**程序说明**

1. 本程序基于Python编写，首先需要安装所需的库文件：

$ pip install pyymal

$ pip install gurobipy

注：Gurobi是数学规划优化的**商业系统**，您需要购买商业许可或使用学术许可来解锁其全部功能（如求解大规模优化问题）。

1. 库文件就绪后，可在Main.py文件中设置程序运行的参数，各参数意义如下：

|  |  |
| --- | --- |
| **参数** | **意义** |
| verbose | 是否输出详细日志 |
| pathData | 输入数据的路径 |
| SetInitCurrency() | 设定起始货币种类 |
| SetTermCurrency() | 设定目标货币种类 |
| SetInitCurrencyQuantity() | 设定起始货币数量 |
| SetFeeLimit() | 设定手续费限制 |

1. 完成设定后，运行Main.py文件即可开始优化并在控制台输出模型信息与优化结果。

**数据结构说明**

Text

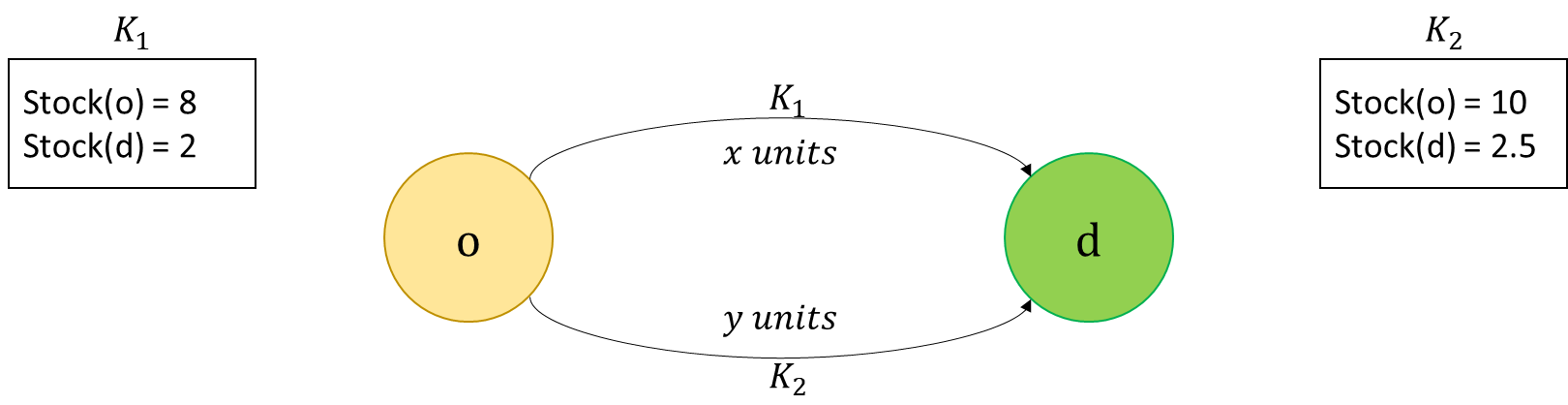
Description automatically generated

数据以yaml格式储存，使用Python读取时会被解析为字典的列表。以右图所示的数据为例，列表中共有两个字典，分别存储两个交易所的信息。所含信息有交易所名称(nameExchange)，货币存量(stocks)，交易所相关手续费(B1)，交易量相关手续费(B2)。手续费的类型为字典的字典，上级字典的键为起始货币种类，下级字典的键为目标货币的种类，值为将起始货币兑换位目标货币所需要的手续费。

**测试案例**

**拟用如下两个测试案例作为检测程序正确与否的工具，同时也做demo。需要说明的是，目前并未考虑手续费约束，但保留子环消除约束。**

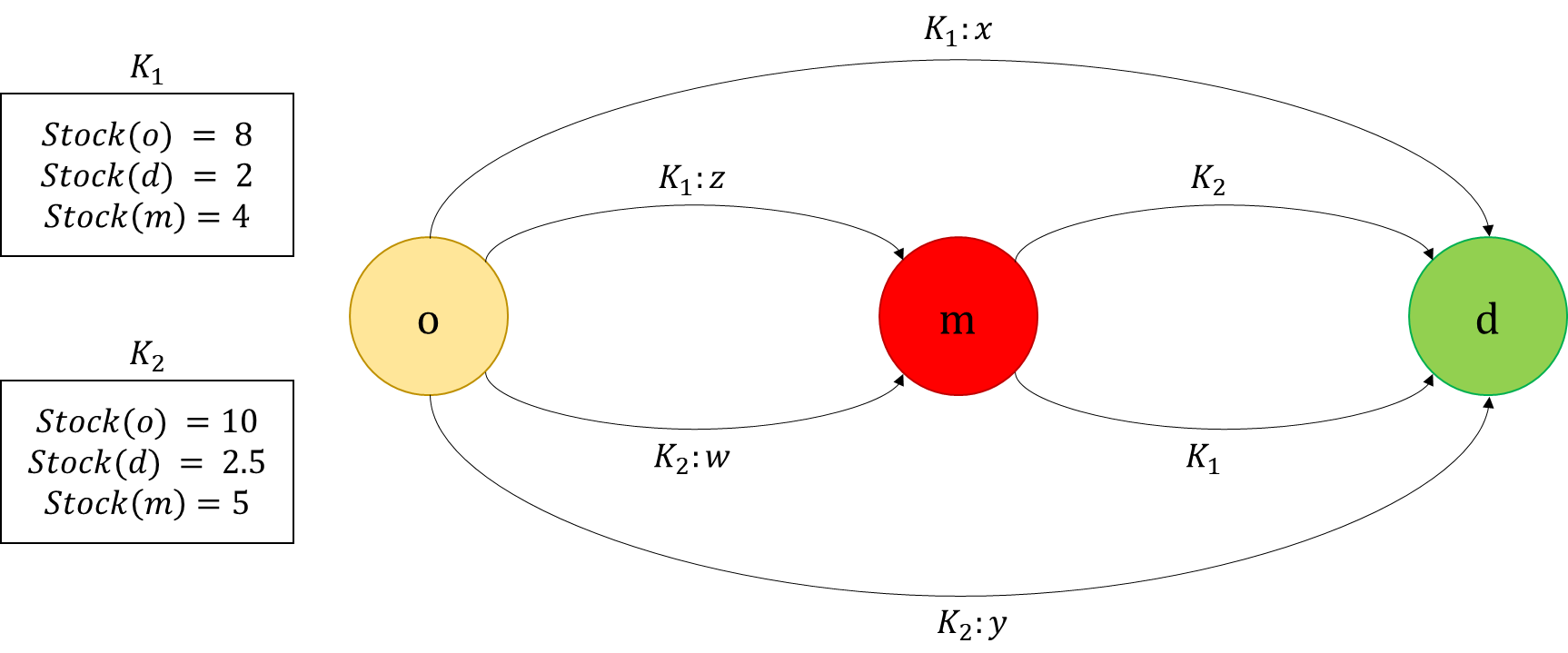
**测试案例1**：仅考虑两个渠道，两种货币，其中的货币存量为，的货币存量为，兑换目标为：将1单位货币全部兑换为货币并最大化所得的货币数量，示意图如下：

**

经程序计算，最优兑换策略为：将0.4477单位和0.5522单位的货币分别经由渠道兑换为货币，总计获得0.2368单位的货币。

**数学推导：**

**测试案例2**：仅考虑两个渠道，三种货币，其中的货币存量为，的货币存量为，兑换目标为：将1单位货币全部兑换为货币并最大化所得的货币数量，示意图如下：

****

经程序计算，使用最优兑换策略总计获得0.2410单位的货币。最优兑换策略如下：

**合理性检查：**本案例相比于案例1引入额外新币种，兑换路径数量增加，且两种货币量保持不变，因而最终收益的货币数量大于案例1

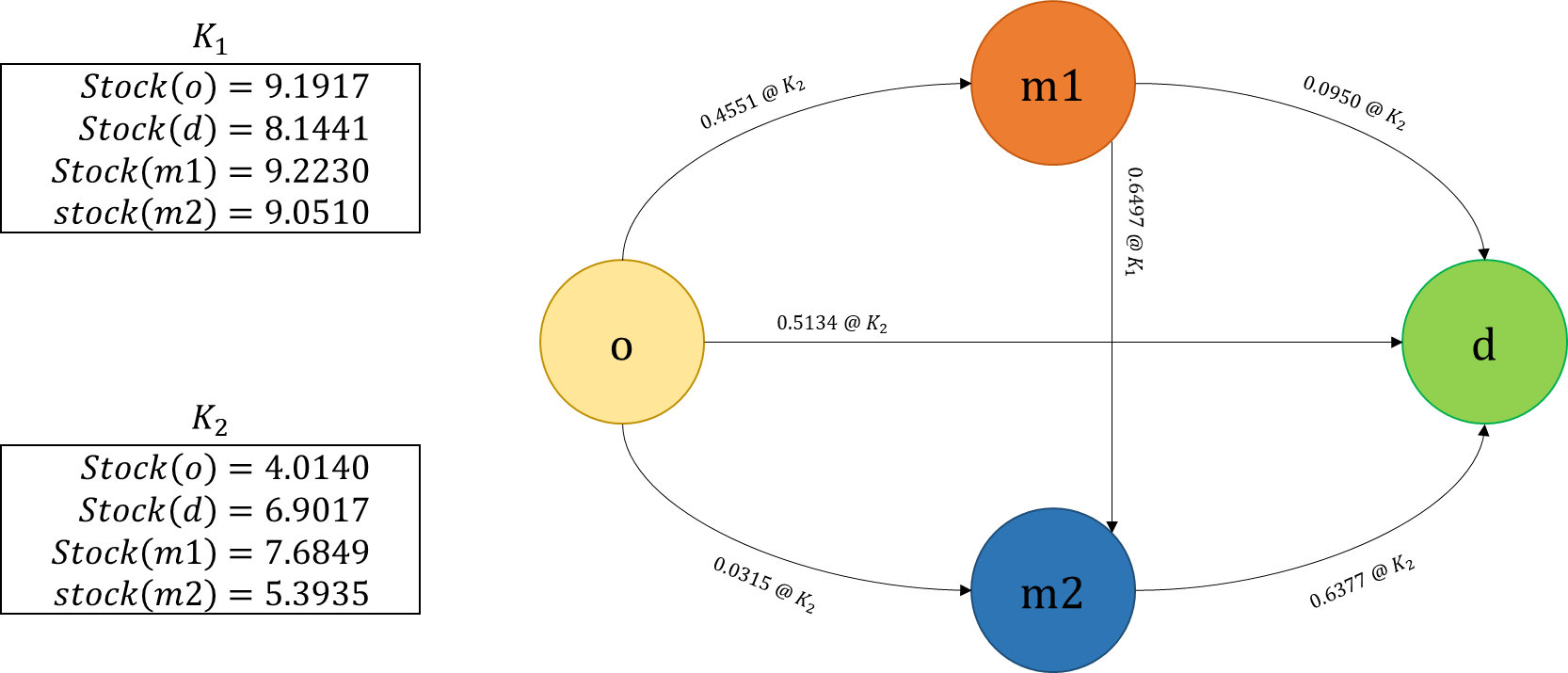
测试案例3：考虑两个渠道，四种货币，其中：

的货币存量为，

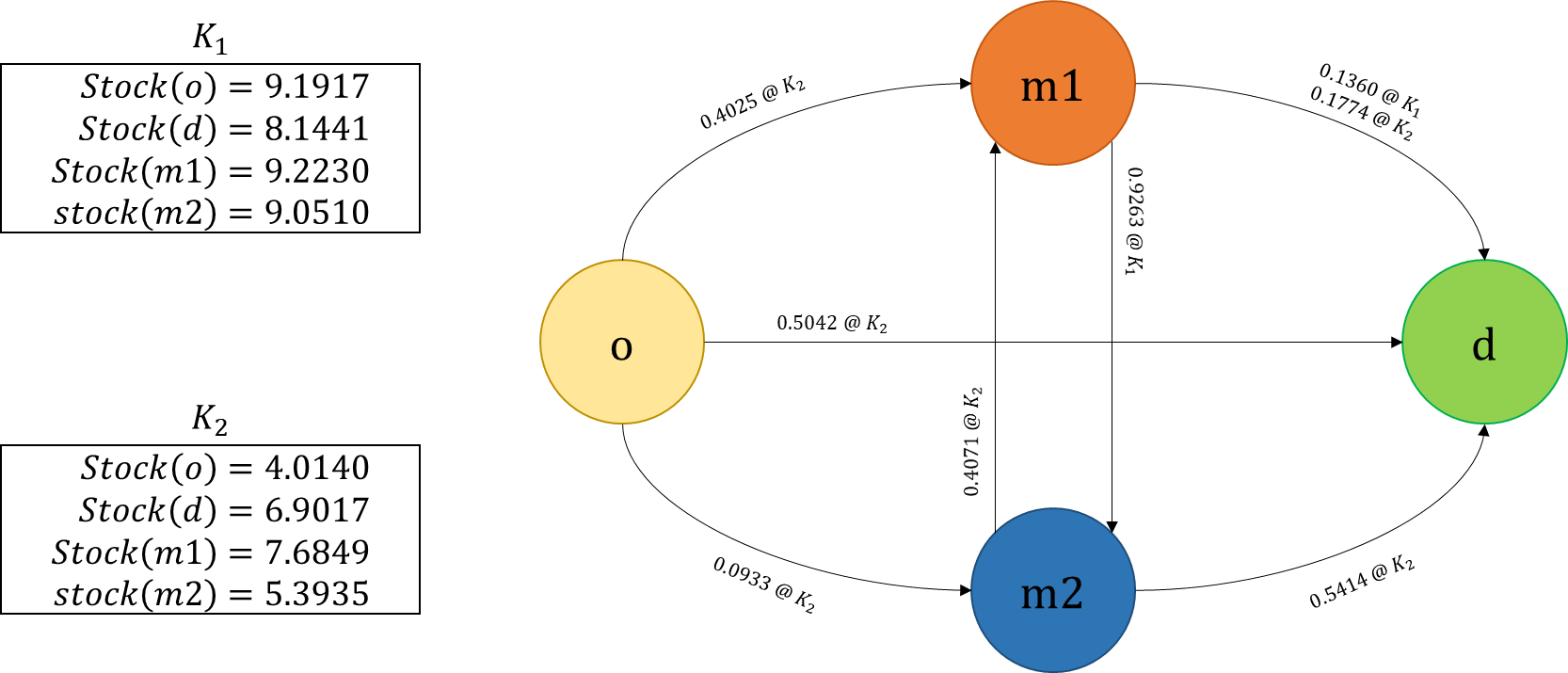
的货币存量为，

兑换目标为：将1单位货币全部兑换为货币并最大化所得的货币数量

在保留“去除子环路”约束时，兑换方案如下图所示，最佳目标值为1.6299单位：



在删除“去除子环路”约束时，兑换方案如下图所示，最佳目标值为1.6739单位：



后者引入了子环路来利用静态模型的缺陷提高了最佳目标值，但是在实际应用中此种行为并不合理，应予以避免。且在多次测试中发现，删除“去除子环路”约束后使得求解时间不减反增，因此在后续实验中应保留“去除子环路”约束。

**算法效率分析**

模型的规模主要由货币种类数量(#E)和交易所数量(#C)决定，以整数对(#E, #C)表示，本节研究以上两种因素对于求解时间的影响。测试方法如下：程序首先根据输入的(#E, #C)生成相应数量的货币种类和交易所，每个交易所中的各类货币库存为1到10间的随机数（均匀分布），手续费初始化为0，初始币种持有量为1。

测试设备信息如下：

中央处理器：Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz

内存：16 GB

操作系统：Windows 10专业版

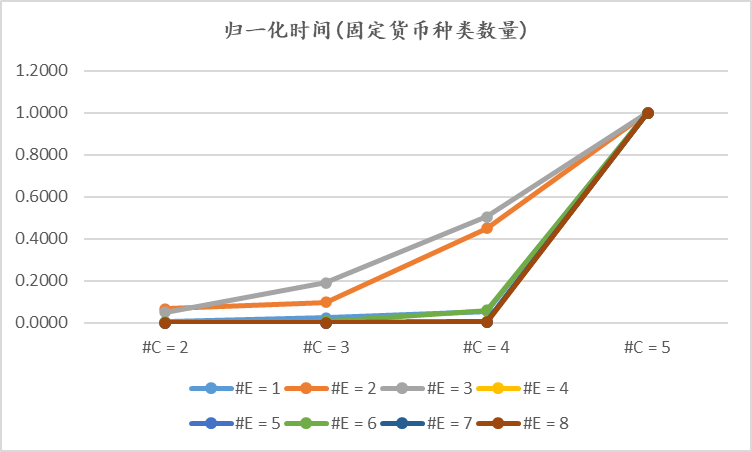
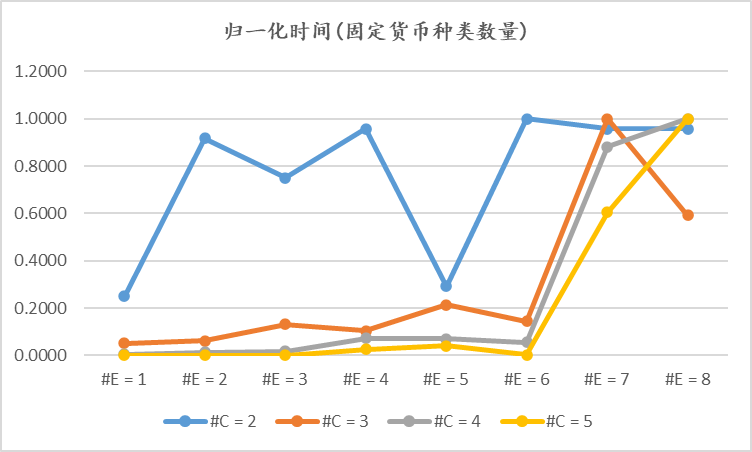
解释器版本：Python 3.8.1

求解软件：Gurobi 9.1.1

求解每组数据并记录求解时间，结果如下表所示，其中每个单元格内的数字代表某(#E, #C)参数下的优化求解时间，单位是秒：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **货币种类数量**  **交易所数量** | **2** | **3** | **4** | **5** |
| **1** | 0.0060 | 0.0269 | 0.0608 | 1.1230 |
| **2** | 0.0219 | 0.0319 | 0.1486 | 0.3312 |
| **3** | 0.0179 | 0.0678 | 0.1795 | 0.3560 |
| **4** | 0.0229 | 0.0538 | 0.7929 | 121.57 |
| **5** | 0.0070 | 0.1107 | 0.7819 | 200.57 |
| **6** | 0.0239 | 0.0748 | 0.6174 | 10.616 |
| **7** | 0.0229 | 0.5196 | 9.7159 | 2980.8 |
| **8** | 0.0229 | 0.3072 | 11.0268 | 4923.1 |

接下来在固定货币种类数量或交易所数量的条件下，对计算时间进行灵敏度分析。具体如下图所示（为表现时间变化趋势，每条曲线用其最大值归一化）：



由此可知：

（1）交易所数量对计算耗时的影响关系较为复杂，并非单调递增。

（2）随着货币数量增加，计算时间单调递增。对于大规模问题，计算成本将无法承受，必须进行简化。

**~~子环路消除约束的探讨~~**

~~建模过程中，为避免兑换路径形成不合理的子环路，曾引入“兑换路径子环路消除”约束，现对比有无该约束对优化过程的影响。下表为有/无“兑换路径子环路消除”约束时的模型求解时间，观察~~**~~可知，子环路消除约束的确对计算时间有一定影响~~**~~。当然，对于大规模问题，不管是否保留子环约束，都无法在有限时间内求解。~~

~~当货币种类数量大于或等于4时，无“兑换路径子环路消除”约束的模型~~**~~无法在合理时间内得到最优解~~**~~，~~**~~甚至在很长时间内无法得到可行解~~**~~。注：上表中每个单元格中的数值为求解时间，单位：秒。每个数值由50次计算平均后得到，且“有子环路消除”和“无子环路消除”使用的数据相同。~~

~~在以上测试中，有/无“兑换路径子环路消除”约束的两种模型所解得的~~**~~最优目标值相同~~**~~，但是~~**~~决策变量取值未必相同~~**~~，因为最优兑换路径不一定唯一。~~

**~~总结：删除“兑换路径子环路消除”约束并未使模型取得更优解，且仅能对微型模型在合理时间内完成求解。究其原因，可能是模型性质不同，求解器所选算法不同。将模型建模成为整数规划，其算法更成熟所以数值稳定性更好。因此，目前先保留子环消除约束。~~**