Optimization of Automated Trading Strategies through Nature-Inspired Computing

Bulat Akhmatov, Alexandra Vabnits, Sofia Shulyak
DS-01

Table of contents

Introduction

Stock (or cryptocurrency) prices are chaotic, noisy, non-stationary, yet not fully randomly distributed numerical data.

Trading strategy – a set of rules and techniques used to determine when to buy or sell stocks, currencies, or other financial instruments with the goal of generating profits.

Automated trading system – a program that executes a defined trading strategy with specific chosen parameters by interacting with stock or currency exchange.

Freqtrade

It is an open-source crypto trading bot. It provides backtesting, plotting and money management functionality.

Strategies are user-defined python classes with automated trading logic, defined by class functions and parameters.

Trading operations of the bot on one of the example strategies:

Problem statement

Question :
How can we choose parameters for a trading strategy to maximize profit?

Proposed solution:
Develop a GA optimization for parameters of Freqtrade strategies to enhance the performance of the trading bot

Problem statement

Candidate and gen types

Strategies

Sample Strategy (4 params),
Diamond Strategy, and Strategy 005 (8 params)

Parameter types

CategoricalParameter, DecimalParameter,
IntParameter, BooleanParameter
from freqtrade.strategy

Why GA is applicable for that problem?

- From one side, crypto market is relatively random,
- On the other hand, large enough search spaces make bruteforce not efficient.
  → GA random nature and convergence speed is well suitable.

Simple strategy with well pre-balanced search space:
more than 6 billion combinations

```
buy_vertical_push = DecimalParameter(0.5, 1.5, decimals=3, default=1, space='buy')
buy_horizontal_push = IntParameter(0, 10, default=0, space='buy')
buy_fast_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='buy')
buy_slow_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='buy')
sell_vertical_push = DecimalParameter(0.5, 1.5, decimals=3,  default=1, space='sell')
sell_horizontal_push = IntParameter(0, 10, default=0, space='sell')
sell_fast_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='sell')
sell_slow_key = CategoricalParameter(['open', 'high', 'low', 'close', 'volume'], default='low', space='sell')
```

Methodology | GA steps

1. Strategy Representation

Python class and a set of user defined parameters. Parameters = genes, candidate = a set of parameters

2. Initial population:

trading strategies with hyperparameters set randomly from a search space.

3. Fitness evaluation:

strategy backtesting on historical market data

4. Genetic operators:

mutations and crossover in hyperparameters

5. Population composition:

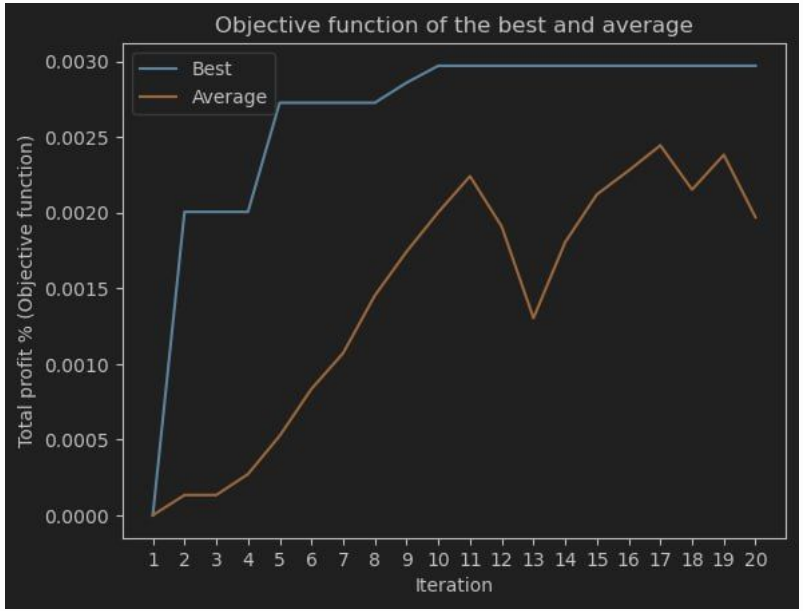
the top 1/3 of candidates +
mutated 1/3 +
crossovered 1/3.
Mutation and crossover chances are equal for all candidates.

Experiment and Results

TABLE I
OPTIMIZATION TECHNIQUES COMPARISON RESULTS

Strategy name	Best of initial population profit	Profit with GA	GA time	Profit with hyper-opt	Hyperopt time
Sample Strategy	0.113%	0.185%	17 min	0.184%	9 min
Diamond	0.00%	0.297%	23 min	0.34%	7 min
Strategy 005	0.133%	0.273%	16 min	0.268%	7 min

epoches: 20
population size: 30
backtesting from 2024-01-01 to 2024-04-05
with 5m timeframe



Diamond strategy optimized with GA

Conclusion

Genetic algorithm is a reasonably suitable solution for optimising algorithmic trading strategies.

Considerations:

- Strategy optimization triggered rarely → GA's slow convergence is not a problem.
- No deep analysis → no understanding of market situation → profit is limited and unstable
  These strategies are simple examples we provided to show how GA can improve strategy performance

Future work:

- Multiprocessing,
- GP-based strategy creation
- Support of built-in Freqtrade functions (ROI, stoploss, e.c.)