晚餐：占总能量的 30-40%

  晚餐应该是一天中最轻盈的一餐，以免影响睡眠质量。晚餐应包括丰富的蔬菜、适量的蛋白质和适量的碳水化合物。选择低脂、低盐、低糖的烹调方式，如蒸、煮、炖、焯等，有助于保持餐食的健康。

3. 满足营养素定量要求：我们的食谱应包含丰富的营养素，包括蛋白质、脂肪、碳水化合物、维生素和矿物质等。合理的营养摄入比例是蛋白质占 10%，脂肪占 20%，碳水化合物占 50%。我们应该选择高质量的蛋白质，如鱼肉、禽肉、豆类和奶制品，并适量摄入健康的脂肪，如橄榄油、坚果等。同时，增加蔬菜和水果的摄入量，以获取丰富的维生素和矿物质，促进身体的健康发展。

男性和女性的饮食需求存在显著差异，这些差异受到生理结构、代谢率、激素水平和生命阶段的影响。了解这些差异并根据性别的不同调整饮食是维护健康和获得最佳营养的关键。每个人都应该采取个性化的饮食策略，以满足其特定的营养需求，确保长期健康和幸福。

因此，对于男女生，我们有如下不同的针对性建议：

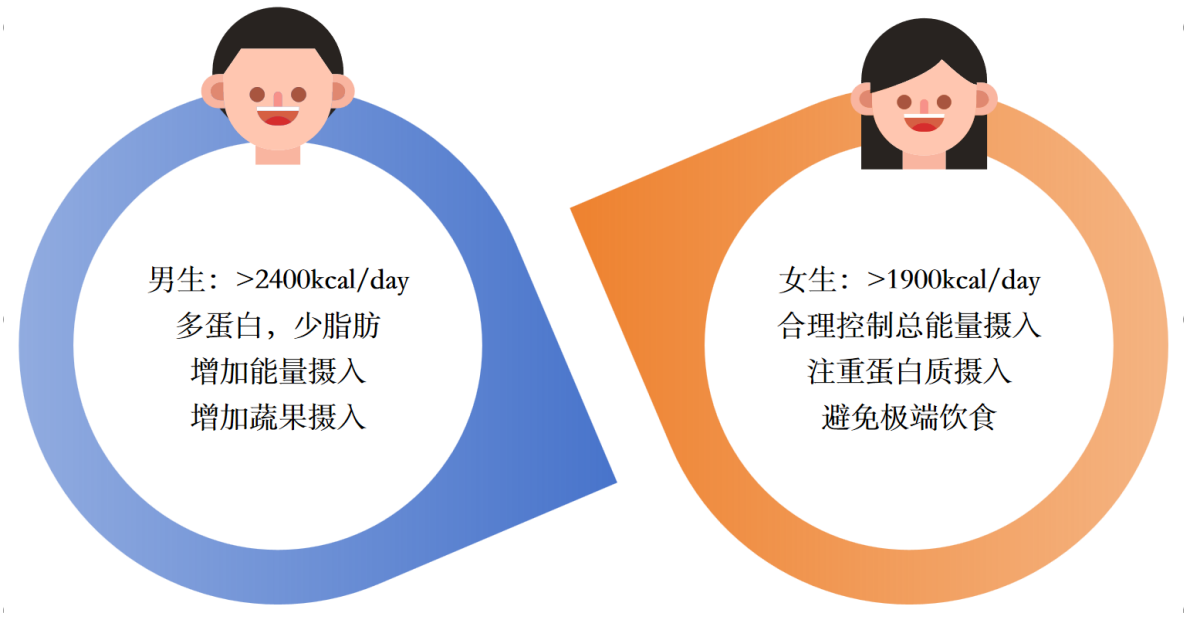


图 16: 男女生膳食建议

合理的饮食结构再加上科学规律的运动可以让男生的“肌肉梦”，女生的“马甲线”走进现实，同时会让激素的分泌保持在正常范围内，不仅会减少脱发的风险而且也会让皮肤更加细腻。

希望以上建议能够帮助大家养成良好的饮食习惯，为健康的身体和学业进步打下坚实的基础！祝大家身体健康，学业顺利！

10 模型评价

10.1 模型优点

- 规划评分体系合理：
  我们使用熵权法确定各个指标权重后进行综合评分，体现了各个因素对于整体食谱的重要程度，可以有效合理地评价一份食谱的优劣程度。

- 优化效果卓越：
  经过多次尝试，我们选择了在解决优化问题上表现良好的遗传算法，可以很好地适应不同的优化目标和约束条件。经过优化后的食谱在满足膳食平衡条件的同时整体评分升高。

- 数据降维，减小模型复杂度：
  采取聚类的方法选出代表性食物作为优化模型输入，提升算法运行速度，同时避免了可能出现的过拟合问题。

10.2 模型缺点

- 针对性不强：
  模型建立的数据基础不够全面，没有考虑到不同能量消耗、不同体质的人群对营养需求不同以及不同的口味偏好。

- 没有考虑可食性：
  优化设计得到的食谱只考虑了营养，没有考虑搭配食用时的体验感。

参考文献

[1] 杨月欣，王光亚，潘兴昌，《中国食物成分表第一册第 2 版》，北京：北京大学医学出版社，2009，4-255

[2] 疯狂绅士, 什么叫做” 基于熵权的 TOPSIS 综合评价法”?,
https://www.zhihu.com/question/546564958/answer/2705275789,2024.5.25

[3] pentiumCM，机器学习—K-Means、K-Means++ 原理及算法实现，
https://blog.csdn.net/pentiumCM/article/details/103657283，2024.5.25

[4] 葛继科，邱玉辉，吴春明，蒲国林，遗传算法研究综述 [J]，计算机应用研究，2008

[5] 刘建军，司光亚，王艳正，何大川，基于模型的多目标优化问题方法研究 [J]，系统仿真学报，2020

A 附录 1

数据预处理代码

```
from utils_algorithm import *

def evaluation(data):
    # start_time = time.time()

    data = [int(text) for text in data]
    if data.count(0)<x_n-20:
        return [-100]

    brk=np.array(data[:brk_len])[:,np.newaxis]
    lun=np.array(data[brk_len:lun_pos])[:,np.newaxis]
    din=np.array(data[lun_pos:])[:,np.newaxis]

    to_preprocess = ['brk','lun','din']
    to_diet = [brk,lun,din]
    to_menu = [menu_brk,menu_lun,menu_din]
    to_food = [comp_brk,comp_lun,comp_din]
    # to_mapp = [mapping_brk,mapping_lun,mapping_din]
    report = np.empty((3,25))

    comps = [0] * 61
    types = [False] * 5

    for index, meal in enumerate(to_preprocess):
        diet = to_diet[index]
        menu = to_menu[index]
        food = to_food[index]
        # mapp = to_mapp[index]

        ans = diet * menu
        ans = np.around(np.sum(ans,axis=0), 2)

        AAS = round(min(ans[13]/ans[2]/40, ans[14]/ans[2]/70, ans[15]/ans
```

```
        [2]/55, ans[16]/ans[2]/35, ans[17]/ans[2]/60, ans[18]/ans
        [2]/40, ans[19]/ans[2]/10, ans[20]/ans[2]/50) * 100, 2)

    report[index,0:22] = ans
    report[index,22] = AAS

    diet = diet.flatten().tolist()
    for indexx, roww in food.iterrows():
        if diet[roww['id']-1]!=0: # origin
            count = roww['amount'] * diet[roww['id']-1]
        # if diet[mapp.index(roww['id'])]!=0: # new
        # count = roww['amount'] * diet[mapp.index(roww['id'])]
            comps[roww['pos']] += count
            types[roww['type']-1] = True

daily = np.around(np.sum(report,axis=0), 2)

Ratio_error_brk = abs(cal_error(round(report[0,0] / daily[0] * 100, 2)
    , 30))
if Ratio_error_brk>25:
    return [-100]
Ratio_error_lun = abs(cal_error(round(report[1,0] / daily[0] * 100, 2)
    , 35))
if Ratio_error_lun>20:
    return [-100]
Ratio_error_din = abs(cal_error(round(report[2,0] / daily[0] * 100, 2)
    , 35))
if Ratio_error_din>20:
    return [-100]

Protein_error = abs(cal_error(round(daily[2] *4 / daily[0], 2), 0.125)
    )
if Protein_error>20:
    return [-100]
Fat_error = abs(cal_error(round(daily[3] *9 / daily[0], 2), 0.25))
if Fat_error>20:
    return [-100]
```

```
CHO_error = abs(cal_error(round(daily[4] *4 / daily[0], 2), 0.575))
# if CHO_error>13:
# return [-100]

Energy_error = abs(cal_error(daily[0], ref[0]))
if Energy_error>10:
    return [-100]
VA_error = abs(cal_error(daily[6], ref[1]))
VB1_error = abs(cal_error(daily[7], ref[2]))
VB2_error = abs(cal_error(daily[8], ref[3]))
VC_error = abs(cal_error(daily[9], ref[4]))
Ca_error = abs(cal_error(daily[10], ref[5]))
Fe_error = abs(cal_error(daily[11], ref[6]))
Zn_error = abs(cal_error(daily[12], ref[7]))

AAS_report = round((report[0,22]*report[0,2] + report[1,22]*report
    [1,2] + report[2,22]*report[2,2]) / daily[2], 2)
Price_report = daily[21]

count = 5 - types.count(False)
if count!=5:
    return [-100]

count = 61 - comps.count(0)
if count<12:
    return [-100]

# runtime = round(time.time() - start_time, 6)


if flag:
    print(Energy_error, Protein_error, Fat_error, CHO_error, VA_error,
         VB1_error, VB2_error, VC_error, Ca_error, Fe_error, Zn_error,
        Ratio_error_brk, Ratio_error_lun, Ratio_error_din, AAS_report,
        Price_report, Type_report, Comps_report, runtime)
else:
    # if Comps_report==False or Type_report==False or Energy_error>10
```

```
            or Protein_error>20 or Fat_error>20 or CHO_error>13 or VA_error
            >30 or VB1_error>30 or VB2_error>30 or VC_error>30 or Ca_error
            >30 or Fe_error>30 or Zn_error>30 or Ratio_error_brk>16.7 or
            Ratio_error_lun>14.3 or Ratio_error_din>14.3:
        # return [AAS_report]
        return [-Price_report]

if __name__=='__main__':
    flag = 1
    brk = pd.read_csv(f'./data/{diet_folder}/{sex}_brk.csv',index_col=
        False)
    brk = brk['num'].tolist()
    lun = pd.read_csv(f'./data/{diet_folder}/{sex}_lun.csv',index_col=
        False)
    lun = lun['num'].tolist()
    din = pd.read_csv(f'./data/{diet_folder}/{sex}_din.csv',index_col=
        False)
    din = din['num'].tolist()

    data = brk + lun + din

    evaluation(data)
```

B 附录 2

遗传算法代码

```
    from utils_algorithm import *
from diet_evaluate_optim import evaluation

NGEN = 10 # 迭代次数
MU = 50000 # 种群大小
CXPB = 0.8 # 交叉概率
NELITE = 10 # 精英个数

def generate_individual():
    individual_brk = [0] * brk_len
    non_zero_indices = rn.sample(range(brk_len), rn.randint(2, 6))
```

```
    for index in non_zero_indices:
        individual_brk[index] = rn.randint(1, 3)

    individual_lun = [0] * lun_len
    non_zero_indices = rn.sample(range(lun_len), rn.randint(2, 6))
    for index in non_zero_indices:
        individual_lun[index] = rn.randint(1, 3)

    individual_din = [0] * din_len
    non_zero_indices = rn.sample(range(din_len), rn.randint(2, 6))
    for index in non_zero_indices:
        individual_din[index] = rn.randint(1, 3)

    individual = individual_brk + individual_lun + individual_din

    return individual

def evaluate_population(population):
    return [evaluation(individual) for individual in population]

def select_elites(population, scores):
    elite_indices = sorted(range(len(scores)), key=lambda i: scores[i],
        reverse=True)[:NELITE]
    return [population[i] for i in elite_indices]

def crossover(parent1, parent2):
    crossover_point = rn.randint(1, x_n - 1)
    child1 = parent1[:crossover_point] + parent2[crossover_point:]
    child2 = parent2[:crossover_point] + parent1[crossover_point:]
    return child1, child2

def mutate(individual):
    mutated_individual = individual[:]
    indices = rn.sample(range(x_n), k=kk)
    for i in indices:
        if mutated_individual[i]==0:
            if rn.random() < MUTPB:
```

```
                mutated_individual[i] = rn.randint(1, 4)
        else:
            mutated_individual[i] = 0
    return mutated_individual

def genetic_algorithm():
    population = [generate_individual() for _ in range(MU)]
    for index in range(NGEN):
        t = time.time()
        scores = evaluate_population(population)
        elites = select_elites(population, scores)
        best_gen = -evaluation(elites[0])[0]
        t = time.time() - t

        for results in elites:
            gen = -evaluation(results)[0]
            print(f'''GEN:{index} runtime:{t}''')
            print(f'''elites:{results}''')
            print(f'''score:{gen}''')
            if gen<50:
                pd.DataFrame(results).to_csv(f'./data/{prob}_report_{gen}.
                    csv')
        if best_gen>50:
            return 0, 0
        next_generation = elites[:]
        while len(next_generation) < MU:
            parent1, parent2 = rn.choices(population, k=2)
            # child1, child2 = crossover(parent1, parent2)
            # child1 = mutate(child1)
            # child2 = mutate(child2)
            child1 = mutate(parent1)
            child2 = mutate(parent2)
            next_generation.extend([child1, child2])
        population = next_generation

    best_individual = max(population, key=lambda x: evaluation(x))
    best_score = -evaluation(best_individual)
```

```
    repeat = False
    return best_individual, best_score

if __name__=='__main__':
    repeat = True
    while(repeat):
        best_solution, best_score = genetic_algorithm()

    print("Best solution:", best_solution)
    print("Best score:", best_score)
```

C 附录 3

数据分析代码

```
    from utils_algorithm import *
from result_report import analysis

mark = '32.5'

file = f'./data/{prob}_report_{mark}.csv'

content = pd.read_csv(file,index_col=False)

diet = content['0'].tolist()

brk = pd.DataFrame(diet[:brk_len],columns=['num'])
lun = pd.DataFrame(diet[brk_len:lun_pos],columns=['num'])
din = pd.DataFrame(diet[lun_pos:],columns=['num'])

brk['id'] = range(1,brk_len+1)
lun['id'] = range(1,lun_len+1)
din['id'] = range(1,din_len+1)

brk = brk.reindex(columns=['id','num'])
lun = lun.reindex(columns=['id','num'])
din = din.reindex(columns=['id','num'])
```

```
brk.to_csv(f'./data/report/brk_{prob}_{mark}.csv',index=False)
lun.to_csv(f'./data/report/lun_{prob}_{mark}.csv',index=False)
din.to_csv(f'./data/report/din_{prob}_{mark}.csv',index=False)

content_path = file[:-4]

analysis(mark)

from utils_algorithm import *

def analysis(mark):
    report = pd.DataFrame(columns=["id", "Energy_kcal", "Water_g", "
        Protein_g", "Fat_g", "CHO_g", "Fiber_g", "VA_ug", "VB1_mg", "
        VB2_mg", "VC_mg", "Ca_mg", "Fe_mg", "Zn_mg", "Ile_mg", "Leu_mg", "
        Lys_mg", "SAA_mg", "AAA_mg", "Thr_mg", "Trp_mg", "Val_mg", "price
        ", "AAS", "Meal_Ratio", "Ratio_error"])

    if sex=='male':
        ref = ref_male
    if sex=='female':
        ref = ref_female

    to_preprocess = ['brk','lun','din']

    for meal in to_preprocess:
        content = pd.read_csv(f'./data/report/{meal}_{prob}_{mark}.csv',
            index_col=False)
        menu = pd.read_csv(f'./data/{menu_folder}/menu_{meal}.csv',
            index_col=False)

        diet = [f'{meal}'] + [0.0] * 25

        for index, row in content.iterrows():
            if row['num']!=0:
                food = row['id']
                num = row['num']
                diet[1]+=menu.loc[menu['id']==food,'Energy_kcal'].values[0]
```

```
    * num
diet[2]+=menu.loc[menu['id']==food,'Water_g'].values[0] *
   num
diet[3]+=menu.loc[menu['id']==food,'Protein_g'].values[0] *
    num
diet[4]+=menu.loc[menu['id']==food,'Fat_g'].values[0] * num
diet[5]+=menu.loc[menu['id']==food,'CHO_g'].values[0] * num
diet[6]+=menu.loc[menu['id']==food,'Fiber_g'].values[0] *
   num
diet[7]+=menu.loc[menu['id']==food,'VA_ug'].values[0] * num
diet[8]+=menu.loc[menu['id']==food,'VB1_mg'].values[0] *
   num
diet[9]+=menu.loc[menu['id']==food,'VB2_mg'].values[0] *
   num
diet[10]+=menu.loc[menu['id']==food,'VC_mg'].values[0] *
   num
diet[11]+=menu.loc[menu['id']==food,'Ca_mg'].values[0] *
   num
diet[12]+=menu.loc[menu['id']==food,'Fe_mg'].values[0] *
   num
diet[13]+=menu.loc[menu['id']==food,'Zn_mg'].values[0] *
   num
diet[14]+=menu.loc[menu['id']==food,'Ile_mg'].values[0] *
   num
diet[15]+=menu.loc[menu['id']==food,'Leu_mg'].values[0] *
   num
diet[16]+=menu.loc[menu['id']==food,'Lys_mg'].values[0] *
   num
diet[17]+=menu.loc[menu['id']==food,'SAA_mg'].values[0] *
   num
diet[18]+=menu.loc[menu['id']==food,'AAA_mg'].values[0] *
   num
diet[19]+=menu.loc[menu['id']==food,'Thr_mg'].values[0] *
   num
diet[20]+=menu.loc[menu['id']==food,'Trp_mg'].values[0] *
   num
diet[21]+=menu.loc[menu['id']==food,'Val_mg'].values[0] *
```

```
                num
            diet[22]+=menu.loc[menu['id']==food,'price'].values[0] *
                num

    diet[23]=round(min(diet[14]/diet[3]/40, diet[15]/diet[3]/70, diet
        [16]/diet[3]/55, diet[17]/diet[3]/35, diet[18]/diet[3]/60, diet
        [19]/diet[3]/40, diet[20]/diet[3]/10, diet[21]/diet[3]/50) *
        100, 2)
    report.loc[len(report)] = diet

report.loc[len(report)] = report.apply(lambda x: x.sum())
report.loc[3, 'id'] = 'total'

report.loc[report['id']=='brk', 'Meal_Ratio'] = round(report.loc[
    report['id']=='brk','Energy_kcal'].values[0] / report.loc[report['
    id']=='total','Energy_kcal'].values[0] *100, 2)
report.loc[report['id']=='lun', 'Meal_Ratio'] = round(report.loc[
    report['id']=='lun','Energy_kcal'].values[0] / report.loc[report['
    id']=='total','Energy_kcal'].values[0] *100, 2)
report.loc[report['id']=='din', 'Meal_Ratio'] = round(report.loc[
    report['id']=='din','Energy_kcal'].values[0] / report.loc[report['
    id']=='total','Energy_kcal'].values[0] *100, 2)

report.loc[report['id']=='brk', 'Ratio_error'] = cal_error(report.loc[
    report['id']=='brk','Meal_Ratio'].values[0], 30)
report.loc[report['id']=='lun', 'Ratio_error'] = cal_error(report.loc[
    report['id']=='lun','Meal_Ratio'].values[0], 35)
report.loc[report['id']=='din', 'Ratio_error'] = cal_error(report.loc[
    report['id']=='din','Meal_Ratio'].values[0], 35)

Protein_Percentage = round(report.loc[report['id']=='total','Protein_g
    '].values[0] * 4 / report.loc[report['id']=='total','Energy_kcal
    '].values[0], 2)
Fat_Percentage = round(report.loc[report['id']=='total','Fat_g'].
    values[0] * 9 / report.loc[report['id']=='total','Energy_kcal'].
    values[0], 2)
CHO_Percentage = round(report.loc[report['id']=='total','CHO_g'].
```

```
    values[0] * 4 / report.loc[report['id']=='total','Energy_kcal'].
    values[0], 2)

AAS = round((report.loc[report['id']=='brk','AAS'].values[0] * report.
    loc[report['id']=='brk','Protein_g'].values[0] + report.loc[report
    ['id']=='lun','AAS'].values[0] * report.loc[report['id']=='lun','
    Protein_g'].values[0] + report.loc[report['id']=='din','AAS'].
    values[0] * report.loc[report['id']=='din','Protein_g'].values[0])
     / report.loc[report['id']=='total','Protein_g'].values[0], 2)

direct = [f'error_%'] + [0] * 25
direct[1] = cal_error(report.loc[report['id']=='total','Energy_kcal'].
    values[0], ref[0])
direct[3] = cal_error(Protein_Percentage, 0.125)
direct[4] = cal_error(Fat_Percentage, 0.25)
direct[5] = cal_error(CHO_Percentage, 0.55)
direct[7] = cal_error(report.loc[report['id']=='total','VA_ug'].values
    [0], ref[1])
direct[8] = cal_error(report.loc[report['id']=='total','VB1_mg'].
    values[0], ref[2])
direct[9] = cal_error(report.loc[report['id']=='total','VB2_mg'].
    values[0], ref[3])
direct[10] = cal_error(report.loc[report['id']=='total','VC_mg'].
    values[0], ref[4])
direct[11] = cal_error(report.loc[report['id']=='total','Ca_mg'].
    values[0], ref[5])
direct[12] = cal_error(report.loc[report['id']=='total','Fe_mg'].
    values[0], ref[6])
direct[13] = cal_error(report.loc[report['id']=='total','Zn_mg'].
    values[0], ref[7])

report.loc[len(report)] = direct


dict = pd.read_csv('./data/all_composition.csv',index_col=False)
comps = [0] * 61
types = [False] * 5
```

```
for meal in to_preprocess:
    diet = pd.read_csv(f'./data/report/{meal}_{prob}_{mark}.csv',
        index_col=False)
    food = pd.read_csv(f'./data/{menu_folder}/{meal}.csv',index_col=
        False)

    diet = diet['num'].tolist()
    for indexx, roww in food.iterrows():
        if diet[roww['id']-1]!=0: # origin
            count = roww['amount'] * diet[roww['id']-1]
        # if diet[mapp.index(roww['id'])]!=0: # new
        # count = roww['amount'] * diet[mapp.index(roww['id'])]
            comps[roww['pos']] += count
            types[roww['type']-1] = True

count = 0
report.loc[5, 'id'] = ''
report.loc[6, 'id'] = 'AAS'
report.loc[6, 'Energy_kcal'] = AAS

report.loc[7, 'id'] = ''
report.loc[8, 'id'] = 'Types'
for ii in range(0,5):
    report.loc[9+ii, 'id'] = f'{ii+1}'
    report.loc[9+ii, 'Energy_kcal'] = types[ii]
    if types[ii]==True:
        count+=1
report.loc[8, 'Energy_kcal'] = count
if count==5:
    report.loc[8, 'Water_g'] = 'GOOD'
else:
    report.loc[8, 'Water_g'] = 'BAD'


report.loc[14, 'id'] = ''
report.loc[15, 'id'] = 'Compos'
```

```
ii = 0
for index, key in enumerate(comps):
    if key!=0:
        report.loc[16+ii, 'id'] = dict.loc[index, 'Id']
        report.loc[16+ii, 'Energy_kcal'] = key
        ii+=1
report.loc[15, 'Energy_kcal'] = ii
if ii>12:
    report.loc[15, 'Water_g'] = 'GOOD'
else:
    report.loc[15, 'Water_g'] = 'BAD'



report.to_csv(f'./data/report/{prob}_{mark}_report.csv',index=False)
```