PipsEvolution Reference Guide

Version 1.0.277

Mariusz Wołoszyn

July 13, 2013

Contents

1 Introduction

This reference guide describes the core components of PipsEvolution, a Genetic Programming platform for MetaTrader. It is not intended as a tutorial.

Please go to official documentation page at http://pipsevolution.org/docs/ to obtain all available documentation.

2 Distributed components

PipsEvolution platform consists of following components distributed within the installation package.

2.1 PipsEvolution.exe

Main application binary responsible for performing GP work flow tasks.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution`

2.2 PipsEvolutionTrain.ex4

The compiled expert used during evolution. It is almost identical to PipsEvolutionExport.mq4 but contains some additional code to communicate with PipsEvolution.exe.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution\experts`

2.3 PipsEvolutionExport.mq4

Source code of the EA. It's mainly used for exporting final strategies or populations for further use.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution\experts`

2.4 PipsEvoFunctionsImport.mqh

Header file containing declarations of functions that can be used as building blocks (build set) during creation and evolution of signal functions along basic arithmetic functions.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution\experts\include`

Not all functions are implemented in the current version despite being declared here.

2.5 PipsEvoFunctionsLib.ex4

Library implementing functions declared in PipsEvoFunctionsImport.mqh.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution\experts`

2.6 PipsEvolutionConfig.ini

Configuration file containing all the settings. This file is optional, if not present the build in default configuration is used.

Location: main application folder, e.g. `C:\Program Files (x86)\PipsEvolution`

3 Files and folders created during execution

3.1 Evolution

This folder holds a copy of PipsEvolution.ini file used to perform respective evolution, log file and file with all strategies results as well as every generation signal functions evolved code (in source and compiled form). Those files can later be used to export given strategy or generation. All the files are saved in a subfolder identifying named evolution. The name of created folder is representing current EVOLUTION_NAME and is PIPS_EVO_01 by default. In that subfolder you may find files like in following example:

```
PIPS_EVO_01_AllGenerations_results.txt
PIPS_EVO_01_log.txt
PIPS_EVO_01_Long_P500_G0.ex4
PIPS_EVO_01_Long_P500_G0.mq4
PIPS_EVO_01_Long_P500_G1.ex4
PIPS_EVO_01_Long_P500_G1.mq4
...
PipsEvolutionConfig.ini
```

The first file contains results of each and every individual in every generation encoded in semicolon separated file. The second file is the log file created during run of this experiment, it contains more or less what was displayed on screen during the evolution. The last file is the PipsEvolutionConfig.ini file used during evolution. You may need it later to consult settings used for this experiment. All the .ex4 and .mq4 files are containing code created via genetic programming and evolutionary selected for the best fitness. Each file is representing single generation in the evolution. The file name encodes experiment name (PIPS_EVO_01), the strategy direction (Long), letter P followed by population size (500) and letter G followed by generation number (0 to 9).

3.2 Export

This folder is used to export desired strategies. It is populated with results of export operation.

4 Configuration options

PipsEvolution is holding its configuration in plain ini file format. The default file name is `PipsEvolutionConfig.ini` and is located in the main application folder. Configuration file consists of 5 sections, each corresponding to different aspects of application. The section name starts in square bracket followed by configuration options. Below is example configuration file:

```
[TERMINAL]
MT4_DIRECTORY=C:\Program Files (x86)\MetaTrader 4
MT4_DATA_DIRECTORY=C:\Program Files (x86)\MetaTrader 4\history\default
[STRATEGY]
EVOLUTION_NAME=PIPS_EVO_01
SYMBOL=EURUSD
TIMEFRAME=240
MODEL=2
FROM=2011.01.01
TO=2011.03.31
TRADING_MODE=0
EXIT_MODE=0
POSITION_SIZE=0.10
STOP_LOSS=0.0050
TAKE_PROFIT=0.0050
TESTER_INIT_DEPOSIT=10000
TESTER_DEPOSIT_CURR=USD
```

Each configuration option is described in following chapters with one chapter dedicated to the respective section.

4.1 TERMINAL

This section holds options identifying MetaTrader 4 location as well as some advanced internal MetaTrader terminal configurations.

4.1.1 MT4_DIRECTORY

Description Full path to the main MetaTrader 4 folder. PipsEvolution uses this path to find `terminal.exe` and other files and folders required during processing.

Type: String.

Default value `C:\Program Files (x86)\MetaTrader 4`

4.1.2 MT4_DATA_DIRECTORY

Description Full path to the desired MetaTrader4 history data folder.

Type: String.

Default value `C:\Program Files (x86)\MetaTrader 4\history\default`

4.1.3 LOGIN

Description Used to pass the MT4 server account name (login). It's advanced option.

Please use it with extreme care as MetaTrader tester at each run obtains spread data from the server so in theory every run of tester might be with different spread causing discrepancies among results obtained within PipsEvolution and directly in MetaTrader tester.

Type: String.

Default value 123456

4.1.4 PASSWORD

Description Used to pass the MT4 server password. It's advanced option. Please see comments to the LOGIN option.

Type: String.

Default value pass

4.2 STRATEGY

This section holds options responsible for the tester configuration like symbol, timeframe, dates and so on and generic evolved strategy properties like its name, trading mode, exit mode, stop loss etc.

4.2.1 EVOLUTION_NAME

Description The name of the current evolution (sometimes called experiment). For each experiment the dedicated folder containing its results and signal functions is created named EVOLUTION_NAME in Evolution folder.

Type: String.

Default value PIPS_EVO_01

4.2.2 SYMBOL

Description The symbol used to run the strategy on. This option is directly passed on to the MetaTrader tester and has to match one of available symbol in historical data folder.

Type: String.

Default value EURUSD

4.2.3 TIMEFRAME

Description Timeframe used to run the strategy on. This option is directly passed on to the MetaTrader tester. (In MetaTrader GUI it's inappropriately called *"Period"*).

Allowed values 1, 5, 15, 30, 60, 240, 1440 corresponding to M1, M5, M15, M30, H1, H4, D1 timeframes. Please refer to MQL4 Refference for details.

Default value 240

4.2.4 MODEL

Description Tester modeling method. This option is directly passed on to MetaTrader tester. Please refer to Client Terminal - User Guide -> Methods of Modeling for more details.

Allowed values

- **0** Every tick (based on all available least timeframes)
- **1** Control points (the nearest less timeframe is used)
- **2** Open prices only (fastest method to analyze the bar just completed)

Default value 2

4.2.5 FROM

Description Tester *"From"* date. This option is directly passed on to the MetaTrader tester. The date the experiment and strategy starts to run from in the format YYYY.MM.DD.

Type: String.

Default value 2011.01.01

4.2.6 TO

Description Tester *"To"* date. This option is directly passed on to the MetaTrader tester. The date the strategy ends to run on in the format YYYY.MM.DD.

Type: String.

Default value 2011.03.31

4.2.7 TRADING_MODE

Description Expert's trading mode: long or short. This option is passed to the PipsEvolutionTrain.ex4 expert (Expert properties -> Inputs) as *tradingMode*. Indicates whether the training expert should perform long or short trades only effectively evolving only long or short playing strategies.

Allowed values

0 MODE_LONG

1 MODE_SHORT

Default value 0

4.2.8 EXIT_MODE

Description Expert's exit mode: constant or signal. This option is passed to the PipsEvolutionTrain.ex4 expert (Expert properties -> Inputs) as *exitMode*. Indicates the way the training expert closes opened position.

Allowed values

0 EXIT_MODE_CONSTANT; Close the position only by hitting stopLoss or *takeProfit*.

1 EXIT_MODE_CONSTANT_OR_SIGNAL; Close the position either by hitting stopLoss or *takeProfit* (if either is set) or by close signal evolved exactly the same way as open signal.

Default value 0

4.2.9 POSITION_SIZE

Description Expert's position size in lots (or mini lots depending on account type used in MetaTrader). This option is passed to the PipsEvolutionTrain.ex4 expert (Expert properties -> Inputs) as positionSize.

Type: Double.

Default value 0.1

4.2.10 STOP_LOSS

Description Expert's stop loss distance. Nonzero value required when EXIT_MODE is set to 0. This option is passed to the PipsEvolutionTrain.ex4 expert (Expert properties -> Inputs) as stopLoss.

Type: Double.

Default value 0.00500

4.2.11 TAKE_PROFIT

Description Expert's take profit distance. Nonzero value required when EXIT_MODE is set to 0. This option is passed to the PipsEvolutionTrain.ex4 expert (Expert properties -> Inputs) as takeProfit.

Type: Double.

Default value 0.00500

4.2.12 TESTER_INIT_DEPOSIT

Description Tester "*Initial deposit*" setting (Expert properties ->Testing). This option is directly passed on to the MetaTrader tester.

Type: Integer.

Default value 10000

4.2.13 TESTER_DEPOSIT_CURR

Description Tester "*Initial deposit*" *currency* setting (Expert properties ->Testing). This option is directly passed on to the MetaTrader tester.

Type: String.

Default value USD

4.3 BUILD_SET

This section holds options responsible for the Genetic Programming building blocks i.e. MQL build in functions used in evolution. All Boolean settings can hold 0 or 1 value indicating that the given function is used to construct strategies during evolution or not respectively.

4.3.1 OPEN

Description Use current bar open price.

Type: Boolean.

Default value 1

4.3.2 HIGH

Description Use previous bar high price.

Type: Boolean.

Default value 0

4.3.3 LOW

Description Use previous bar low price.

Type: Boolean.

Default value 0

4.3.4 CLOSE

Description Use previous bar close price.

Type: Boolean.

Default value 0

4.3.5 HIGHEST

Description Use MQL4 iHighest() function. Please refer to MQL4 Reference iHighest.

Type: Boolean.

Default value 0

4.3.6 LOWEST

Description Use MQL4 iLowest() function. Please refer to MQL4 Reference iLowest.

Type: Boolean.

Default value 0

4.3.7 BID

Description Use current bid price.

Type: Boolean.

Default value 0

4.3.8 ASK

Description Use current Ask price.

Type: Boolean.

Default value 0

4.3.9 IND_MA

Description Use moving average indicator (MQL4 iMA() function). Please refer to MQL4 Reference iMA.

Type: Boolean.

Default value 1

4.3.10 IND_MACD

Description Use moving averages convergence/divergence indicator (MQL iMACD() function). Please refer to MQL4 Reference iMACD.

Type: Boolean.

Default value 1

4.3.11 IND_RSI

Description Use relative strength index indicator (MQL iRSI() function). Please refer to MQL4 Reference iRSI.

Type: Boolean.

Default value 1

4.3.12 MAX_LOOKBACK

Description Indicates how many bars back can an indicator be looked back. In most cases it is equivalent to indicator's *shift* argument (index of the value from the indicator buffer relative to the current bar. For price arrays like OPEN[], HIGH[] it's an array index. Value must be greater then 1.

Type: Integer.

Default value 12

4.3.13 MAX_IND_PERIOD

Description Indicates the maximal value of the indicator's *period* argument.

Type: Integer.

Default value 20

4.4 TARGETS

This section holds options responsible for defining the way the fitness function is calculated. Each available target is represented by three options indicating:

1. whether the target is enabled,
2. whether the target is maximized or minimized.
3. what weight is associated to given target.

4.4.1 TOTAL_NET_PROFIT

Description Indicates to optimize total net profit of a system.

Allowed values 0, 1 (off, on)

Default value 1

4.4.2 TOTAL_NET_PROFIT_OPT

Description Indicates whether to maximize or minimize total net profit of a system.

Allowed values 0, 1 (minimize, maximize)

Default value 1

4.4.3 TOTAL_NET_PROFIT_W

Description Weight associated with total net profit target.

Type: Integer.

Default value 5

4.4.4 PROFIT_FACTOR

Description Indicates to optimize profit factor of a system.

Allowed values 0, 1 (off, on)

Default value 1

4.4.5 PROFIT_FACTOR_OPT

Description Indicates whether to maximize or minimize profit factor of a system.

Allowed values 0, 1 (minimize, maximize)

Default value 1

4.4.6 PROFIT_FACTOR_W

Description Weight associated with profit factor target.

Type: Integer.

Default value 3

4.4.7 EXPECTED_PAYOFF

Description Indicates to optimize expected payoff of a system.

Allowed values 0, 1 (off, on)

Default value 0

4.4.8 EXPECTED_PAYOFF_OPT

Description Indicates whether to maximize or minimize expected payoff of a system.

Allowed values 0, 1 (minimize, maximize)

4.4.9 EXPECTED_PAYOFF_W

Description Weight associated with expected payoff target.

Type: Integer.

Default value 0

4.5 GP_CONFIG

This section holds options responsible for defining how the genetic and evolutionary engine is operating.

4.5.1 POPULATION

Description Number of strategies initially created and evolved. Number of individuals in population – population size.

Type: Integer.

Minimum value 50

Maximum value 500

Default value 500

4.5.2 GENERATIONS

Description Number of generation that has to pass before the evolution is finished. Generations are number starting from 0 to GENERATIONS-1.

Type: Integer.

Minimum value 1

Maximum value 20

Default value 10

4.5.3 INIT_MAX_DEPTH

Description Maximal depth of program syntax trees created during seeding process.

Type: Integer.

Minimum value 3

Maximum value 6

Default value 4

4.5.4 MAX_DEPTH

Description Maximal depth of syntax tree allowed during evolution.

Type: Integer.

Minimum value 6

Maximum value 12

Default value 10

4.5.5 SEED_MODE

Description This option defines seeding mode used during initial creation of syntax tree.

Allowed values

0 SEED_FULL: create trees of INIT_MAX_DEPTH;

1 SEED_GROW: create trees of growing depth from 3 to INIT_MAX_DEPTH;

Default value 1

4.5.6 SELECTION_MODE

Description This option defines selection algorithm used to pick best individuals for reproduction, crossover and mutation.

Allowed values

0 SELECTION_ROULETTE: selection of individual strategy is performed at random with selection probability proportional to fitness value;

1 SELECTION_TOURNAMENT: in tournament selection a TOURNAMENT_SIZE number of individuals is first chosen at random and then the best one is selected;

2 SELECTION_RANDOM: random.

Default value 0

4.5.7 FITNESS_BASE

Description This option defines the way the fitness function is applied

Allowed values

0 FITNESS_BASE_RANK

1 FITNESS_BASE_NORMALISED

Default value 0

4.5.8 TOURNAMENT_SIZE

Description The number of individuals selected to each tournament size if SELECTION_MODE of type SELECTION_MODE is used.

Type: Integer.

Minimum value 20

Maximum value Population size.

Default value 2

4.5.9 REPRODUCTION_RATE

Description Relative rate of reproduction. The number of reproduced individuals is calculated according to following equation:

reproduced_individuals = (REPRODUCTION_RATE / (REPRODUCTION_RATE + CROSSOVER_RATE + MUTATION_RATE)) * POPULATION

Type: Integer.

Minimum value 10

Maximum value 100

Default value 10

4.5.10 CROSSOVER_RATE

Description Relative rate of crossover. The number of crossed individuals is calculated according to following equation:

crossed_individuals = (CROSSOVER_RATE / (REPRODUCTION_RATE + CROSSOVER_RATE + MUTATION_RATE)) * POPULATION

Type: Integer.

Minimum value 10

Maximum value 100

Default value 80

4.5.11 MUTATION_RATE

Description Relative rate of mutation. The number of mutated individuals is calculated according to following equation:

mutated_individuals = POPULATION - (reproduced_individuals + crossed_individuals)

Type: Integer.

Minimum value 10

Maximum value 100

Default value 10

4.5.12 ALPHA

Description

Type: Double.

Minimum value

Maximum value

Default value

5 PipsEvolutionTrain Expert Input Variables

5.1 evolutionName

Description Name of evolution. Used to distinguish experiments and name Evolution file. During training is is set automatically to value of EVOLUTION_NAME option.

Type: String.

5.2 strategyNumber

Description Number of strategy to execute. During training process the terminal tester iterates over this variable to obtain results of every strategy.

Type: Integer.

5.3 tradingMode

Description Expert's trading mode: long or short. Indicates whether the training expert should perform long or short trades only effectively evolving only long or short playing strategies. During training is is set automatically to value of TRADING_MODE option.

Allowed values

0 TRADING_MODE_LONG

1 TRADING_MODE_SHORT

5.4 exitMode

Description Indicates the way the strategy closes opened positions. During training is is set automatically to value of EXIT_MODE option.

Allowed_values:

0 EXIT_MODE_CONSTANT: close by stopLoss and takeProfit only. stopLoss and takeProfit cannot be zero.

1 EXIT_MODE_CONSTANT_AND_SIGNAL: close by evolved signal as well as stopLoss and takeProfit. In this mode stopLoss and takeProfit can be equal zero.

5.5 stopLoss

Description Distance from opening price used as *stoploss* argument for OrderSend. During training it is set automatically to value of STOP_LOSS option.

Type: Double.

Allowed_values: Positive. Nonzero if exitMode is set to 0.

5.6 takeProfit

Description Distance from opening price used as *takeprofit* argument for OrderSend. During training it is set automatically to value of TAKE_PROFIT option.

Type: Double.

Allowed_values: Positive. Nonzero if exitMode is set to 0.

5.7 positionSize

Description Number of lots, used as *volume* argument for OrderSend. During training it is set automatically to value of POSITION_SIZE option.

Type: Double.

Allowed_values: Positive.

6 PipsEvolutionExport Expert Input Variables

6.1 longStrategyNumber

Description Number of strategy to execute.This argument is used when whole generation evolved with TRADING_MODE is exported.

Type: Integer.

6.2 shortStrategNumber

Description Number of strategy to execute.This argument is used when whole generation evolved with TRADING_MODE is exported.

Type: Integer.

6.3 stopLossStrategyLong

Description Distance from opening price used as *stoploss* argument for OrderSend. This argument is used when strategy evolved with TRADING_MODE is exported.

Type: Double.

Allowed_values: Positive. Nonzero if strategy was evolved with EXIT_MODE set to 0.

6.4 stopLossStrategyShort

Description Distance from opening price used as *stoploss* argument for OrderSend. This argument is used when strategy evolved with TRADING_MODE is exported.

Type: Double.

Allowed_values: Positive. Nonzero if strategy was evolved with EXIT_MODE set to 0.

6.5 takeProfitStrategyLong

Description Distance from opening price used as *takeprofit* argument for OrderSend. This argument is used when strategy evolved with TRADING_MODE is exported.

Type: Double.

Allowed_values: Positive. Nonzero if strategy was evolved with EXIT_MODE set to 0.

6.6 takeProfitStrategyShort

Description Distance from opening price used as *takeprofit* argument for OrderSend. This argument is used when strategy evolved with TRADING_MODE is exported.

Type: Double.

Allowed_values: Positive. Nonzero if strategy was evolved with EXIT_MODE set to 0.

6.7 positionSize

Description Number of lots, used as *volume* argument for OrderSend.

Type: Double.

Allowed_values: Positive.

6.8 slippage

Description Maximum price slippage, used as *slippage* argument for OrderSend.

Type: Integer.

Allowed_values: Positive.

6.9 magicNumber

Description User defined strategy identifier. Used as *magic* argument for OrderSend.

Type: Integer.

Allowed_values: Positive.