华泰证券 HUATAI SECURITIES

遗传规划因子挖掘的 GPU 加速

华泰研究

2024 年 2 月 19 日 | 中国内地　　深度研究

研究员　林晓明
SAC No. S0570516010001　linxiaoming@htsc.com
SFC No. BPY421　+(86) 755 8208 0134

研究员　何康，PhD
SAC No. S0570520080004　hekang@htsc.com
SFC No. BRB318　+(86) 21 2897 2039

人工智能系列之 73：使用 GPU 算力对遗传规划因子挖掘进行加速

本研究使用 GPU 算力对遗传规划因子挖掘进行加速。遗传规划的痛点之一是运算效率低，因子进化不具备方向性，需要反复迭代筛选。我们基于 DEAP 开源框架，使用 PyTorch 拓展算子和适应度函数。结果表明，在 Intel Core i9-13900KF 和 NVIDIA GeForce RTX 4090 测试环境下，对于遗传规划全流程及核心的因子值计算和 IC 值计算环节，GPU 能提供 7~9 倍的加速效果。5 轮挖掘每轮进化 3 代总耗时 12.4 小时，得到 350 个符合入库条件的因子。

大规模矩阵运算主要涉及因子值计算与 IC 值计算，优化空间大

分析遗传规划全流程时间开销，大规模矩阵运算主要涉及因子值计算与 IC 值计算环节，均针对个股层面，每轮迭代需要重复数千次（取决于种群数量），并迭代若干轮，时间开销大，具备较大优化空间。而后代选择、交叉变异等环节仅作用于算子表达式，即针对因子层面，不涉及大规模矩阵计算，时间开销低。本研究测试场景下，单次计算单因子全历史的因子值和 IC 值，CPU 平均时间开销分别为 8.5 秒和 0.8 秒，GPU 平均时间开销分别为 1 秒和 0.0007 秒，GPU 加速效果显著。

技术路线的选择：PyTorch+CUDA 较易实现，通信成本低，加速效果好

遗传规划的 GPU 加速有三种可行的技术路线：RAPIDS、Numba+CUDA 和 PyTorch+CUDA。RAPIDS 代码改写工程量小，但实证表明加速效果不如 PyTorch。Numba+CUDA 在大规模运算时加速效果更好，但代码改写工程量大，且部分复杂函数仍需在 CPU 执行，通信成本高。PyTorch+CUDA 代码改写工程量小，上手容易，全部运算均可在 GPU 上完成，数据流连贯，通信成本低。本研究采用 PyTorch+CUDA 路线，不改写 DEAP 底层代码，而是新定义一批基于 PyTorch 的算子和适应度函数，添加到 DEAP 框架中。

合理种群数量条件下，GPU 能够实现 7~9 倍加速效果

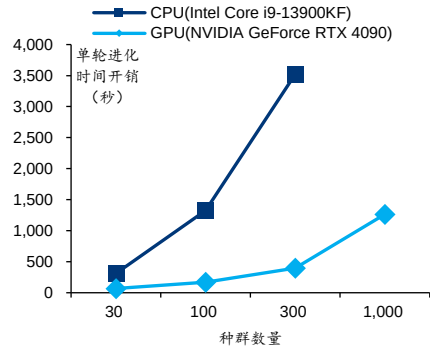
本研究构建了以残差收益率为优化目标的因子挖掘流程。每一轮挖掘以剔除上一轮新因子的残差收益率为优化目标，适应度为样本内年化 RankICIR。因子入库条件为适应度高且与已有因子相关度低。定义 29 个初始特征和 31 个算子。当种群数量分别为 30/100/300 时，因子挖掘全流程 GPU 相比 CPU 的加速倍数分别为 4.4/7.9/8.1 倍。其中核心的因子计算和 IC 值计算环节，GPU 相比 CPU 的加速倍数分别为 4.6/7.7/8.9 倍。考虑到种群数量较低时不具备参考意义，我们认为 GPU 提供的 7~9 倍加速是相对合理的结果。

优质因子中 ts_grouping_sortavg 算子和 turn 特征的组合出现频次高

正式实验中种群数量设为 3000，进行 5 轮挖掘，每轮进化 3 代，共挖掘出 350 个符合入库条件的因子，总时间开销约 12.4 小时，平均单轮进化时间开销约 0.8 小时。对部分样本外多头表现较好的因子表达式进行解读，我们发现 ts_grouping_sortavg 算子和 turn 特征的组合出现频次较高，其含义是基于过去一段时间每日换手率进行筛选，针对换手率最高或最低的部分交易日的对应指标求均值。相比直接求指标的区间均值，换手率最高或最低的交易日可能蕴含更丰富的信息。

风险提示：遗传规划挖掘的选股因子是历史经验的总结，存在失效可能。遗传规划得到的因子可解释性较低，使用需谨慎。机器学习模型存在过拟合的风险。

遗传规划核心环节时间开销对比



资料来源：Wind，华泰研究

正文目录

遗传规划 GPU 加速的工程实现

运算效率决定了量化策略的执行和迭代速度。随着行业数据粒度不断细化，数据体量日益增加，量化策略的比拼从框架完善逐渐下沉到细节打磨和工程实现，如何充分利用算力提升开发效率，已成为量化机构竞争的新赛道。

遗传规划是量化投资常用的因子挖掘算法。站在深度学习已成为主流的今天，遗传规划的“灰箱”属性仍有其特殊价值：挖掘出的因子既超越人脑想象，又具备一定的可解释性。然而，遗传规划的“痛点”之一是运算效率低。基因的进化不具备方向性，只不过是好的变异经历漫长的时间被自然选择；因子的进化同样充满随机性，需要通过反复迭代从海量候选因子中筛选出优质因子。

本研究尝试使用 GPU 算力提升遗传规划的运算效率。我们基于 DEAP 框架，使用 PyTorch 拓展算子和适应度函数，实现因子挖掘的 GPU 加速。结果表明，在 Intel Core i9-13900KF 和 NVIDIA GeForce RTX 4090 测试环境下，对于遗传规划全流程以及核心的因子计算值和 IC 值计算环节，GPU 能提供 7~9 倍的加速效果，5 轮挖掘每轮进化 3 代总耗时 12.4 小时，共得到 350 个符合入库条件的因子。

成熟开源框架介绍

遗传规划算法拥有大量成熟开源框架。基于 CPU 平台的 DEAP、gplearn、geppy 等在业界被广泛使用。基于 GPU 平台的 KarooGP、TensorGP 也逐渐受到关注，这两个框架均基于 TensorFlow 开发。2021 年，NVIDIA 印度团队采用 cuML 实现遗传规划算法，发现其运算效率优于 gplearn、KarooGP 和 TensorGP，在两组数据集上的运算效率分别是 gplearn 的 119 和 40 倍，但代码未开源。

图表1：基于 Python 的遗传规划开源框架

名称	运算平台	GitHub 地址	初代发布年份
DEAP	CPU	https://github.com/DEAP/deap	2010
gplearn	CPU	https://github.com/trevorstephens/gplearn	2015
geppy	CPU	https://github.com/ShuhuaGao/geppy	2018
KarooGP	CPU/GPU(TensorFlow)	https://github.com/kstaats/karoo_gp	2017
TensorGP	CPU/GPU(TensorFlow)	https://github.com/AwardOfSky/TensorGP	2021

资料来源：GitHub 官网，华泰研究

华泰金工研报《人工智能 21：基于遗传规划的选股因子挖掘》（2021-06-10）等研究采用 gplearn 框架。由于该框架仅支持二维数据，无法实现三维数据（股票×时间×初始特征）的传入，当时我们针对因子挖掘场景进行了大量深度定制改造。

本文选择 DEAP 框架进行定制改造。相比于 gplearn，DEAP 的功能架构更具延展性。DEAP 采用模块化编程的设计思路，可使用简练的编程语言，“注册（register）”初始特征、初始种群、算子树等功能模块。代码结构清晰，后续运维方便。后文我们将基于 PyTorch 实现 DEAP 框架 CPU 至 GPU 的拓展。

图表2：DEAP 框架主要特点

模块名称	主要特点
初始因子传入	可传入任意格式数据（array、dataframe、graph、audio），且支持自定义分类（如定义初始因子 1-10 为量价因子，因子 11-20 为基本面因子），以便后续对算子开展个性化定制
算子定义	可附加数据格式约束，如某算子(a/b).rolling(int_n)可约束 a 为量价因子，b 为基本面因子，int_n 为随机整型变量，“引导”计算机算法按照人类设定的思路开展因子挖掘
适应度选择	支持定义复杂的适应度函数，可将风格市值中性化、单因子测试等嵌入其中
交叉变异	提供丰富的交叉变异工具包，如交叉包括单点交叉、混合交叉，变异包括高斯突变、乱序突变、均匀整数突变等
后代选择	支持采用锦标赛选择、轮盘赌选择、有放回的随机选择等后代选取方式

资料来源：DEAP 项目官网，华泰研究

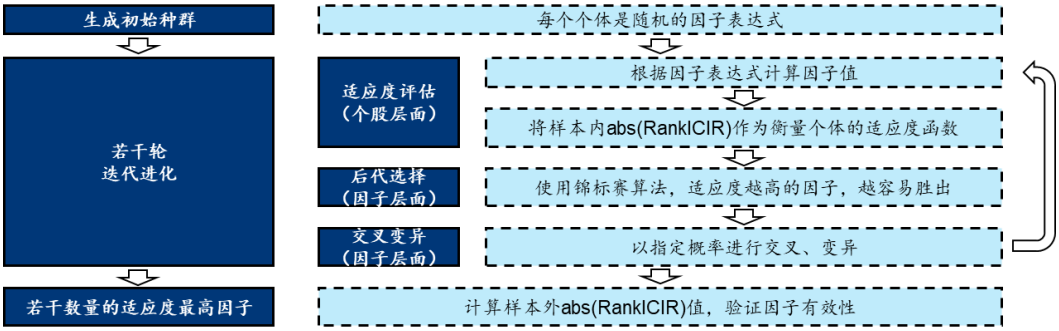
加速优化空间分析

遗传规划的基础概念请详见华泰金工研报《人工智能 21：基于遗传规划的选股因子挖掘》（2021-06-10），本文不再展开介绍。

从工程实现角度，遗传规划因子挖掘的哪些环节存在加速优化空间？下图展示了遗传规划全流程。算法的输入为初始特征，每个特征可以视作个股×日期的二维 pandas.DataFrame 矩阵，假设样本为全 A 股连续 10 年日频数据，那么每个特征包含的元素数量在千万数量级。

大规模 DataFrame 矩阵运算主要涉及“适应度评估”模块的因子值计算与 IC 值计算两个环节。该模块针对个股层面，每轮迭代需要重复数千次（取决于种群数量），并在此基础上迭代若干轮，时间开销较大。而“后代选择”、“交叉变异”模块仅作用于算子表达式，即针对因子层面，不涉及大规模 DataFrame 矩阵计算，时间开销较低。

图表3：遗传规划流程分析



资料来源：华泰研究

我们进一步测试遗传规划各模块 CPU 运算的时间开销，如下表：

1. 全部初始特征读取耗时较长约 30 秒，但仅执行一次；
2. 中性化耗时不短约 7 秒，表达式交叉变异耗时较短约 0.7 秒，两者仅执行有限次；
3. 对单因子进行因子值和 IC 值计算分别耗时约 8.5 秒和 0.8 秒，两者执行次数约为迭代次数×种群数量。假设种群数量在千数量级，那么两者将执行上万次，有较大优化空间。

针对不同环节、不同运算类型，可分类采取差异化的 GPU 加速优化措施，后文将展开介绍。

另外需要说明，中性化环节单次时间开销并不低，是否需要加速优化取决于中性化的对象。如果针对预期收益 y 做中性化，那么该运算执行次数仅取决于迭代轮数，可以不做优化。例如第 1 轮对 y 做行业市值中性取残差，第 2 轮对该残差做上一轮的因子中性取残差……但如果针对因子 x 做中性化，那么该运算执行次数将达数万次，很有必要进行 GPU 加速。

图表4：遗传规划各环节加速优化空间分析

执行次数	模块名称	CPU 时间开销	GPU 时间开销	GPU 加速优化空间
一次	数据读取	33s（全部数据）	暂不优化	仅重复一次，优化必要性不大，且优化空间有限
迭代轮数	收益中性化	7s（全历史）	暂不优化	解析解：可通过 CuPy、Numba+CUDA、torch.Tensor 实现； 数值解：可通过 torch.NN 梯度下降实现
	表达式交叉变异	0.7s	暂不优化	轻量级数据运算，耗时较少，不需要优化
迭代轮数×种群数量（上万次）	因子值计算（元素级）	~8.5s（单因子全历史；取决于因子复杂度）	~1s（单因子全历史；取决于因子复杂度）	如矩阵加法 A+B 等，各元素逐一加和，可通过 torch.Tensor 实现加速
	因子值计算（非元素级）			如加权移动平均等，涉及矩阵不同行列的复杂变换，可通过 Numba+CUDA 实现加速
	IC 值计算（非元素级）	0.8s（单因子全历史）	0.0007s（单因子全历史）	可通过 torch.Tensor 相关函数（如 sum/mean/sqrt 等）实现加速

注：测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090；全历史对应 10 年样本内和 4 年 1 个月样本外
资料来源：Wind，华泰研究

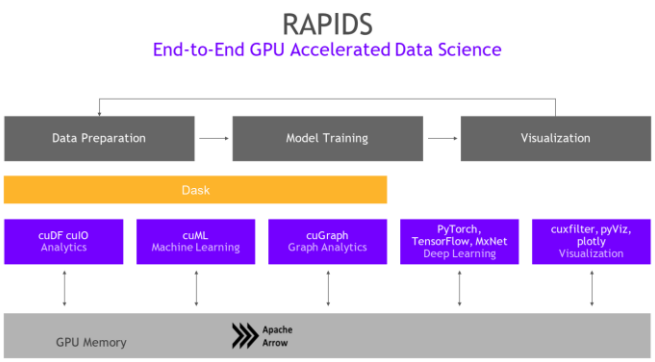
GPU 加速技术路线

如何实现遗传规划的 GPU 加速？下面介绍三种可行的技术路线。

RAPIDS

华泰金工研报《人工智能 70：高频因子计算的 GPU 加速》（2023-10-16）中，我们测试了 NVIDIA RAPIDS 的加速效果。RAPIDS 是 NVIDIA 针对数据科学与机器学习推出的 GPU 加速平台。其重要特性是将基于 CUDA 底层代码的优化以 Python 高级语言的形式体现。常用 API 包括 CuPy（对标 NumPy）、cuDF（对标 Pandas）、cuML（对标 scikit-learn）等，由于 API 语法几乎相同，仅需要轻量级的代码修改，即可实现 CPU 到 GPU 运算的迁移。

图表5：RAPIDS：数据科学的 GPU 加速



资料来源：NVIDIA 官网，华泰研究

图表6：RAPIDS 加速因子计算代码范例

```
import numpy as np
def return_downward_var(_return):
    _return_down = _return * (_return<0)
    mask = (_return_down!=0).sum(axis=0)==0
    _return_down[_return_down==0] = np.nan
    RDV = np.nanvar(_return_down, axis=0)
    RDV[mask]=0
    return RDV
```

```
import cupy as cp
def return_downward_var(_return):
    _return_down = _return * (_return<0)
    mask = (_return_down!=0).sum(axis=0)==0
    _return_down[_return_down==0] = cp.nan
    RDV = cp.nanvar(_return_down, axis=0)
    RDV[mask]=0
    return RDV
```

资料来源：华泰研究

针对遗传规划场景，RAPIDS 技术路线的改造思路如下：

1. 考虑到 pandas.DataFrame 的运算速度低于 numpy.ndarray，并且参与因子挖掘运算的初始特征均为“股票×时间”的标准同源结构，不涉及数据对齐需求，故采用 numpy.ndarray 统一替代 pandas.DataFrame。
2. 采用 CuPy 替代 NumPy，通过轻量化改写实现 GPU 加速。

Numba+CUDA

Numba 是常用的 Python 加速方式。Numba 是 Python 函数的即时（Just-in-time，简称 JIT）编译器。C/C++属于编译型语言，代码首次运行前，经编译器转换为可执行机器码。而 Python 属于解释性语言，代码每次运行前都需要重新编译，拖累了运行速度。JIT 技术可在运行时将调用的函数或程序段编译成机器码载入内存，以提升 Python 代码运行速度。

Numba 的 CPU 加速的实现方式是在待编译的函数前加装饰器，如：@numba.vectorize，@numba.jit。Numba 同样支持 GPU 加速，修改装饰器即可，如：@numba.cuda.vectorize，@numba.cuda.jit。

针对遗传规划场景，Numba+CUDA 技术路线的改造思路如下：

1. 以前述 RAPIDS 技术路线为基础，对简单函数采用 CuPy 替代。
2. 对部分复杂函数（如线性衰减加权 decay_linear 等）采用 Numba+CUDA 手工复写。

图表7：Numba+CUDA 代码范例

```
import math
import scipy.stats
import numpy as np
from numba import vectorize

# 从均匀分布中随机采样 1 亿次
x = np.random.uniform(-3, 3, size=100000000).astype(np.float32)

# 常数
SQRT_2PI = np.float32((2*math.pi)**0.5)

@vectorize(['float32(float32, float32, float32)'], target='cuda')
def gaussian_pdf(x, mean, sigma):
    '''计算给定高斯分布的概率密度函数.'''
    return math.exp(-0.5 * ((x - mean) / sigma)**2) / (sigma * SQRT_2PI)

# GPU
gaussian_pdf(x, 0.0, 1.0)

# CPU
scipy.stats.norm.pdf(x, loc=np.float32(0.0), scale=np.float32(1.0))
```

资料来源：百度飞桨官网《基于 Numba 的 CUDA Python 编程简介》，华泰研究

PyTorch+CUDA

PyTorch 是目前学界和业界常用的开源机器学习和深度学习框架。Pytorch 的加速功能不仅体现在神经网络训练的反向传播上，也可用于 Tensor 计算的前向传播上。

针对遗传规划场景，PyTorch 技术路线的改造思路如下：

1. 采用 torch.Tensor 结构替代原 NumPy、CuPy 结构，采用 Tensor 张量“计算图”代替原 NumPy 数据流。
2. 对 NumPy 中的简单函数，直接替换为 Tensor 结构适用的新函数，如 numpy.concatenate 替换为 torch.cat。
3. 对 NumPy 中的复杂函数（如 rolling、correlation、rankpct 等），基于 Tensor 基础函数进行手动改写。
4. 将 Tensor 张量使用.to(device)传至显存，实现全数据流的 GPU 运算。

图表8：NumPy 和 PyTorch 等价函数对比

NumPy 函数	PyTorch 函数	NumPy 函数	PyTorch 函数	NumPy 函数	PyTorch 函数
np.array()	torch.tensor()	np.random.randn()	torch.randn()	np.exp()	torch.exp()
np.zeros()	torch.zeros()	np.sum()	torch.sum()	np.log()	torch.log()
np.ones()	torch.ones()	np.mean()	torch.mean()	np.dot()	torch.dot()
np.arange()	torch.arange()	np.max()	torch.max()	np.matmul()	torch.matmul()
np.linspace()	torch.linspace()	np.min()	torch.min()	np.transpose()	torch.transpose()
np.eye()	torch.eye()	np.argmax()	torch.argmax()	np.reshape()	torch.reshape()
np.random.rand()	torch.rand()	np.argmin()	torch.argmin()	np.concatenate()	torch.cat()

资料来源：ChatGPT，华泰研究

技术路线对比和最终方案

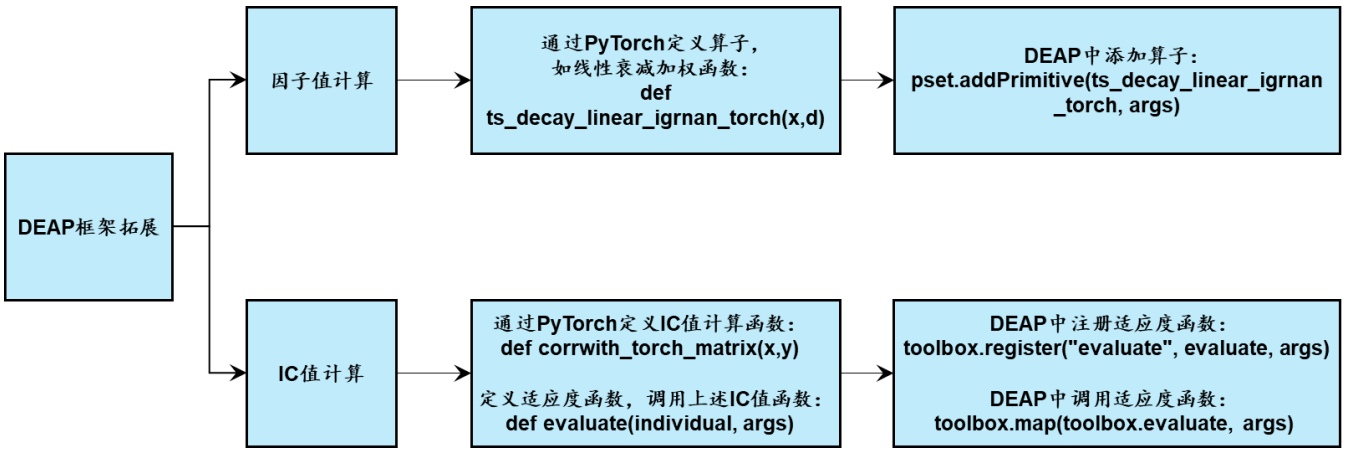
对比上述三种技术路线：

1. RAPIDS 代码改写工程量小，但实证表明加速效果不如 PyTorch。
2. Numba+CUDA 在大规模运算时加速效果更好，但代码改写工程量大，且部分复杂函数可能仍需在 CPU 上执行，拉高了通信成本。
3. PyTorch+CUDA 代码改写工程量小，上手较容易，全部运算均可在 GPU 上完成，且数据流较连贯，通信成本低。

需要说明的是，三种技术路线并不冲突。在开发能力允许情况下，可以将 CuPy、cuDF、Numba、PyTorch 等多个库穿插使用。

综合考虑性能和工作量，本研究采用第三种 PyTorch+CUDA 路线。重点针对 DEAP 框架中的因子值计算和 IC 值计算两个环节进行代码改造，方案如下图。与其称为“改造”，不如称为“拓展”，我们并没有改写 DEAP 底层代码，而是新定义了一批函数，“添加”到 DEAP 框架中。

图表9：基于 DEAP 框架的 GPU 加速方案



资料来源：华泰研究

GPU 加速案例分析

本章将展示三个 GPU 加速案例，作为前述三种技术路线对比的代码展示和结果补充。测试环境为 CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090。

元素级运算：矩阵加法

元素级运算对矩阵中每个元素做相同运算操作。案例一为矩阵加法 A+B。下图分别展示：(1)原始 pandas.DataFrame 加法代码；(2)技术路线一和二结合，即 CuPy+Numba+CUDA 代码；(3)技术路线三 PyTorch+CUDA 代码。

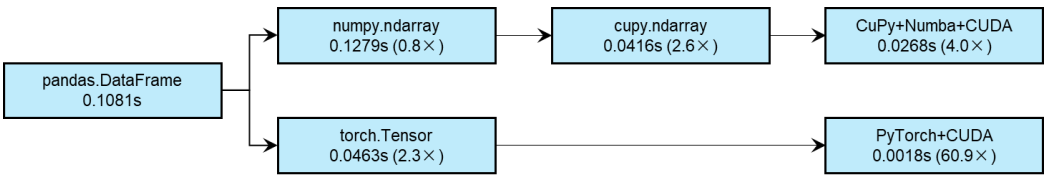
图表10：矩阵加法 GPU 加速测试部分代码

```
# 初始化数据
data = pd.DataFrame(np.random.randn(5000,5000))

# pandas.DataFrame 直接相加
def df_plus(a,b):
    return a+b

for i in range(10):
    out = df_plus(data,data)

# CuPy+Numba+CUDA
@vectorize(["float32(float32, float32)"], target='cuda')
def numba_plus(a,b):
    return a+b

data_cp = cp.array(data,dtype='float32')
for i in range(10):
    out = numba_plus(data_cp,data_cp)

# PyTorch+CUDA
def torch_plus(a,b):
    return a+b

data_cuda = torch.tensor(np.array(data,dtype='float32')).to('cuda')
for i in range(10):
    out = torch_plus(data_cuda,data_cuda)
```

注：为区分不同加速算法性能，加法操作重复 10 次进行
资料来源：华泰研究

假设矩阵维度为 5000×5000，即元素数量在千万数量级，与 A 股日度数据量相匹配。结果显示，该数量级规模运算下，技术路线三 PyTorch+CUDA 加速效果约为 61 倍，技术路线一和二相结合的 CuPy+Numba+CUDA 加速效果约为 4 倍。PyTorch+CUDA 性能优于其他方案。另外需要说明，我们的测试结果仅反映该数据量下各技术路线的性能。实际上如果数据量进一步扩大，Numba+CUDA 因其预编译的优势，相比 PyTorch+CUDA 速度或更快。

图表11：矩阵加法 GPU 加速测试结果



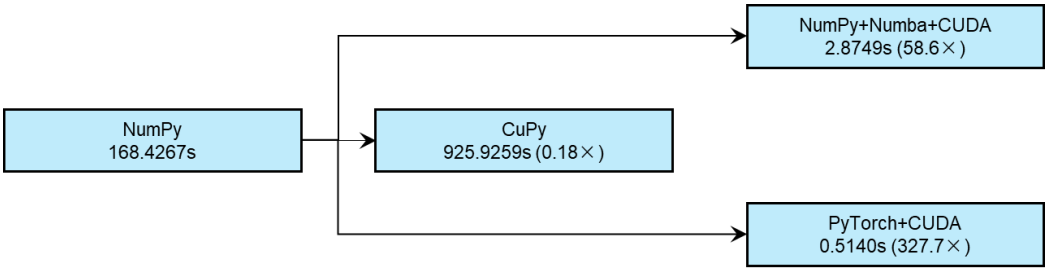
注：括号内数字代表相对 pandas.DataFrame 加速倍数；测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090
资料来源：华泰研究

非元素级运算：线性衰减加权 decay_linear

非元素级运算涉及矩阵不同行列的复杂变换。案例二中，对单个矩阵 A 的每一列，以长度 d 为滚动窗口，与向量[d,d-1,…1]进行加权求和。该运算常用于因子表达式的构建。我们分别测试 NumPy、NumPy+Numba+CUDA、CuPy、PyTorch+CUDA 方案。由于代码较冗长，此处不做展示。

假设矩阵维度为 5000×5000，结果表明，PyTorch+CUDA 加速效果约为 328 倍，优于其他方案；NumPy+Numba+CUDA 次之，加速效果约为 59 倍；CuPy 运算效率反而低。这里需要补充说明，限于篇幅我们没有测试 CuPy+Numba+CUDA 方案，实际上经过更精细的改造，可能实现优于 PyTorch+CUDA 的效果。

图表12：线性衰减加权 GPU 加速测试结果



注：括号内数字代表相对 pandas.DataFrame 加速倍数；测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090
资料来源：华泰研究

非元素级运算：计算相关系数 corrwith

案例三中，对两个矩阵 A 和 B 逐行计算相关系数，得到相关系数序列，对应因子挖掘中 IC 值计算的场景。通过 pandas.DataFrame.corrwith 函数可以在 CPU 上高效实现。PyTorch 中没有内置直接等价的函数（torch.corrcoef 只接受单个矩阵输入），需手工实现。可以通过 for 循环逐行计算，也可采用矩阵运算形式。下图展示：(1)原始 pandas.DataFrame.corrwith 代码；(2)PyTorch 改写为矩阵运算并结合 CUDA 加速的代码。

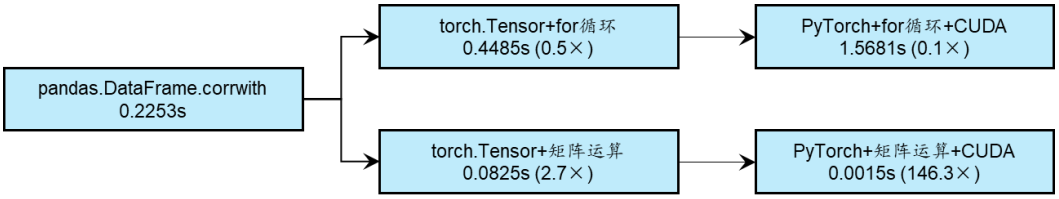
图表13：计算相关系数 GPU 加速测试部分代码

```
# 初始化数据
data_a = pd.DataFrame(np.random.randn(5000,5000))
data_b = pd.DataFrame(np.random.randn(5000,5000))

# pandas.DataFrame.corrwith
corr = data_a.corrwith(data_b,axis=1)

# PyTorch+矩阵运算+CUDA
def corrwith_torch_matrix(x,y):
    assert (x.shape == y.shape)
    x_mean = x - torch.nanmean(x, 1, keepdim=True) # [TS, CS]
    y_mean = y - torch.nanmean(y, 1, keepdim=True) # [TS, CS]
    nomi = torch.nansum(x_mean*y_mean, 1)
    denomi = torch.sqrt(torch.nansum(x_mean**2, 1)) * torch.sqrt(torch.nansum(y_mean**2, 1))
    corr = nomi / denomi # [TS, 1]
    return corr

data_a_cuda = torch.tensor(np.array(data_a,dtype='float32')).to('cuda')
data_b_cuda = torch.tensor(np.array(data_b,dtype='float32')).to('cuda')
corr = corrwith_torch_matrix(data_a_cuda,data_b_cuda)
```

资料来源：华泰研究

结果显示，千万个矩阵元素的数量级规模下，无论是否采用 CUDA，PyTorch+for 循环相比 pandas 的效率反而更低；改为矩阵运算后，PyTorch 展现出优势，PyTorch+矩阵运算+CUDA 加速效果约为 146 倍。

图表14：计算相关系数 GPU 加速测试结果



注：括号内数字代表相对 pandas.DataFrame 加速倍数；测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090
资料来源：华泰研究

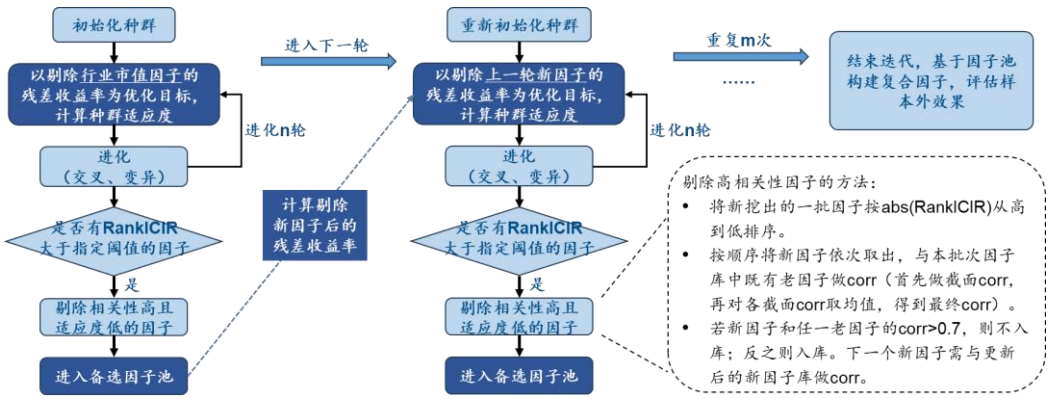
因子挖掘测试

本章介绍 GPU 加速遗传规划因子挖掘的方法，展示 CPU 和 GPU 时间开销对比及挖掘出的单因子测试结果。

方法

参考华泰金工研报《人工智能 28：基于量价的人工智能选股体系概览》(2020-02-18)，本研究构建了以残差收益率为优化目标的因子挖掘流程。具体而言，我们以挖掘增量信息为目标：第 1 轮挖掘以剔除行业市值因子的残差收益率为优化目标，计算适应度；从第 2 轮开始，以剔除上一轮新因子的残差收益率为优化目标，计算适应度；共重复 5 轮。每一轮挖掘(大循环)内部，共进行 3 代进化(小循环)。因子入库条件为：因子样本内年化 RankICIR 的绝对值大于 2.5，并且与因子库已有因子截面相关系数均值的绝对值低于 0.7。

图表15： 以残差收益率为优化目标的遗传规划因子挖掘流程



资料来源：华泰研究

由于算力限制，本研究采用静态训练集，以 2010-2019 年共 10 年数据为样本内时段，挖掘得到的因子用于 2020 年 1 月至 2024 年 1 月的样本外测试。

图表16： 遗传规划参数

参数名称	参数设置
样本内时段	2010-01-04 至 2019-12-31
样本外时段	2020-01-02 至 2024-01-31
选股域	中证全指成分股
种群数量	3000：每轮迭代的公式数量
精英数量	500：该数量的公式被随机选中，其中适应度最高的公式能进行交叉或变异生成下一代公式
大循环次数	5：每轮大循环重新开始挖掘，创立全新种群，以上一轮残差收益率为优化目标
小循环次数	3：每轮大循环内部进行几次挖掘
最大深度	3：公式树最高复杂度
交叉概率	0.35：两个个体之间进行交叉变异的概率
变异概率	0.25：个体进行突变变异的概率
优化目标	T+1 至 T+11 区间残差收益率
适应度函数	样本内年化 RankICIR
因子入库条件	适应度绝对值>2.5，且和已有因子相关度绝对值<0.7
测试环境	CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090

资料来源：华泰研究

设置 31 个算子，可分为加减乘除等元素级运算、滚动平均和切分等时序运算、截面排名等截面运算共三大类。设置 29 个初始特征，分为时序标准化特征和原始特征共两类。其中，原始特征名称后缀为 ori；时序标准化特征为滚动 60 日 zscore 时序标准化处理后的因子，在不引入未来信息的前提下，使得因子可在同一量纲下进行加减乘除操作。对于相关系数等算子，需要以原始特征作为数据，从而保留特征量纲信息，避免算子丧失解释意义。

图表17：31 个算子

算子分类	算子名称	算子定义	输入参数
元素级运算	add(X,Y)	X,Y 两因子相加	X(或 Y): 时序标准化特征
	sub(X,Y)	X,Y 两因子相减	
	mul(X,Y)	X,Y 两因子相乘	
	div(X,Y)	X,Y 两因子相除	
	log_torch(X)	log(X)	
	sqrt_torch(X)	sqrt(X)	
	neg(X)	-X	
	sigmoid_torch(X)	sigmoid(X) = 1 / (1+exp(-X))	
	sign_torch(X)	X 的符号函数，当 X 为正时，输出 1；X 为负时，输出-1；X 为 0 时，输出 0	X(或 Y): 原始特征
时序运算	ts_delay_torch(X,d)	X 因子 d 日前的取值	X(或 Y): 时序标准化特征 d: 1-10 的正整数
	ts_delta_torch(X,d)	X 因子 t-d 日至 t 日的变动数值	
	ts_delaypct_torch(X,d)	X 因子 t-d 日至 t 日的变动百分比	
	ts_correlation_torch(X,Y,d)	X,Y 两因子在过去 d 日滚动时序上的相关系数	X(或 Y): 原始特征 d: 2-10 的正整数
	ts_argmin_torch(X,d)	X 因子在过去 d 日滚动时序上最小值对应的日期编号（按时间倒序分别为 d, d-1, ……, 2, 1）	
	ts_argmax_torch(X,d)	X 因子在过去 d 日滚动时序上最大值对应的日期编号（编号定义同上）	
	ts_rank_torch(X,d)	X 因子在过去 d 日滚动时序上的百分比排名	
	ts_covariance_torch(X,Y,d)	X,Y 两因子在过去 d 日滚动时序上的协方差	X(或 Y): 时序标准化特征 d: 2-10 的正整数
	ts_decay_linear_igrnan_torch(X,d)	X 因子在过去 d 日滚动时序上的加权平均（权重按时间倒序分别为 d, d-1, ……, 2, 1）	
	ts_SubPosDecayLinear_torch(X,Y,d)	relu(X-Y)在过去 d 日滚动时序上的加权平均（权重取值同上）	
	ts_min_torch(X,d)	X 因子在过去 d 日滚动时序上的最小值数值	
	ts_max_torch(X,d)	X 因子在过去 d 日滚动时序上的最大值数值	
	ts_stddev_torch(X,d)	X 因子在过去 d 日滚动时序上的标准差数值	
	ts_sum_torch(X,d)	X 因子在过去 d 日滚动时序上的累计求和	
	ts_rankcorr_torch(X,Y,d)	X 因子的截面百分比排名与 Y 因子的截面百分比排名在过去 d 日滚动时序上的相关系数	
	ts_grouping_ascsortavg_torch(X,Y,d,n)	在过去 d 日滚动时序分组上，Y 因子最小的 n 天范围内对应的 X 因子的平均值	X: 时序标准化特征 Y: 原始特征 d: 正整数 10、15、20、40 n: 1-10 的正整数
	ts_grouping_decsortavg_torch(X,Y,d,n)	在过去 d 日滚动时序分组上，Y 因子最大的 n 天范围内对应的 X 因子的平均值	
	ts_grouping_diffsortavg_torch(X,Y,d,n)	在过去 d 日滚动时序分组上，Y 因子最大的 n 天范围内对应的 X 因子的平均值- Y 因子最小的 n 天范围内对应的 X 因子的平均值	
截面运算	rank_pct_torch(X)	X 因子在每一时间截面上的百分比排名	X(或 Y): 时序标准化特征
	rank_sub_torch(X,Y)	X 因子的截面百分比排名-Y 因子的截面百分比排名	
	rank_div_torch(X,Y)	X 因子的截面百分比排名/Y 因子的截面百分比排名	
	rank_add_torch(X,Y)	X 因子的截面百分比排名+Y 因子的截面百分比排名	

资料来源：华泰研究

图表18：29 个初始特征

特征含义	滚动 60 日时序标准化特征名称	原始特征名称
收盘价（后复权）	close	close_ori
开盘价（后复权）	open1	open1_ori
最高价（后复权）	high	high_ori
最低价（后复权）	low	low_ori
均价（后复权）	vwap	vwap_ori
成交量	volume	volume_ori
成交额	amt	amt_ori
换手率	turn	turn_ori
市净率倒数	bp	bp_ori
市盈率 TTM 倒数	ep	ep_ori
经营性现金流 TTM/总市值	ocfp	ocfp_ori
近 12 个月股息率	dp	dp_ori
当日收益率	/	return1_ori
过去 20 日成交量均值	adv20	adv20_ori
过去 60 日成交量均值	adv60	adv60_ori

资料来源：Wind，华泰研究

时间开销对比

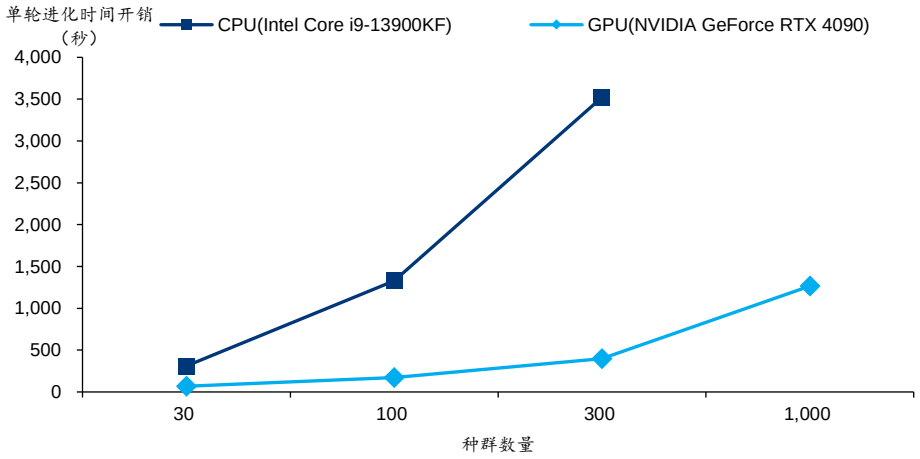
首先对比不同种群数量下，单轮进化全部环节 CPU 和 GPU 时间开销。当种群数量分别为 30、100、300 时，因子挖掘全流程 GPU 相比 CPU 的加速倍数分别为 4.4、7.9、8.1 倍。其中核心的遗传规划环节（因子值计算+IC 值计算）时间占比高，GPU 相比 CPU 加速倍数分别为 4.6、7.7、8.9 倍。我们未测试种群数量为 1000 时 CPU 时间开销。考虑到种群数量较少时结果不具备代表性，我们认为在种群数量 100、300 情况下，GPU 提供的 7~9 倍加速是相对合理的结果。

图表19：单轮进化全部环节 CPU 和 GPU 时间开销对比

运算平台	种群数量	精英数量	设置参数（秒）	遗传规划（秒）	因子入库和全流程（秒；导出（秒）除读取数据）		遗传规划 GPU 加速倍数	全流程（除读取数据）GPU 加速倍数
CPU	30	5	5.0E-04	310.9	24.0	334.8	/	/
CPU	100	17	1.7E-03	1326.5	248.0	1574.4	/	/
CPU	300	50	1.8E-03	3522.4	888.6	4411.0	/	/
CPU	1000	167	/	/	/	/	/	/
GPU	30	5	1.0E-03	68.3	7.2	75.5	4.6	4.4
GPU	100	17	1.6E-03	171.6	28.2	199.7	7.7	7.9
GPU	300	50	1.8E-03	397.9	148.7	546.6	8.9	8.1
GPU	1000	167	1.2E-02	1265.6	315.5	1581.1	/	/

注：测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090
资料来源：Wind，华泰研究

图表20：单轮进化核心环节（因子计算+IC 值计算）CPU 和 GPU 时间开销对比



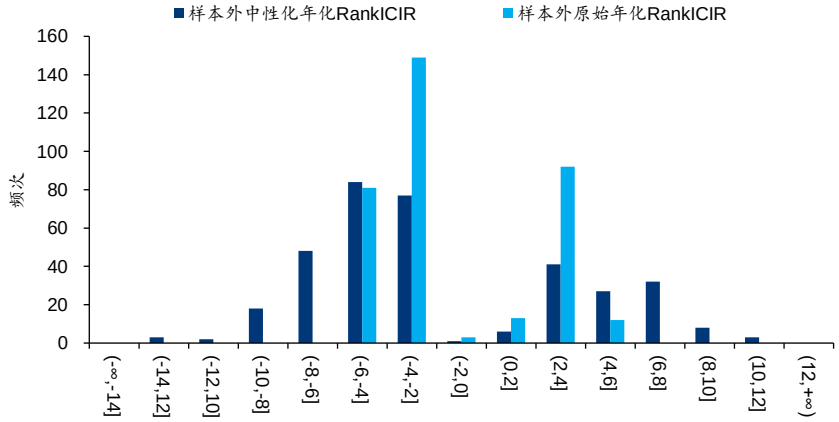
注：测试环境：CPU Intel Core i9-13900KF，GPU NVIDIA GeForce RTX 4090
资料来源：Wind，华泰研究

单因子展示

正式实验中，我们将种群数量设为 3000，进行 5 轮挖掘，每轮进化 3 代。5 轮分别挖掘出 136、87、42、47、38 个符合入库条件因子，合计 350 个因子，总时间开销约 12.4 小时，平均单轮进化时间开销约 0.8 小时。

以下展示全部因子样本外中性化年化 RankICIR 和原始年化 RankICIR 频次分布。

图表21：全部因子样本外年化 RankICIR 频次分布



注：股票池：中证全指成分股；回测区间：2020-01-02 至 2024-01-31；
资料来源：Wind，华泰研究

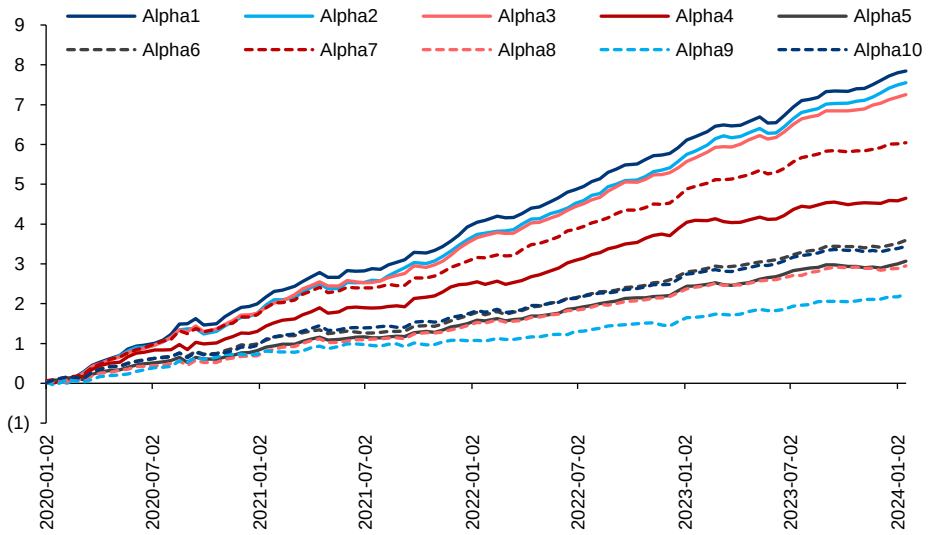
图表22：挖掘得到的部分因子表达式

因子编号	因子表达式
Alpha1	ts_grouping_diffsortavg_torch(sigmoid_torch(high), turn_ori, 15, 7)
Alpha2	ts_grouping_diffsortavg_torch(ts_rank_torch(close_ori, 9), turn_ori, 20, 1)
Alpha3	ts_grouping_diffsortavg_torch(close, amt_ori, 10, 6)
Alpha4	add(amt, ts_sum_torch(ts_rankcorr_torch(low, volume, 10), 6))
Alpha5	log_torch(ts_rankcorr_torch(volume, close, 7))
Alpha6	sigmoid_torch(ts_rankcorr_torch(amt, high, 7))
Alpha7	ts_grouping_diffsortavg_torch(bp, volume_ori, 10, 3)
Alpha8	sigmoid_torch(ts_rankcorr_torch(amt, close, 5))
Alpha9	rank_div_torch(sqrt_torch(adv20), rank_pct_torch(adv20))
Alpha10	sigmoid_torch(ts_rankcorr_torch(volume, bp, 9))

资料来源：Wind，华泰研究

上表展示部分因子表达式，下图展示这部分因子样本外累计 RankIC，调仓频率为 10 日，因子均做行业市值中性化。

图表23：部分因子双周频累计 RankIC



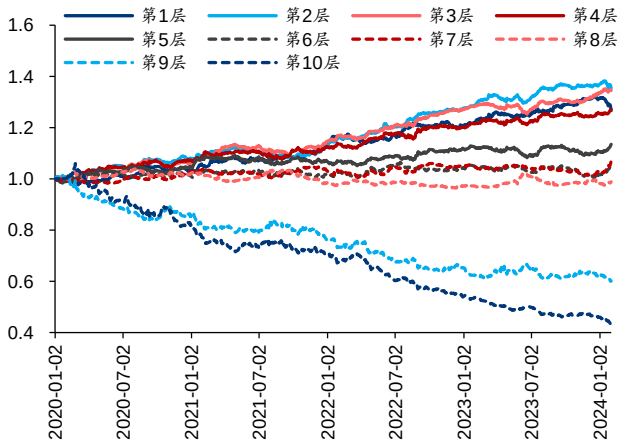
注：股票池：中证全指成分股；回测区间：2020-01-02 至 2024-01-31；因子进行行业市值中性；预测收益使用 T 至 T+10 日收盘价区间收益率
资料来源：Wind，华泰研究

由于我们以 RankICIR 为适应度指标，部分因子存在空头区分度高但多头收益不佳的现象。以下有选择地展示部分多头表现较好的因子样本外分层测试结果。

Alpha1：ts_grouping_diffsortavg(sigmoid(high), turn_ori,15, 7)。(1)计算时序标准化最高价，做 sigmoid 归一化；(2)近 15 日中换手率最高与最低的 7 日内，求(1)步结果各自均值之差；本质是精细化的动量因子，反向因子。

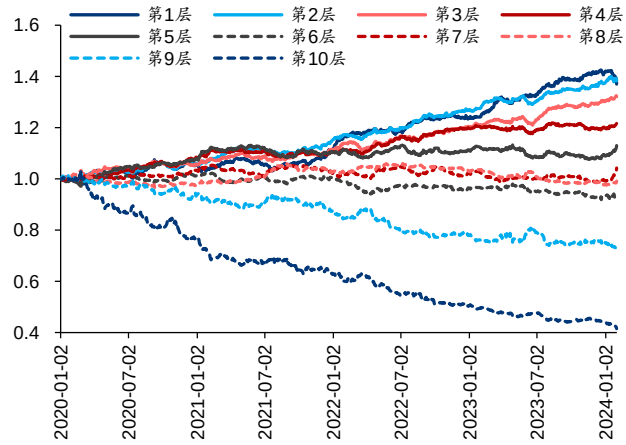
Alpha12：ts_grouping_diffsortavg(rank_add(volume, vwap), turn_ori, 40, 7)。(1)计算时序标准化成交量与 vwap 截面排序之和；(2)近 40 日中换手率最高与最低的 7 日内，求(1)步结果各自均值之差；本质是精细化的动量和流动性因子，反向因子。

图表24：Alpha1 因子样本外分层测试相对净值



注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

图表25：Alpha12 因子样本外分层测试相对净值

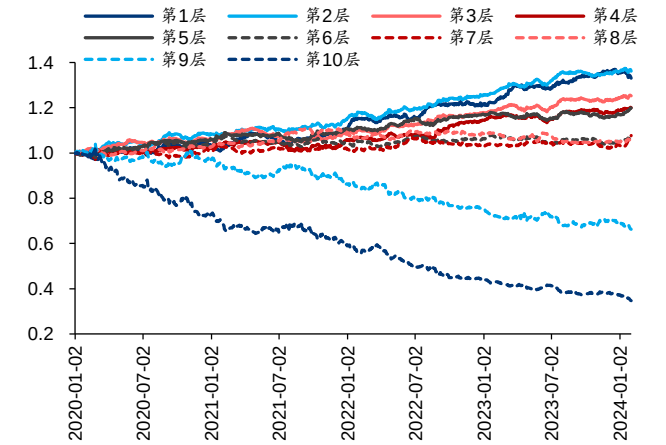


注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

Alpha224：ts_grouping_decsortavg(amt, turn_ori, 15, 4)。(1)计算时序标准化成交额；(2)近 40 日中换手率最低的 4 日内，求(1)步结果均值；本质是精细化的流动性因子，反向因子。

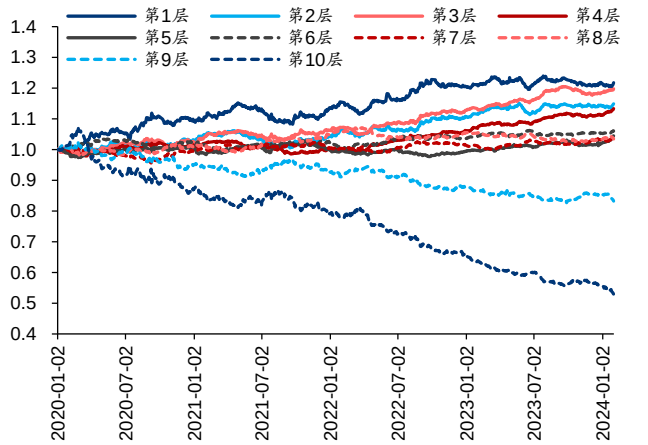
Alpha226：rank_sub(log(ts_argmin(adv20_ori, 2)), scale(ts_grouping_decsortavg(amt, turn_ori, 20, 8)))。(1)计算 20 日均成交量；(2)计算(1)步结果近 2 日最大值距今日期，取对数；(3) 近 20 日中换手率最低的 8 日内，求时序标准化成交额的均值，做截面标准化；(4)计算(2)和(3)步结果截面排序之差；逻辑较复杂，本质是精细化的低流动性因子，正向因子。

图表26： Alpha224 因子样本外分层测试相对净值



注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

图表27： Alpha226 因子样本外分层测试相对净值

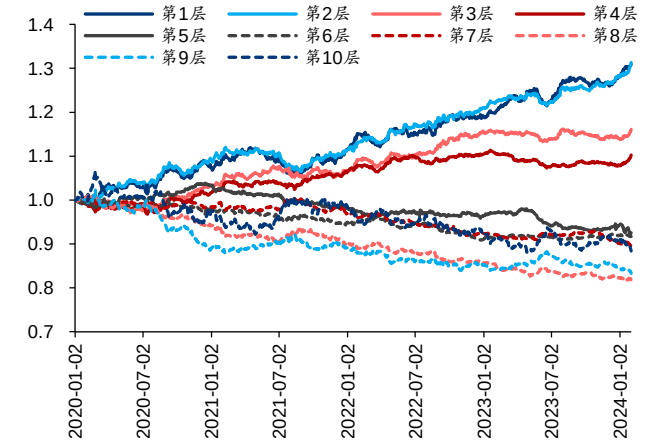


注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

Alpha229 ： neg(rank_sub(ts_grouping_decsortavg(vwap, turn_ori, 40, 7), ts_argmax(turn_ori, 6)))。(1)计算时序标准化 vwap；(2)近 40 日中换手率最低的 7 日内，求(1)步结果均值；(3)计算近 6 日换手率最高日距今日期；(4)计算(2)和(3)步结果截面排序之差；(5)取相反数；逻辑较复杂，本质是精细化的反转和低流动性因子，正向因子。

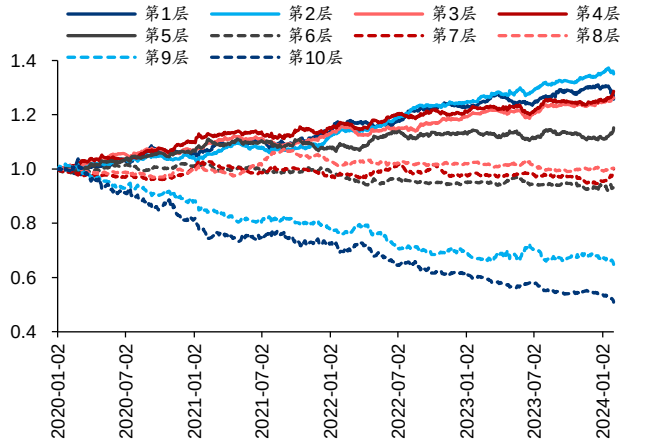
Alpha230：ts_grouping_diffsortavg(vwap, turn_ori, 40, 2)。(1)计算时序标准化 vwap；(2)近 40 日中换手率最高和最低的 2 日内，求(1)步结果各自均值之差；本质是精细化的动量因子，反向因子。

图表28： Alpha229 因子样本外分层测试相对净值



注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

图表29： Alpha230 因子样本外分层测试相对净值

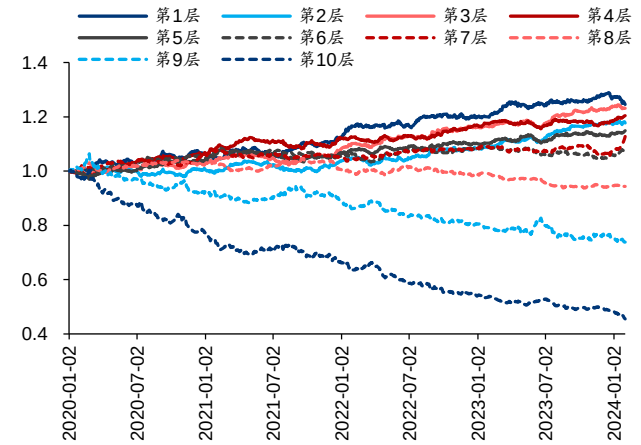


注：测试区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

Alpha271：ts_grouping_ascsortavg(ts_delaypct(low, 6), turn_ori, 40, 1)。(1)计算时序标准化最低价，计算近 6 日变动百分比；(2)取近 40 日中换手率最高日的(1)步结果；本质是精细化的动量因子，反向因子。

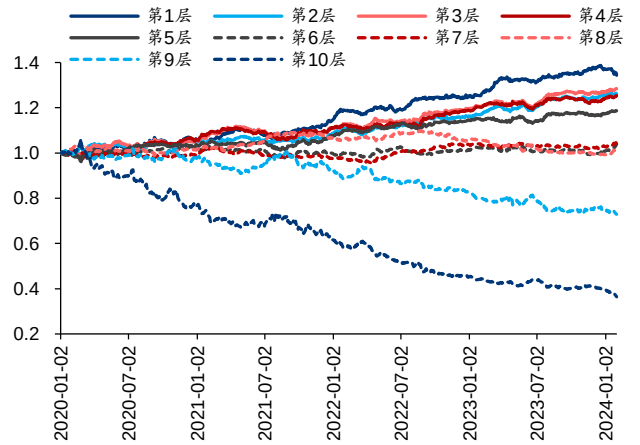
Alpha313：ts_grouping_decsortavg(scale(ts_min(adv60, 2)), turn_ori, 20, 6)。(1)计算时序标准化 60 日均成交量，求近 2 日最小值，做截面标准化；(2)近 20 日中换手率最低的 6 日内，求(1)步结果均值；本质是精细化的流动性因子，反向因子。

图表30： Alpha271 因子样本外分层测试相对净值



注：回测区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

图表31： Alpha313 因子样本外分层测试相对净值



注：回测区间为 2020-01-02 至 2024-01-31
资料来源：Wind，华泰研究

通过上述因子表达式的解读，我们发现 ts_grouping_sortavg 算子和 turn 特征的组合出现频次较高，其含义是基于过去一段时间每日换手率进行筛选，针对换手率最高或最低的部分交易日的对应指标求均值。相比直接求指标的区间均值，换手率最高或最低的交易日可能蕴含更丰富的信息。

以下展示上述 8 个因子样本外测试指标，调仓频率为 10 日，因子均做行业市值中性化。

图表32： 部分因子样本外测试指标

因子	RankIC 均值	RankICIR（未年化）	RankIC 胜率	T 均值	Top 层年化超额收益	Bottom 层年化超额收益	对冲组合年化超额收益	Top 层夏普比率	Bottom 层夏普比率	对冲组合夏普比率	Top 层信息比率	Top 层年化换手率
Alpha1	7.9%	1.11	86.0%	2.54	12.4%	-14.2%	13.3%	0.56	-0.56	1.47	1.57	34.5
Alpha12	7.9%	0.97	81.9%	2.57	14.2%	-15.0%	14.6%	0.67	-0.58	1.38	1.76	17.3
Alpha224	9.0%	0.99	82.6%	2.96	13.4%	-19.4%	16.4%	0.65	-0.73	1.37	1.57	19.1
Alpha226	6.6%	0.90	81.9%	1.80	10.7%	-9.2%	10.0%	0.57	-0.37	0.88	0.96	30.6
Alpha229	4.3%	0.59	71.7%	1.09	12.6%	3.7%	4.5%	0.66	0.15	0.47	1.41	33.9
Alpha230	6.8%	0.99	84.1%	2.18	12.2%	-10.0%	11.1%	0.55	-0.40	1.28	1.46	21.6
Alpha271	6.2%	0.87	79.9%	2.11	11.7%	-12.9%	12.3%	0.56	-0.52	1.43	1.32	17.4
Alpha313	8.4%	0.89	80.4%	2.78	13.5%	-18.1%	15.8%	0.66	-0.67	1.25	1.67	14.2

注：股票池：中证全指成分股；回测区间：2020-01-02 至 2024-01-31；因子进行行业市值中性；预测收益使用 T 至 T+10 日收盘价区间收益率
资料来源： Wind，华泰研究

总结和讨论

本研究使用 GPU 算力对遗传规划因子挖掘进行加速。遗传规划的痛点之一是运算效率低，因子进化不具备方向性，需要反复迭代筛选。我们基于 DEAP 开源框架，使用 PyTorch 拓展算子和适应度函数。结果表明，在 Intel Core i9-13900KF 和 NVIDIA GeForce RTX 4090 测试环境下，对于遗传规划全流程及核心的因子值计算和 IC 值计算环节，GPU 能提供 7~9 倍的加速效果。5 轮挖掘每轮进化 3 代总耗时 12.4 小时，得到 350 个符合入库条件的因子。

分析遗传规划全流程时间开销，大规模矩阵运算主要涉及因子值计算与 IC 值计算环节，均针对个股层面，每轮迭代需要重复数千次（取决于种群数量），并迭代若干轮，时间开销大，具备较大优化空间。而后代选择、交叉变异等环节仅作用于算子表达式，即针对因子层面，不涉及大规模矩阵计算，时间开销低。本研究测试场景下，单次计算单因子全历史的因子值和 IC 值，CPU 平均时间开销分别为 8.5 秒和 0.8 秒，GPU 平均时间开销分别为 1 秒和 0.0007 秒，GPU 加速效果显著。

遗传规划的 GPU 加速有三种可行的技术路线：RAPIDS、Numba+CUDA 和 PyTorch+CUDA。RAPIDS 代码改写工程量小，但实证表明加速效果不如 PyTorch。Numba+CUDA 在大规模运算时加速效果更好，但代码改写工程量大，且部分复杂函数仍需在 CPU 执行，通信成本高。PyTorch+CUDA 代码改写工程量小，上手容易，全部运算均可在 GPU 上完成，数据流连贯，通信成本低。本研究采用 PyTorch+CUDA 路线，不改写 DEAP 底层代码，而是新定义一批基于 PyTorch 的算子和适应度函数，添加到 DEAP 框架中。

本研究构建了以残差收益率为优化目标的因子挖掘流程。每一轮挖掘以剔除上一轮新因子的残差收益率为优化目标，适应度为样本内年化 RankICIR。因子入库条件为适应度高且与已有因子相关度低。定义 29 个初始特征和 31 个算子。当种群数量分别为 30/100/300 时，因子挖掘全流程 GPU 相比 CPU 的加速倍数分别为 4.4/7.9/8.1 倍。其中核心的因子计算和 IC 值计算环节，GPU 相比 CPU 的加速倍数分别为 4.6/7.7/8.9 倍。考虑到种群数量较低时不具备参考意义，我们认为 GPU 提供的 7~9 倍加速是相对合理的结果。

正式实验中种群数量设为 3000，进行 5 轮挖掘，每轮进化 3 代，共挖掘出 350 个符合入库条件的因子，总时间开销约 12.4 小时，平均单轮进化时间开销约 0.8 小时。对部分样本外多头表现较好的因子表达式进行解读，我们发现 ts_grouping_sortavg 算子和 turn 特征的组合出现频次较高，其含义是基于过去一段时间每日换手率进行筛选，针对换手率最高或最低的部分交易日的对应指标求均值。相比直接求指标的区间均值，换手率最高或最低的交易日可能蕴含更丰富的信息。

本研究存在以下未尽之处：(1)以 ICIR 为适应度挖掘得到的部分因子存在多头端表现不佳的问题，可使用 PyTorch 实现因子多头收益计算，优化适应度指标；(2)丰富初始特征和算子，提升因子表现；(3)受限于算力，本研究采用固定样本内时段的测试框架，可改为滚动样本内时段以应对因子失效现象；(4)有别于团队过往研究，本研究引入时序标准化特征，作为原始特征的补充，时序标准化操作能带来多大的边际提升，需要更严谨的消融实验证明；(5)其他枚举式因子挖掘算法如 OpenFE 同样面临运算效率低的缺点，或存在 GPU 加速空间。

参考资料

Sathia, V., Ganesh, V., & Nanditale, S. R. T. (2021). Accelerating Genetic Programming using GPUs. arXiv preprint arXiv:2110.11226.

百度飞桨《基于 Numba 的 CUDA Python 编程简介》：https://aistudio.baidu.com/projectdetail/5965667

风险提示

遗传规划挖掘的选股因子是历史经验的总结，存在失效可能。遗传规划得到的因子可解释性较低，使用需谨慎。机器学习模型存在过拟合的风险。

免责声明

分析师声明

本人，林晓明、何康，兹证明本报告所表达的观点准确地反映了分析师对标的证券或发行人的个人意见；彼以往、现在或未来并无就其研究报告所提供的具体建议或所表达的意见直接或间接收取任何报酬。

一般声明及披露

本报告由华泰证券股份有限公司（已具备中国证监会批准的证券投资咨询业务资格，以下简称“本公司”）制作。本报告所载资料是仅供接收人的严格保密资料。本报告仅供本公司及其客户和其关联机构使用。本公司不因接收人收到本报告而视其为客户。

本报告基于本公司认为可靠的、已公开的信息编制，但本公司及其关联机构(以下统称为“华泰”)对该等信息的准确性及完整性不作任何保证。

本报告所载的意见、评估及预测仅反映报告发布当日的观点和判断。在不同时期，华泰可能会发出与本报告所载意见、评估及预测不一致的研究报告。同时，本报告所指的证券或投资标的的价格、价值及投资收入可能会波动。以往表现并不能指引未来，未来回报并不能得到保证，并存在损失本金的可能。华泰不保证本报告所含信息保持在最新状态。华泰对本报告所含信息可在不发出通知的情形下做出修改，投资者应当自行关注相应的更新或修改。

本公司不是 FINRA 的注册会员，其研究分析师亦没有注册为 FINRA 的研究分析师/不具有 FINRA 分析师的注册资格。

华泰力求报告内容客观、公正，但本报告所载的观点、结论和建议仅供参考，不构成购买或出售所述证券的要约或招揽。该等观点、建议并未考虑到个别投资者的具体投资目的、财务状况以及特定需求，在任何时候均不构成对客户私人投资建议。投资者应当充分考虑自身特定状况，并完整理解和使用本报告内容，不应视本报告为做出投资决策的唯一因素。对依据或者使用本报告所造成的一切后果，华泰及作者均不承担任何法律责任。任何形式的分享证券投资收益或者分担证券投资损失的书面或口头承诺均为无效。

除非另行说明，本报告中所引用的关于业绩的数据代表过往表现，过往的业绩表现不应作为日后回报的预示。华泰不承诺也不保证任何预示的回报会得以实现，分析中所做的预测可能是基于相应的假设，任何假设的变化可能会显著影响所预测的回报。

华泰及作者在自身所知情的范围内，与本报告所指的证券或投资标的不存在法律禁止的利害关系。在法律许可的情况下，华泰可能会持有报告中提到的公司所发行的证券头寸并进行交易，为该公司提供投资银行、财务顾问或者金融产品等相关服务或向该公司招揽业务。

华泰的销售人员、交易人员或其他专业人士可能会依据不同假设和标准、采用不同的分析方法而口头或书面发表与本报告意见及建议不一致的市场评论和/或交易观点。华泰没有将此意见及建议向报告所有接收者进行更新的义务。华泰的资产管理部门、自营部门以及其他投资业务部门可能独立做出与本报告中的意见或建议不一致的投资决策。投资者应当考虑到华泰及/或其相关人员可能存在影响本报告观点客观性的潜在利益冲突。投资者请勿将本报告视为投资或其他决定的唯一信赖依据。有关该方面的具体披露请参照本报告尾部。

本报告并非意图发送、发布给在当地法律或监管规则下不允许向其发送、发布的机构或人员，也并非意图发送、发布给因可得到、使用本报告的行为而使华泰违反或受制于当地法律或监管规则的机构或人员。

中国香港

本报告由华泰证券股份有限公司制作,在香港由华泰金融控股（香港）有限公司向符合《证券及期货条例》及其附属法律规定的机构投资者和专业投资者的客户进行分发。华泰金融控股（香港）有限公司受香港证券及期货事务监察委员会监管，是华泰国际金融控股有限公司的全资子公司，后者为华泰证券股份有限公司的全资子公司。在香港获得本报告的人员若有任何有关本报告的问题,请与华泰金融控股（香港）有限公司联系。

香港-重要监管披露

- 华泰金融控股（香港）有限公司的雇员或其关联人士没有担任本报告中提及的公司或发行人的高级人员。
- 有关重要的披露信息，请参华泰金融控股（香港）有限公司的网页 https://www.htsc.com.hk/stock_disclosure

其他信息请参见下方 **“美国-重要监管披露”**。

美国

在美国本报告由华泰证券（美国）有限公司向符合美国监管规定的机构投资者进行发表与分发。华泰证券（美国）有限公司是美国注册经纪商和美国金融业监管局（FINRA）的注册会员。对于其在美国分发的研究报告，华泰证券（美国）有限公司根据《1934 年证券交易法》（修订版）第 15a-6 条规定以及美国证券交易委员会人员解释，对本研究报告内容负责。华泰证券（美国）有限公司联营公司的分析师不具有美国金融监管（FINRA）分析师的注册资格，可能不属于华泰证券（美国）有限公司的关联人员，因此可能不受 FINRA 关于分析师与标的公司沟通、公开露面和所持交易证券的限制。华泰证券（美国）有限公司是华泰国际金融控股有限公司的全资子公司，后者为华泰证券股份有限公司的全资子公司。任何直接从华泰证券（美国）有限公司收到此报告并希望就本报告所述任何证券进行交易的人士，应通过华泰证券（美国）有限公司进行交易。

美国-重要监管披露

- 分析师林晓明、何康本人及相关人士并不担任本报告所提及的标的证券或发行人的高级人员、董事或顾问。分析师及相关人士与本报告所提及的标的证券或发行人并无任何相关财务利益。本披露中所提及的“相关人士”包括 FINRA 定义下分析师的家庭成员。分析师根据华泰证券的整体收入和盈利能力获得薪酬，包括源自公司投资银行业务的收入。
- 华泰证券股份有限公司、其子公司和/或其联营公司，及/或不时会以自身或代理形式向客户出售及购买华泰证券研究所覆盖公司的证券/衍生工具，包括股票及债券（包括衍生品）华泰证券研究所覆盖公司的证券/衍生工具，包括股票及债券（包括衍生品）。
- 华泰证券股份有限公司、其子公司和/或其联营公司，及/或其高级管理层、董事和雇员可能会持有本报告中所提到的任何证券（或任何相关投资）头寸，并可能不时进行增持或减持该证券（或投资）。因此，投资者应该意识到可能存在利益冲突。

评级说明

投资评级基于分析师对报告发布日后 6 至 12 个月内行业或公司回报潜力（含此期间的股息回报）相对基准表现的预期（A 股市场基准为沪深 300 指数，香港市场基准为恒生指数，美国市场基准为标普 500 指数，台湾市场基准为台湾加权指数，日本市场基准为日经 225 指数），具体如下：

行业评级

增持：预计行业股票指数超越基准
中性：预计行业股票指数基本与基准持平
减持：预计行业股票指数明显弱于基准

公司评级

买入：预计股价超越基准 15%以上
增持：预计股价超越基准 5%~15%
持有：预计股价相对基准波动在-15%~5%之间
卖出：预计股价弱于基准 15%以上
暂停评级：已暂停评级、目标价及预测，以遵守适用法规及/或公司政策
无评级：股票不在常规研究覆盖范围内。投资者不应期待华泰提供该等证券及/或公司相关的持续或补充信息

法律实体披露

中国：华泰证券股份有限公司具有中国证监会核准的“证券投资咨询”业务资格，经营许可证编号为：91320000704041011J

香港：华泰金融控股（香港）有限公司具有香港证监会核准的“就证券提供意见”业务资格，经营许可证编号为：AOK809

美国：华泰证券（美国）有限公司为美国金融业监管局（FINRA）成员，具有在美国开展经纪交易商业务的资格，经营业务许可编号为：CRD#:298809/SEC#:8-70231

华泰证券股份有限公司

南京
南京市建邺区江东中路 228 号华泰证券广场 1 号楼/邮政编码：210019

电话：86 25 83389999/传真：86 25 83387521
电子邮件：ht-rd@htsc.com

北京
北京市西城区太平桥大街丰盛胡同 28 号太平洋保险大厦 A 座 18 层/邮政编码：100032
电话：86 10 63211166/传真：86 10 63211275
电子邮件：ht-rd@htsc.com

深圳
深圳市福田区益田路 5999 号基金大厦 10 楼/邮政编码：518017
电话：86 755 82493932/传真：86 755 82492062
电子邮件：ht-rd@htsc.com

上海
上海市浦东新区东方路 18 号保利广场 E 栋 23 楼/邮政编码：200120
电话：86 21 28972098/传真：86 21 28972068
电子邮件：ht-rd@htsc.com

华泰金融控股（香港）有限公司
香港中环皇后大道中 99 号中环中心 58 楼 5808-12 室
电话：+852-3658-6000/传真：+852-2169-0770
电子邮件：research@htsc.com
http://www.htsc.com.hk

华泰证券（美国）有限公司
美国纽约公园大道 280 号 21 楼东（纽约 10017）
电话：+212-763-8160/传真：+917-725-9702
电子邮件: Huatai@htsc-us.com
http://www.htsc-us.com