Barra Optimizer 8.7 User Guide

July 2017

1 About This Guide 1
2 Portfolio Optimization with Barra Optimizer 2
2.1 Introducing Barra Optimizer 2
2.2 Key Features 3
3 Basic Optimization Concepts 6
3.1 Portfolio Optimization Definition 6
3.1.1 Problem Formulation 6
3.1.2 Problem Classification 8
3.2 Units 9
3.3 Portfolio Balance Constraint 9
3.4 Asset Types 10
3.4.1 Regular Assets 10
3.4.2 Cash Assets 10
3.4.3 Currencies 11
3.4.4 Futures-Type Assets 11
3.4.5 Composites 11
3.5 Return 11
3.6 Bounds and Linear or Piecewise Linear Constraints 12
3.6.1 Asset and Factor Bounds 12
3.6.2 General Linear Constraints 12
3.6.3 Constraint Bounds and Slack Variables 12
3.6.4 Beta Constraint 12
3.6.5 General Piecewise Linear Constraints 13
3.6.6 Total Active Weights Constraint 13
3.6.7 Issuer Constraints 14
3.7 Risk Control in Optimization—Aversions, Constraints, and Grouping 15
3.7.1 Factor Risk Models 15
3.7.2 Asset-Covariance Risk Models 16
3.7.3 Benchmarks—Multiple and With Non-Investment Universe Assets 16
3.7.3.1 Benchmark With Non-Investment Universe Assets 16

3.7.3.2 Multiple Benchmarks 17
3.7.4 Risk Aversion 17
3.7.5 Total Risk vs. Active Risk 17
3.7.6 Multiple Risk Terms 17
3.7.7 Secondary Risk Model 18
3.7.8 Risk Constrained Optimization 19
3.7.9 Risk Target Optimization 24
3.8 Transaction Cost, Holding Cost, and Turnover Control 25
3.8.1 Piecewise Linear Transaction Costs 25
3.8.2 Non-Linear Transaction Costs 26
3.8.3 Fixed Transaction Costs 27
3.8.4 Fixed Holding Costs 27
3.8.5 Turnover Functions and Constraints 28
3.8.5.1 Turnover Base Value 29
3.8.5.2 Upper Bound on Buy or Sell Turnover 29
3.8.5.3 Upper Bound on Turnover by Group 30
3.8.5.4 Option to Exclude Cash 30
3.9 Constraint Flexibility—Penalties, Soft Bounds, and Hierarchy 30
3.9.1 Penalty Functions 30
3.9.1.1 Quadratic vs. Linear Penalties 31
3.9.1.2 Symmetric vs. Asymmetric Penalties 32
3.9.1.3 Penalty Details 32
3.9.1.4 Pure Linear Penalties with a Single Target vs. with a Free Region 33
3.9.2 Soft Constraints 33
3.9.3 Constraint Hierarchy 34
3.10 Optimization Inspection 37
3.10.1 Optimality Conditions 37
3.10.2 Optimality Tolerances 37
3.10.3 Maximum Time Limit 38
3.10.4 Indication of the Optimization Problem Type 38
3.10.5 Indication of the Output Portfolio as Heuristic or Optimal 39
3.10.6 Trade List Information 39
3.10.7 Solver Information 39

3.10.8 Impact to Utility 40
3.10.9 Small Weights or Trades 40
3.10.10 Portfolios and Messages Returned 40
4 Advanced Optimization Topics 41
4.1 Threshold and Cardinality Constraints—Paring Optimization 41
4.1.1 Minimum Holding Thresholds/Holding Level Paring 42
4.1.2 Minimum Trade Thresholds/Transaction Level Paring 42
4.1.3 Holding Cardinality/Asset Paring 43
4.1.4 Trade Cardinality/Paring 43
4.1.5 Paring Tolerances 44
4.1.6 Linear Penalties on Paring Constraints 44
4.1.7 Upper Bound on Utility and Branch-and-Bound Algorithm for Paring Cases 47
4.2 Roundlotting—Optimal and Post-Optimization 48
4.2.1 Optimal Roundlotting or Constraint-Aware Roundlotting 48
4.2.2 Post-Optimization 48
4.3 Risk Parity 49
4.3.1 Definition 49
4.3.2 Risky and Risk-free Assets 50
4.3.3 Inputs and Scope 50
4.3.4 Usage Recommendations 50
4.3.5 Risk Parity Portfolio Construction 51
4.4 Diversification Control 51
4.4.1 Portfolio Concentration Limit Constraint 51
4.4.2 Diversification Ratio Limit Constraint 52
4.5 Residual Alpha 53
4.6 By-Side Optimization—Leverage, Turnover by Side, Cardinality by Side 54
4.6.1 Leverage (Hedge) Constraints 54
4.6.1.1 Hedge Scaling Constraint 55
4.6.1.2 Weighted Long or Short Leverage Constraints 55
4.6.1.3 Total Leverage Constraint 56
4.6.1.4 Short-Long Leverage Ratio Constraint 56
4.6.1.5 Net-Total Leverage Ratio Constraint 56

4.6.1.6 Weighted Total Leverage Constraints 56
4.6.2 Penalty on Leverage Constraints 57
4.6.3 Turnover-by-Side Constraints 57
4.6.4 Threshold and Cardinality-by-Side Constraints 57
4.6.5 Short Rebates/Short Costs 58
4.6.6 Parametric Optimization with Leverage Constraints 59
4.7 Solving a General Linear or Convex Quadratic Program 59
4.8 Tax-Aware Optimization 60
4.8.1 Tax Rates 60
4.8.1.1 One-Rate 60
4.8.1.2 Two-Rate 61
4.8.2 Netting Different Types of Gains and Losses 63
4.8.3 Tax Arbitrage 63
4.8.4 Tax Harvesting 63
4.8.5 Tax Lot Trading Rules 64
4.8.6 Wash Sales 64
4.8.6.1 Ignore Wash Sales 64
4.8.6.2 Trade-Off Wash Sales 64
4.8.6.3 Disallow Wash Sales 65
4.8.7 Capital Gain Tax 65
4.8.8 Overlap Constraint—Control of Total Active Benchmark Holdings 65
4.8.9 Non-Convexity 66
4.8.10 Cardinality and Threshold Constraints in Tax-Aware Optimization 66
4.8.11 Tax-Aware Risk Target Optimization 66
4.9 5-10-40 Rule 66
4.10 Maximizing the Sharpe or Information Ratio 67
4.10.1 Maximizing the Sharpe Ratio 67
4.10.2 Maximizing the Information Ratio 67
4.11 Optimal Frontiers—Parametric Optimization 67
4.11.1 Risk-Return Efficient Frontiers 68
4.11.1.1 Varying Return 68
4.11.1.2 Varying Risk 68
4.11.2 Constraint-Utility Frontiers 69

4.11.2.1 Varying Turnover 69
4.11.2.2 Varying Tax Cost 69
4.11.2.3 Varying Leverage 70
4.12 Multiple-Period Optimization 70
4.13 Multiple-Account Optimization 72
4.13.1 Joint Market-Impact Transaction Costs 72
4.13.2 Cross-Account Constraints 73
4.13.3 Two Basic Approaches 73
4.13.3.1 Total-Welfare Maximization Approach 74
4.13.3.2 Nash Equilibrium (Game Theory) Approach 74
4.13.4 Post-Optimization Transaction Cost Allocation 74
4.13.5 Scopes and Limitations 75
5 Using the Barra Optimizer APIs 76
5.1 Create a Workspace 76
5.2 Add Assets into Workspace 77
5.3 Construct Initial Portfolio, Benchmark, and Trade Universe 78
5.4 Define Risk Model 78
5.5 Prepare Optimization Case 83
5.5.1 Link Risk Models 83
5.5.2 Set Up the Objective 83
5.5.3 Set Up Constraints 84
5.5.4 Set Up Efficient Frontier 89
5.5.5 Set Up Tax-Aware Optimization Parameters 90
5.6 Run Optimization 90
5.7 Run Multiple-Period Optimization 91
5.8 Run Multiple-Account Optimization 93
5.9 Inspect Output 96
5.9.1 Non-Interactive Approach 96
5.9.2 Interactive Approach 97
5.9.3 Basic Output Information 97
5.9.4 Additional Statistics for Optimal Portfolio or Initial Portfolio 98
5.9.5 Constraint Slack Information 99

5.9.6 Error Handling 99
5.9.7 Information for Infeasible Cases 102
5.9.8 KKT Table Items 103
5.9.9 Asset-level Transaction Costs 103
5.10 Data Serialization 103
5.11 Release Resources 104
5.12 Thread Safety 104
5.13 Reference and Tutorial Sample Codes 104
6 Using MATLAB, R, and SAS Interface 105
6.1 Working with MATLAB Interface 105
6.1.1 Setting Up MATLAB 105
6.1.1.1 Verify Your Java Version 105
6.1.1.2 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux) 106
6.1.1.3 Quick Start to Set Up the Java Class Path and Library Path 106
6.1.1.4 Manually Setting Up Java Class Path and Library Path 107
6.1.2 MATLAB Tutorials 108
6.1.3 Java APIs to for setting up workspace data 109
6.1.4 MATLAB Toolbox 110
6.2 Working with R Language Interface 111
6.2.1 Setting Up the R Language Environment 111
6.2.1.1 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux) 111
6.2.1.2 Initialize Barra Optimizer Access 112
6.2.2 R Tutorials and Sample Code 112
6.3 Working with SAS Interface 113
6.3.1 Setting Up SAS Java 115
6.3.1.1 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux) 115
6.3.1.2 Set the Java Class Path 115
6.3.2 SAS Sample 116
7 Using Python Interface 118
8 Using the Barra Optimizer XML 119
8.1 <WorkSpace> 122
8.2 <Data> 122

8.2.1 <Attribute> ... 122
8.2.2 <Grouping_Scheme> ... 124
8.2.3 <Transaction_Costs> ... 125
8.2.4 <Attribute_Assignment> ... 126
8.2.5 <Data_File> Attribute ... 127
8.3 <Portfolios> ... 127
8.4 <Risk_Models> ... 127
8.5 <Rebalance_Profiles> ... 128
8.5.1 <Utility> ... 128
8.5.2 <Linear_Constraint> ... 130
8.5.2.1 <Cash_Asset_Bound> ... 130
8.5.2.2 <NonCash_Asset_Bound> ... 130
8.5.2.3 <Asset_Bounds> ... 131
8.5.2.4 <Asset_Bounds_By_Group> ... 132
8.5.2.5 <Factor_Constraint> ... 132
8.5.2.6 <Factor_Constraint_By_Group> ... 133
8.5.2.7 <Custom_Constraint > ... 133
8.5.2.8 <Constraint_By_Group> ... 133
8.5.2.9 <Beta_Constraint> ... 133
8.5.2.10 <Noncash_Asset_Penalty> ... 133
8.5.2.11 <Asset_Penalty_By_Group> ... 133
8.5.2.12 <Asset_Penalty> ... 134
8.5.2.13 <Asset_Free_Range_Penalty> ... 134
8.5.2.14 <Asset_Trade_Size> ... 135
8.5.3 <Cardinality_Constraint> ... 135
8.5.4 <Cardinality_Constraint_By_Group> ... 135
8.5.5 <Threshold_Constraint> ... 135
8.5.6 <Threshold_Constraint_By_Group> ... 136
8.5.7 <Threshold_Constraint_By_Asset> ... 136
8.5.8 <Transaction_Cost_Constraint> ... 136
8.5.9 <Roundlot_Constraint> ... 137
8.5.10 <Turnover_Constraint> ... 137
8.5.11 <Turnover_Constraint_By_Group> ... 137
8.5.12 <Turnover_Definition> ... 137

8.5.13 <Leverage_Constraint> ... 137
8.5.14 <Weighted_Total_Leverage_Constraint> ... 138
8.5.15 <Risk_Constraint> ... 139
8.5.16 <Risk_Constraint_By_Group> ... 139
8.5.17 <Risk_Parity_Constraint> ... 139
8.5.18 <Five_Ten_Forty_Rule> ... 140
8.5.19 <Constraint_Priority> ... 140
8.5.20 <Penalty> ... 140
8.5.21 <Free_Range_Penalty> ... 141
8.5.22 <Portfolio_Concentration_Constraint> ... 141
8.5.23 <Diversification_Ratio > ... 141
8.5.24 <Total_Active_Weight_Constraint> ... 141
8.5.25 <Total_Active_Weight_Constraint_By_Group> ... 141
8.5.26 <Issuer_Constraint> ... 141
8.5.27 Multiple-Period Optimization: Cross-Period Constraints ... 142
8.5.28 <General_PWLI_Constraint> ... 142
8.5.29 <Overlap_Constraint> ... 142
8.5.30 <Include_Futures> ... 142
8.6 <Rebalance_Job> ... 143
8.6.1 <Rebalance_Profile_ID> ... 143
8.6.2 <Initial_Portfolio_ID> ... 143
8.6.3 <Universe_Portfolio_ID> ... 143
8.6.4 <Reference_Portfolios> ... 143
8.6.5 <Portfolio_Base_Value> ... 143
8.6.6 <Cashflow_Weight> ... 143
8.6.7 <Primary_Risk_Model_ID> ... 143
8.6.8 <Secondary_Risk_Model_ID> ... 143
8.6.9 <Attribute_Assignment> ... 144
8.6.10 <Optimization_Setting> ... 144
8.7 <Rebalance_Result> ... 145
8.8 Multiple-Period Optimization Using XML ... 145
8.10 Using XML APIs ... 149

8.10.1 Setting Up in C# 151
8.10.2 Setting Up in MATLAB 152
9 Using the Command Line Executable 154
9.1 Loading Barra Models Direct Data 154
9.2 Running Optimization 154
9.2.1 Optimize With Each Element of the Workspace 154
9.2.2 Optimize With a Single Workspace 155
9.3 Converting Protobuf ⇔ XML 155
9.4 Converting WSP → Protobuf/XML 156
9.5 Converting WorkSpace Protobuf/XML → WSP 156
9.6 Converting WSP => Legacy XML format 156
9.7 Additional Options 156
10 Frequently Asked Questions 157
10.1 Why do I get different portfolios from different Optimizer versions? 157
10.2 Why do I get a suboptimal or counterintuitive solution? 157
10.3 Which features may lead to suboptimal or counterintuitive solutions? 157
10.4 Which feature combinations are supported in Barra Open Optimizer? 157
10.5 Is there any limit on the number of risk terms in the objective function? 157
10.6 Is there any limit on the number of risk constraints? 157
10.7 Is there any limit on the number of risk models? 158
10.8 Is there any limit on the number of benchmarks? 158
Appendix A: Explanation of Outputs 159
Appendix B. Availability of Features and Functions 161
Appendix C: Glossary 163
Appendix D: Object Model 164
Appendix E: API Class References 167
Appendix F: API Diagram 169
References 170

1 About This Guide

Barra Optimizer is a software package that handles a wide variety of portfolio construction and optimization problems. Barra Open Optimizer is the stand-alone version of the same solver (Barra Optimizer) available in Barra PortfolioManager, BarraOne, and Barra Aegis. Barra Open Optimizer provides C++, Java, C#, COM, R, and Python APIs to help you to develop your applications using C++, Java, C#, Visual Basic, Excel VBA, MATLAB, Python, R, and SAS.

This guide explains the various types of portfolio optimizations that Barra Optimizer can handle and includes the following information about Barra Optimizer:

- Brief problem definitions and high-level descriptions of the features and functionalities.
- Insightful mathematical formulations and illustrative examples.
- API information for programmers who would like to develop advanced portfolio optimization applications with ease by taking advantage of the powerful features offered by Barra Optimizer.

The problem definitions and mathematical formulations in this guide can be used as a reference when either using the Barra Open Optimizer library directly or using Barra Optimizer via Barra's portfolio and risk management products. However, the updated API information here only applies to the standalone Barra Open Optimizer 8.7.

Note: The actual implementation in a given product may also be more or less restrictive than what is described here. If you are interested in the theoretical background of optimization, you may find references listed at the end of this guide.

Typographic Conventions

Throughout this guide, vectors and matrices are in bold fonts, whereas scalar variables are in italicized fonts.

2 Portfolio Optimization with Barra Optimizer

Optimization is the process of locating the best possible answer given a model of your goals (objectives or utility) and the rules and restrictions you need to follow (constraints).

Creating an optimization involves the following three important steps:

- Assess the goals and restrictions for your portfolio
- Quantify and model the goals and restrictions
- Formulate Step 2 into objectives and constraints for use in an optimizer

Step 1 is specific to the portfolio and investor. Step 2 is a key component of Barra's modeling offerings, and is discussed extensively on the MSCI Support site. This guide discusses the third step in creating an optimization problem.

Once the optimization problem has been created, it needs to be run and then, analyzed.

Barra Optimizer contains numerous introspective features. For information about the features, see the Key Features section.

2.1 Introducing Barra Optimizer

Barra Optimizer is a software library designed to solve a wide variety of portfolio optimization problems. The goal of optimization is to construct an optimal portfolio, balancing various competing objectives (such as maximizing return, minimizing risk, etc.) while taking into account specified constraints. It is specially designed to create portfolios and trades that maximize an objective function subject to a set of practical trading or investment constraints. The objective function consists of optional return, risk, transaction cost, and other terms. Utility function is an alternative name for the objective function. Barra Optimizer is an integral part of many MSCI products and services.

Barra Optimizer incorporates proprietary solvers developed in-house, resulting from over two decades of dedicated research at MSCI in the area of financial optimization. It provides fast and reliable results for well-defined, convex portfolio construction and rebalancing cases. It also applies innovative, high-quality heuristic techniques to certain ill-behaved, complex portfolio optimization problems. It takes advantage of the special structure of multi-factor risk models employed by many portfolio managers.

Studies show that Barra Optimizer can handle 30,000 assets and more than 4,000 constraints. Its only capacity limitation is the size of available computing memory. Barra Optimizer currently supports the following types of problems:

- Standard Mean-Variance Optimization
- Maximizing the Sharpe Ratio or Information Ratio
- Threshold and Cardinality Optimization
- Risk Parity Portfolio Construction
- Risk Target

- Risk Constrained Optimization
- Portfolio Construction with Diversification Control
- Long/Short (Hedge) Optimization
- Tax-Aware Optimization
- Optimal Roundlotting
- 5-10-40 Rules
- Parametric Optimization / Efficient Frontiers
- Multiple-Period Optimization
- Multiple-Account Optimization

However, a particular release of Barra Optimizer (either stand-alone or in a particular Barra product, such as BarraOne™) may offer only a subset of the full capabilities. The easiest way to use Barra Optimizer is through Barra's portfolio and risk management products, such as Barra Aegis, BarraOne, and Barra PortfolioManager. These applications automatically package user inputs into the formats required by Barra Optimizer.

The standalone Barra Open Optimizer, on the other hand, accommodates alternative risk models and offers a much wider range of features and functionalities. It also affords sophisticated users more control over the selection of internal parameters, which makes the product more flexible and powerful.

2.2 Key Features

The following list highlights some of the most prominent features of Barra Optimizer:

- Consists of a fast, efficient, and reliable convex solver for mean-variance optimization
- Employs innovative heuristics to tackle non-convex, non-linear, or discrete problems
- Offers maximizing Sharpe ratio or information ratio as alternative objective functions
- Supports various risk models
- Facilitates multi-criteria optimization by allowing different multipliers for different terms in the objective function
- Supplies two separate risk aversion parameters for the common factor risk and specific risk
- Allows up to two risk models, multiple risk terms, and multiple benchmarks in a given problem
- Permits the benchmarks or covariance matrices in the objective function to be different from those in the constraints
- Accepts any separable, convex, piecewise linear transaction cost function, both in the objective function and in the constraints
- Accommodates nonlinear and fixed transaction costs in the objective function
- Incorporates residual alpha as an optional term in the objective function

- Assists in tilting portfolios towards specific targets through the use of various penalty functions, including:
  - Symmetric penalties
  - Asymmetric penalties
  - Quadratic penalties
  - Linear-only penalties
- Supports a variety of constraint types and combinations of constraint types:
  - Bounds on assets, factors, or constraint slacks
  - User-defined general linear constraints
  - Beta constraint
  - 5/10/40 rules
  - Turnover upper limit
  - Transaction cost upper limit
  - Maximum or minimum number of assets
  - Maximum or minimum number of trades
  - Minimum holding or transaction size level
  - Risk constraints
  - Leverage (Hedge) constraints
  - Round lot constraints
  - Portfolio concentration limit
  - Diversification ratio limit
  - General piecewise linear constraints
  - Total active weights constraint
- Affords more flexible user control by offering soft constraints, constraint hierarchy, and an optimality tolerance multiplier
- Enables roundlotting either during the optimization process or post optimization
- Provides comprehensive and flexible modeling choices in By-Side Optimization:
  - Various types of leverage constraints
  - Leverage-by-side constraints
  - Turnover-by-side constraints
  - Cardinality-by-side
  - Modeling of short rebates or costs
  - Threshold-by-side
  - Various risk constraints
  - Risk target optimization
- Provides sophisticated tax-aware optimization consideration:
  - Grants a choice of one or two tax rates
  - Adapts to HIFO, LIFO, or FIFO trading rules
  - Allows targeting on or netting of different types of gains and losses
  - Affords control over the bounds of net as well as gross gains or losses
  - Offers three wash sales handling options
- Generates efficient frontiers for a range of return, risk, turnover limit, tax, or other constraint bounds at the user's request

- Controls total risk or tracking error via a variety of flexible mechanisms:
  - Upper bound on specific risk or factor risk
  - Upper bound on fractional contribution of specific risk or factor risk
  - Upper bound on overall risk from a subgroup of assets
  - Upper bound on fractional contribution of overall risk from a subgroup of assets or factors
  - Risk parity constraint
- Showcases a novel mean-variance approach to Multiple-Period Optimization for look-ahead portfolio construction and trade scheduling.
- Offers two approaches to Multiple-Account Optimization—Total Welfare Maximization and Nash Equilibrium—for dealing with joint market-impact transaction costs and cross-account constraints.

3 Basic Optimization Concepts

This section presents the traditional mean-variance portfolio optimization problem, discusses its classifications, and describes the individual terms in the objective function, also known as the utility function, and constraint categories. In addition, it describes the basic settings and constraints that you can view or edit in the Barra Optimizer workspace.

3.1 Portfolio Optimization Definition

3.1.1 Problem Formulation

From the user's perspective, a portfolio optimization problem in Barra Optimizer can be represented as:

Maximize:

$$\text{Objective Function} = \text{Return Terms} - \text{Risk Terms} - \text{Transaction Cost} - \text{Other Terms}$$

Subject to:

$$\text{Standard Constraints} \begin{cases} \text{Portfolio Balance Constraint} \\ \text{General Linear Constraints} \\ \text{Factor Constraints} \\ \text{Beta Constraint} \\ \text{Turnover Limit Constraints} \\ \text{Transaction Cost Limit Constraints} \\ \text{General Convex Piecewise Linear Constraints} \\ \text{Bounds on Assets/Factors/Constraints} \end{cases}$$

$$\text{Special Constraints} \begin{cases} \text{Paring Constraints} \\ \text{Round Lot Constraints} \\ \text{5/10/40 Constraints} \\ \text{Leverage (Hedge) Constraints} \\ \text{General Non-Convex Piecewise Linear Constraints} \\ \text{Tax-Related Constraints} \\ \text{Risk Constraints} \end{cases}$$

Among all terms in the objective function, the return ones are linear, the risk ones are quadratic, and the transaction cost ones can be pure linear, piecewise linear, quadratic, fixed per trade, or even special convex nonlinear. The additional terms include the capital-gain tax cost, residual alpha, and so on, which may be complicated and sometimes highly non-linear depending on the additional constraints that define them. Most terms allow a multiplier in the front to adjust its impact in the objective function. All terms are optional.

Similarly, all constraints are optional except the Portfolio Balance Constraint. The standard constraints consist of only linear or convex piecewise linear constraints. The special constraints are more complicated, usually discrete, or nonlinear in nature.

Mathematically, a typical mean-variance portfolio optimization can be formulated as follows:

$$\underset{\mathbf{h}}{\text{Maximize}}: \quad f(\mathbf{h}) = \lambda_r \mathbf{r}^{\mathrm{T}}\mathbf{h} - 100(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_B) - \lambda_{Tc}Tc(\mathbf{h},\mathbf{h}_0) - \lambda_{Pen}Pen(\mathbf{s},\mathbf{w}) - g(\mathbf{h}) \tag{1}$$

Subject to:

$\sum_{i\neq futures} h_i = \sum_{i\neq futures} h_{0i} + \mathrm{CFW}$	Portfolio Balance Constraint	(2)
$\mathbf{l}_h \le \mathbf{h} \le \mathbf{u}_h$	Asset ranges	(3)
$\mathbf{s} = \mathbf{A}\mathbf{h}$	Definitional Constraint for General Slack Variable	(4)
$\mathbf{l}_s \le \mathbf{s} \le \mathbf{u}_s$	Range for Slack Variables	(5)
$\mathbf{w} = \mathbf{X}^{\mathrm{T}}\mathbf{h}$	Definitional Constraint for Factor Slack Variables	(6)
$\mathbf{l}_X \le \mathbf{w} \le \mathbf{u}_X$	Factor Exposure Range	(7)
$l_\beta \le \beta(\mathbf{h}) \le u_\beta$	Beta Constraint	(8)
$To(\mathbf{h},\mathbf{h}_0) \le u_{To}$	Turnover Constraint	(9)
$Tc(\mathbf{h},\mathbf{h}_0) \le u_{Tc}$	Transaction Cost Constraint	(10)
$L_p^i \le P^i(\mathbf{h},\mathbf{h}_0) \le U_p^i, \quad i = 1, 2, \cdots, n_p$	General Piecewise Linear Constraints	(11)
$\mathbf{l}_{Con} \le Con(\mathbf{h}) \le \mathbf{u}_{Con}$	Additional Constraints	(12)

where,

λ_F	Common factor risk aversion	
λ_D	Specific risk aversion	
λ_r	Return multiplier	
λ_{TC}	Transaction cost multiplier	
λ_{pen}	Penalty multiplier	
$\mathbf{h}$	Portfolio holding weights	(nx1)
$\mathbf{h}_0$	Initial Portfolio weights	(nx1)
$\mathbf{h}_B$	Benchmark holding weights	(nx1)
$\mathbf{r}$	Alpha or expected asset return	(nx1)
$\mathbf{w}$	Factor exposure slacks	(kx1)
$\mathbf{A}$	Coefficient matrix for general constraints	(mxn)

$\mathbf{D}$	Specific covariance matrix	(nxn)
$\mathbf{X}$	Factor exposures	(nxk)
$\mathbf{F}$	Factor covariance matrix	(kxk)
$\mathbf{l}$	Lower bounds, per constraint	
$\mathbf{u}$	Upper bounds, per constraint	
CFW	Cash Flow/Base Value	
$Pen(\mathbf{s},\mathbf{w})$	Penalty functions on constraint and factor slacks	
$Tc(\mathbf{h},\mathbf{h}_0)$	Transaction cost function	
$To(\mathbf{h},\mathbf{h}_0)$	Turnover function	
$g(\mathbf{h})$	Additional non-standard objective terms	
$Con(\mathbf{h})$	Additional non-standard constraints	
$P^i(\mathbf{h},\mathbf{h}_0)$	The i^{th} general piecewise linear function	
n	Number of assets in investment universe	
m	Number of general constraints	
k	Number of factors	
n_p	Number of general piecewise linear constraints	

Notes:

- All the above notations are used throughout this guide, unless otherwise noted.
- All the constraints are optional, except for the portfolio balance constraint, which is required and automatically created.
- Currently, $g(\mathbf{h})$ is only supported in the form of a power function, although any differentiable function may be considered in the future.

Some of the other constraints (for example, threshold, cardinality, and long/short leverage constraints, etc.) that are referenced as non-standard constraints are described later in this document.

3.1.2 Problem Classification

We often distinguish a long-short optimization from a standard one. Typically, Standard (Portfolio) Optimization consists of only the standard constraints. For information about standard constraints, see the Portfolio Optimization Definition section.

The presence of any special constraints would render the optimization non-standard, or a special type. On many occasions, we also classify optimization problems into convex and non-convex categories. It should be noted that

standard optimization might not be convex. Similarly, convex optimization may not be standard. We also want to emphasize that Barra Optimizer cannot guarantee a global optimal solution in the case of non-convex problems.

A convex portfolio optimization problem enjoys a special property—any convex combination of two feasible portfolios will remain a feasible portfolio. Convexity is desirable because it makes the search for an optimal solution easy and predictable. Convex problems are well-behaved, can be solved with less difficulty, and are guaranteed to produce a global optimal portfolio, if one exists. Non-convex problems are usually very complicated mathematically and more challenging computationally.

All the standard constraints are convex, and many special constraints are non-convex. Convex functions, convex constraints, and convex problems are separate yet related concepts. To qualify an optimization problem as convex, not only must all constraints be convex, the objective function must also be convex if it is a minimization problem or concave if it is a maximization problem.

Because of their complexity and difficulty, most non-convex portfolio optimization problems are tackled by heuristic procedures. Even though Barra Optimizer strives to employ innovative, intuitive, efficient, and effective heuristics, and the solution quality is usually good in many cases, there is no guarantee that the heuristic solution it generates for a particular non-convex case is indeed global optimal.

3.2 Units

Barra Optimizer uses *fractions* rather than *percentages* for units of input. This means that all input, including the alphas, variances, covariances, as well as holdings, transaction costs, and tax costs are entered as decimals relative to the base value used to calculate portfolio weights. For example, an alpha of 2% is input as 0.02. Similarly, if the specific variance of an asset is $(30\%)^2$, then it is entered as 0.09. Raw alpha scores need to be scaled or normalized before inputting into the Barra Optimizer.

Everything associated with the holdings (for example, upper and lower bounds on assets, coefficients of constraints, and so on) should be scaled accordingly.

Transaction costs and tax costs should be expressed as the fraction of their dollar amount to the base value. An exception to this is the breakpoints for piecewise linear transaction costs, which are entered in high-level APIs as dollar amounts.

3.3 Portfolio Balance Constraint

The portfolio balance constraint is an important constraint that is automatically enforced by Barra Optimizer. Portfolio optimization will not change the value of the initial portfolio plus the cash flow value:

$$PV = PV_0 + CF \tag{13}$$

where,

- PV is the optimal portfolio value
- PV_0 is the initial portfolio value
- CF is the cash flow value

Because, $PV = BV * \sum_{j \in J} \mathbf{h}_j$ where, BV is the base value used for computing asset weights, the optimal holding $\mathbf{h}$ must satisfy the following portfolio balance constraint:

$$\sum_{i \neq futures} h_i = \frac{PV_0 + CF}{BV} \tag{14}$$

Alternatively, the same constraint can be expressed as a function of the initial weights:

$$\sum_{i \neq futures} h_i = \sum_{i \neq futures} h_{0i} + CF / BV \tag{15}$$

Notes:

- Typically, you can select the initial portfolio value or the initial portfolio value plus the cash flow value as the base value.
- You can also select an assigned value as the base value. To avoid numerical issues during optimization, the base value should not be far away from $PV_0 + CF$.
- You do not set the portfolio balance constraint explicitly as **Barra Optimizer will add it automatically**. This constraint can be disabled and replaced with a user-defined linear constraint (although this is not recommended).
- The portfolio balance constraint enforces the total net trade to be equal to the cash flow if no cash asset is used. If no cash flow is set, the total net trade will be 0. A cash asset is used to control the net total trade value. For more information, see the Turnover Functions and Constraints section.

3.4 Asset Types

3.4.1 Regular Assets

An asset that is not classified as any other asset type is referred to as a regular asset in Barra Optimizer. It can represent an equity asset, a non-equity instrument, or even an asset class. The only requirement is that the asset must be consistent with the risk model used.

3.4.2 Cash Assets

To leave some of the portfolio as un-invested, a cash asset should be added to the workspace. Bounds on the cash asset can be set in the same way as bounds on any other asset, and can be used to specify a maximum or minimum level of cash. The cash asset can have a negative weight. The optimal results from the optimization will obey the specified cash bounds. However, post-optimal roundlotting may lead to a breach of the bounds.

By default, cash flow and change in cash asset position both affect the amount of turnover. An option is available to exclude cash in turnover definitions. For more information, see the Turnover Functions and Constraints section.

3.4.3 Currencies

Currencies are treated as a regular asset for all purposes outside non-linear transaction costs. Normally a currency asset has exposure to just its currency factor. Multiple currencies can occur within a single portfolio, for example, when using a global or regional model.

3.4.4 Futures-Type Assets

Futures-type assets are used to reflect the special nature of futures and other contracts that are entered into with 0 or minimal investment. They are treated differently than other assets. Major differences include the following:

- Futures-type assets are not counted in the portfolio balance constraint. The portfolio market value will not change even though the position of futures may change before and after optimization.
- Futures-type assets, by default, are not counted in turnover computation.
- Futures-type assets, by default, are not taken into account when considering threshold or cardinality constraints.
- Futures-type assets, by default, are not counted in leverage constraints in hedge (long/short) optimization.
- Futures-type assets, by default, are not taken into account when considering roundlotting.

We have added separate options to allow futures in turnover, threshold or cardinality, hedge and roundlotting constraints.

3.4.5 Composites

Composites are used to reflect the look-through exposure gained when using an instrument that has exposure to underlying components of the risk model. For example, S&P 500 ETF or Future would be a composite when used with a US-stock model. By using the composite functionality, the optimizer correctly accounts for the exposure to the underlying stocks within the risk control functions. Effectively, the weight in the composite is passed through to the weight of the underlying assets when using the risk aversion or risk constraints. For other constraints and objective terms, the composite is treated as a normal asset.

An asset can be both a composite and futures-type asset.

3.5 Return

The (portfolio) return in Barra Optimizer is generally defined as the inner product of the asset return vector ($\mathbf{r}$) and the asset holding vector ($\mathbf{h}$)— $\mathbf{r}^{\mathrm{T}}\mathbf{h}$. As such, the return term is linear in $\mathbf{h}$. For cases with no benchmark, the individual asset returns are total return. For cases with a benchmark, the individual asset returns may represent either a total or an active return with respect to the corresponding return of a benchmark—the difference between the two is a just a constant, which should not affect the optimization outcome. In the latter case, the asset return vector $\mathbf{r}$ is also known as the asset alpha, denoted as $\boldsymbol{\alpha}$.

3.6 Bounds and Linear or Piecewise Linear Constraints

By definition, standard constraints are linear or convex piecewise linear constraints. In Barra Optimizer, they are classified into a number of categories according to their roles or applications, as described in the following sections.

3.6.1 Asset and Factor Bounds

Barra Optimizer allows users to set lower and upper bounds on each asset's weight in the portfolio. For example, one can require the portfolio weight of IBM, $\mathbf{h}_1$, to satisfy the inequality $0 \le \mathbf{h}_1 \le 0.05$ or the weight of Intel, $\mathbf{h}_2$ to be in the range $-0.01 \le \mathbf{h}_1 \le 0.02$.

Users can also control the magnitude of a factor exposure by using an upper or lower bound or both. For instance, one can demand that a portfolio's exposure to the "size" factor be less than one standard deviation of the benchmark's exposure to the same factor.

3.6.2 General Linear Constraints

Many real-world investment rules or restrictions may be modeled or approximated by a general linear constraint. For example, the requirement that the sum of weights of all assets that have exposure to the Oil Industry factor must be less than 3% of the total portfolio value can be represented by a general linear constraint. Barra Optimizer supports all general linear constraints. These constraints may be set directly or by using a slack variable.

Barra Optimizer may accommodate user-defined linear constraints. For example, the return constrained problems can be easily set up with a user-defined constraint of the form; $return \ge \text{Lower Bound}$, where $return$ is defined as $\mathbf{r}^{\mathrm{T}}\mathbf{h}$.

3.6.3 Constraint Bounds and Slack Variables

In the context of optimization, a **slack variable** is a variable that is added to an inequality constraint to transform it to equality. Introducing a slack variable replaces an inequality constraint with an equality constraint and a bound constraint on the slack variable. Slack variables are introduced internally in Barra Optimizer to make user-defined inequality constraints conform to a standard equality format.

For example, equations (4) - (5) are internal replacements of the user-specified general linear constraints

$$\mathbf{l}_s \le \mathbf{A}\mathbf{h} \le \mathbf{u}_s \tag{16}$$

Similarly, equations (6)-(7) are replacements of the factor constraints

$$\mathbf{l}_X \le \mathbf{X}^{\mathrm{T}}\mathbf{h} \le \mathbf{u}_X \tag{17}$$

The lower bound, upper bound, both, or neither can be set for any linear constraint, either in inequality or equality format. If a fixed value is desired on the constraint, set both the lower bound and the upper bound to that value.

3.6.4 Beta Constraint

A Beta constraint is, technically, a general linear constraint. However, Barra Optimizer does provide users with an option to set a beta constraint with the betas calculated within the optimization. Beta measures a portfolio's

volatility, or systematic risk, with respect to the benchmark. Beta is calculated using the risk model and benchmark:

$$\boldsymbol{\beta} = (\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}} + \mathbf{D})\mathbf{h}_B \,/\, \mathbf{h}_B{}^{\mathrm{T}}(\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}} + \mathbf{D})\mathbf{h}_B \tag{18}$$

The $Beta(\mathbf{h})$ term in the Beta constraint (8) is given by:

$$Beta(\mathbf{h}) = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{h} \tag{19}$$

Note that a benchmark may contain assets outside the investment universe. For more information, see the Benchmarks—Multiple and With Non-Investment Universe Assets section.

3.6.5 General Piecewise Linear Constraints

Many useful constraints such as the turnover limit constraint, transaction cost limit constraints, leverage constraints, and tax-related constraints are special cases of the general, separable piecewise linear constraints. These constraints are discussed in detail in the Transaction Cost, Holding Cost, and Turnover Control section and the Tax-Aware Optimization section.

Mathematically, these constraints can be represented by:

$$l_{gpw} \le gpw(h_1, h_2, \cdots, h_n) = \sum_{i=1}^{n} gpw(h_i) \le u_{gpw} \tag{20}$$

where,

$$gpw(h_i) = \sum_{j=1}^{N_i} \delta_{ij}\, c_{ij} \min(h_i - b_{ij}, b_{i(j+1)} - b_{ij}) \tag{21}$$

and,

$$\delta_{ij} = \begin{cases} 1, & if\ h_i - b_{ij} \ge 0 \\ 0, & otherwise \end{cases} \tag{22}$$

for any sets of c_{ij} and b_{ij} satisfying $b_{i,-k_{i1}} < \cdots < b_{i,-2} < b_{i,-1} < b_{i,0} < b_{i,1} < b_{i,2} < \cdots < b_{i,k_{i2}}$.

Conceptually, the b_{ij}'s are the break points and the c_{ij}'s are the slopes of the individual piecewise linear functions. The special break point $b_{i,0}$ is where the piecewise linear function for h_i is zero, i.e., $gpw(h_i) = 0$. There are k_{i1} breaks points below $b_{i,0}$, and k_{i2} break points above $b_{i,0}$. In total, there are $N_i + 1$ break points and N_i slopes for each decision variable h_i, $i = 1, 2, \cdots, n$, where $N_i = k_{i1} + k_{i2}$.

For convex piecewise linear constraints, either the lower bound l_{gpw} is negative infinity and the slopes c_{ij}'s are monotonically increasing, i.e., $c_{i,-k_{i1}} \le \cdots \le c_{i,-2} \le c_{i,-1} \le c_{i,1} \le c_{i,2} \le \cdots \le c_{i,k_{i2}}$, or the upper bound u_{gpw} is infinity and the slopes c_{ij}'s are monotonically decreasing.

3.6.6 Total Active Weights Constraint

The total active weights constraint is another special case of the general, separable piecewise linear constraints. Mathematically, it is defined as:

$$l_{taw} \le \left|\mathbf{h} - \mathbf{h}_B\right| \le u_{taw} \tag{23}$$

where l_{taw} and u_{taw} are the lower and upper bound of the constraint, respectively. Intuitively, total active weights represent the sum of the absolute active weights with respect to the benchmark. A binding lower bound may render this constraint non-convex.

3.6.7 Issuer Constraints

Issuer constraints refer to the bounds on net or absolute total holdings of an issuer. Let h_{K_i} be the portfolio weight of the i^{th} asset of issuer *K*, and n_K be the number of distinctive assets belonging to issuer *K*. Then, a "net" issuer constraint on issuer *K* is defined as

$$l_K \le \sum_{i=1}^{n_K} h_{K_i} \le u_K \tag{24}$$

while an "absolute" issuer constraint on issuer K is given by:

$$l_K \le \sum_{i=1}^{n_K} \left|h_{K_i}\right| \le u_K \tag{25}$$

where l_K and u_K are the lower and upper bound of the constraint, respectively.

Mathematically, (24) is a linear constraint and (25) is a piecewise linear constraint. For long-only optimization problems, all issuer constraints are convex. However, if shorting is allowed, the "absolute" issuer constraints may not be convex when the lower bound is positive.

Barra Optimizer's definition of an "issuer" is much broader than the traditional meaning of a security issuer. It extends the concept of an issuer to any group of assets based on certain rules. In other words, issuer constraints can be used to handle certain group constraints.

Users can set both global bounds (that are applied to all issuers) and individual bounds (that are applied to specific issuers) on a particular issuer simultaneneously. However, only the more restrictive one prevails.

Issuer constraints can be combined with almost all optimization problems except the following:

- Multiple-Period Optimization
- Multiple-Account Optimization
- Tax-Aware Optimization
- Maximizing Sharpe Ratio or Information Ratio

Notes:

- For each issuer, users are only allowed to set at most one net total holding constraint AND at most one absolute total holding constrraint. If users set multiple issuer constraints of the same type on an issuer, only the last one will prevail.
- Both upper and lower bounds on net total holding are supported. However, only upper bound on absolute total holding is supported.
- Soft bounds or penalties on issuer constraints are not supported.

3.7 Risk Control in Optimization—Aversions, Constraints, and Grouping

Risk control in Barra Optimizer can be achieved in a number of ways. The simplest and recommended method is to use a risk aversion within the objective. This requires two aversion parameters λ_D and λ_F, which can be set to the same value. Risk control using aversions is a Quadratic Programming Problem and produces a more stable problem than risk control using constraints.

Barra Optimizer also supports risk control using constraints. In addition, it supports using multiple benchmarks, risk models, or factor/asset groupings to achieve fine control of risk in the portfolio.

3.7.1 Factor Risk Models

Barra Optimizer accepts risk models in either standard factor model form, or in asset-covariance form. See the next section for asset-covariance matrix details.

Factor model form can be written as:

$$\sum = \mathbf{D} + \mathbf{XFX}^{\mathrm{T}} \tag{26}$$

Barra Optimizer takes advantage of this special structure in its internal algorithms.

The asset-specific covariance matrix can be further represented by:

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_r & \mathbf{E}^{\mathrm{T}} \\ \mathbf{E} & \mathbf{V} \end{bmatrix} \tag{27}$$

where,

$\mathbf{D}_r$ = $n_r \times n_r$ (block) diagonal covariance matrix representing specific risk for non-composite assets, in which n_r is the number of non-composite assets in the investment universe[1].

$\mathbf{E}$ = $n_c \times n_r$ matrix representing the covariance of composites with non-composite assets, in which n_c is the number of composites in the investment universe and $n_r + n_c = n$.

$\mathbf{V}$ = $n_c \times n_c$ matrix representing the covariance of composites with composites.

The matrices $\mathbf{E}$ and $\mathbf{V}$ in (27) are usually computed using the following formulae:

$$\mathbf{E} = \mathbf{WD}_m\mathbf{M}^{\mathrm{T}} \tag{28}$$

$$\mathbf{V} = \mathbf{WD}_m\mathbf{W}^{\mathrm{T}} \tag{29}$$

where

$\mathbf{D}_m$ = $n_m \times n_m$ (block) diagonal covariance matrix representing specific risk for all non-composite assets in the market, in which n_m is the number of non-composite assets in the market[2].

$\mathbf{W}$ = $n_c \times n_m$ composition (or weight) matrix for composites in which $\mathbf{W}_{ij}$ is the fraction of composite i made up by regular asset j.

[1] "Investment universe" refers to all assets that can be held in the portfolio.

[2] "Market" refers to the collection of all assets in existence.

$\mathbf{M}$ = $n_r \times n_m$ matrix which selects assets from the market to the investment universe. $\mathbf{M}_{ij} = 1$ indicates that asset j in the market is selected as asset i in the investment universe, and $\mathbf{M}_{ij} = 0$ implies otherwise.

The non-composite assets' specific covariance matrix $\mathbf{D}_r$ is usually diagonal. It becomes block diagonal only when there are "linked" assets, which are groups of non-composite assets with common exposure to a source of specific risk. Typical linked assets include different classes of stocks of the same company (for example, common stocks, or preferred stocks) or different bonds issued by the same company. The common, underlying source of specific risk is the root—in this case, the underlying company.

Without composite and linked assets, the specific covariance matrix $\mathbf{D}$ reduces to a simple diagonal matrix $\mathbf{D}_r$.

3.7.2 Asset-Covariance Risk Models

In addition to factor risk models, Barra Optimizer also supports risk models using asset-by-asset matrices. The recommended approach is to use the specific covariance functionality to set $\mathbf{D}$ to a general covariance matrix, which is an n x n positive semi-definite matrix containing total covariance among the assets, and use no factors.

3.7.3 Benchmarks—Multiple and With Non-Investment Universe Assets

The presence of a benchmark affects only the risk component of the objective function. The other components in the objective function—return, transaction cost, and other terms—all remain the same with or without a benchmark.

3.7.3.1 Benchmark With Non-Investment Universe Assets

Benchmark assets that are not in the investment universe are still included in the risk calculation. Internally, Barra Optimizer is often able to reduce the number of decision variables by efficient treatment of the non-investment universe assets.

More specifically, it is straightforward to show that:

$$(\bar{\mathbf{h}} - \bar{\mathbf{h}}_B)^{\mathrm{T}} (\lambda_D \bar{\mathbf{D}} + \lambda_F \bar{\mathbf{X}} \mathbf{F} \bar{\mathbf{X}}^{\mathrm{T}})(\bar{\mathbf{h}} - \bar{\mathbf{h}}_B) \tag{30}$$

$$= \underbrace{\mathbf{h}^{\mathrm{T}} (\lambda_D \mathbf{D} + \lambda_F \mathbf{X} \mathbf{F} \mathbf{X}^{\mathrm{T}}) \mathbf{h}}_{quadratic} - \underbrace{2[\lambda_D (\mathbf{h}_B{}^{\mathrm{T}} \mathbf{D} + \tilde{\mathbf{h}}_B{}^{\mathrm{T}} \mathbf{C}) + \lambda_F \bar{\mathbf{h}}_B{}^{\mathrm{T}} \bar{\mathbf{X}} \mathbf{F}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}}] \mathbf{h}}_{linear} + \underbrace{\bar{\mathbf{h}}_B{}^{\mathrm{T}} (\lambda_D \bar{\mathbf{D}} + \lambda_F \bar{\mathbf{X}} \mathbf{F} \bar{\mathbf{X}}^{\mathrm{T}}) \bar{\mathbf{h}}_B}_{constant} \tag{31}$$

where,

$\tilde{\mathbf{h}}_B$ = the $\tilde{n} \times 1$ vector of benchmark weights containing only those out-of-universe assets, where $\tilde{n}$ is the number of assets inside the benchmark yet outside the universe.

$\bar{\mathbf{h}}_B = \begin{bmatrix} \mathbf{h}_B \\ \tilde{\mathbf{h}}_B \end{bmatrix}$ = the $\bar{n} \times 1$ augmented vector of all benchmark weights, where $\bar{n} = n + \tilde{n}$.

$\bar{\mathbf{h}} = \begin{bmatrix} \mathbf{h} \\ \mathbf{0} \end{bmatrix}$ = the $\bar{n} \times 1$ augmented vector of asset holdings in which an element is zero if the corresponding asset is outside the investment universe.

$\bar{\mathbf{D}} = \begin{bmatrix} \mathbf{D} & \mathbf{C} \\ \mathbf{C}^{\mathrm{T}} & \tilde{\mathbf{D}} \end{bmatrix}$ = the $\bar{n} \times \bar{n}$ augmented specific covariance matrix. The off-diagonal term $\mathbf{C}$ is often zero, but need not be so (e.g., when linked assets or composites are present).

$\bar{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \tilde{\mathbf{X}} \end{bmatrix}$ = the $\bar{n} \times k$ augmented factor exposure matrix.

From(31), we see that out-of-universe assets only appear in the linear and constant parts. In other words, they do not affect the quadratic component of the risk terms. Therefore, only the linear component needs to be adjusted for these assets. The adjustment to constant component is optional but not essential since it would not affect either the optimization process or optimal solution.

3.7.3.2 Multiple Benchmarks

Multiple benchmarks are allowed in risk control terms. See Section 3.7.6 for more details.

3.7.4 Risk Aversion

In contrast to the return terms, the risk terms are quadratic in $\mathbf{h}$:

$$100(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_B)$$
$$=100\left(\underbrace{\mathbf{h}^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})\mathbf{h}}_{quadratic}-\underbrace{2\mathbf{h}_B{}^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})\mathbf{h}}_{linear}+\underbrace{\mathbf{h}_B{}^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})\mathbf{h}_B}_{constant}\right) \quad (32)$$

where, the benchmark vector $\mathbf{h}_B$ may contain assets outside the investment universe. The impact of risk on objective in Barra Optimizer is influenced by the two risk aversion parameters λ_D and λ_F, and amplified by the multiplier 100. For more information about the effects of risk aversion on portfolio optimization, see Liu and Xu (2010).

3.7.5 Total Risk vs. Active Risk

When the benchmark vector $\mathbf{h}_B$ is empty, the risk terms reduce to:

$$100\mathbf{h}^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})\mathbf{h} \quad (33)$$

This represents the variance of the portfolio's total return or its return with respect to cash. Otherwise, the risk terms represent the variance of the portfolio's active return with respect to the benchmark. The same distinction between Total Risk vs. Active Risk in the objective is true for risk constraints.

3.7.6 Multiple Risk Terms

Additional control can be achieved by deriving new risk terms from the existing risk models and by weighting specific risks, assets, or factors to form new covariance terms.

For accuracy, the additional risk terms not expressed explicitly in (1) already are absorbed into the term $g(\mathbf{h})$. In general, these terms take on one of the following forms:

Weighted Specific Risk: $$100\,\lambda_{D_i}\,(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}\,\mathbf{W}_{D_i}\,\mathbf{D}_i\,\mathbf{W}_{D_i}^{T}\,(\mathbf{h}-\mathbf{h}_B) \quad (34)$$

Factor-Weighted Factor Risk: $$100\,\lambda_{F_i}\,(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}\,\mathbf{X}_i\,\mathbf{W}_{F_i}\,\mathbf{F}_i\,\mathbf{W}_{F_i}^{T}\,\mathbf{X}_i^{\,T}\,(\mathbf{h}-\mathbf{h}_B) \quad (35)$$

Asset-Weighted Factor Risk: $$100\,\lambda_{X_i}\,(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}\,\mathbf{W}_{X_i}\,\mathbf{X}_i\,\mathbf{F}_i\,\mathbf{X}_i^{\,T}\,\mathbf{W}_{X_i}^{T}\,(\mathbf{h}-\mathbf{h}_B) \quad (36)$$

where,

- $\mathbf{D}_i$ is the specific covariance matrix or the full asset-by-asset covariance matrix.
- $\mathbf{F}_i$ is the factor covariance matrix.
- $\mathbf{X}_i$ is the factor exposure matrix.
- $\mathbf{D}_i$, $\mathbf{F}_i$, and $\mathbf{X}_i$ can belong to either of the primary risk model or the secondary risk model.
- $\mathbf{W}_i$ is a diagonal weight matrix with appropriate dimensions.
- λ_i is risk aversion for the covariance term that further adjusts the impact of the individual risk terms in the objective function.
- $\mathbf{h}_B$ for each term can be unique, same as other terms, or empty

Multiple covariance terms of each type can be added, as long as they reference either the primary or the secondary model.

3.7.7 Secondary Risk Model

Barra Optimizer supports a secondary risk model. The secondary risk model may appear in the objective function as an additional risk term as shown below.

$$-100(\mathbf{h}-\mathbf{h}_{B_2})^{\mathrm{T}}(\lambda_{D_2}\mathbf{D}_2+\lambda_{F_2}\mathbf{X}_2\mathbf{F}_2\mathbf{X}_2^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_{B_2}) \quad (37)$$

where,

λ_{D_2}	Specific risk aversion parameter
λ_{F_2}	Common factor risk aversion parameter
$\mathbf{h}_{B_2}$	Secondary benchmark holding weights (may or may not be the same as the primary one)
$\mathbf{D}_2$	Specific covariance matrix of the secondary risk model
$\mathbf{X}_2$	Factor exposures matrix of the secondary risk model
$\mathbf{F}_2$	Factor covariance matrix of the secondary risk model

Notes:

- Secondary risk model may also be used in risk constraints. For more information, see the Subgroup Risk Constraints section.
- General linear constraints must be used instead of factor constraints for the secondary risk model.
- Benchmark may contain assets outside the investment universe.

3.7.8 Risk Constrained Optimization

In standard portfolio optimization, risk relative to return may be adjusted implicitly by altering the risk aversion values in the objective function.

Barra Optimizer allows multiple risk terms and multiple benchmarks in the objective function. The impact of each additional risk term can be adjusted using its risk aversion. For more information, see the Multiple Risk Terms section.

Alternatively, users may impose explicit constraints on risk. These constraints could be either convex or non-convex, but would all be nonlinear and may make the optimization problem considerably more difficult to solve. Non-convexity may also lead to suboptimal solutions.

The risk terms in a risk constraint can be either identical to or different from the ones in the objective. When different, the terms may represent risk from a different risk model or different aspects of risk from the same risk model—for example, with respect to different benchmarks or from different subgroups.

Prior to version 8.7, Barra Optimizer employed the definition of subgroup risk that is simple and convex. However, under this definition, the sum of risks from all subgroups may not equal to the overall risk. This subgroup risk definition is now denoted as “non-additive definition” and is still supported. Starting with version 8.7, Barra Optimizer also supports the additive definition of the subgroup risk, in addition to the non-additive one. Note that the choice of the definition matters only when the risk constraint is defined on the subgroup of assets (or factors). Portfolio-level risks are the same under both definitions.

Note that Barra Optimizer supports the combination of risk constraints with threshold, cardinality, and hedge constraints.

Multiple Risk Sources Examples

Barra Optimizer can be used to constrain one risk term while minimizing another, or constrain multiple sources of risk. In general, upper bounds on portfolio level risk are convex and can be set on the overall risk, factor risk or specific risk. Up to two risk models can be used in constraints.

Overall Risk

Mathematically, minimizing the active risk relative to an index while constraining the portfolio’s derivation from an in-house model portfolio can be represented as:

$$\text{Maximize: } \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \lambda\ (\mathbf{h}-\mathbf{h}_{B_1})^{\mathrm{T}}\mathbf{V}_1(\mathbf{h}-\mathbf{h}_{B_1}) - \text{Other Terms} \tag{38}$$

$$\text{Subject to: } \sqrt{(\mathbf{h}-\mathbf{h}_{B_2})^{\mathrm{T}}\sum(\mathbf{h}-\mathbf{h}_{B_2})} \le u_\sigma \tag{39}$$

$$\text{Other Constraints}$$

where, $\mathbf{h}_{B_1}$ and $\mathbf{h}_{B_2}$ are the weight vectors for two benchmarks, in this case the index and in-house reference portfolio, but could alternatively be either identical or different, $\mathbf{V}_1$ and $\sum$ are two asset-by-asset covariance matrices, which could also be either identical or different, and could be from the same or different risk models, λ is a risk aversion parameter, and u_σ is the given upper bound on risk.

Note that one or both of the benchmark vectors could be empty, in which case the corresponding risk term would represent the portfolio's total risk instead of the tracking error.

Factor Risk

An example of constraining two risks, one of which is the portfolio's factor risk, is given below:

$$\text{Maximize: } \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Other Terms} \tag{40}$$

$$\text{Subject to: } \sqrt{(\mathbf{h}-\mathbf{h}_{B_1})^{\mathrm{T}}\mathbf{V}_1(\mathbf{h}-\mathbf{h}_{B_1})} \le u_\sigma \tag{41}$$

$$\sqrt{(\mathbf{h}-\mathbf{h}_{B_2})^{\mathrm{T}}\mathbf{X}_2\mathbf{F}_2\mathbf{X}_2^{T}(\mathbf{h}-\mathbf{h}_{B_2})} \le u_{F_2} \tag{42}$$

$$\text{Other Constraints}$$

where, $\mathbf{F}_2$ is a factor covariance matrix and $\mathbf{X}_2$ is an exposure matrix. Constraint (42) requires that the factor risk of the portfolio cannot be greater than u_{F_2}. The benchmarks could be empty or identical, and the $\mathbf{F}_2$ and $\mathbf{X}_2$ matrices could be either related to or independent of the $\mathbf{V}_1$ matrix.

Specific Risk

The following is an example where we minimize a factor risk while constraining an overall risk and a specific risk:

$$\text{Maximize: } \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \lambda(\mathbf{h}-\mathbf{h}_{B_1})^{\mathrm{T}}\mathbf{X}_1\mathbf{F}_1\mathbf{X}_1^{\mathrm{T}}(\mathbf{h}-\mathbf{h}_{B_1}) - \text{Other Terms} \tag{43}$$

$$\text{Subject to: } \sqrt{(\mathbf{h}-\mathbf{h}_{B_2})^{\mathrm{T}}\mathbf{V}_2(\mathbf{h}-\mathbf{h}_{B_2})} \le u_\sigma \tag{44}$$

$$\sqrt{(\mathbf{h}-\mathbf{h}_{B_3})^{\mathrm{T}}\mathbf{D}(\mathbf{h}-\mathbf{h}_{B_3})} \le u_{D_3} \tag{45}$$

$$\text{Other Constraints}$$

where, $\mathbf{D}$ is a specific risk covariance matrix, which could be either related to or independent from the matrices $\mathbf{F}_1$ and $\mathbf{V}_2$, as long as they all come from at most two risk models. Again, one or more of the three benchmarks here could be empty, and they could all be identical.

Subgroup Risk Constraints

Sometimes, investors are more concerned with the risk from a subgroup than with the overall risk of the entire portfolio. For example, users may wish to limit the risk attributed to the style factor to be below 10 basis points, or

alternatively, set an upper bound on the risk coming from the technology sector at 0.05. Risk here refers to either the total risk or the tracking error, depending on whether there is a benchmark present in the constraint.

As described above, Barra Optimizer now supports two definitions of subgroup risk: a non-additive definition and an additive definition. Risk from subgroups under the non-additive definition may not sum to overall risk. This definition ignores covariances outside of the subgroup, which may or may not be desirable to the user. The advantage of the non-additive definition is convexity. More details are given below.

On the other hand, subgroup risks under the additive definition sum to overall risk. Subgroup risks computed under this definition are the same as computed using the x-sigma-rho paradigm. The additive definition, however, makes the problem non-convex and hence more difficult to solve, among other consequences.

Subgroup Risk – Non-Additive Definition

Let I be a subset of asset indexes and J be a subset of factor indexes. The risk from sub-group { I , J } can be defined as:

$$\sigma(\mathbf{h}, I, J) = \sqrt{(\mathbf{h}_I - \mathbf{h}_B)^{\mathrm{T}}(\mathbf{D} + \mathbf{X}\mathbf{F}_J\mathbf{X}^{\mathrm{T}})(\mathbf{h}_I - \mathbf{h}_B)} \tag{46}$$

where,

- $\mathbf{h}_I$ is a $n \times 1$ holding vector in which all positions are zeroes if they are not in I
- $\mathbf{F}_J$ is the modified factor covariance matrix $\mathbf{F}$ in which rows and columns are all zeroes if their indexes are not in J
- $\mathbf{h}_B$ contains all benchmark assets

Note: When defining risk from a subgroup under the non-additive definition, assets that do not belong to the subgroup are removed from the portfolio, but not from the benchmark. If the users want to subgroup the benchmark weights as well, they should form a custom benchmark containing weights for just the subgroup assets. Additionally, factors that do not belong to the subgroup are removed from both the portfolio exposures and the benchmark exposures.

The specific risk from sub-group I can be defined as:

$$\sigma_s(\mathbf{h}, I) = \sqrt{(\mathbf{h}_I - \mathbf{h}_B)^{\mathrm{T}}\mathbf{D}(\mathbf{h}_I - \mathbf{h}_B)} \tag{47}$$

The factor risk from sub-group { I , J } can be defined as:

$$\sigma_F(\mathbf{h}, I, J) = \sqrt{(\mathbf{h}_I - \mathbf{h}_B)^{\mathrm{T}}\mathbf{X}\mathbf{F}_J\mathbf{X}^{\mathrm{T}}(\mathbf{h}_I - \mathbf{h}_B)} \tag{48}$$

To construct a risk-constrained problem, one or more risk constraints of the following format are to be added to the standard portfolio optimization problem:

$$risk(\mathbf{h}, I, J) \le u \tag{49}$$

where,

- u is upper bound of the risk constraint,

- $risk(\mathbf{h}, I, J)$ is one of $\sigma(\mathbf{h}, I, J)$, $\sigma_S(\mathbf{h}, I)$ or $\sigma_F(\mathbf{h}, I, J)$

The non-additive definition of the subgroup risk in Barra Optimizer is simply based on elimination of the entries not belonging to the subgroup from the covariance matrices. For example, let $J = \{2, 5, 8\}$ be a subset of asset or factor indexes.

Then, the covariance matrix for subgroup J would be:

$$\mathbf{V}_J = \begin{bmatrix} \sigma_{2,2}^2 & \sigma_{2,5}^2 & \sigma_{2,8}^2 \\ \sigma_{5,2}^2 & \sigma_{5,5}^2 & \sigma_{5,8}^2 \\ \sigma_{8,2}^2 & \sigma_{8,5}^2 & \sigma_{8,8}^2 \end{bmatrix} \tag{50}$$

where, $\sigma_{i,j}^2$ is an element in the i^{th} row and j^{th} column of the original covariance matrix $\mathbf{V}$.

A benefit of this simple definition is that an upper bound on subgroup risk defined this way would be a convex constraint. A drawback of this definition is that risk is non-additive—the sum of risks from all subgroups does not equal to the overall risk.

Fractional Risk Contribution from a Subgroup – Non-Additive Definition

In addition to constraining the risk from a subgroup, Barra Optimizer permits users to set a constraint on the fractional risk contribution from a subgroup, which is the ratio of the subgroup risk to the overall portfolio risk. For instance, users may request that the fractional risk attributed to asset selection be no more than 30%.

Barra Optimizer allows constraining fractional risk contribution in the following form:

$$\frac{risk(\mathbf{h}, I, J)}{\sigma(\mathbf{h})} \le u \tag{51}$$

where,

$$\sigma(\mathbf{h}) = \sqrt{(\mathbf{h} - \mathbf{h}_B)^{\mathrm{T}}(\mathbf{D} + \mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h} - \mathbf{h}_B)} \tag{52}$$

and $risk(\mathbf{h}, I, J)$ is one of $\sigma(\mathbf{h}, I, J)$, $\sigma_S(\mathbf{h}, I)$ or $\sigma_F(\mathbf{h}, I, J)$ as defined in (46)-(48).

The presence of the fractional risk constraints renders the optimization problem both non-convex and nonlinear. It increases the complexity of the problem and the likelihood that only a sub-optimal solution, as opposed to the true global optimum, may be generated. Caution should be exercised when imposing such a constraint.

Subgroup Risk – Additive Definition

Let, as mentioned above, I be a subset of asset indexes, J be a subset of factor indexes, and subscripting a vector by I (or J) denote replacing its components with zeroes if their positions are not in I (or J).

Let $\sigma(\mathbf{h}) = \sqrt{(\mathbf{h} - \mathbf{h}_B)^{\mathrm{T}}(\mathbf{D} + \mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h} - \mathbf{h}_B)}$ as in (52) above, $\sigma_s(\mathbf{h}) = \sqrt{(\mathbf{h} - \mathbf{h}_B)^{\mathrm{T}}\mathbf{D}(\mathbf{h} - \mathbf{h}_B)}$, and $\sigma_F(\mathbf{h}) = \sqrt{(\mathbf{h} - \mathbf{h}_B)^{\mathrm{T}}\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}}(\mathbf{h} - \mathbf{h}_B)}$

Then, the risk (or tracking error, when $\mathbf{h}_B$ is not empty) from asset sub-group I under the additive definition is

$$\sigma^A(\mathbf{h}, I) = \frac{(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}(\mathbf{D}+\mathbf{XFX}^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_B)_I}{\sigma(\mathbf{h})} \tag{53}$$

The specific risk (or tracking error) from asset sub-group I under the additive definition is

$$\sigma_s^A(\mathbf{h}, I) = \frac{(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}\mathbf{D}(\mathbf{h}-\mathbf{h}_B)_I}{\sigma_s(\mathbf{h})} \tag{54}$$

The factor risk (or tracking error) from factor sub-group J under the additive definition is

$$\sigma_F^A(\mathbf{h}, J) = \frac{(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}\mathbf{XF}(\mathbf{X}^{\mathrm{T}}(\mathbf{h}-\mathbf{h}_B))_J}{\sigma_F(\mathbf{h})} \tag{55}$$

To construct a risk-constrained problem, one or more risk constraints of the following format are to be added to the standard portfolio optimization problem:

$$risk^A(\mathbf{h}, I, J) \le u \tag{56}$$

where,

- u is upper bound of the risk constraint,
- $risk^A(\mathbf{h}, I, J)$ is one of $\sigma^A(\mathbf{h}, I)$, $\sigma_s^A(\mathbf{h}, I)$ or $\sigma_F^A(\mathbf{h}, J)$ as defined above,

Note that the presence of these constraints renders the optimization problem both non-convex and nonlinear, with the consequences described above.

Fractional Risk Contribution from a Subgroup – Additive Definition

In addition to constraining the risk from a subgroup according to the additive definition, Barra Optimizer permits users to set a constraint on the fractional risk contribution from a subgroup, which is the ratio of the subgroup risk to the corresponding portfolio risk (overall, factor, or specific) . For instance, users may request that no more than 10% of overall tracking error be attributed to assets from a certain group, or no more than 60% of factor risk be attributed to style factors.

Barra Optimizer supports fractional risk contribution constraints under the additive definitions in the following forms:

$$\frac{\sigma^A(\mathbf{h}, I)}{\sigma(\mathbf{h})} \le u \tag{57}$$

$$\frac{\sigma_s^A(\mathbf{h}, I)}{\sigma_S(\mathbf{h})} \le u \tag{58}$$

$$\frac{\sigma_F^A(\mathbf{h}, J)}{\sigma_F(\mathbf{h})} \leq u \tag{59}$$

where $\sigma^A(\mathbf{h}, I)$, $\sigma_s^A(\mathbf{h}, I)$, $\sigma_F^A(\mathbf{h}, J)$, $\sigma(\mathbf{h})$, $\sigma_S(\mathbf{h})$ and $\sigma_F(\mathbf{h})$ are as defined previously in this section. Again, note that the presence of these constraints render the optimization problem both non-convex and nonlinear, with the consequences described above.

Notes:

- A secondary risk model ($\mathbf{D}$ $\mathbf{F}$ $\mathbf{X}$) can be used to constrain risk that is different from the model used in the objective function. For more information, see the Secondary Risk Model section.
- Constraints may be defined both with and without a benchmark. The benchmark in any risk constraint is specific to that constraint and as such may be different from the benchmark in the objective function.

***Current Limitations**:*

- In general, from version 1.3 onwards, Barra Optimizer no longer supports lower bounds on risk. However, starting with version 8.2, for problems with only one convex risk constraint defined by the primary risk model, Barra Optimizer allows a soft lower bound on the total risk or tracking error.
- Barra Optimizer may fail or return a sub-optimal solution when solving a case containing non-convex risk constraints (e.g. constraining on percentage risk contribution).

3.7.9 Risk Target Optimization

Risk Target Optimization allows users to set a specific target on the portfolio risk level. Mathematically, the problem can be represented as:

$$\text{Maximize:} \quad \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Transaction Cost} \tag{60}$$

$$\text{Subject to:} \quad \sigma \leq \sigma_{Target} \tag{61}$$

$$\text{Other Constraints}$$

where, σ is the standard deviation of the portfolio return or active return. Risk Target Optimization is a special case of Varying Risk in the Risk-Return Efficient Frontiers section.

On exit, Barra Optimizer reports an implied risk aversion, λ_{imp} for each risk target problem. This value is the risk aversion that makes the optimal solution to the following standard optimization problem identical to the one of the above risk target problem, when using convex constraints and objectives.

$$\text{Maximize:} \quad \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \lambda_{imp} \cdot \sigma^2 - \text{Transaction Cost} \tag{62}$$

$$\text{Subject to:} \quad \text{Other Constraints}$$

The risk target optimization problem differs from the general risk-constrained problem in several aspects, including:

- It has no risk term in the objective. If the risk target is on the efficient frontier, then constraint (61) must be binding.

- Users do not supply the risk aversion values. Instead, Barra Optimizer will derive and output an implied risk aversion value.
- When the risk target is not on the efficient frontier (i.e., it is either too high or too low), Barra Optimizer provides better information for the risk target problem than for the risk constrained problem.

For information about the risk-constrained problem, see the Risk Constrained Optimization section.

For more information about risk target optimization and its relationship with mean-variance optimization, see Kopman and Liu (2009).

The risk term in risk target optimization has the following flexible form:

$$\sigma(\mathbf{h}) = \sqrt{(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}(\theta_D\mathbf{D}+\theta_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_B)} \tag{63}$$

where θ_D and θ_F are two scalars for controlling the relative impact of specific and factor risk, and the benchmark $\mathbf{h}_B$ could be empty.

3.8 Transaction Cost, Holding Cost, and Turnover Control

Transaction cost (in decimals) may appear not only in the objective function, but also in a constraint. Barra Optimizer permits an upper limit on the total transaction cost in the form of constraint.(10) For example, users may require that the portfolio's total transaction cost be no more than 3% of the portfolio value. Separate transaction cost limits on buying and selling equities may also be imposed by using a number of general piecewise linear constraints. However, transaction cost limit constraints may contain only piecewise linear functions. Constraints on nonlinear or fixed transaction costs are currently not supported.

Barra Optimizer accommodates a wide variety of transaction cost functions in the objective. As explained more in detail in the subsequent subsections, transaction costs can be modeled either by traditional piecewise linear functions ($Tc_p(\mathbf{h},\mathbf{h}_0)$) or by nonlinear power functions ($Tc_n(\mathbf{h},\mathbf{h}_0)$), or as a fixed cost per trade ($Tc_f(\mathbf{h},\mathbf{h}_0)$). The transaction cost term in (1) can be written as the sum of three optional terms:

$$Tc(\mathbf{h},\mathbf{h}_0) = Tc_p(\mathbf{h},\mathbf{h}_0) + Tc_n(\mathbf{h},\mathbf{h}_0) + Tc_f(\mathbf{h},\mathbf{h}_0) \tag{64}$$

3.8.1 Piecewise Linear Transaction Costs

A simple piecewise linear transaction cost function has the following form:

$$Tc_p(\mathbf{h},\mathbf{h}_0) = \sum_{i=1}^{n}\left[c_{ibuy}\cdot\max(\mathbf{h}_i-\mathbf{h}_{0i},0)+c_{isell}\cdot\max(\mathbf{h}_{0i}-\mathbf{h}_i,0)\right] \tag{65}$$

where, c_{ibuy} and c_{isell} are unit transaction costs for buying and selling, respectively. To ensure convexity, it must be true that $c_{ibuy} \geq 0$ and $c_{isell} \geq 0$. The Barra Market Impact Model provides more sophisticated transaction cost handling, and may generate complicated piecewise linear functions with multiple break points at both the buy and sell sides. Figure 1 illustrates these functions:

Figure 1: Piecewise Linear Transaction Cost

3.8.2 Non-Linear Transaction Costs

Barra Optimizer allows non-linear transaction costs in the following form:

$$Tc_n(\mathbf{h},\mathbf{h}_0)=\sum_{i=1}^{n} r_i \left|\mathbf{h}_i-\mathbf{h}_{0i}\right|^{p_i} \tag{66}$$

where $r_i > 0$ is the non-linear transaction cost multiplier and $p_i > 1$ is the exponent of the asset-level power function. Figure 2 provides an illustration of such a function.

Figure 2: Non-linear Transaction Cost

To specify a proper value for the coefficient r_i, you may use the following formula:

$$r_i=\frac{c_i}{q_i^{\,p_i}} \tag{67}$$

This means that the transaction cost would be c_i (in decimal), if the trading amount $\left|\mathbf{h}_i-\mathbf{h}_{0i}\right|$ is q_i (in decimal).

Notes:

- Cash and currencies are not included in non-linear transaction costs.
- Barra Optimizer currently does not support upper bounds on non-linear transaction costs.

3.8.3 Fixed Transaction Costs

Barra Optimizer also supports general fixed transaction costs in the following form:

$$Tc_f(\mathbf{h},\mathbf{h}_0) = \sum_{i=1}^{n} fc_i \tag{68}$$

where, fc_i is the fixed transaction cost for asset i and is defined as:

$$fc_i = fc_i = \begin{cases} c_{fb_i}, & h_i > h_{0i} \\ 0, & h_i = h_{0i} \\ c_{fs_i}, & h_i < h_{0i} \end{cases} \tag{69}$$

In (69), c_{fb_i} and c_{fs_i} are two constants representing the fixed cost for asset i at the buy side and sell side, respectively. Figure 3 shows an example of the impact of such a function.

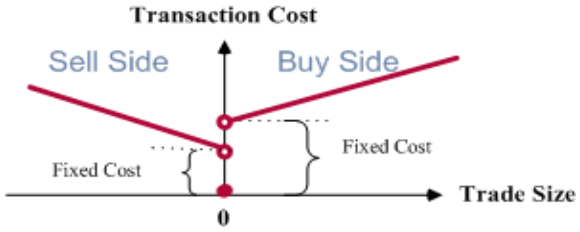


Figure 3: Piecewise Linear Plus Fixed Transaction Cost

A crossover trade makes an asset change from long position to short position, or vice versa. A crossover trade may be counted as one trade or two trades. For example, if you count a crossover trade as two trades, an asset with an initial position of 2% is sold with a final position of -1%, and the fixed transaction costs would double the amount of the sell_cost.

Note: Barra Optimizer currently does not support upper bounds on fixed transaction costs.

3.8.4 Fixed Holding Costs

Similar to the definition of fixed transaction costs, fixed holding costs can be modeled as follows:

$$Hc_f(\mathbf{h},\mathbf{h}_0) = \sum_{i=1}^{n} fh_i \tag{70}$$

where, fh_i is the fixed holding cost for asset i and is defined as:

$$fh_i = \begin{cases} c_{u_i}, & h_i > h_{ri} \\ 0, & h_i = h_{ri} \\ c_{d_i}, & h_i < h_{ri} \end{cases} \tag{71}$$

In (71), c_{u_i} (or c_{s_i}) is a constant representing the fixed cost for holding asset i more (or less) than asset i's weight in a referent portfolio—h_{ri}. The reference portfolio could be the initial or a benchmark portfolio, or even empty. Figure 4 shows an example of the impact of such a function.

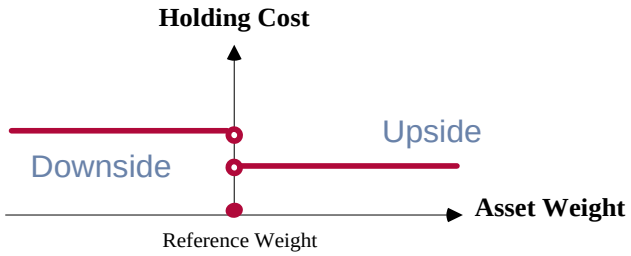


Figure 4: Fixed Holding Cost

The fixed holding costs $Hc_f(\mathbf{h},\mathbf{h}_0)$ enter into the objective (1) as part of the term $g(\mathbf{h})$.

Note: Barra Optimizer currently supports fixed holding costs only in the objective function, not in the constraints. Additionally, combination of fixed transaction costs and fixed holding costs is not allowed.

3.8.5 Turnover Functions and Constraints

Barra Optimizer allows users to set an upper bound on the amount of overall portfolio turnover[3] or turnover by group. When there is no cash flow (either cash infusion or cash withdrawal), the total amount of equity buys must equal the total amount of equity sells. Under such circumstances, the overall turnover in Barra Optimizer is defined as:

$$\frac{Equity\ Buys + Equity\ Sells + |Change\ in\ Cash\ Position|}{2 \cdot TurnoverBaseValue} \tag{72}$$

When cash flow does occur, the overall turnover is defined as:

$$\frac{Equity\ Buys + Equity\ Sells + |Change\ in\ Cash\ Position| - |Cash\ Flow|}{2 \cdot TurnoverBaseValue} \tag{73}$$

In terms of weights, a turnover function can be rewritten as shown below:

$$TO(\mathbf{h}) = \frac{1}{2}\left(\sum_{j \in J}\left|h_j - h_{0_j}\right| - |CFW|\right) / TBVW \tag{74}$$

where, $TBVW = TurnoverBaseValue\ /\ BV$ is the weight of the turnover base value

[3] In API, overall turnover is also called net turnover, in contrast to long-side turnover and short-side turnover.

3.8.5.1 Turnover Base Value

Barra Optimizer offers the following two possible definitions for TurnoverBaseValue:

1. Default definition:

   TurnoverBaseValue = InitialLongPositions + | InitialShortPositions| + |InitialCashPositions| + CashFlow (75)

   $TBVW = \sum_{j \in J} \left| h_{0j} \right| + CFW$ and the turnover function can be written as:

   $$TO(\mathbf{h}) = \frac{1}{2}\left(\sum_{j \in J} \left| h_j - h_{0j} \right| - \left| CFW \right| \right) / \left(\sum_{j \in J} \left| h_{0j} \right| + CFW \right) \quad (76)$$

   Optionally, futures can be included.

2. Use the same base value BV that is used for asset weight computation:

   $$\text{TurnoverBaseValue} = \text{BV} \quad (77)$$

   $TBVW = 1$ and the turnover function can be written as shown below:

   $$TO(\mathbf{h}) = \frac{1}{2}\left(\sum_{j \in J} \left| h_j - h_{0j} \right| - \left| CFW \right| \right) \quad (78)$$

   When there is no cash flow, the numerator of the turnover can be written as:

   $$\max(Equity\,buys, Equity\,sales) \quad (79)$$

For multiple-period optimization, only the second option is supported.

3.8.5.2 Upper Bound on Buy or Sell Turnover

In addition to the overall turnover, Barra Optimizer also supports buy-side and sell-side turnover limit constraints:

Buy-Side Turnover Constraint:

$$\frac{Equity\ Buys}{TurnoverBaseValue} \le MaxBuyTurnover; \quad (80)$$

Sell-Side Turnover Constraint:

$$\frac{Equity\ Sells}{TurnoverBaseValue} \le MaxSellTurnover. \quad (81)$$

In terms of weights, these constraints are:

$$\sum_{i \in I} Max(h_i - h_{0i}, 0.0) \le MaxBuySideTurnover \cdot TBVW$$
$$\sum_{i \in I} Max(h_{0i} - h_i, 0.0) \le MaxSellSideTurnover \cdot TBVW \quad (82)$$

where i is the index set of assets, excluding cash and futures. Optionally, futures can be included.

Furthermore, for long/short optimization, Barra Optimizer allows you to specify additional turnover limits by side. For details, see the By-Side Optimization section.

3.8.5.3 Upper Bound on Turnover by Group

In addition to an upper bound on turnover at the whole portfolio-level, Barra Open Optimizer also supports an upper bound on turnover at the group-level. Examples of such constraints are, "turnover from asset group K is at most 20%", or "total buys in asset group J is at most 10%". The definition for turnover by group is same as the definition for overall turnover as defined above except that the numerator will be defined on the given asset group only.

Note that users can set multiple turnover-by-group constraints simultaneously. Futures are excluded by default and can be included optionally.

3.8.5.4 Option to Exclude Cash

In the default definitions above, cash is included in the numerator.

Barra Open Optimizer now provides an option to exclude cash. With this option, both cash position change and cash flow will be excluded from the numerator of the turnover definition. In other words, turnover will be simply defined as:

$$\frac{Equity\ Buys + Equity\ Sells}{2 \cdot TurnoverBaseValue} \quad (83)$$

To further exclude cash from the denominator of the turnover definition, you may use a user-defined turnover base value that does not include any cash.

3.9 Constraint Flexibility—Penalties, Soft Bounds, and Hierarchy

3.9.1 Penalty Functions

Penalty functions are used to tilt portfolios toward user-specified targets on constraint or factor slacks, as well as targets on asset weights (either ordinary or active weights). Deviation away from a target causes a negative impact on the objective function. Some of the penalty terms are part of the $g(\mathbf{h})$ in (1).

One method to specify a penalty function is to use three user-specified parameters—Target, Upper, and Lower. Target is the desired value of the constraint slack in question. Upper (Lower) is a given value above (below) Target at which a 1% deduction of the objective value, before any multiplier adjustment, will be imposed. At values other than these three parameters, different penalty functions usually affect the objective value differently.

Barra Optimizer supports both quadratic penalties and pure linear penalties. For pure linear penalties, the target can be either a single point or a free range. Figure 5 illustrates the six types of penalty functions that Barra Optimizer supports. The following sections provide more details on these penalty functions.

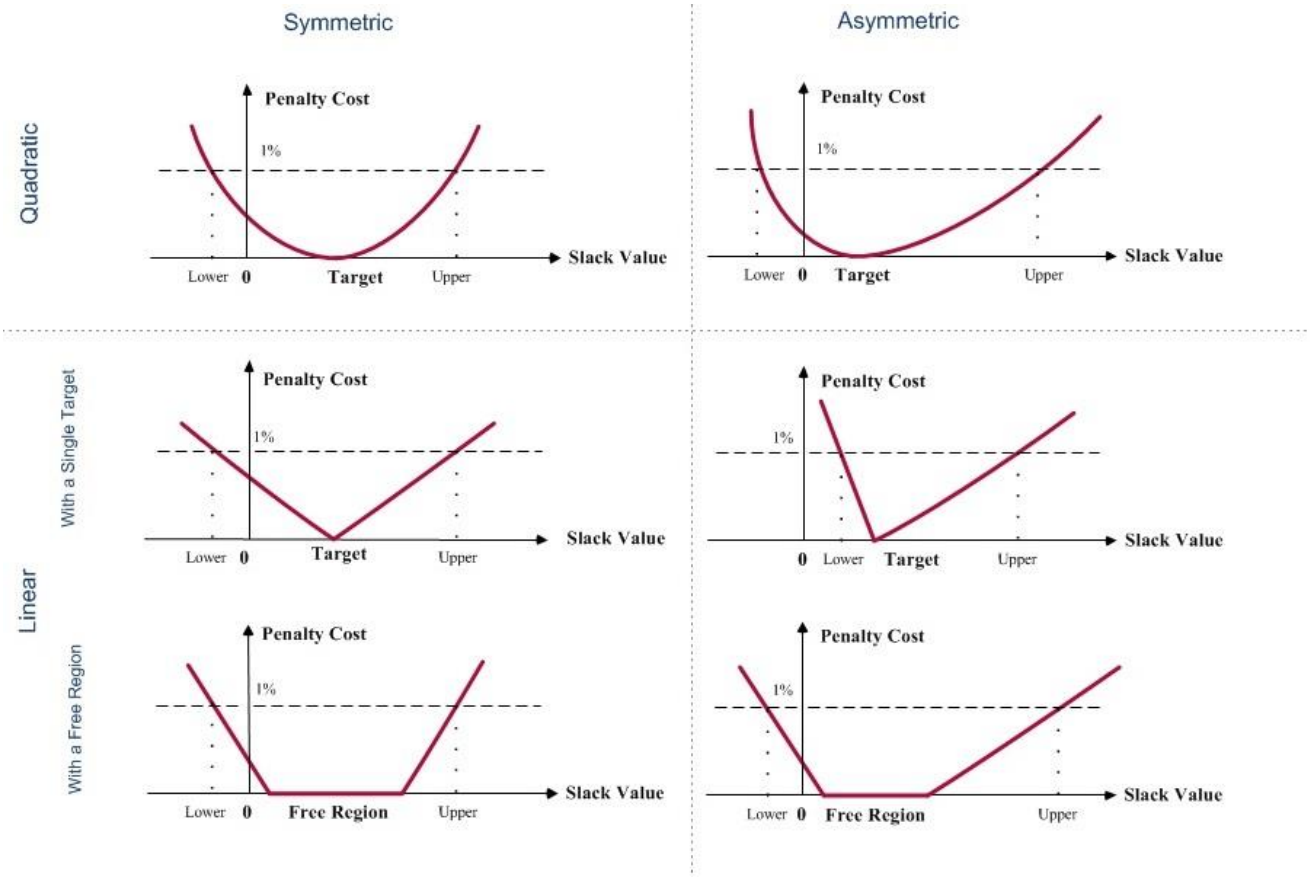


Figure 5: Different Types of Penalty Functions

3.9.1.1 Quadratic vs. Linear Penalties

Let v denote a generic constraint slack variable. Then, pure quadratic penalties are in the form:

$$Pen_s(v) = \rho_q \cdot (v - Target)^2 \tag{84}$$

whereas, pure linear penalties with a single target are:

$$Pen_u(v) = \rho_{up} \cdot \max(v - Target,\ 0) \tag{85}$$

$$Pen_d(v) = \rho_{down} \cdot \max(Target - v,\ 0) \tag{86}$$

To ensure the 1% deduction in objective at $v = Upper$ or $v = Lower$, it must be true that the slopes in the above penalties functions are given by:

$$\rho_q = 0.01/(Upper - Target)^2 \tag{87}$$

$$\rho_{up} = 0.01/(Upper - Target) \tag{88}$$

$$\rho_{down} = 0.01/(Target - Lower) \tag{89}$$

In the case of pure quadratic penalties, "$Upper - Target = Target - Lower$" must also hold.

$Upper \neq Target \neq Lower$, therefore, ρ_{up} and ρ_{down} cannot be simultaneously zero.

3.9.1.2 Symmetric vs. Asymmetric Penalties

A symmetric penalty is characterized by the property:

$$Upper - Target = Target - Lower \tag{90}$$

By definition, a pure quadratic penalty is a symmetric penalty. The converse is not true. A symmetric penalty is not necessarily quadratic because it can be pure linear or piecewise linear.

In contrast, an asymmetric penalty has the property:

$$Upper - Target \neq Target - Lower \tag{91}$$

It is usually synthesized by a pure (symmetric) quadratic penalty plus two pure linear penalties, and can be represented by:

$$Pen_a(v) = Pen_s(v) + Pen_u(v) + Pen_d(v) \tag{92}$$

Let $BigRange = \max(Upper - Target,\ Target - Lower)$. It is not difficult to show that to ensure the 1% deduction in objective at $v = Upper$ and $v = Lower$, the slopes implied in the three penalty terms in (92) must be the following:

$$\rho_q = 0.01/BigRange^2 \tag{93}$$

$$\rho_{up} = \begin{cases} 0, & if\ Upper - Target = BigRange \\ 0.01\left[1/(Upper - Target) - (Upper - Target)/BigRange^2\right], & otherwise \end{cases} \tag{94}$$

$$\rho_{down} = \begin{cases} 0, & if\ Target - Lower = BigRange \\ 0.01\left[1/(Target - Lower) - (Target - Lower)/BigRange^2\right], & otherwise \end{cases} \tag{95}$$

3.9.1.3 Penalty Details

When penalty target is a single point, a penalty function $Pen(v)$ has the following characteristics:

- lower < target < upper
- Pen(target) = 0
- Pen(lower) = Pen(upper) = 0.01
- The smaller target – lower (i.e. the closer of lower to target), larger the penalty value $Pen(v)$ will be when $v < Target$

- The smaller upper – target (i.e. the closer of upper to target), larger the penalty value $Pen(v)$ will be when $v > Target$

The user may apply penalties for the following constraint categories only:

- Asset bounds (i.e., bounds on ordinary or active asset weights)
- General linear constraints (including beta constraint and group linear constraints)
- Factor constraints
- Hedge (leverage) constraints
- Total leverage constraints

3.9.1.4 Pure Linear Penalties with a Single Target vs. with a Free Region

Pure linear penalties with a single target are given by (85) -(86). Similarly, pure linear penalties with a free region are in the form:

$$Pen_u(v) = \rho_{up} \cdot \max(v - TargetHigh,\ 0) \tag{96}$$

$$Pen_d(v) = \rho_{down} \cdot \max(TargetLow - v,\ 0) \tag{97}$$

Note that when $TargetHigh = TargetLow$, the penalty target is reduced from a free range to a single point.

To specify a pure linear penalty function with a free range, users are asked to provide the two end points of the free region— $TargetLow$ and $TargetHigh$, as well as the slopes at each side of the free region— ρ_{up} and ρ_{down} directly. To ensure the convexity of the penalty function, the slopes should satisfy the condition $-\rho_{down} \le 0 \le \rho_{up}$. It is permissible for both slopes to be zero in this case.

3.9.2 Soft Constraints

Soft constraints are useful for reducing the possibility of Barra Optimizer not finding a feasible solution. They are another set of valuable tools for users to manage their constraints. All constraints in the usual sense are hard requirements that an optimal portfolio ought to obey. Hard constraints are usually inflexible and may often lead to infeasibility, leaving the portfolio optimization problem unsolved. On many occasions, infeasibility is due to conflicts among a small group of constraints. Relaxing or "softening" some of the hard constraints may help yield a feasible solution.

Most constraints can be specified as soft constraints in Barra Optimizer. The few exceptions include asset bounds, 5/10/40 rules, and tax constraints. The upper and lower bounds of a soft constraint are treated as a nice-to-have but not a must. For well-behaved convex problems, Barra Optimizer will return an optimal portfolio satisfying the bounds if such a portfolio exists. Otherwise, it will try to find a portfolio with minimal violation of the constraint. For non-convex problems, Barra Optimizer will return the best available feasible solution if it can find one. Otherwise, it will return the portfolio that has the least violation of the constraint among all the portfolios at its disposal. Again, there is no guarantee that the solutions are globally optimal in the non-convex cases.

A soft constraint may be violated due to conflicts with other constraints, particularly in cases where a large number of other constraints and penalties are defined. Barra Optimizer internally applies a penalty to the violation of the soft constraint to discourage the constraint slack moving away from the soft bound. This is not applicable to soft constraints on thresholds, cardinality, roundlotting, and risk constraints.

Barra Optimizer produces a log of the violated soft constraints as part of the optimization output. It supports soft constraints for the following constraint categories:

- General linear constraints (including beta constraint and group constraints)
- Factor constraints
- Turnover constraints
- Transaction cost constraint
- Hedge (leverage) constraints, including total leverage constraint, short/long ratio constraint, and net/total leverage constraint
- Threshold constraints
- Cardinality constraints
- Risk constraints
- Roundlotting constraint

3.9.3 Constraint Hierarchy

A constraint hierarchy is designed to provide users with greater flexibility in arranging and managing their constraints. It allows users to specify constraint priorities in the event of infeasibility.

When no solution can be found that satisfies all constraints simultaneously, it will systematically and sequentially soften the constraints, one, or several at a time, according to their relaxation priorities. In such cases, it will search for a solution with minimal violation with respect to the original constraints. Constraints in the lower levels of the hierarchy are softened first. Constraints not pre-set in the hierarchy will not be relaxed at all. Constraints softened in a lower level of the hierarchy will remain soft for higher levels.

This procedure continues until a feasible solution to the softened problem for a given level of the hierarchy is found, or until all levels of the hierarchy are exhausted.

Barra Optimizer allows you prioritize the following constraint categories:

- General linear constraints
- Factor constraints
- Turnover constraints
- Transaction cost constraint
- Holding cardinality constraints
- Holding threshold constraints

- Transaction threshold constraints
- Trade cardinality constraints
- Paring constraints, which include all of the above four threshold and cardinality categories
- Hedge (leverage) constraints, including total leverage constraint, short/long ration constraint, and net/total leverage constraint
- Risk constraints
- Roundlotting constraints

Barra Optimizer also allows you to set the priority for individual constraints from categories of general linear constraints and factor constraints. You can categorize your constraints based on priority. They are relaxed in the following order:

Priority	Relax Order
Low	Relax first
Medium	Relax next
High	Relax last
Not in Hierarchy	Never Relax

With constraint hierarchies defined, Barra Optimizer will resolve optimization problems sequentially as follows:

1. Tries to solve the original problem; returns successfully if the problem is solved.
2. If finding an optimal solution fails, relaxes constraints with low priority and tries to solve the problem; returns successfully if the problem is solved.
3. If finding an optimal solution still fails, relaxes constraints with medium priority and tries to solve the problem; returns successfully if the problem is solved.
4. If finding an optimal solution still fails, relaxes constraints with high priority and tries to solve the problem; returns successfully if the problem is solved.
5. If finding an optimal solution still fails, terminates and reports failure.

Notes:

- More than one constraint category may be in the same hierarchy (with the same priority)
- When relaxing a constraint (except threshold, cardinality, roundlotting, and risk constraints), Barra Optimizer will set soft bounds for the category so that violations of these constraints are discouraged by soft-bound penalties.
- If the problem has no constraints in a category that is specified in the constraint hierarchies, Barra Optimizer will ignore that category.
- Constraint categories not specified in the hierarchy will not be relaxed at all.

- This feature is especially useful if you do not want a lengthy back-testing process interrupted by infeasibilities caused by restrictive constraints

Constraint Hierarchy Example 1:

Constraint Category	Priority	Relax Order
General linear constraints	Low	Relax first
Turnover constraints	High	Relax last
Transaction cost constraint	Low	Relax first
Hedge constraints	Medium	Relax next
Trade cardinality constraints	Medium	Relax next
Paring constraints	High	Relax last

Considering Example 1:

1. Barra Optimizer first attempts to solve the problem without relaxing any constraints.
2. If the first attempt fails, it tries to solve a modified problem with general linear constraints and the transaction cost constraint relaxed.
3. If the second attempt fails, it tries to solve a modified problem with hedge constraints and trade cardinality constraints relaxed (in addition to constraints relaxed in the first attempt).
4. If the third attempt fails, it tries to solve a modified problem with turnover constraints and all threshold and cardinality constraints relaxed (in addition to constraints relaxed in the previous attempts).
5. Other constraints (for example, factor constraints) are not in the hierarchy and are not relaxed at all.

Constraint Hierarchy Example 2:

Constraint Category/Individual Constraints	Priority	Relax Order
Holding cardinality	Medium	Relax next
Holding threshold	Medium	Relax next
Transaction threshold	Low	Relax first
Factor Momentum constraint	High	Relax last

Considering Example 2:

1. Barra Optimizer first attempts to solve the problem without relaxing any constraints.

2. If the first attempt fails, it tries to solve a modified problem with minimum transaction size threshold relaxed.
3. If the second attempt fails, it tries to solve a modified problem with all three types of threshold and cardinality constraints relaxed.
4. If the third attempt fails, it tries to solve a modified problem with the Momentum factor constraint relaxed (in addition to constraints relaxed in the previous attempts).
5. Instead of softening the threshold and cardinality constraints during the optimization, they are simply relaxed.

3.10 Optimization Inspection

3.10.1 Optimality Conditions

An optimal portfolio ought to fulfill the following three sets of conditions, also known as the Karush-Kuhn-Tucker (KKT) conditions:

- Primal Feasibility Conditions—All primal constraints defined by the original portfolio optimization problem should be satisfied;
- Dual Feasibility Conditions—All conditions for the implied dual variables should be satisfied; and
- Complementary Slackness Conditions—All relationships between the slackness in a primal constraint and the sign of the associated dual variable should be satisfied.

On the other hand, a heuristic portfolio may not satisfy some of the conditions in the last two sets.

Barra Optimizer employs a number of solution methods, some of which are primal algorithms. These algorithms start with a portfolio that satisfies the primal and complementary slackness conditions, and keep searching for better portfolios within the primal feasible region. The algorithms terminate when a portfolio satisfying the dual feasibility conditions is found.

In contrast, dual algorithms start with a portfolio that satisfies the dual feasibility conditions as well as the complementary slackness conditions. These algorithms keep searching for better portfolios within the dual feasible space until they find a portfolio that also satisfies the primal feasibility conditions.

Primal algorithms have an advantage over others in that they always return a portfolio satisfying all original constraints, even if they have to be terminated prematurely. For more information about the algorithms employed in Barra Optimizer, see Liu (2004).

3.10.2 Optimality Tolerances

Optimality tolerances affect the termination criteria of the optimization. Let $\mathbf{h}^*$ be the true theoretical optimal solution to a general Convex Quadratic Programming (CQP) problem, and $u(\mathbf{h}^*)$ be the corresponding true maximum objective. Suppose the optimality tolerance used by an optimizer is ε. Then, the optimizer may terminate its process at any time as long as the solution ($\mathbf{h}^{**}$) it returns satisfies:

$$\left|u(\mathbf{h}^{**}) - u(\mathbf{h}^*)\right| \le \varepsilon \tag{98}$$

In general, the tighter the optimality tolerance, the higher the quality of the optimal solution in terms of the objective function value, but the longer it may take for optimizer to reach its "tolerance-adjusted" optimality.

In practice, however, the true optimal objective is unknown. Different solvers may adopt different surrogate optimality evaluation criteria in lieu of (98). The optimality tolerances used may also be different. The impact of these optimality tolerances on speed will depend on the solver used.

To afford users some control over the tradeoffs between the speed and solution quality, Barra Optimizer offers an optimality tolerance multiplier to scale the default optimality tolerance. The default optimality tolerance multiplier is 1.0. To prevent unreasonable or unpractical settings, there is a permissible range for this multiplier. This range depends on the solver being used. Specifically, the permissible range for our quadratic solver and nonlinear programming solver is [0.01, 10000] and for our second-order cone programming solver is [0.01, 100].

If the user-defined optimality tolerance multiplier is outside the permissible range, the closest permissible value for the selected solver will be used. For example, if users set the multiplier to 1000, and the optimizer decides that the second-order cone programming solver is best suited for the given case, then the optimizer will reset the multiplier to 100 before applying it.

The effectiveness of the optimality tolerance multiplier will depend on the solver used as well as the case characteristics. In some cases, the effect may be counter-intuitive. For example, with cardinality or threshold constraints, larger optimality tolerance multipliers may result in a slower performance because the searching paths might be different.

Under usual circumstances, **leaving the default optimality tolerance multiplier unchanged is strongly recommended.**

3.10.3 Maximum Time Limit

Users can set a maximum time limit on solving a given optimization case. However, this limit is only treated as a guideline by the Barra Optimizer. Sometimes, an internal procedure or process may not be stopped once it starts, causing the time limit to be exceeded.

3.10.4 Indication of the Optimization Problem Type

Barra Optimizer can indicate whether an optimization problem is convex.

An optimization problem is convex if all of the following conditions hold true:

- Objective function is quadratic (not information ratio or Sharpe ratio).
- Fixed transaction costs are absent.
- Roundlotting constraint is not enabled.
- Threshold constraints are absent.
- Cardinality constraints are absent.
- 5/10/40 rule is not enabled.
- Optimization problem is not a tax-aware optimization.
- Turnover by side constraint is absent.

- L/S leverage constraint is absent or all of the following conditions hold:
  - There is no S/L leverage ratio constraint.
  - There is no net/total leverage ratio constraint.
  - There is no positive lower bound on any of the long side leverage constraint(s).
  - There is no negative upper bound on any of the short side leverage constraint(s).
  - There is no positive lower bound on the total leverage constraint.
- There are no fractional contribution risk constraints and no subgroup risk constraints using additive definition.

3.10.5 Indication of the Output Portfolio as Heuristic or Optimal

Barra Optimizer can indicate whether the output portfolio is heuristic or optimal. An output portfolio is optimal if the following conditions hold true:

- The problem is convex and there is no soft bound violation, or
- The objective is information/Sharpe ratio and the objective value is non-negative

3.10.6 Trade List Information

Barra Optimizer outputs the following trade list information for the optimal or roundlotted portfolio for a given asset:

- Trade type (hold, buy, sell, covered buy, short sell, crossover buy, and crossover sell)
- Initial shares
- Traded shares
- Final shares
- Price
- Traded value
- Traded value in percentage
- Fixed transaction cost
- Non-linear transaction cost
- Piecewise linear transaction cost
- Total transaction cost

3.10.7 Solver Information

Starting with Barra Optimizer 8.0, users can obtain the solver information associated with the final solution. Such information may be used to understand why a case takes a long time and may also help decide which constraints to drop to speed up the optimization process. For example, if the solver associated with the final solution is the

"Non-Linear Programming Solver", risk constraints may be the cause for excessive run time. If they are not essential, consider relax or remove them.

3.10.8 Impact to Utility

Impact-to-utility is a value computed by the optimizer, based on duality theory, estimating the instantaneous rate of utility change if a binding bound is relaxed.

- It is a diagnostic tool helping to identify the most influential binding constraint.
- It is not always available for non-convex cases.

3.10.9 Small Weights or Trades

Due to numerical precisions used by different optimization engines, the optimal portfolio may contain small holdings or small trades that are within the tolerances specified. This phenomenon may be especially prominent with interior-point-method-based solvers. Starting with Barra Optimizer 8.6, users have three options if they want to eliminate these small weights or trades post-optimization:

- Add the dropped weights to cash position or treat them as a cash flow.
- Allocate the dropped holdings or trades to the remaining holdings or trades equally.
- Allocate the dropped holdings or trades to the remaining holdings or trades proportionally to their relative sizes.

Note that the resulting portfolio after these adjustments may slightly violate some constraints or asset bounds.

3.10.10 Portfolios and Messages Returned

In general, for well-defined convex portfolio optimization problems, Barra Optimizer will return an optimal portfolio if the problem is feasible. For others, only a heuristic portfolio will be provided.

- APPENDIX A summarizes the types of portfolios and return messages that will be provided for various portfolio optimization problems.
- APPENDIX B shows which combination of features and functions are currently supported in Barra Optimizer.

4 Advanced Optimization Topics

4.1 Threshold and Cardinality Constraints—Paring Optimization

Threshold constraints are used to ensure non-zero trades or positions are sufficiently large. Cardinality constraints restrict the number of non-zero trades or positions to be less than or greater than a set bound. Both can be used to ensure the optimizer makes economically meaningful decisions. Barra Optimizer uses similar techniques for solving these two types of constraints, and thus they are often grouped together.

Inclusion of these constraints makes the optimization problem discrete and non-convex. These problems are well known to be difficult to solve to optimality. For a convex optimization problem, Barra Optimizer will be able to either correctly detect the problem as infeasible, or return an optimal portfolio. This is not true for non-convex problems. The following analogy compares the nature and complexity of the two problems.

Imagine a glass marble is slipping down the side of an empty pool. If the pool is smoothly curved without any craters in the concrete, the initial position of the marble does not matter. One can always count on the marble to settle into a single spot, which must be the lowest point in the pool. This process resembles solving a well-behaved optimization case without discrete or non-convex constraints.

Introducing discrete constraints is like roughing up the smooth surface of the empty pool. The marble slipping down the side of such a pool might settle into a crater in the shallow end without ever reaching the deepest spot in the entire pool. In other words, it might get stuck at a local minimum without ever reaching the global minimum. Where the marble will finally end depends on its initial position and the height of the pool, as well as the configuration details of the pool bottom. Locating the lowest point in the cratered pool with a marble becomes trickier and less reliable in such cases.

Barra Optimizer often refers to threshold and cardinality constraints as paring constraints and the optimization problems that contain them as pairing problems, because Barra Optimizer uses a paring algorithm to solve them. These terms originated from Barra. Barra Optimizer's paring algorithm is a combination of heuristic procedures aimed at finding an approximate optimal solution within a reasonable amount of time. The traditional pare-down heuristic emphasizes optimality. It starts with a standard optimal solution and iteratively pares down excessive assets. By contrast, the build-up heuristic stresses feasibility. It forms a small feasible portfolio that satisfies the paring constraints first, and then gradually adds more assets to improve the objective, or utility. For more detailed discussions of the pare-down and build-up heuristics, as well as their performance results, see Liu and Xu (2016), Liu and Xu (2011) and Xu and Liu (2010).

The quality of heuristic paring solutions can be evaluated using a Branch-and-Bound algorithm in Barra Optimizer based on Lagrangian relaxation and cost splitting. This algorithm is also capable of providing truly optimal solutions for small-sized paring problems, as well as possibly improved heuristic solutions for large ones. For more information, see Kopman, Liu, and Shaw (2009).

The following sections describe cardinality and threshold constraints in detail.

Notes:

- Cash and futures are not counted in the threshold and cardinality constraints. Futures can optionally be counted in the threshold and cardinality constraints.

- Paring is implemented using heuristic approaches that try to find a good solution within a reasonable amount of time; it is not guaranteed that global optimal solutions will always be achieved.
- All combinations of cardinality and threshold constraints are allowed.
- Soft min/max bounds are allowed for cardinality constraints
- Soft thresholds are allowed for minimum level constraints.
- A crossover trade makes an asset change from a long position to a short position, or vice versa, and a crossover trade may be counted as one trade or two trades.

4.1.1 Minimum Holding Thresholds/Holding Level Paring

Minimum holding threshold constraints specify that the holding level for any asset cannot be non-zero and below a certain level. They are also known as holding level paring constraints in Barra Optimizer. Mathematically, they may be represented as:

$$h_i \geq \xi_i^+ \ \ or \ \ h_i \leq -\xi_i^- \ \ or \ \ h_i = 0, \qquad i = 1, 2, \cdots n \qquad \text{(Holding Threshold Constraints)} \qquad (99)$$

where ξ_i^+ and ξ_i^- are positive real constants. Note that constraints (99) require that the minimum holding level for asset i be ξ_i^+ if it takes on a long position, or ξ_i^- if it takes on a short position.

However, starting with Barra Optimizer 8.4, an "EnableGrandfatherRule" option is provided. If you select this option, $h_i \geq h_{0i}$ is always considered feasible when $h_{0i} > 0$, even if $h_{0i} < \xi_i^+$; and similarly, $h_i \leq h_{0i}$ is always considered feasible when $h_{0i} < 0$, even if $|h_{0i}| < \xi_i^-$.

Mathematically, the holding threshold constraints with the grandfather rule can be represented by:

$$\begin{cases} h_i \geq \text{Min}(\xi_i^+, h_{0i}) \ \ or \ \ h_i \leq -\xi_i^- \ \ or \ \ h_i = 0, \ \ if \ \ h_{0i} > 0, \\ h_i \geq \xi_i^+ \ \ or \ \ h_i \leq \text{Max}(-\xi_i^-, h_{0i}) \ \ or \ \ h_i = 0, \ \ if \ \ h_{0i} < 0, \qquad i = 1, 2, \cdots n \\ h_i \geq \xi_i^+ \ \ or \ \ h_i \leq -\xi_i^- \ \ or \ \ h_i = 0, \ \ if \ \ h_{0i} = 0, \end{cases} \qquad (100)$$

4.1.2 Minimum Trade Thresholds/Transaction Level Paring

Minimum trade threshold constraints specify that the transaction size for any non-zero buy or sell trades cannot be below a certain level. These constraints are also known as Transaction Level Paring. Mathematically, they may be represented as:

$$h_i - h_{0i} \geq \zeta_i^+ \ \ or \ \ h_i - h_{0i} \leq -\zeta_i^- \ \ or \ \ h_i - h_{0i} = 0, \qquad i = 1, 2, \cdots n \qquad \text{(Trade Threshold Constraints)} \qquad (101)$$

where ζ_i^+ and ζ_i^- are positive real constants. Note that constraints (101) require that the minimum transaction size for asset i be ζ_i^+ if the transaction is a buy trade, or ζ_i^- if it is a sell trade. Under this definition, $h_i = 0$ may not be feasible if $0 \leq h_{0i} < \zeta_i^-$ or $-\zeta_i^+ < h_{0i} \leq 0$.

However, starting with Barra Optimizer 8.0, an "Allow Close-Out" option is provided. If you select this option, $h_i = 0$ is always considered feasible even if the transaction size is below the minimum requirement.

Mathematically, the trade threshold constraints with the close-out option can be represented by:

$$h_i - h_{0i} \geq \zeta_i^+ \quad or \quad h_i - h_{0i} \leq -\zeta_i^- \quad or \quad h_i - h_{0i} = 0 \quad or \quad h_i = 0, \qquad i = 1, 2, \cdots n \tag{102}$$

4.1.3 Holding Cardinality/Asset Paring

Holding cardinality constraints specify that the number of assets in the whole portfolio or in an asset group cannot be more or less than a certain number. These are also known as asset paring constraints.

Mathematically, they may be represented as:

$$K_{\min}^J \leq \sum_{i \in G_j} \delta_i \leq K_{\max}^J, \ \forall J, \ \delta_i = \begin{cases} 0, & if \ h_i = 0 \\ 1, & otherwise \end{cases} \qquad \text{(Holding Cardinality Constraints)} \tag{103}$$

where, $K_{\max}^J$ and $K_{\min}^J$ are positive integers indicating the maximum or minimum number, respectively, for group G_J. An asset group can be comprised of the whole portfolio, or its predefined subset. Barra Optimizer also supports counting long versus short assets separately, through a more complex optimization procedure known as hedge optimization. For more information, see the By-Side Optimization—Leverage, Turnover by Side, Cardinality by Side section.

Note that long or short assets are NOT pre-defined. Rather, they are the output of optimization.

4.1.4 Trade Cardinality/Paring

Trade cardinality constraints specify that the number of trades involved in transacting an initial portfolio into an optimal one cannot be greater or less than a certain number. These are also known as trade paring constraints. Mathematically, they can be represented as:

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 0, & if \ h_i - h_{0i} = 0 \\ 1, & otherwise \end{cases} \qquad \text{(Trade Cardinality Constraints)} \tag{104}$$

where, $T_{\max}^J$ and $T_{\min}^J$ are positive integers representing the maximum and minimum, respectively, for group G_J.

A trade can either be a buy or sell trade. In addition to constraining the total number of trades, Barra Optimizer allows setting a maximum or minimum on the number of buy or sell trades separately.

Mathematically, these constraints are variations of (102), namely,

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 1, & if \ h_i > h_{0i} \\ 0, & otherwise \end{cases} \qquad \text{(Trade Cardinality Constraints for Buy Trades)} \tag{105}$$

OR

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 1, & if \ h_i < h_{0i} \\ 0, & otherwise \end{cases} \qquad \text{(Trade Cardinality Constraints for Sell Trades)} \tag{106}$$

For more information, see Xu and Liu (2011).

4.1.5 Paring Tolerances

Numerical tolerances are used by the Barra Optimizer to determine whether a paring constraint is satisfied or violated. Users have the flexibility to use three different tolerances for different types of paring constraints:

- torA—Tolerance for Asset Paring
- torL—Tolerance for Level Paring
- torT—Tolerance for Trade Paring

The default value for all three tolerances is 5.0e-6. With these tolerances, the paring constraints can be re-written as follows:

- Holding Threshold Constraints:

$$h_i \geq \xi_i^+ - torL \quad or \quad h_i \leq torL - \xi_i^- \quad or \quad -torL \leq h_i \leq torL, \qquad i = 1,2,\cdots n \tag{107}$$

- Trade Threshold Constraints:

$$h_i - h_{0i} \geq \zeta_i^+ - torL \quad or \quad h_i - h_{0i} \leq torL - \zeta_i^- \quad or \quad -torL \leq h_i - h_{0i} \leq torL, \qquad i = 1,2,\cdots n \tag{108}$$

- Holding Cardinality Constraints:

$$K_{\min}^J \leq \sum_{i \in G_j} \delta_i \leq K_{\max}^J, \ \forall J, \ \delta_i = \begin{cases} 1, & if \ h_i > torA \ or \ h_i < -torA \\ 0, & otherwise \end{cases} \tag{109}$$

- Trade Cardinality Constraints:

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 1, & if \ h_i - h_{0i} > torT \quad or \quad h_i - h_{0i} < -torT \\ 0, & otherwise \end{cases} \tag{110}$$

- Trade Cardinality Constraints for Buy Trades:

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 1, & if \ h_i > h_{0i} + torT \\ 0, & otherwise \end{cases} \tag{111}$$

- Trade Cardinality Constraints for Sell Trades:

$$T_{\min}^J \leq \sum_{i \in G_J} \sigma_i \leq T_{\max}^J, \quad \forall J, \quad \sigma_i = \begin{cases} 1, & if \ h_i < h_{0i} - torT \\ 0, & otherwise \end{cases} \tag{112}$$

4.1.6 Linear Penalties on Paring Constraints

In the earlier Barra Optimizer versions, cardinality and threshold constraints could be specified either as hard or soft constraints, but no penalty functions were allowed on these constraints. When a cardinality or threshold constraint was specified as hard, then any solution violating the constraint would be deemed as infeasible. On the other hand, when a constraint was specified as soft, all else being equal, a solution satisfying the constraint would be deemed as infinitely better than a solution violating the constraint, regardless of the utility values of the two solutions.

Starting with Barra Optimizer 8.4, users are allowed to set linear penalties on cardinality and threshold constraints. Violations on these constraints are translated through penalty rates into reductions on the utility level. More specifically, the optimization problem with penalties on paring constraints becomes:

$$\begin{aligned} &\underset{\mathbf{h}}{\text{Maximize:}} && U(\mathbf{h}) - Pen_{paring}(\mathbf{h}) \\ &\text{Subject to:} && \text{Paring constraints}^{4} \\ & && \text{Other Constraints} \end{aligned} \tag{113}$$

where,

$U(\mathbf{h})$ = the original utility function when paring penalties were not allowed

$Pen_{paring}(\mathbf{h})$ = the sum of paring penalties penalizing the violations of all cardinality or threshold constraints

Different penalty rates can be applied to different types of cardinality or threshold constraints. Let us assume the following:

P_A = penalty per extra asset or penalty due to lack of one asset

P_T = penalty per extra trade or penalty due to lack of one trade

P_{HL} = penalty per unit below the minimum holding threshold

P_{TL} = penalty per unit below the minimum trade threshold

Then, the total paring penalties can be represented by:

$$\begin{aligned} Pen_{paring}(\mathbf{h}) = {} & P_A \cdot \{Max(\sum_{i=1}^{n}\delta_i - K^J_{\max}, 0.0) + Max(K^J_{\min} - \sum_{i=1}^{n}\delta_i, 0.0)\} \\ & + P_T \cdot \{\max(\sum_{i=1}^{n}\sigma_i - T^J_{\max}, 0.0) + \max(T^J_{\min} - \sum_{i=1}^{n}\sigma_i, 0.0)\} \\ & + P_{HL} \cdot \sum_{i=1}^{n}\left(\theta_i^+ \cdot (\xi_i^+ - h_i) + \theta_i^- \cdot (h_i - \xi_i^-)\right) \\ & + P_{TL} \cdot \sum_{i=1}^{n}\left(\varphi_i^+ \cdot (h_{0i} + \zeta_i^+ - h_i) + \varphi_i^- \cdot (h_i - h_{0i} + \zeta_i^-)\right) \end{aligned} \tag{114}$$

where,

$K^J_{\max}$ and $K^J_{\min}$ = the upper and lower bounds of the holding cardinality constraint, respectively

$T^J_{\max}$ and $T^J_{\min}$ = the upper and lower bounds of the trade cardinality constraint, respectively

ξ_i^+ and ξ_i^- = the long- and short-side minimum holding thresholds, respectively

ζ_i^+ and ζ_i^- = the buy- and sell-side minimum trade thresholds, respectively

[4] Some of these paring constraints will be relaxed if the penalty rate corresponding to their paring type is positive.

$$\delta_i = \begin{cases} 1, & if\ h_i > torA\ \ or\ h_i < -torA \\ 0, & otherwise \end{cases}$$

$$\sigma_i = \begin{cases} 1, & if\ h_i > h_{0i} + torT\ \ or\ h_i < h_{0i} - torT \\ 0, & otherwise \end{cases}$$

$$\theta_i^+ = \begin{cases} 1, & if\ torL < h_i < \xi_i^+ - torL \\ 0, & otherwise \end{cases}$$

$$\theta_i^- = \begin{cases} 1, & if\ -\xi_i^- + torL < h_i < -torL \\ 0, & otherwise \end{cases}$$

$$\varphi_i^+ == \begin{cases} 1, & if\ h_{0i} + torL < h_i < h_{0i} + \zeta_i^+ - torL \\ 0, & otherwise \end{cases}$$

$$\varphi_i^- == \begin{cases} 1, & if\ h_{0i} - \zeta_i^- + torL < h_i < h_{0i} - torL \\ 0, & otherwise \end{cases}$$

Notes:

- For group-level paring constraints, the summations in (114) should be over all assets in the particular group rather than over all assets in the entire portfolio.
- When the penalty rate for a specific paring type is positive, all paring constraints of that type will be relaxed, no matter whether they were input as hard or soft constraints. In other words, violations on the paring constraints with a positive penalty rate will not trigger infeasibility. Instead, they will be penalized according to the penalty rates provided by the user. When comparing two paring solutions involving positive penalty rates, all else equal, the one with higher utility will always be preferred.
- When the penalty rate for a specific paring type is zero, then paring penalties for that type will be ignored, and all paring constraints of that type will behave the same as before. That is, when a paring constraint is specified as hard, then any violation of it will trigger infeasibility. When it is specified as soft, then a solution violating the constraint will be deemed as infinitely worse than a solution satisfying the constraint, all else equal, regardless of the utility values of the two solutions.

4.1.7 Upper Bound on Utility and Branch-and-Bound Algorithm for Paring Cases

Starting with Barra Optimizer 8.4, users have the option to obtain an upper bound on utility for some paring cases. This upper bound is estimated by Barra Optimizer using the perspective reformulation approach. It can be used to estimate the utility gap between the unknown optimal and known heuristic solutions:

$$Gap_{uti} = UB_{uti} - Uti(\mathbf{h}^*) \tag{115}$$

where $\mathbf{h}^*$ is the heuristic portfolio returned by Barra Optimizer, $Uti(.)$ is the utility function of the paring case in question, and UB_{uti} is the upper bound on utility.

Note: Currently, the scope for option is limited.

This option only applies to cases that are simply standard portfolio optimization cases and limited cardinality or threshold constraints (see the Problem Classification section for the definition of a standard case). In other words, no other special constraints are allowed besides paring. For example, the following features CANNOT be present in the cases:

- Non-convex leverage constraints
- Non-linear transaction costs
- Risk constraints, with one exception described below
- Tax-aware optimization
- Roundlotting constraints
- Parametric optimization

In addition, note the following:

- The specific covariance matrix must be diagonal or block-diagonal. In particular, composite assets are not allowed in the specific covariance matrix of these cases.
- Specific risk aversion must be positive, and/or there must be exactly one hard constraint on risk or tracking error.
- Only hard maximum cardinality constraints and hard threshold constraints are currently supported. Minimum cardinality constraints, penalties on cardinality and threshold constraints, soft cardinality, and threshold constraints are not supported.

Starting with Barra Optimizer 8.6, users have an option to solve a paring problem by calling a newly developed branch-and-bound (BB) algorithm. The BB algorithm is based on the bounding routine mentioned above and currently has the same scope limitations. The BB algorithm may take longer than the heuristics described in the beginning of this section, but may potentially produce better results, as was demonstrated in internal testing.

The BB algorithm is tuned, however, to produce a good solution as fast as possible, so it is not guaranteed to produce the optimal solution. Additionally, failure of the BB algorithm does not mean that the feasible solution does not exist. BB algorithm might exit earlier than the user-defined timeout if it deems it is not likely to find a feasible solution. The users will need to explicitly specify that they want BB algorithm to run, instead of the standard heuristics. This option is mutually exclusive with the option of getting an upper bound on utility. If both

are specified, the one specified last will take effect. Both options will be ignored when the problem features are out of scope.

4.2 Roundlotting—Optimal and Post-Optimization

In general, solutions to portfolio optimization problems are real numbers. The resulting trade list may contain many small or odd trades—for example, to buy 11.51 shares of IBM, or to sell 3603 shares of AAPL. In practice, however, stocks are typically traded in multiples of round lots rather than in sporadic odd ones. The size of a round lot may vary. For most stocks on the New York Stock Exchange, one round lot is 100 shares.

4.2.1 Optimal Roundlotting or Constraint-Aware Roundlotting

Round lot constraints impose restrictions on the size of the trades leading to an optimal portfolio. These constraints may be represented by:

$$h_i = h_{0i} + \omega_i r_i, \qquad i = 1, 2, \ldots, n \tag{116}$$

where,

- r_i = round lot trade size (in terms of portfolio weight) for assets i
- ω_i = an integer number to be determined

Enforcing the above n constraints ensures that the trade list contains only round lot trades. Note that the final optimal holdings may not be round lot shares.

Unless h_{0i} is a round lot position itself, $h_i = 0$ may not be a feasible solution to an optimal roundlotting case because $h_i - h_{0i}$ may not be a round lot trade. However, if the option "AllowOddCloseOut" is set to be true, then the optimizer will consider $h_i = 0$ as a feasible solution. Note that this is just an option, not a requirement for the optimizer to close out positions. The optimizer will only "close out" a position if doing so improves utility.

Similar to the threshold and cardinality constraints discussed in the Threshold and Cardinality Constraints—Paring Optimization section, the presence of the round lot constraints makes the feasible region discontinuous and complicated. Optimal Roundlotting is a hard-to-solve mixed-integer programming problem. A solution may not exist. Even if an optimal solution does exist, it is usually difficult to find.

Barra Optimizer employs heuristic algorithms to solve the Optimal Roundlotting problem. These algorithms seek a portfolio that is near optimal within a reasonable amount of time. The solution portfolios, in general, are feasible yet not globally optimal. For more information about constraint-aware roundlotting, see Xu and Liu (2013).

4.2.2 Post-Optimization

Instead of imposing the round lot constraints, Barra Optimizer now provides users with a utility function to round the optimal trades to their nearest round lots after the optimization process. This is a simple, one-step mechanical procedure. The resultant trade list will have no odd lots, yet one or more of the original constraints in the problem may be violated.

Notes:

- One or more of the original constraints in the problem may be violated.

- The in-optimization roundlotting constraint need not be enabled to run post-optimization roundlotting.
- Post-optimization roundlotting is not applicable to frontier optimization.
- If crossover trade is not allowed and the nearest roundlot trade results in a crossover trade:
  - If closeout with odd lot trade is allowed, the final position is set to zero
  - If closeout is not allowed, make a roundlot trade in reverse direction to avoid crossover
- Cash position may be adjusted to offset those post optimization roundlotting adjustments.

4.3 Risk Parity

4.3.1 Definition

A portfolio with an equal contribution to risk from each risky asset satisfies the risk parity condition, and is also known as an equal risk weighted portfolio. Contribution to risk is computed as the asset weight multiplied by the asset's marginal contribution to risk. Each asset is expected to contribute the same magnitude to portfolio's return at the end of the period.

Mathematically, it can be represented as follows:

$$c_i = \frac{\mathbf{h}_i (\mathbf{Qh})_i}{\sqrt{\mathbf{h}^t \mathbf{Qh}}} \tag{117}$$

where,

- $\mathbf{h}$ is the asset holding vector
- $\mathbf{Q}$ is a covariance matrix of asset returns
- c_i is the contribution of asset *i* to portfolio risk

Risk parity (also called equal risk contribution) requires the following:

$$c_i = c_j \quad \forall i, j \in S$$

We call the set S of assets included in risk parity condition the risk parity set. It can be used when the portfolio manager needs a weighting schema within a group of assets, and does not have a view on the expected return for the assets within that group. While the minimum volatility portfolio places most of the weight on low-risk assets, and equal weighting is indifferent as to the differences in risk between assets, a risk-parity approach weights each asset inversely proportional to its risk contribution.

4.3.2 Risky and Risk-free Assets

Risk parity set S has the following meaning:

- Only assets included in S are allowed to contribute to risk
- Their contributions are equal

Hence,

- Risky assets not included in S are not going to be held
- Risk-free assets (such as cash) not included in S are allowed to be held
- If a risk-free asset is included in S, portfolio risk is going to be 0, hence only risk-free assets are going to be held

When there are no risk-free assets in the problem and the asset covariance matrix is non-singular, the long-only risk parity portfolio is unique. This uniqueness has the following practical implications:

- Imposing additional constraints besides the ones describing risk parity has a definite potential of making the portfolio construction problem infeasible.
- Trying to influence portfolio weights in any way by introducing extra terms in the objective function will have no effect if the solution is unique.

When there is a unique risk-free asset in the problem, any risk parity portfolio will be uniquely determined by the weight of the risk-free asset.

4.3.3 Inputs and Scope

Inputs for risk parity consist of risk parity indicator, the set S, and the risk model, giving rise to the asset covariance matrix (i.e. primary or secondary).

Risk parity portfolio construction is supported by Barra Optimizer with the following limitations:

- The problem is long-only for all risky assets (shorting risk free assets is allowed).
- Equal risk contribution is imposed on asset level for total portfolio risk (not tracking error).
- All constraints are convex and non-soft.
- There are no discrete elements in the problem (i.e. no cardinality, threshold, roundlotting, 5/10/40 constraints, or fixed transaction costs).
- The objective function is linear or convex quadratic.

4.3.4 Usage Recommendations

Because the risk parity condition is very restrictive, users are discouraged from imposing additional unnecessary constraints or objective function terms in the risk parity portfolio construction problem.

4.3.5 Risk Parity Portfolio Construction

Risk parity portfolio construction is often utilized as a step in a multi-stage process. One common use case is to use the portfolio formed as a benchmark for further construction. This can be quite helpful if additional portfolio rules or constraints need to be satisfied that may violate risk parity.

The tradeoff between assets as the optimizer moves away from risk parity to satisfy constraints can be controlled by the objective function for the second optimization.

- Minimize risk relative to risk parity => Use a variance aversion with the risk model
- Minimize the weight deviation => Use a quadratic or linear penalty.
- Maximize alpha, minimize risk, minimize transaction cost => Use a standard quantitative process

The solution to the risk parity problem can also be used to scale weights in an underlying benchmark. For example, an aggregate sector weighting can be determined through risk parity. In addition, an industry rotation strategy can leverage risk parity portfolio construction to select the weighting within a sector, as an alternative to market cap weighting the sector components.

4.4 Diversification Control

Several measures of the diversification level of a portfolio have been proposed in literature. The two important ones valuable in quantitative portfolio construction are:

- Portfolio concentration
- Diversification ratio

Barra Optimizer supports an upper bound on portfolio concentration or a lower bound on diversification ratio.

4.4.1 Portfolio Concentration Limit Constraint

For long-only portfolios, portfolio concentration is defined as the sum of weights of the top k_c largest holdings. A lower concentration implies a higher diversification. To prevent a portfolio from being too concentrated, one may want to set an upper bound on portfolio concentration.

Mathematically, a portfolio concentration limit constraint can be written as:

$$\sum_{j=1}^{k_c} h_{cj} \le U_{ConP} \tag{118}$$

where, $h_{c1} \ge h_{c2} \ge \cdots \ge h_{cn}$ are the asset weights ranked from the largest to smallest and $1 \le k_c \le n$.

For example, a client may want to cap the top 5 holdings of his portfolio at 45% of the total portfolio value. His asset universe is 500. In this case, $k_c = 5$, $n = 500$ and $U_{ConP} = 0.45$. Thus, his portfolio concentration limit constraint is essentially $h_{c1} + h_{c2} + h_{c3} + h_{c4} + h_{c5} \le 0.45$.

The portfolio concentration limit constraint can be used to meet any regulatory guidelines that limit the concentration of the top holdings.

Note that this constraint is not a simple linear constraint in $\mathbf{h}$, because assets that have the largest holdings are not known in advance.

Notes:

- This constraint is applied to (total) asset weights directly. An exclusion list can be used to specify the assets not to be considered when applying this constraint.
- Shorting is allowed. Negative weights, instead of absolute weights, are used for short positions in this constraint.
- A composite is counted as an asset. In other words, there is no "look through ability".
- In general, this constraint can be combined with the analytic features that can be used in a standard portfolio optimization.

4.4.2 Diversification Ratio Limit Constraint

The diversification ratio of a portfolio is the ratio of its weighted average asset volatility to its actual volatility. To ensure a portfolio has at least certain level of diversification, one would want to set a lower bound on the diversification ratio.

Mathematically, the diversification ratio limit constraint can be represented as:

$$L_{DR} \le \frac{\sum_{i=1}^{n} h_i \sigma_i}{\sigma_p} \tag{119}$$

where, h_i is the portfolio weight for asset i, σ_i is the volatility risk for asset i, σ_p is the total risk of the portfolio, and n is as usual the total number of assets in universe.

When assets in a portfolio are perfectly correlated to each other, the total risk of the portfolio equals to its weighted average asset volatility. In such a case, the diversification ratio of the portfolio is 1. In general, however, assets are not perfectly correlated to each other, implying that the total risk of a portfolio is less than its weighted average asset volatility and the portfolio diversification ratio is greater than 1. A well-diversified portfolio has a smaller total risk, and hence a larger diversification ratio.

Notes:

- This constraint can only be used in long-only optimization.
- It applies to total risk only (i.e., no benchmark in the definition of the diversification ratio).
- It can be combined with the analytic features that can be combined with a convex risk constraint.

4.5 Residual Alpha

An asset's active return, or alpha, may be decomposed into a part that is spanned by the risk exposures— $\boldsymbol{\alpha}_R$ and a part that is residual (orthogonal) to them— $\boldsymbol{\alpha}_{R_\perp}$

$$\boldsymbol{\alpha} = \underbrace{\mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\boldsymbol{\alpha}}_{\boldsymbol{\alpha}_R} + \underbrace{\left(\mathbf{I} - \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\right)\boldsymbol{\alpha}}_{\boldsymbol{\alpha}_{R_\perp}} \tag{120}$$

To penalize the residual alpha in the portfolio optimization, a quadratic penalty term is added to Barra Optimizer's objective function $f(\mathbf{h})$:

$$Maximize \quad f(\mathbf{h}) - \lambda_R \cdot \left(\tilde{\alpha}'_{R_\perp}(\mathbf{h} - \mathbf{h}_B)\right)^2 \tag{121}$$

where,

- $\tilde{\alpha}_{R_1} = \dfrac{\alpha_{R_\perp}}{std\left(\alpha_{R_\perp}\right)}$ is scaled residual alpha
- $std\left(\alpha_{R_\perp}\right) = \sqrt{\dfrac{\sum_{i=1}^{n}\left(\alpha_{R_\perp,i} - \bar{\alpha}_{R_\perp}\right)^2}{n-1}}$ $\bar{\alpha}_{R_\perp}$ is the average of $\alpha_{R_\perp}$ elements
- $\alpha_{R_1} = \left(I - \mathbf{X}\left(\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\right)\alpha$ is a non-scaled residual alpha
- W is a diagonal matrix that represents a weighting scheme user chooses
- λ_R is a user-specified multiplier for the residual alpha penalty term. The default value of λ_R can be computed using the following formula:

$$\lambda_R = 100 * \lambda_F * {\sigma_{\alpha_{R1}}}^2, \quad \text{where } \sigma_{\alpha_{R1}} = 0.02 \tag{122}$$

For background information and application issues regarding residual alpha, see Bender et al. (March 2009) and Bender et al. (June 2009).

4.6 By-Side Optimization—Leverage, Turnover by Side, Cardinality by Side

Long-Short (Hedge) Optimization is not characterized by just allowing shorting. Instead, it is characterized by at least one of the leverage constraints defined in the Leverage (Hedge) Constraints section. From Barra Optimizer's perspective, an optimization problem with short positions yet without leverage or other special constraints is simply a Standard Optimization problem. For a summary of Long-Short Optimization in the Barra Optimizer, see Liu and Xu (2014).

Long-Short Optimization and Standard Optimization have many features and functions in common. In particular, variance of either the portfolio's total return or active return with respect to a benchmark may be specified in the objective function. Long—Short Optimization can also be combined with Risk Constrained Optimization and Round Lot Optimization. Certain features—for example, turnover-by-side constraints, threshold-by-side constraints, cardinality-by-side constraints, and short rebates/costs—are currently unique to Long-Short Optimization and not available in Standard Optimization, though.

Long-Short Optimization is generally more difficult to solve than Standard Optimization due to the presence of the leverage constraint(s). The algorithms involved are more computationally expensive. Moreover, a lower bound on a leverage constraint may make the problem non-convex. For these reasons, it is best to use Standard Optimization instead of Long-Short Optimization whenever possible.

Notes:

- Tax-aware optimization is currently not supported with hedge optimization.
- You may set penalties on slacks s_{long}, s_{short} and s_{Hi}.
- Barra Optimizer will take into account short rebate costs if you specify them. For information about short rebate costs, see the Short Rebates/Short Costs section.
- By default, futures are excluded from leverage constraints and by side constraints. However, optionally, futures can be included in such constraints.

4.6.1 Leverage (Hedge) Constraints

Leverage constraints limit the total amount of long positions (L) or short positions (S) or both in a portfolio. By definition,

$$L = \sum_{i=1, i\neq cash, i\neq futures}^{n} \max(h_i, 0) \tag{123}$$

and

$$S = \sum_{i=1, i\neq cash, i\neq futures}^{n} \max(-h_i, 0) \tag{124}$$

Note that the Portfolio Balance Constraint can now be represented by:

$$C + L - S = 1 \tag{125}$$

where, C is the total amount of cash in the portfolio. Fixing any two of the three values L, S, and C implicitly determines the third if the holding constraint is enforced. The cash asset C is optional.

From time to time, you may not want to have a cash asset in the investable universe, nor have the typical portfolio balance constraint. Instead, you may want to constrain on the net leverage, which is the difference between total long and total short. In this case, you can drop the portfolio balance constraint and add a general linear constraint as a replacement:

Lower Bound <= Long – Short <= Upper Bound

For example, for dollar-neutral portfolios, you can add a linear constraint:

Long –Short = 0.

The following sections present several types of leverage constraints currently supported in Barra Optimizer. We use l and u to denote the lower and upper bounds throughout the sections.

4.6.1.1 Hedge Scaling Constraint

The Hedge Scaling Constraint has the following form:

$$l_L \le L \le u_L \tag{126}$$

In essence, it is a leverage constraint on long positions only.

4.6.1.2 Weighted Long or Short Leverage Constraints

The Weighted Long Leverage Constraints have the following form:

$$l_{L_i} \le \sum_{j=1,\, j\neq cash,\, j\neq futures}^{n} a_{ij} \max(h_j, 0) \le u_{L_i}, \quad i = 1, 2, \ldots, n \tag{127}$$

where a_{ij} is the weight for h_j in the i^{th} weighted long hedge constraint. Similarly, the Weighted Short Leverage Constraints have the following form:

$$l_{s_i} \le \sum_{j=1,\, j\neq cash,\, j\neq futures}^{n} b_{ij} \min(h_j, 0) \le u_{S_i}, \quad i = 1, 2, \ldots, n \tag{128}$$

where b_{ij} is the weight for h_j in the i^{th} weighted short hedge constraint.

Note that the scaling constraint (126) can be considered as a special case of the constraints in (127). For example, by setting $a_{ij} = 1$ for $j = 1, 2, \ldots, n$, the i^{th} constraint of (127) reduces to (126). Another example of (127) is a constraint that requires the total long exposure to the banking industry be at most 10% of the given base value. Similarly, constraining the total short exposure to the size factor to be at least 0.02 is an example of (128).

4.6.1.3 Total Leverage Constraint

The Total Leverage Constraint has the following form:

$$l_T \le L + S \le u_T \tag{129}$$

In essence, it is a leverage constraint on the sum of long and short positions.

Note: You can set soft bound and/or penalty on total leverage.

4.6.1.4 Short-Long Leverage Ratio Constraint

The Short-Long Leverage Ratio Constraint has the following form:

$$l_r \le \frac{S}{L} \le u_r \tag{130}$$

Notes:

- This constraint is not defined if $L = 0$.
- You can set a soft bound on the short/long ratio.
- You cannot set a penalty on the short/long ratio.

4.6.1.5 Net-Total Leverage Ratio Constraint

The Net-Total Leverage Ratio Constraint has the following form:

$$l_r \le \frac{L - S}{L + S} \le u_r \tag{131}$$

Note that the numerator of the ratio is the net leverage, whereas the denominator is the total leverage.

From (125), we see that:

$$L - S = 1 - C \tag{132}$$

Thus, any limit on the net leverage $L - S$ can be translated into a bound on the cash asset when holding constraint is present.

Notes:

- You can set a soft bound on the net/total leverage ratio.
- You cannot set a penalty on the net/total leverage ratio.

4.6.1.6 Weighted Total Leverage Constraints

The Weighted Total Leverage Constraints have the following form:

$$l_{T_i} \le \sum_{j \ne cash,\, j \ne futures}^{n} c_{ij} \max(h_j, 0) - \sum_{j \ne cash,\, j \ne futures}^{n} d_{ij} \min(h_j, 0) \le u_{T_i}, \quad i = 1, 2, \ldots, n \tag{133}$$

where the c_{ij} and d_{ij} are the long and short weights, respectively, for h_j in the i^{th} weighted total hedge constraint. The total leverage constraint (129) is actually a special case of the constraints in (133).

4.6.2 Penalty on Leverage Constraints

Barra Optimizer supports penalties on the hedge constraints (126)-(129) and (133). These penalties can be either quadratic or pure linear and either symmetric or asymmetric, as described in the Constraint Flexibility—Penalties, Soft Bounds, and Hierarchy section. This feature helps users to tilt their portfolios towards specific leverage targets, meeting their compliance requirements set either by regulation or investment policy.

Penalty on either of the leverage ratio constraints(130) and (131) is not currently supported.

4.6.3 Turnover-by-Side Constraints

In Long-Short Optimization, in addition to limiting the overall turnover, users may constrain turnover at either the long or the short side. The long side turnover constraint is defined as:

$$\frac{\max(Long\ buys, Long\ sales)}{Initial\ long\ positions} \le TurnoverLimit_{long} \tag{134}$$

Whereas, the short side turnover constraint is defined as:

$$\frac{\max(Short\ covers, Short\ sales)}{|Initial\ short\ positions|} \le TurnoverLimit_{short} \tag{135}$$

Alternatively, the following definitions may be used:

$$\frac{Long\ buys + Long\ sales}{2 \cdot Initial\ long\ positions} \le TurnoverLimit_{long} \tag{136}$$

$$\frac{Short\ covers + Short\ sales}{2 \cdot |Initial\ short\ positions|} \le TurnoverLimit_{short} \tag{137}$$

Let $h_{0i}^{+} = max(0, h_{0i})$ and $h_{0i}^{-} = min(0, h_{0i})$.

Then, in more rigorous mathematical terms:

- $Long\ buys = \sum_i max(h_i - h_{0i}^{+}, 0)$, $Long\ sales = \sum_i max(h_{0i}^{+} - \max(0, h_i), 0)$
- $Short\ sales = \sum_i max(h_{0i}^{-} - h_i, 0)$, $Short\ covers = \sum_i \max(\min(0, h_i) - h_{0i}^{-}, 0)$
- $Initial\ long\ positions = \sum_i h_{0i}^{+}$, $|Initial\ short\ positions| = -\sum_i h_{0i}^{-}$

4.6.4 Threshold and Cardinality-by-Side Constraints

Barra Optimizer allows multiple threshold or cardinality constraints to appear in the Long-Short Optimization. In addition to the usual constraints with respect to the overall holdings—maximum or minimum number of assets,

minimum holding levels or transaction sizes, and maximum or minimum number of trades—Barra Optimizer supports these constraints by the long or short side separately.

Maximum or Minimum Number of Assets by Side

For holding cardinality, Barra Optimizer allows users to constrain the number of long names and short names separately. For instance, one may limit the maximum number of long assets to 100, and the minimum number of short assets to 10. As in Standard Optimization, one may also set a limit on the total number of assets in the portfolio.

Minimum Holding Level by Side

For holding thresholds, users may set different thresholds on the long and short side holdings simultaneously. For example, users may require that any long position be at least 1%, but any short position be at most -0.5%.

Minimum Transaction Level by Side

For trade thresholds, users may simultaneously set different thresholds on the long side transactions (i.e., for long buys and long sales) and short side transactions (i.e., for short covers and short sales). For example, one may specify that any trade size on the long side should be at least 0.5%, and any trade size on the short side should be at least 1%.

By definition, if an asset has an initial long position, the minimum long side transaction level is applied to its buy and sell transactions. Similarly, if any asset has an initial short position, then the minimum short side transaction level is applied to all its transactions. However, if an asset's initial position is zero, then the minimum long side transaction level will be applied to its buy transactions, while the minimum short side transaction level will be applied to its sell transactions.

Maximum or Minimum Number of Trades by Side

For trade cardinality, users have the flexibility to limit the number of long trades or short trades separately. A long trade occurs when an asset has a positive initial position and its final position differs from this initial, or when the initial position is zero and the final position is positive. Conversely, a short trade arises when an asset has a negative initial position with a different final position, or a zero initial position but a negative final position. In the case where an asset has a negative initial position and a positive final position, the transaction actually involves two segments—a short trade (covering a short position) plus a crossover long trade (buying a long position). Similarly, the transaction going from a positive initial position to a negative final position consists of a long trade (selling a long position) plus a crossover short trade (initiating a short position). Barra Optimizer gives the user a choice to treat a transaction with a crossover segment either as one trade or as two trades.

4.6.5 Short Rebates/Short Costs

In a short sale of a security, a short-seller first borrows and then sells the stock. Typically, the proceeds from the short sale earn interest for the seller, but the seller also pays various fees to the lender and prime broker for providing the stock. The amount of the interest earned on proceeds, minus the sum of all costs associated with short-selling, is the short rebate the seller actually gets back. A negative short rebate implies a net short cost to the short-seller.

Short rebates can be modeled directly in Barra Optimizer's Long-Short Optimization. When short rebates are present, the objective function becomes:

$$\text{Objective Function} = \text{Return} - \text{Risk} - \text{Transaction Cost} + \text{Short Rebates} - \text{Others} \quad (138)$$

For individual assets, the short rebate is defined as:

$$ShortRebate_i = SRate_i.\text{MAX}(-h_i, 0) \quad (139)$$

where $SRate_i$ is the user-provided short-rebate rate for asset i. Either $SRate_i$ can be passed directly to the optimizer, or the user can pass its components:

$$SRate_i = r_c - c - p_i \quad (140)$$

where,

r_c = the interest rate earned on cash proceeds from short-selling this stock.

c = the cost of leverage, referring to the standard fees applying uniformly to all short stocks.

p_i = the hard-to-borrow penalty specific to this stock, referring to the additional borrowing cost incurred by small cap or illiquid stocks.

Explicitly modeling short rebates/short costs in Long-Short Optimization allows users to see how optimal portfolios vary with different short-rebate rates, or different components of the rate. This feature helps users fine tune their long-short investment strategies.

4.6.6 Parametric Optimization with Leverage Constraints

Parametric optimization (discussed in the Optimal Frontiers—Parametric Optimization section) is now supported in the presence of piece-wise linear leverage constraints. In other words, risk-return efficient frontiers and risk target or return target problems can be combined with the constraints (126)-(129) and (133) in long-short optimization. For Leverage-Utility frontiers, only varying the upper bound of a (weighted) long-leverage, the lower bound of a (weighted) short-leverage, or the upper bound of a (weighted) total-leverage constraint is allowed. See the Varying Leverage section for more details.

4.7 Solving a General Linear or Convex Quadratic Program

A general Convex Quadratic Programming (CQP) problem has the following mathematical form:

$$\underset{\mathbf{x}}{\text{Maximize}}: \quad \mathbf{c}^{\mathrm{T}}\mathbf{x} - \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} \quad (141)$$

$$\text{Subject to}: \quad \mathbf{A}\mathbf{x} \le \mathbf{b} \quad (142)$$

$$\mathbf{l}_{\mathbf{x}} \le \mathbf{x} \le \mathbf{u}_{\mathbf{x}} \quad (143)$$

where, $\mathbf{c}$, $\mathbf{x}$, $\mathbf{l}_{\mathbf{x}}$, $\mathbf{u}_{\mathbf{x}}$ are $n \times 1$ vectors, $\mathbf{Q}$ is an $n \times n$ semi-positive definite matrix, $\mathbf{A}$ is an $m \times n$ coefficient matrix, and $\mathbf{b}$ is an $m \times 1$ vector. If $\mathbf{Q} = \mathbf{0}$, the problem reduces to a Linear Programming (LP) problem.

Barra Optimizer can be used to solve a generic LP or CQP problem by transforming it into the format of (1)-(12) with $\mathbf{h} = \mathbf{x}$, $\boldsymbol{\alpha} = \mathbf{c}$, $\mathbf{G} = \mathbf{Q}$, and so on.

4.8 Tax-Aware Optimization

Barra Optimizer's Tax-Aware Optimization extends the features and functions of Standard Optimization by taking the capital gain taxes into consideration. More specifically, the Tax-Aware Optimization allows users to:

- Penalize the realization of any capital gain tax in the objective.
- Place bounds on the gross gains or losses, net gains or losses, as well as the total capital gain tax incurred.
- Target net gains or losses and reap benefits from tax harvesting.
- Account for, disallow, or ignore wash sales.
- Examine the summary gains/losses/tax information and detailed trading information for each tax lot in the output.

Threshold and cardinality constraints are allowed in Tax-Aware Optimization (more details are in the Cardinality and Threshold Constraints in Tax-Aware Optimization section). Tax-utility frontiers (defined in the Constraint-Utility Frontiers section) for Tax-Aware Optimization are also offered.

In general, a Tax-Aware Optimization problem has the following form:

$$\underset{\mathbf{h}}{\text{Maximize:}} \quad \mathbf{r}^{\mathrm{T}}\mathbf{h} - 100(\mathbf{h}-\mathbf{h}_B)^{\mathrm{T}}(\lambda_D\mathbf{D}+\lambda_F\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h}-\mathbf{h}_B) - \lambda_{Tc}\,Tc(\mathbf{h},\mathbf{h}_0) - g(\mathbf{h}) - \lambda_{Tax}\,Tax(\mathbf{h}) \qquad (144)$$

$$\text{Subject to:} \quad Tax-Related\ Constraints$$

$$\text{Other Constraints}$$

where,

- $Tax(\mathbf{h})$ = total tax cost (as a fraction of the portfolio base value) triggered by capital gains from trading. It is defined more precisely by the additional constraints in the Tax Rates or Wash Sales sections.
- λ_{Tax} = multiplier specifying the importance of the tax cost relative to other terms in the objective. It is a scaling factor under the user's control.

4.8.1 Tax Rates

Depending on whether long-term and short-term capital gains are treated differently, the Tax-Aware Optimization problems can be classified into two categories—one-rate or two-rate.

4.8.1.1 One-Rate

No distinction between short-term and long-term capital gains is made here. Only one capital gains tax rate is considered. To construct such a problem, the following additional constraints are added to the Standard Optimization:

$$gg = \sum_{i=1}^{n} \sum_{l \in \{l:G_{il} \geq 0\}}^{L_i} G_{il}\mathbf{h}_{il}^{-} \qquad \text{(Gross Gain Definition)} \qquad (145)$$

$$gl = \sum_{i=1}^{n} \sum_{l \in \{l: G_{il} < 0\}}^{L_i} G_{il} \mathbf{h}_{il}^{-} \quad \text{(Gross Loss Definition)} \quad (146)$$

$$ng = gg + gl - LC \quad \text{(Net Gain Definition)} \quad (147)$$

$$Tax = tr \cdot \max(ng, 0) \quad \text{(Total Tax Cost Definition)} \quad (148)$$

$$l_{gg} \le gg \le u_{gg} \quad \text{(Tax Arbitrage Constraint on Gross Gain)} \quad (149)$$

$$l_{gl} \le gl \le u_{gl} \quad \text{(Tax Arbitrage Constraint on Gross Loss)} \quad (150)$$

$$l_{ng} \le ng \le u_{ng} \quad \text{(Tax Arbitrage Constraint on Net Gain)} \quad (151)$$

$$Tax \le u_{tax} \quad \text{(Total Tax Cost Constraint)} \quad (152)$$

where,

L_i = total number of tax lots of asset i

$\mathbf{h}_{il}^{-}$ = sales amount of tax lot l of asset i

G_{il} = capital gain or loss per dollar sold of tax lot l of asset i = $(Price_i - CostBasis_{il}) / Price_i$

gg = gross capital gains

gl = gross capital losses

ng = net capital gains

LC = loss carryover (a non-negative constant)

tr = capital gains tax rate

l_{gg}, u_{gg} = lower and upper bounds on gross gains (non-negative constants)

l_{gl}, u_{gl} = lower and upper bounds on gross losses (non-negative constants)

l_{ng}, u_{ng} = lower and upper bounds on net gains (constants that can be negative)

u_{tax} = upper bound on total tax cost (a non-negative constant)

4.8.1.2 Two-Rate

Short-term and long-term capital gains are treated differently with two tax rates here. To construct such a problem, the following additional constraints are added to the Standard Optimization:

$$lgg = \sum_{i=1}^{n} \sum_{l \in \{l: G_{il} \ge 0, A_{il} > 365\}}^{L_i} G_{il} \mathbf{h}_{il}^{-} \quad \text{(Long-Term Gross Gain Definition)} \quad (153)$$

$$lgl = \sum_{i=1}^{n} \sum_{l \in \{l: G_{il} < 0, A_{il} > 365\}}^{L_i} G_{il} \mathbf{h}_{il}^{-} \quad \text{(Long-Term Gross Loss Definition)} \quad (154)$$

$$sgg = \sum_{i=1}^{n} \sum_{l \in \{l: G_{il} \geq 0, A_{il} \leq 365\}}^{L_i} G_{il} \mathbf{h}_{il}^{-}$$ (Short-Term Gross Gain Definition) (155)

$$sgl = \sum_{i=1}^{n} \sum_{l \in \{l: G_{il} < 0, A_{il} \leq 365\}}^{L_i} G_{il} \mathbf{h}_{il}^{-}$$ (Short-Term Gross Loss Definition) (156)

$$lng = lgg + lgl - LLC$$ (Long-Term Net Gain Definition) (157)

$$sng = sgg + sgl - SLC$$ (Short-Term Net Gain Definition) (158)

$$lcng = \max(lng, 0) + \delta \min(0, sng)$$ (Long-Term Cross Net Gain Definition) (159)

$$scng = \max(sng, 0) + \delta \min(0, \text{lng})$$ (Short-Term Cross Net Gain Definition) (160)

$$Tax = ltr \cdot \max(lcng, 0) + str \cdot \max(scng, 0)$$ (Total Tax Cost Definition) (161)

$$l_{lgg} \leq lgg \leq u_{lgg}$$ (Tax Arbitrage Constraint on Long-Term Gross Gain) (162)

$$l_{lgl} \leq lgl \leq u_{lgl}$$ (Tax Arbitrage Constraint on Long-Term Gross Loss) (163)

$$l_{sgg} \leq sgg \leq u_{sgg}$$ (Tax Arbitrage Constraint on Short-Term Gross Gain) (164)

$$l_{sgl} \leq sgl \leq u_{sgl}$$ (Tax Arbitrage Constraint on Short-Term Gross Loss) (165)

$$l_{lng} \leq lng \leq u_{lng}$$ (Tax Arbitrage Constraint on Long-Term Net Gain) (166)

$$l_{sng} \leq sng \leq u_{sng}$$ (Tax Arbitrage Constraint on Short-Term Net Gain) (167)

$$Tax \leq u_{tax}$$ (Total Tax Cost Constraint) (168)

where,

A_{il}	=	age (in terms of days away from the analysis date) of tax lot l of asset i
lgg	=	long-term gross capital gains
lgl	=	long-term gross capital losses
sgg	=	short-term gross capital gains
sgl	=	short-term gross capital losses
lng	=	long-term net capital gains
sng	=	short-term net capital gains
$lcng$	=	long-term cross net capital gains
$scng$	=	short-term cross net capital gains
LLC	=	long-term loss carryover (a non-negative constant)
SLC	=	short-term loss carryover (a non-negative constant)
ltr	=	long-term capital gain tax rate

str	=	short-term capital gain tax rate
δ	=	$\begin{cases} 1, & \text{netting between long-term and short-term capital gains and losses is allowed} \\ 0, & \text{otherwise} \end{cases}$
l_{lgg}, u_{lgg}	=	lower and upper bounds on long-term gross gains (non-negative constants)
l_{lgl}, u_{lgl}	=	lower and upper bounds on long-term gross losses (non-negative constants)
l_{sgg}, u_{sgg}	=	lower and upper bounds on short-term gross gains (non-negative constants)
l_{sgl}, u_{sgl}	=	lower and upper bounds on short-term gross losses (non-negative constants)
l_{lng}, u_{lng}	=	lower and upper bounds on short-term net gains (constants that can be negative)
l_{sng}, u_{sng}	=	lower and upper bounds on short-term net gains (constants that can be negative)

4.8.2 Netting Different Types of Gains and Losses

Barra Optimizer offers two variants of the Two-Rate Tax-Aware Optimization problem, each tailored to a particular investment situation.

The first is characteristic of a limited partnership or a mutual fund. Here, it is not appropriate to focus on the total net tax of a portfolio, either because the portfolio will be combined with others before computing the tax, or because the gains or losses will be passed through to the investors.

The second is more characteristic of an individual investor managing all his taxable (and possibly non-taxable) capital assets as a single portfolio. Here, the focus is on the total net tax of the portfolio. There may also be institutional portfolios for which this is the appropriate model.

The difference in how Barra Optimizer models these situations is straightforward. In the first case, no attempt is made to net short-term losses against long-term gains or vice-versa. In the second case, netting between different types of gains and losses takes place. In terms of constraints (159) and (160), the first situation corresponds to $\delta = 0$, whereas the second corresponds to $\delta = 1$. Barra Optimizer sets the first case as the default, and makes the second case an option.

4.8.3 Tax Arbitrage

Tax arbitrage refers to the trading activities that take advantage of a difference in tax rates or tax systems as the basis for profit. Barra Optimizer's Tax-Aware Optimization offers users numerous ways to explore their tax arbitrage strategies. In particular, users can set both upper and lower bounds on any of the gross gains, gross losses, and net gains amount, either long-term or short-term.

4.8.4 Tax Harvesting

Tax harvesting usually refers to the practice of selling stocks that have lost money to offset a capital-gains tax liability. To facilitate tax-harvesting strategies, Barra Optimizer allows users to set a specific target on the amount

of net capital gains, either short-term or long-term. A penalty function of the form (84),(85), or (86) is then used to penalize any deviations away from the target.

For example, users may target the net short-term capital gains at $1 million in a Two-Rate Tax-Aware Optimization, and use a quadratic penalty function of the form (84) with the penalty value ρ_q set at 100. In contrast to the usual practice of letting users specify the Upper and Lower parameters of the penalty function, here the users are required to supply the values of the penalties— ρ_q, ρ_{up} or ρ_{down} —explicitly. For information about the penalty functions, see the Constraint Flexibility—Penalties, Soft Bounds, and Hierarchy section.

Many investors desire to take full advantage of their annual capital loss deduction allowance and harvest the resulting tax benefits. In such cases, they could explicitly set the net capital loss target at the allowance level. In the US, the annual allowance is typically $3000. Thus, the tax-harvesting problem is a special case of the net capital gains target problem.

4.8.5 Tax Lot Trading Rules

Under the default AUTO setting, Barra Optimizer automatically determines the order in which tax lots are traded according to the marginal contribution of each tax lot to the objective. The resulting trades usually obey the HIFO rule, but may not be strictly so.

Users may override the AUTO setting with the following trading rules:

- FIFO—Tax lots bought earlier are traded before those bought later.
- LIFO—Tax lots bought later are traded before those bought earlier.
- HIFO—Tax lots with higher cost basis are traded before those with lower cost basis.

4.8.6 Wash Sales

According to the Internal Revenue Service Publication 550 (2011), “A wash sale occurs when you sell or trade stock or securities at a loss and, within 30 days before or after the sale, you buy substantially identical stock or securities…” Your loss is disallowed for the income tax purpose because of the wash sale rules. However, you can add the disallowed loss to the cost basis of the substantially identical stock or securities you acquire. Your holding period for these substantially identical stock or securities includes the period you held the old stock or securities.

The details of the wash sale rules are rather complicated. Barra Optimizer offers three options in handling wash sales, as described in the following sections.

4.8.6.1 Ignore Wash Sales

With this option, all wash sales are ignored, even if they did occur. Rules pertaining to wash sales are not enforced or accounted for.

4.8.6.2 Trade-Off Wash Sales

Here, wash sales are allowed to take place, and their tax consequences and their ultimate effect on the objective are modeled internally, according to the IRS rules. Barra Optimizer employs a number of heuristic procedures to solve such problems.

4.8.6.3 Disallow Wash Sales

Here, wash sales are not allowed to occur. In particular, this rule implies the following:

- If no sales or purchases of an asset occurred within the last 30 days, then there is no restriction on the purchase or sale of this asset.
- If an asset was sold for a loss at least once within the last 30 days, and if that loss had not been disallowed previously, then the user would be prohibited from purchasing any shares of this asset.
- If two or more lots of an asset have been purchased within the last 30 days, then the user may only sell those tax lots of the same asset with unrealized gains but none with unrealized losses.
- If exactly one lot of an asset has been purchased within the last 30 days, then the user would be free to sell this tax lot (even at a loss) and any tax lots of the same asset with unrealized gains. Furthermore, after selling this one lot (resulting in either a gain or loss), there would be no restriction on the sales of any lots with losses.

In the last scenario above, Barra Optimizer may adopt a cherry-picking strategy—to pick that one lot and sell it first—to be free to sell other tax lots with unrealized losses and improve the objective value.

4.8.7 Capital Gain Tax

Barra Optimizer allows users to penalize the realization of capital gain taxes. In such a case, the objective function becomes:

$$\underset{\mathbf{h}}{\text{Maximize}}: \quad f(\mathbf{h}) - \lambda_{Tax} Tax(\mathbf{h}) \tag{169}$$

where, $f(\mathbf{h})$ is the simple, usual objective function; $Tax(\mathbf{h})$ is the total capital gain tax defined by multiple auxiliary constraints in the Tax Rates section; and λ_{Tax} is a multiplier that may be construed as a linear penalty on the capital gains tax.

We often refer to this type of problems involving capital gains and taxes as the Tax-Aware Optimization. For more information, see the Tax-Aware Optimization section.

4.8.8 Overlap Constraint—Control of Total Active Benchmark Holdings

For various reasons, a portfolio manager may want to control the sum of the benchmark weights corresponding to his non-zero portfolio weights. We call this an "overlap" constraint. Mathematically, it can be represented as:

$$\sum_{i \in J} h_{B_i} \le U_{overlap}, \qquad J = \{ j \mid h_i \ne 0.0 \} \tag{170}$$

OR

$$\sum_{i} \delta_i \cdot h_{B_i} \le U_{overlap}, \qquad \delta_i = \begin{cases} 1, & \text{if } h_i \ne 0.0 \\ 0, & \text{otherwise} \end{cases} \tag{171}$$

where,

h_i and h_{B_i} are the optimal portfolio weight and benchmark weight, respectively, for asset i. $U_{overlap}$ is an upper bound for the overlap.

The overlap constraint is a hard-to-solve, non-convex, discrete constraint similar to the cardinality constraints. The Barra Optimizer employs heuristics to handle it.

4.8.9 Non-Convexity

In general, tax-aware optimization problems are non-convex and solutions are heuristic as well as local in nature. Global optimality is not guaranteed.

4.8.10 Cardinality and Threshold Constraints in Tax-Aware Optimization

With a few exceptions, such as paring by groups, asset-level thresholds, and grandfather rule for level paring, the majority types of cardinality and threshold constraints that are available in standard optimization or long-short optimization are also allowed in tax-aware optimization. More specifically, Barra Optimizer now supports the following constraints in tax-aware optimization:

- Maximum/minimum number of names
- Maximum/minimum number of trades
- Maximum/minimum number of buy- or sell-side trades
- Minimum long- or short-side holding threshold
- Minimum long- or short-side trade threshold

4.8.11 Tax-Aware Risk Target Optimization

Tax-Aware Risk Target Optimization allows users to set a specific target on the portfolio risk level while taking into account the impact of capital gain taxes. Mathematically, the problem can be represented as:

$$\text{Maximize:} \quad \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Transaction Cost} - \text{Tax Cost} \tag{172}$$

$$\text{Subject to:} \quad \sigma \le \sigma_{Target} \tag{173}$$

$$\text{Other Constraints}$$

where, σ is the standard deviation of the portfolio return or active return, and "Other Constraints" includes standard and tax-related constraints.

Note that the "trade-off wash sales" option is currently not supported with this feature.

4.9 5-10-40 Rule

The 5/10/40 Rule is a shorthand term for the European Community Directive on Undertakings for Collective Investment in Transferable Securities (UCITS). It stems from a German law, but similar regulations exist in France, Switzerland, and Austria. It requires that a balanced portfolio satisfy the following conditions:

- The maximum weight of securities of a single issuer cannot exceed 10% of the portfolio value; and

- The sum of the weights of all issuers representing more than 5% of the portfolio value cannot exceed 40%.

A more flexible definition of the rule is implemented in Barra Optimizer as follows:

- The maximum weight of securities of a single issuer cannot exceed Q% of the portfolio value; and
- The sum of the weights of all issuers representing more than P% of the portfolio value cannot exceed R%.

This general rule is referred to as the "P/Q/R" rule, and the default values are P = 5, Q = 10, and R = 40. It is important to note that Barra Optimizer treats composites separately from other types of assets. A composite is considered its own issuer. The component weights of a composite are not combined with the non-composite asset weights of the same issuer. For example, assume that a portfolio consists of only three equally weighted assets—X, Y, and Z. Assets X and Y are regular assets, both issued by company A. Asset Z is a composite comprised of 30% X and 70% Y. Then, from the perspective of Barra Optimizer, this portfolio has two issuers—A and Z. Issuer A accounts for two-thirds of the portfolio weight, and Z one-third.

Portfolio optimization problems with 5-10-40 rules are discrete and not convex in general. Barra Optimizer employs branch and bound, or heuristic algorithms to tackle the 5/10/40 rules. The solution portfolios, particularly for large-sized problems, are not globally optimal in general.

4.10 Maximizing the Sharpe or Information Ratio

4.10.1 Maximizing the Sharpe Ratio

The Sharpe Ratio (SR) is defined as the excess return of a portfolio divided by its total risk. This can be expressed mathematically as:

$$\underset{\mathbf{h}}{\text{Maximize}} \quad \frac{\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - r_f - \lambda_{Tc}\, Tc(\mathbf{h})}{\sqrt{\mathbf{h}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{h}}} \tag{174}$$

where r_f is the risk-free interest rate, $\boldsymbol{\Sigma}$ is defined in (26), and the other variables are defined in (1).

4.10.2 Maximizing the Information Ratio

The Information Ratio (IR) is defined as the active return of a portfolio with respect to its benchmark divided by its tracking error. It can be expressed mathematically as:

$$\underset{\mathbf{h}}{\text{Maximize}} \quad \frac{\boldsymbol{\alpha}^{\mathrm{T}}(\mathbf{h} - \mathbf{h}_B) - \lambda_{Tc}\, Tc(\mathbf{h})}{\sqrt{(\mathbf{h} - \mathbf{h}_B)^{\mathrm{T}}\boldsymbol{\Sigma}(\mathbf{h} - \mathbf{h}_B)}} \tag{175}$$

For more information, see Kopman & Liu (2009).

4.11 Optimal Frontiers—Parametric Optimization

Standard Optimization, Paring Problems, Risk Constrained Optimization, Long-Short Optimization, and Tax-Aware Optimization can all be regarded as single portfolio optimization because the optimal solution for any one of them is a single portfolio. In contrast, Parametric Optimization consists of sequences of single portfolio optimization. Its output usually contains multiple portfolios.

Generally, Parametric Optimization treats a certain component in a single portfolio optimization problem as a parameter. The optimal solution is a function of the parameter chosen. As the parameter takes on different values, so does the optimal solution. The trajectory of the optimal solution forms a frontier, each point on which indicates the best possible portfolio given a value of the parameter.

Barra Optimizer offers two broad categories of Parametric Optimization—the Risk-Return Efficient Frontiers and the Constraint-Objective Frontiers.

4.11.1 Risk-Return Efficient Frontiers

A Risk-Return Efficient Portfolio has the smallest level of risk among all portfolios with the same level of expected return. It is also the portfolio having the highest expected return among all portfolios with the same level of risk. Return here refers to alpha minus any transaction costs present in the objective function. The (risk-return) efficient frontier is a graph of the risk-return pairs for all efficient portfolios.

Either return or risk may be varied. Assume:

- r is the portfolio net return
- σ is the standard deviation of the portfolio return or active return.

Mathematically, the two basic types of risk-return efficient frontier problems are:

4.11.1.1 Varying Return

$$\begin{aligned} &\text{Minimize:} && \sigma^2 + \text{Penalties} \\ &\text{Subject to:} && \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Transaction Costs} = r\ ,\ r \in [r_{min}, r_{max}] \\ & && \text{Other constraints} \end{aligned} \tag{176}$$

4.11.1.2 Varying Risk

$$\begin{aligned} &\text{Maximize:} && \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Transaction Costs} - \text{ Penalties} \\ &\text{Subject to:} && \sigma \in [\sigma_{min}, \sigma_{max}] \\ & && \text{Other Constraints} \end{aligned} \tag{177}$$

The Risk Target Problem in the Risk Target Optimization section is a special case of the Varying Risk type of Risk-Return frontier problem where the risk range is reduced to a fixed point.

Thus, the variance term could be for either total risk or tracking error. Net return may incorporate penalties when needed. Users may specify a section of the frontier to examine by entering either the return range (r_{min}, r_{max}) or the risk range $(\sigma_{min}, \sigma_{max})$. If no user input is given, a broad section of the frontier will be provided in the output by default.

Barra Optimizer returns the following information about selected efficient portfolios along the frontier:

- Objective function value
- Net return (%), i.e., $\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h} - \text{Transaction Costs}$

- Predicted portfolio risk (%)
- Portfolio turnover (%)
- Implied risk aversion

The implied risk aversion is the risk aversion value that, if applied to a single portfolio optimization with an equivalent problem setting, would result in an optimal portfolio with the specified predicted risk[5]. The reported objective is computed by using the implied risk aversion.

4.11.2 Constraint-Utility Frontiers

The Constraint-Utility Frontier, (or in other words, the Constraint-Objective Frontier), is a graph of the objective levels of an optimal portfolio against different values of a constraint bound. It illustrates the effect of varying the bound of a constraint on the optimal objective value.

For example, how would the objective value change when the turnover limit is increased from 5% to 30%? What would happen to the objective value when the portfolio's exposure to the "growth" factor varies from -1 to 1 relative to the benchmark? The Turnover-Utility Frontier, Tax-Utility Frontier, and Leverage-Utility Frontier are special cases of the Constraint-Utility Frontier.

To use the Constraint-Utility Frontier feature, users need to indicate which constraint is to vary (i.e., whether it is the turnover constraint limit constraint) and over what range. The constraint bound could be an upper bound, a lower bound, or equality bound. For piecewise linear constraints, however, only the upper bound is permitted to vary.

For a given number of points on the efficient frontier, $\mathrm{N}_{\text{points}}$, Barra Optimizer picks $\mathrm{N}_{\text{points}}$ values of variable t in the user-specified range [lower, upper] and solves a series of optimization problems as shown below:

4.11.2.1 Varying Turnover

$$\begin{aligned} \text{Maximize:} \quad & f(\mathbf{h}) \\ \text{Subject to:} \quad & TO(\mathbf{h}) \le t \\ & Other\,Constraints \end{aligned} \tag{178}$$

4.11.2.2 Varying Tax Cost

$$\begin{aligned} \text{Maximize:} \quad & f(\mathbf{h}) \\ \text{Subject to:} \quad & Tax(\mathbf{h}) \le t \\ & Other\,Constraints \end{aligned} \tag{179}$$

[5] If the problem is not convex, or it has a penalty or short rebate cost term in the objective function, or soft bound is involved, then this may not be true.

4.11.2.3 Varying Leverage

$$\begin{aligned} \text{Maximize:} \quad & f(\mathbf{h}) \\ \text{Subject to:} \quad & \begin{cases} L_{LLev} \le LongLeverage(\mathbf{h}) \le t & or \\ t \le ShortLeverage(\mathbf{h}) \le U_{SLev} & or \\ L_{TLev} \le TotalLeverage(\mathbf{h}) \le t \end{cases} \\ & Other\ Constraints \end{aligned} \tag{180}$$

Note that the leverage constraint in (180) is only limited to the types described in sections 4.6.1.1 -4.6.1.3 and section 4.6.1.6. In other words, only varying the upper bound of a (weighted) long-leverage, the lower bound of a (weighted) short-leverage, or the upper bound of a (weighted) total-leverage constraint is allowed. No ratio-type leverage constraints are supported in leverage-utility frontiers.

Similar to the output for Risk-Return Frontiers, Barra Optimizer returns the following information about selected optimal portfolios along a Constraint-Utility Frontier:

- Objective function value
- Net return (%), i.e., $\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{h}$ – Transaction Costs
- Predicted portfolio risk (%)
- Portfolio's exposure to the constraint

4.12 Multiple-Period Optimization

As its name suggests, Multiple-Period Optimization takes into consideration the input information (i.e., return, risk, transaction cost, etc.) over a number of successive periods and produces a set of optimal portfolios for each of these periods. This is in contrast to the traditional single portfolio optimization, which only considers the input information for a single period and produces one optimal portfolio for that period. It also differs from Parametric Optimization, which solves a sequence of single portfolio optimization problems, all for a single period. Multiple-Period Optimization solves one single portfolio optimization problem combining multiple periods.

Mathematically, a mean-variance multiple-period optimization can be formulated as follows:

$$\begin{aligned} \underset{\mathbf{h}^1,\mathbf{h}^2,\cdots,\mathbf{h}^{N_t}}{\text{Maximize:}} \quad f(\mathbf{h}^1,\mathbf{h}^2,\cdots,\mathbf{h}^{N_t}) = & \sum_{t=1}^{N_t} \lambda_r^t (\mathbf{r}^t)^{\mathrm{T}} \mathbf{h}^t - 100 \sum_{t=1}^{N_t} \lambda^t \sigma^2(\mathbf{h}^t, \mathbf{h}_B) - \sum_{t=1}^{N_t} \lambda_{Tc}^t Tc(\mathbf{h}^t, \mathbf{h}_0^t) \\ & - \sum_{t=1}^{N_t} \lambda_{Pen}^t Pen(\mathbf{s}^t, \mathbf{w}^t) \end{aligned} \tag{181}$$

Subject to:

$$\mathbf{e}^T\mathbf{h}^t = 1 \qquad \text{Portfolio Holding Constraints} \tag{182}$$

$$\mathbf{l}_h^t \le \mathbf{h}^t \le \mathbf{u}_h^t \qquad \text{Asset ranges} \tag{183}$$

$\mathbf{s}^t = \mathbf{A}^t\mathbf{h}^t$	Definitional Constraints for General Slack Variables	(184)
$\mathbf{l}_s^t \le \mathbf{s}^t \le \mathbf{u}_s^t$	Ranges for Constraint Slacks	(185)
$\mathbf{w}^t = (\mathbf{X}^t)^{\mathrm{T}}\mathbf{h}^t$	Definitional Constraints for Factor Slack Variables	(186)
$\mathbf{l}_X^t \le \mathbf{w}^t \le \mathbf{u}_X^t$	Factor Exposure Ranges	(187)
$To\left(\mathbf{h}^t, \mathbf{h}_0^t\right) \le u_{To}^t$	Turnover Constraint for Period t	(188)
$\sum_{t=1}^{N_t} To\left(\mathbf{h}^t, \mathbf{h}_0^t\right) \le u_{To}$	Total (Cross-Period) Turnover Constraint	(189)
$Tc\left(\mathbf{h}^t, \mathbf{h}_0^t\right) \le u_{Tc}^t$	Transaction Cost Constraint for Period t	(190)
$\sum_{t=1}^{N_t} Tc\left(\mathbf{h}^t, \mathbf{h}_0^t\right) \le u_{Tc}$	Total (Cross-Period) Transaction Cost Constraint	(191)

for all periods $t = 1, 2, \cdots, N_t$, where N_t is the total number of periods. All variables, terms, and constraints above are essentially the same as defined in Section 3.1.1 except that they are now period-specific.

Notes:

- The total number of periods is at most 5. i.e. $N_t \le 5$.
- The only cross-period constraints allowed are the turnover and transaction cost limit constraints.
- The objective terms and constraints (except the cross-period ones) for each period are separable. The objective for each period is a convex quadratic function. The standard constraints for each period are either linear or convex piecewise linear.
- No discrete constraints or variables are allowed except cardinality and threshold constraints with limited scope. More specifically, holding cardinality and minimum holding threshold constraints are supported for all periods, but trade cardinality and minimum trade threshold constraints are only supported for the first period.
- No nonlinear constraints are allowed.
- Users specify portfolio base value for the first period and cash flow weights for each period. This base value should equal to the initial portfolio value plus cash flow value for the first period. Barra Optimizer adjusts the portfolio base values and asset initial weights for subsequent periods internally according to the cash flow weights provided. Asset initial weights for each period sum to 1.
- Users are responsible for providing alpha and transaction cost information for all periods.
- Only one risk model is allowed in each period. The same risk model is used for all periods.
- At most one benchmark is allowed in each period.
- Overall as well as buy- and sell-side turnover are supported in per-period turnover constraints. However, buy- and sell-side cross-period turnover constraints are not supported.

For more information about multiple-period optimization, see Liu and Xu (2017).

4.13 Multiple-Account Optimization

Multiple-Account Optimization (or Multiple-Portfolio Optimization) simultaneously optimizes a number of accounts based on their individual account information as well as their joint market-impact transaction costs or any cross-account constraints.

Mathematically, a mean-variance Multiple-Account Optimization that maximizes the total-welfare of all accounts can be formulated as follows:

$$\underset{\mathbf{h}^1,\mathbf{h}^2,\cdots,\mathbf{h}^{N_p}}{\text{Maximize:}} \quad f(\mathbf{h}^1,\mathbf{h}^2,\cdots,\mathbf{h}^{N_p}) = \sum_{p=1}^{N_p}\lambda_r^p(\mathbf{r}^p)^{\mathrm{T}}\mathbf{h}^p - 100\sum_{p=1}^{N_p}\lambda^p\sigma^2(\mathbf{h}^p,\mathbf{h}_B^p) - \sum_{p=1}^{N_p}\lambda_{Tc}^p Tc(\mathbf{h}^p,\mathbf{h}_0^p) - \sum_{p=1}^{N_p}\lambda_{Pen}^p Pen(\mathbf{s}^p,\mathbf{w}^p) - \lambda_{MTC}MTC(\mathbf{h}^1-\mathbf{h}_0^1,\mathbf{h}^2-\mathbf{h}_0^2,\cdots,\mathbf{h}^{N_p}-\mathbf{h}_0^{N_p}) \tag{192}$$

Subject to:

$\mathbf{e}^T\mathbf{h}^p = 1$	Portfolio Holding Constraints	(193)
$\mathbf{l}_h^p \le \mathbf{h}^p \le \mathbf{u}_h^p$	Asset ranges	(194)
$\mathbf{s}^p = \mathbf{A}^p\mathbf{h}^p$	Definitional Constraints for General Slack Variables	(195)
$\mathbf{l}_s^p \le \mathbf{s}^p \le \mathbf{u}_s^p$	Ranges for Constraint Slacks	(196)
$\mathbf{w}^p = (\mathbf{X}^p)^{\mathrm{T}}\mathbf{h}^p$	Definitional Constraints for Factor Slack Variables	(197)
$\mathbf{l}_X^p \le \mathbf{w}^p \le \mathbf{u}_X^p$	Factor Exposure Ranges	(198)
$To\left(\mathbf{h}^p,\mathbf{h}_0^p\right) \le u_{To}^p$	Turnover Constraint for Account p	(199)
$Tc\left(\mathbf{h}^p,\mathbf{h}_0^p\right) \le u_{Tc}^p$	Transaction Cost Constraint for Account p	(200)
$\mathbf{l}_g \le \boldsymbol{g}(\mathbf{h}^1,\mathbf{h}^2,\cdots,\mathbf{h}^{N_p}) \le \mathbf{u}_g$	Cross-Account Constraints	(201)

for all accounts $p = 1, 2, \cdots, N_p$, where N_p is the total number of accounts or portfolios. What sets Multiple-Account Optimization apart from single-portfolio optimization is the extra joint market-impact term at the end of the objective function (192) and the cross-account constraints (201). We will discuss these two items in detail in the following sections. All other variables, terms, and constraints above are essentially the same as defined in the Problem Formulation section except that they are now account-specific.

4.13.1 Joint Market-Impact Transaction Costs

Different accounts may buy or sell the same asset at the same time. Depending on whether cross-account trading is allowed, the joint market-impact transaction cost term could be different. In most cases, especially with ERISA

compliance though, cross-account trading is not allowed. All buy and sell trades may need to be pooled together and trade in the open market.

The joint market-impact transaction cost function for aggregate buys and sells is currently limited to piecewise linear function. In other words, nonlinear joint transaction cost (power function) is not supported.

Let $x_i^{p+} \geq 0$ and $x_i^{p-} \geq 0$ be the buy and sell dollar amounts for asset i in account p, and $\mathbf{x}^p = \left(x_1^{p+} + x_1^{p-} \quad x_2^{p+} + x_2^{p-} \quad \cdots \quad x_n^{p+} + x_n^{p-}\right)$ be the trade amount vector for account p. Let $t_i^+ \geq 0$ and $t_i^- \geq 0$ be the total buy and sell dollar amounts for asset i for all accounts. Then, we have the following relationship:

$$\begin{cases} x_i^{p+} = P^p \max(h_i^p - h_{0i}^p, 0) \\ x_i^{p-} = P^p \max(h_{0i}^p - h_i^p, 0) \end{cases}, \qquad i = 1, 2, \cdots, n, \qquad p = 1, 2, \cdots, N_p \tag{202}$$

where P^p is the portfolio value for account p, and

$$\begin{cases} t_i^+ = \sum_{p=1}^{N_p} x_i^{p+}, \\ t_i^- = \sum_{p=1}^{N_p} x_i^{p-}, \end{cases} \qquad i = 1, 2, \cdots, n \tag{203}$$

Transaction costs are separable among assets. Hence, the joint market-impact transaction cost term can be further written as:

$$\begin{aligned} &MTC(\mathbf{h}^1 - \mathbf{h}_0^1, \mathbf{h}^2 - \mathbf{h}_0^2, \cdots, \mathbf{h}^{N_p} - \mathbf{h}_0^{N_p}) \\ &= MTC(\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{N_p}) = \sum_{i=1}^{n} MTC_i(t_i^+, t_i^-) = \sum_{i=1}^{n} \left[MTC_i(t_i^+) + MTC_i(t_i^-) \right] \end{aligned} \tag{204}$$

4.13.2 Cross-Account Constraints

For cross-account constraints, users will provide input in terms of dollars. The following three types of cross-account constraints are the only ones currently supported:

- Bounds on global position for an asset. For example, total positions for MSCI should be no more than $1b.
- Maximum total buy, sell, or trade size per asset over all accounts. For example, aggregate total buys for MSCI should be less than or equal to $500m, or the total trades for IBM should not be more than $280m.
- Cross-Account Turnover constraint

Note especially that cross-account transaction cost constraints are NOT currently supported.

4.13.3 Two Basic Approaches

We offer two approaches to Multiple-Account Optimization; both approaches are well established in financial research literature. For details, see Yang et al (2013) and O'Cinneide et al (2006).

4.13.3.1 Total-Welfare Maximization Approach

This approach considers the optimization problem (192) - (201) as one single large problem. The objective is to maximize the total-welfare (i.e., the joint utility function) of all accounts. The solution of this approach is called a Pareto optimal solution. Multiple optimal solutions may exist.

Although the solution of this approach maximizes the total welfare for all accounts, it may do so at the expense of some individual accounts. In other words, some accounts may be sacrificed more than others for the benefit of maximizing total welfare. For this reason, a Pareto solution may not be a fair solution to all accounts.

4.13.3.2 Nash Equilibrium (Game Theory) Approach

This approach decomposes the optimization problem (192)-(201) into successive account-level sub-problems, and solves them iteratively, one at a time, until the trade amounts do not change for each individual account. When solving the sub-problem for account p, we assume that the trades for other accounts are known already and treat them as merely constants. In other words, the objective function of the sub-problem for account p can be written as:

$$\begin{aligned}\underset{\mathbf{h}^p}{\text{Maximize:}}\quad f(\mathbf{h}^p) = \lambda_r^p(\mathbf{r}^p)^{\mathrm{T}}\mathbf{h}^p - 100\lambda^p\sigma^2(\mathbf{h}^p,\mathbf{h}_B^p) - \lambda_{Tc}^p Tc(\mathbf{h}^p,\mathbf{h}_0^p) \\ - \lambda_{Pen}^p Pen(\mathbf{s}^p,\mathbf{w}^p) - \lambda_{MTC} MTC_p(\mathbf{x}^p,\gamma^{p+},\gamma^{p-}),\end{aligned} \tag{205}$$

where $\mathbf{x}^p$ is the trade amount for account p, and both $\gamma^{p+} = \sum_{i=1}^{n}(t_i^+ - x_i^{p+})$ and $\gamma^{p-} = \sum_{i=1}^{n}(t_i^- - x_i^{p-})$ are constants representing the sum of buy or sell amounts for all other accounts.

When there are no cross-account constraints, Yang et al (2013) have shown that a unique Nash Equilibrium solution exists and can be obtained by simple iterative procedures. If cross-account constraints are present, more sophisticated methods, such as the Largrange Multipler and Sub-Gradient Algorithms, will be used.

4.13.4 Post-Optimization Transaction Cost Allocation

After solving the Multiple-Account Optimization problem, for various reasons, the joint market-impact transaction costs, if any, may need to be allocated back to each account. The following pro-rata allocation formula is used as the current default approach:

$$TC_p = \sum_{i=1}^{n} Buy_p(i)\,\frac{TC_{buy}(\mathrm{i})}{Total_{buy}(\mathrm{i})} + \sum_{i=1}^{n} Sell_p(i)\,\frac{TC_{sell}(\mathrm{i})}{Total_{sell}(\mathrm{i})} \tag{206}$$

where

- $TC_{buy}(\mathrm{i})$ and $TC_{sell}(\mathrm{i})$ are the total buy cost and total sell cost for asset i.
- $Total_{buy}(\mathrm{i})$ and $Total_{sell}(\mathrm{i})$ are the total buy and total sell amount for asset i for all accounts.
- $Buy_p(\mathrm{i})$ and $Sell_p(\mathrm{i})$ are the buy and sell amounts for asset i in account p.

This allocation method does not guarantee fairness. Users are free to apply any other allocation method of their choice because this procedure is post-optimization.

4.13.5 Scopes and Limitations

In this first release of Multiple-Account Optimization, we limit the scope as follows:

- The total number of accounts will be at most 20. i.e. $N_p \leq 20$.
- The objective terms and constraints (except the joint and cross-account ones) for each account are separable. The objective for each account is a convex quadratic function. The standard constraints for each account are either linear or convex piecewise linear. Most features for a standard single-portfolio optimization are supported at the individual account level. An exception to this is that constraint hierarchy will not be supported for Multiple-Account Optimization. Some other exceptions are given below.
- Users will be able to set transaction costs EITHER for each individual account (the third term in Eq.(192)) OR for the joint market-impact term (the last term in Eq.(192)), but not BOTH.
  In other words, when the joint market-impact term is present, users will NOT be able to set any transaction cost limit constraint for each individual account.
- When transaction cost terms are set for individual accounts, we assume that the same asset will have the same transaction cost function for all accounts. There is no limit on number of break points for an asset or for aggregate buys or sells.
- We assume cross-account trading is not allowed.
- Besides linear and piecewise linear constraints, no other type constraints are allowed for individual accounts with the exception of cardinality and threshold constraints. Penalty on cardinality and threshold constraints are not supported. The “EnableGrandfatherRule” and ““Allow Close-Out” options for paring apply to all accounts uniformly.
- Only one risk model is allowed in each account. The same risk model is used for all accounts. However, different accounts may have different benchmarks and risk aversions.
- Each account may have a different portfolio value. However, the portfolio base value for each account must be the same as its portfolio value.
- All constraints and objective function terms (except the break-points for transaction costs) that are for individual accounts only will be given in terms of weights. Joint transaction cost terms and cross-account constraints will be given in terms of dollar amounts.
- It is currently assumed that asset alphas are the same for all accounts.
- Overall turnover is supported as per-account as well as cross-account constraints.
- Universal maximum trade size per asset for all accounts; for example, no accounts can buy more than $10m MSCI; should be enforced by asset bounds rather than cross-account constraints for efficiency reasons.

5 Using the Barra Optimizer APIs

Barra Open Optimizer provides C++, Java, C#, and COM APIs to enable you to develop your applications using C++, Java, C#, Visual Basic, Excel VBA, MATLAB, Python, R, and SAS. To use the Barra Optimizer API for running a portfolio optimization, your application typically needs to perform the following steps:

- Create a workspace
- Add assets into the workspace and set per-asset data
- Construct initial portfolio, benchmark(s), and trade universe
- Define risk model(s)
- Prepare an optimization case, including setting up the objective, constraints, etc.
- Run optimization
- Inspect outputs
- Release resources

The following sections describe each step in detail for the C++, Java, and C# APIs. The steps for the COM API are similar except that the class and enumerator names are different.

For example:

- The class that represents a workspace is represented as:
  - C++, Java, and C# APIs — `CWorkSpace`
  - COM API — `OptWorkSpace`
- The enumerator type is represented as:
  - C++, Java, and C# APIs — `EAssetType`
  - COM API — `EOptAssetType`

We use <Class Name>::<Function Name> to reference a member function <Function Name> of a class <Class Name>. For example, `CWorkSpace::CreateInstance()` stands for the `CreateInstance()` function of the `CWorkSpace` class.

For applications using Java API, you need to load the native Barra Optimizer C++ library by calling `COptJavaInterface.Init()`.

5.1 Create a Workspace

The first step to run a portfolio optimization is to create an instance of the `CWorkSpace` object by calling `CWorkSpace::CreateInstance()`.

5.2 Add Assets into Workspace

To add assets into the `CWorkSpace` object, call `CWorkSpace::CreateAsset()`. Assets added into the workspace should at least cover those in the initial portfolio, the benchmark(s), and the trade universe.

For each of the added assets, call methods of `CAsset` to set per-asset data, such as alpha, price (optional for non-roundlotting cases), hard-to-borrow penalty, roundlot size, buy/sell costs, tax lots, transaction costs, residual alpha weights, whether excluded from risk parity, fixed holding cost, etc. To load per-asset data using a CSV-formatted file, do the following:

1. Create a CSV file with the first column as assetID, the second column as attribute type, and the third column as the value for the attribute type. The following attribute types are available:

Attribute Type	Description
FBTC	Fixed Buy Transaction Cost in decimal format
FSTC	Fixed Sell Transaction Cost in decimal format
P	Price
A	Expected return in decimal format
LS	Round-lot size
AW	Residual alpha weight
ISS	Issuer ID; used in 5-10-40 rule
NSC	Total net cost on short position
TYPE	Asset type 0 = Cash; 1 = Futures; 2 = Currency; 3 = Regular; 4 = Composite; 5 = Composite futures
EXRP	Whether to exclude from risk parity 0 = Include; 1 = Exclude
NLTC_C	Multiplier of the nonlinear transaction cost function
NLTC_P	The exponent of the nonlinear transaction cost function
NLTC_Q	The traded amount in the nonlinear transaction cost function
USFHC	Upside fixed holding cost
DSFHC	Downside fixed holding cost
RWFHC	Reference weight for fixed holding cost

In addition to the above attribute types, the second column may take a format of `GroupName.GroupAttribute` (with the ".GroupAttribute" part being optional) to specify the asset's

group association. In this case, the optional third column is the group coefficient. If the third column is absent, the group coefficient defaults to 1.0.

2. Call `CWorkSpace::LoadAssetData()` to load the data from the CSV file into `CWorkSpace`. The assets will be automatically added to `CWorkSpace` eliminating the need to call `CWorkSpace::CreateAsset()` for these assets.

Sample CSV file contents:

```
!AssetID,AttributeType,Value
USA11I1,A,0.00231
USA11I1,P,35.65
USA11I1,LS,1000
USA11I1,NSC,0.003193
USA11I1,FBTC,0.0005
USA13Y1,GICS_SECTOR.Information Technology,
USA1LI1,GICS_SECTOR.Information Technology,
USA3351,GICS_SECTOR.Utilities,0.5
```

5.3 Construct Initial Portfolio, Benchmark, and Trade Universe

To construct an initial portfolio, do the following:

1. Create a `CPortfolio` object by calling `CWorkSpace::CreatePortfolio()`.
2. For optimization problems that are not tax-aware, populate the portfolio object with asset IDs and weights by calling `CPortfolio::AddAsset()`.
3. For tax-aware optimization, populate the portfolio object with tax lot purchase information by calling `CPortfolio::AddTaxLot()`.

   Note: The optimizer will calculate asset weights from tax lot information, so do not call `CPortfolio::AddAsset()` to add asset weights. Add wash sale records by calling `CPortfolio::AddWashSaleRec()`.Such wash sale records have recent sales information and will be used by the optimizer to count wash sales against new purchases.

Benchmark portfolios and the trade universe can be constructed by calling `CPortfolio::AddAsset()`. Your application does not need to add weights for the trade universe.

5.4 Define Risk Model

To define a risk model, do the following:

1. Create a `CRiskModel` object by calling `CRiskModel::CreateRiskModel()`.
2. Call member functions of the `CRiskModel` object to populate the object with model data such as D, F, X, and so forth.

Notes:

- To construct another risk model, repeat the above steps.

- To specify a dense risk model without factors, set only the specific covariance.
- To save time for setting up risk model, you may skip setting 0 value covariance and exposure data in D, F, and X.
- For all composite-type assets, do the following:
  - Ensure that covariances and exposures among non-composite assets have been set.
  - Construct a CPortfolio object and populate the constituents and weights representing the composite.
  - Call `CAsset::SetCompositePort()` to compute factor exposures of the composite and covariances of the composite with other assets in the universe.
- Optionally, you may provide factor-blocking structure by calling `CRiskModel::AddFactorBlock()`. Barra Optimizer may improve performance by taking advantage of different scarcity of each factor block in the factor exposure matrix X.

To load ModelsDirect data into `CRiskModel`, do the following:

1. Unzip the ModelsDirect zip files into a root directory containing all the ModelsDirect flat files, or a subdirectory with same name as the model name. A BIMe model name shall always have a format of "BIMeXY" with "X" being series number and "Y" being the long/short specifier. For example, the model name of a long–term, 3 series BIMe model will be "BIMe3L")
2. For non-BIMe models, the following files should be extracted:
   - <Model>_<Version>_Asset_Data.YYYYMMDD**
   - <Model>_<Version>_Asset_Exposure.YYYYMMDD**
   - <Model>_<Version>_Asset_LSR.YYYYMMDD*
   - <Model>_<Version>_Covariance.YYYYMMDD
   - <Model>_Factors.dat*

   **Optional files*

   *** Not available with BIMe model.*

   For BIMe models, the following additional file should be extracted, which specifies the model names of sub-models that the ModelsDirect loader uses to retrieve exposure data:

   - <Model>_<Version>_Models.dat

   Sub-model zip files should be extracted the same way as non-BIMe models.
3. Call `CreateRiskModel()` with same model name as the ModelsDirect model name.
4. Call `LoadModelsDirectData()` with the following parameters:
   - Root model directory: Path where the flat files are located. The ModelsDirect loader looks for files under the root model directory, or in a subfolder with the same name as the model name. For example, if the root model directory for USE3L data is \ModelsDirectRepo, then the flat files can be extracted to \ModelsDirectRepo or \ModelsDirectRepo\USE3L.

- Analysis date: Date starting from which, flat files will be loaded. If a flat file with the exact date is not found, the loader goes back at most 45 calendar days before the analysis date.
- AssetIDs: List of the asset's BARRIDs for which the asset factor exposures and specific risks are to be loaded. The AssetIDs should contain all of the assets in the investible universe

5. For risk models having a currency block, call `SetNumeraire()` to change the numeraire of the factor covariance matrix. <Model>_Factors.dat is required as it contains factor block information.

To load asset exposures into `CRiskModel` using CSV file:

1. Create a CSV file with first column as asset ID, second column as factor ID, and third column as exposure value. For example:

```
! AssetID,FactorID,Exposure
USA11I1,Factor_1A,0.144
USA11I1,Factor_1B,-0.666
USA11I1,Factor_1C,1.646
USA11I1,Factor_1D,0.21
USA11I1,Factor_1E,-0.361
```

2. Call `LoadAssetExposures()` with the full path of the CSV file.
3. To compute the marginal contribution to tracking error or to active risk, do the following:
   - Call `ComputeAssetMCTE()` to compute the MCTE for a single asset.
   - Call `ComputePortAssetMCTE()` to compute the asset MCTE for all assets in the specified portfolio.
   - Call `ComputeFactorMCAR()` to compute the MCAR for a single factor.
   - Call `ComputePortFactorMCAR()` to compute the factor MCAR for all factors in the specified portfolio.
   - Asset-level MCTE computation is defined as following:

     $$Asset\ MCTE(i) = \frac{[(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D})(\mathbf{h} - \mathbf{h_B})\,]_{\mathrm{i}} - \mathbf{h_B}(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D})(\mathbf{h} - \mathbf{h_B})}{\sqrt{(\mathbf{h} - \mathbf{h_B})^{\mathrm{T}}(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D})(\mathbf{h} - \mathbf{h_B})}}$$

     where,
     - $\mathbf{X}$ = Asset factor exposures matrix
     - $\mathbf{F}$ = Factor covariance matrix
     - $\mathbf{D}$= Specific covariance matrix
     - $[\dots]_i$ = the i component of the vector

- Factor-level MCAR computation is defined as following:

$$Factor\ MCAR(j) = \frac{[\mathbf{FX^T(h - h_B)}\,]_j}{\sqrt{(\mathbf{h - h_B})^{\mathrm{T}}(\mathbf{XFX}^T + \mathbf{D})(\mathbf{h - h_B})}}$$

  where,

  - $\mathbf{X}$ = Asset factor exposures matrix
  - $\mathbf{F}$ = Factor covariance matrix
  - $[\ldots]_j$ = the j component of the vector

- Call `ComputeAssetPCTE()` to compute the percent contribution to tracking error for a single asset.
- Call `ComputePortAssetPCTE()` to compute the percent contributions to tracking error for all assets in the specified portfolio.
- Call `ComputeFactorPCAR()` to compute the percent contribution to active risk for a single factor.
- Call `ComputePortFactorPCAR()` to compute the percent contributions to active risk for all factors in the specified portfolio.
- Call `ComputeAssetBeta()` to compute the beta for a single asset.
- Call `ComputePortAssetBeta()` to compute the betas for all assets in the specified portfolio.
- Call `ComputePortCorrelation()` to compute the correlation between two portfolios.
- Call `ComputePortBeta()` to compute the beta for a portfolio.
- Asset-level PCTE computation is defined as following:

$$Asset\ PCTE(i) = \left(\mathbf{h}_i - \mathbf{h}_{\mathrm{B}_i}\right)\frac{\left[\left(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D}\right)\left(\mathbf{h} - \mathbf{h}_{\mathrm{B}}\right)\right]_i - \mathbf{h}_{\mathrm{B}}\left(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D}\right)\left(\mathbf{h} - \mathbf{h}_{\mathrm{B}}\right)}{\left(\mathbf{h} - \mathbf{h}_{\mathrm{B}}\right)^{\mathrm{T}}\left(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D}\right)\left(\mathbf{h} - \mathbf{h}_{\mathrm{B}}\right)} \cdot 100$$

  where,

  - $\mathbf{X}$ = Asset factor exposures matrix
  - $\mathbf{F}$ = Factor covariance matrix
  - $\mathbf{D}$= Specific covariance matrix
  - $[\ldots]_i$ = the i component of the vector

- Factor-level PCAR computation is defined as following:

$$Factor\ PCAR(j) = \left(\mathbf{X}_j - \mathbf{X}_{\mathrm{B}_j}\right)\frac{\left[\mathbf{F}\mathbf{X}^{\mathrm{T}}(\mathbf{h}-\mathbf{h}_{\mathrm{B}})\right]_j}{(\mathbf{h}-\mathbf{h}_{\mathrm{B}})^T\left(\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}}+\mathbf{D}\right)(\mathbf{h}-\mathbf{h}_{\mathrm{B}})}\cdot 100$$

  where,

  - $\mathbf{X}$ = Asset factor exposures matrix
  - $\mathbf{F}$ = Factor covariance matrix
  - $[\ldots]_j$ = the j component of the vector

- Asset beta computation is defined as following:

$$\beta = \frac{\left(\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}}+\mathbf{D}\right)\mathbf{h}_{\mathrm{B}}}{\mathbf{h}_{\mathbf{B}}\left(\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}}+\mathbf{D}\right)\mathbf{h}_{\mathrm{B}}}$$

where,

- $\mathbf{X}$ = Asset factor exposures matrix
- $\mathbf{F}$ = Factor covariance matrix
- $\mathbf{D}$= Specific covariance matrix

- Portfolio correlation computation is defined as following:

$$\rho(P,Q) = \frac{\mathbf{X}_{\mathrm{p}}\mathbf{F}\mathbf{X}_{\mathrm{q}} + \mathbf{h}_{\mathrm{p}}\mathbf{D}\mathbf{h}_{\mathrm{q}}}{\left(\sqrt{\mathbf{X}_{\mathrm{p}}\mathbf{F}\mathbf{X}_{\mathrm{p}} + \mathbf{h}_{\mathrm{p}}\mathbf{D}\mathbf{h}_{\mathrm{p}}}\right)\left(\sqrt{\mathbf{X}_{\mathrm{q}}\mathbf{F}\mathbf{X}_{\mathrm{q}} + \mathbf{h}_{\mathrm{q}}\mathbf{D}\mathbf{h}_{\mathrm{q}}}\right)}$$

  where,

  - $\mathbf{X}_{\mathrm{p}}$= P (or Q) portfolio factor exposures
  - $\mathbf{F}$ = Factor covariance matrix
  - $\mathbf{D}$= Specific covariance matrix

- Portfolio beta computation is defined as following:

$$\mathrm{b}_{\mathrm{p}} = \frac{\mathbf{X}_{\mathrm{p}}\mathbf{F}\mathbf{X}_{\mathbf{b}} + \mathbf{h}_{\mathbf{p}}\mathbf{D}\mathbf{h}_{\mathrm{b}}}{\mathbf{X}_{\mathbf{b}}\mathbf{F}\mathbf{X}_{\mathbf{b}} + \mathbf{h}_{\mathbf{b}}\mathbf{D}\mathbf{h}_{\mathbf{b}}}$$

  where,

  - $\mathbf{X}_{\mathrm{p}}$= P (or B) portfolio factor exposures
  - $\mathbf{F}$ = Factor covariance matrix
  - $\mathbf{D}$= Specific covariance matrix

5.5 Prepare Optimization Case

To prepare an optimization case, call `CWorkSpace::CreateCase()` to create a `CCase` object.

When calling the `CreateCase()` function, your application needs to specify initial portfolio, trade universe, base value (used to compute asset weights) and, optionally, cash flow weight. For long-short portfolios that result in net value of zero, set the base value to long side value of the portfolio for a 100% long portfolio.

Then, call methods of the `CCase` objects to do the following:

- Link risk models
- Set up the objective
- Set up constraints
- Set up efficient frontier
- Set up tax-aware optimization parameters
- Set a risk target or a return target.
- Define a non-linear transaction cost function at portfolio level.
- Test whether the optimization problem is convex by calling `CCase::IsConvex()`. For information about the convex portfolio optimization problem, see the Problem Classification section.
- Set the risk scalar for the return-risk frontier or risk-target optimization.

Note the following when preparing an optimization case:

- Unless specified otherwise, use decimals rather than percentages as units of input parameters.
- Use the `OPT_INF` constant when specifying an infinity value.

5.5.1 Link Risk Models

Barra Optimizer supports up to two risk models per optimization case.

- To link a primary risk model to the case, call `CCase::SetPrimaryRiskModel()`.
- Optionally, to link a secondary risk model to the case, call `CCase::SetSecondaryRiskModel()`.

5.5.2 Set Up the Objective

To set up the objective function, also known as the utility function for the optimization case, call `CCase::InitUtility()` to get a `CUtility` object.

When calling this function, your application can pass an appropriate parameter to specify the type of the objective function: standard quadratic, Information Ratio, or Sharpe Ratio. Then, call methods of the `CUtility` object to populate the object with the following parameters used in the objective function:

- Link a benchmark to the primary risk term and/or the optional secondary risk term

- Risk aversions λ_F and λ_D
- Risk aversions λ_{D_2} and λ_{F_2} if the objective has the variance term of the secondary risk model
- Multiplier for alpha λ_α
- Multiplier for tax costs λ_{TAX}
- Multiplier for transaction cost λ_{TC}
- Multiplier for short costs λ_{SC}
- Multiplier for penalties λ_{pen}
- Whether the penalty term is in the numerator or the denominator depends on whether the objective function is Information Ratio or Sharpe Ratio
- Multiplier for the residual alpha penalty term λ_R
- Risk free rate (a parameter used in Sharpe Ratio)
- Cost of leverage
- Additional covariance terms

5.5.3 Set Up Constraints

To set up constraints, call `CCase::InitConstraint()` to get a `CConstraints` object. Then, call Init methods (for example, `InitLinearConstraints()` of the `CConstraints` object to get an object of a specific type of constraint (for example, `CLinearConstraints`). Methods of such objects usually return a `CConstraintInfo` object, which in turn can be used for setting up the following:

- Values of lower bound and upper bound
- Soft bounds
- Penalties
- Free range linear penalties
- Whether the bounds are relative to a reference portfolio, for example, benchmark or initial portfolio

Two types of penalty functions are supported. To set up a penalty function specified in the Penalty Functions section, call `CConstraintInfo::SetPenalty()` to specify the parameters Target, Upper, and Lower. To set up a free-range linear penalty, call `CConstraintInfo::SetFreeRangeLinearPenalty()` to specify the lower and upper bound of the free range and slopes for each side outside the range.

Note that not all constraints can have lower bounds, soft bounds, penalties or relative bounds, as listed in the following table:

CConstraintInfo Object Returned By	Lower Bound	Upper Bound	Relativity	Softness	Penalty	Free Range Linear Penalty
CConstraints::SetTransactionCostConstraint		Y		Y		
CConstraints::SetCrossPeriodTransactionCostConstraint		Y		Y		
CHedgeConstraints::AddLongSideConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddLongSideFactorConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddLongSideGroupConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddShortSideConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddShortSideFactorConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddShortSideGroupConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::SetLongSideLeverageRange	Y	Y	Y	Y	Y	Y
CHedgeConstraints::SetShortSideLeverageRange	Y	Y	Y	Y	Y	Y
CHedgeConstraints::SetTotalLeverageRange	Y	Y		Y	Y	Y
CHedgeConstraints::AddWeightedTotalLeverageConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddTotalLeverageGroupConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::AddTotalLeverageFactorConstraint	Y	Y	Y	Y	Y	Y
CHedgeConstraints::SetShortLongLeverageRatioRange	Y	Y		Y		
CHedgeConstraints::SetNetTotalLeverageRatioRange	Y	Y		Y		
CLinearConstraints:: SetBetaConstraint	Y	Y		Y	Y	Y
CLinearConstraints::AddGeneralConstraint	Y	Y	Y	Y	Y	Y
CLinearConstraints::AddGroupConstraint	Y	Y	Y	Y	Y	Y
CLinearConstraints::SetAssetRange	Y	Y	Y		Y	Y
CLinearConstraints::SetFactorRange	Y	Y	Y	Y	Y	Y
CLinearConstraints::SetAssetTradeSize		Y				
CRiskConstraints::AddFactorConstraint		Y				
CRiskConstraints::AddSpecificConstraint		Y				
CRiskConstraints::AddTotalConstraint		Y				
CRiskConstraints::AddFactorConstraintByGroup		Y				
CRiskConstraints::AddSpecificConstraintByGroup		Y				

CConstraintInfo Object Returned By	Lower Bound	Upper Bound	Relativity	Softness	Penalty	Free Range Linear Penalty
CRiskConstraints::AddTotalConstraintByGroup		Y				
CTaxConstraints::SetLongGainArbitrageRange	Y	Y				
CTaxConstraints::SetLongLossArbitrageRange	Y	Y				
CTaxConstraints::SetLongNetArbitrageRange	Y	Y				
CTaxConstraints::SetShortGainArbitrageRange	Y	Y				
CTaxConstraints::SetShortLossArbitrageRange	Y	Y				
CTaxConstraints::SetShortNetArbitrageRange	Y	Y				
CTaxConstraints::SetTaxLimit()		Y				
CTurnoverConstraints::SetLongSideConstraint		Y		Y		
CTurnoverConstraints::SetNetConstraint		Y		Y		
CTurnoverConstraints::SetShortSideConstraint		Y		Y		
CTurnoverConstraints::SetBuySideConstraint		Y		Y		
CTurnoverConstraints::SetSellSideConstraint		Y		Y		
CTurnoverConstraints::SetCrossPeriodNetConstraint		Y		Y		
CTurnoverConstraints::AddNetConstraintByGroup		Y		Y	Y	Y
CTurnoverConstraints::AddLongSideConstraintByGroup		Y		Y	Y	Y
CTurnoverConstraints::AddShortSideConstraintByGroup		Y		Y	Y	Y
CTurnoverConstraints::AddBuySideConstraintByGroup		Y		Y	Y	Y
CTurnoverConstraints::AddSellSideConstraintByGroup		Y		Y	Y	Y
CGeneralPWLinearConstraint::SetConstraint	Y	Y		Y	Y	Y
CConstraints::SetOverlapConstraint		Y				
CConstraints::SetPortConcentrationConstraint		Y				
CConstraints::SetDiversificationRatio	Y					
CConstraints::SetTotalActiveWeightConstraint	Y	Y		Y	Y	Y
CConstraints::AddTotalActiveWeightConstraintByGroup	Y	Y		Y	Y	Y
CCrossAccountConstraints::SetAssetBoundOnTotalPosition	Y	Y		Y	Y	
CCrossAccountConstraints::SetMaxAssetTotalTrades		Y		Y	Y	

CConstraintInfo Object Returned By	Lower Bound	Upper Bound	Relativity	Softness	Penalty	Free Range Linear Penalty
`CCrossAccountConstraints::SetMaxAssetTotalBuys`		Y		Y	Y	
`CCrossAccountConstraints::SetMaxAssetTotalSells`		Y		Y	Y	
`CCrossAccountConstraints::SetNetTurnoverConstraint`		Y		Y		
`CIssuerConstraints::AddHoldingConstraint`	Y[1]	Y	Y	N	N	N

Note: [1] Both upper and lower bounds on net total holding are supported. However, only upper bound on absolute total holding is supported.

1. To set up linear constraints, call `CConstraints::InitLinearConstraints()` to get a `CLinearConstraints` object. Then, call methods of the `CLinearConstraints` object to set up the following:
   - Asset ranges: $\boldsymbol{l_h} \leq \mathbf{h} \leq \boldsymbol{u_h}$
   - Factor exposure constraints: $\boldsymbol{l_w} \leq \boldsymbol{w} \leq \boldsymbol{u_w}$
   - General linear constraints: $\boldsymbol{l_s} \leq \boldsymbol{s} \leq \boldsymbol{u_s}$
   - Beta constraint: $\boldsymbol{l_\beta} \leq \boldsymbol{\beta}'\boldsymbol{h} \leq \boldsymbol{u_\beta}$, where $\boldsymbol{\beta} = \frac{(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D})\mathbf{h}_B}{\mathbf{h}_B(\mathbf{XFX}^{\mathrm{T}} + \mathbf{D})\mathbf{h}_B}$
   - Transaction limit: Buy_None, Sell_None, etc.
   - Group constraints
   - Option to enable/disable crossovers
2. To set up risk constraints, call `CConstraints::InitRiskConstraints()` to get a `CRiskConstraints` object. Call the following functions with appropriate parameters (e.g. pass true as value of the last parameter *additiveDef* to use additive definition). Then, use methods of the returned `CConstraintInfo` object to set up upper bounds, etc.:
   - `CRiskConstraints::AddFactorConstraint()`
   - `CRiskConstraints::AddSpecificConstraint()`
   - `CRiskConstraints::AddTotalConstraint()`
   - `CRiskConstraints::AddFactorConstraintByGroup()`
   - `CRiskConstraints::AddSpecificConstraintByGroup()`
   - `CRiskConstraints::AddTotalConstraintByGroup()`

3. To set up risk parity constraints, call `CConstraints::InitRiskConstraints()` to get a `CRiskConstraints` object and call `CRiskConstraints::SetRiskParity()`. Note that an asset can be excluded from risk parity constraints by calling `CAsset::SetExcludeFromRiskParity()`.
4. To set up hedge constraints (leverage constraints), call `CConstraints::InitHedgeConstraints()` to get a `CHedgeConstraints` object. Then call methods of the `CHedgeConstraints` object to set up various leverage constraints. For example, call `CHedgeConstraints::AddWeightedTotalLeverageConstraint()` to get a `CConstraintInfo` object and use its methods to set up bounds, penalty, etc. for a weighted total leverage constraint
5. Call `CConstraints::InitTurnoverConstraint()` to get a `CTurnoverConstraints` object. Then, call methods of this object to get `CConstraintInfo` objects for various turnover constraints and use their methods to set up upper bounds, softness, etc.
6. Call `CConstraints::InitTaxConstraint()` to get a `CTaxConstraints` object. Then, call methods of this object to get `CConstraintInfo` objects and use their methods to set up lower/upper bounds, etc.
7. To set up transaction costs constraint, call `CConstraints::SetTransactionCostConstraint()` to get a `CConstraintInfo` object and then call methods of the `CConstraintInfo` object to set up a transaction costs upper bound, and optionally its softness.
8. To enable roundlotting optimization, call `CConstraints::EnableRoundLotting()`.
9. To set up threshold or cardinality constraints, also known as a paring constraints, call `CConstraints::InitParingConstrinats()` to get a `CParingConstraints` object. Then, call methods of the `CParingConstraints` object to populate the object with parameters such as min/max number of assets/trades, min/max number of buys/sells, minimum holding threshold, minimum transaction size threshold, etc. Limits on the number of assets/trades can be specified on the group-level by calling `AddAssetTradeParingByGroup()` or `AddLevelParingByGroup()`. Minimum holding threshold and minimum transaction size threshold can be specified on the asset-level by calling `AddLevelParingByAsset()`.
    - To allow close out positions if the transaction size is below the minimum threshold, call `CParingConstraints::SetAllowCloseOut()`.
    - To set penalty on cardinality constraints, call `SetPenaltyPerExtraAsset()` or `SetPenaltyPerExtraTrade()`.
    - To set penalty on threshold constraints, call `SetPenaltyPerUnitBelowHoldingThreshold()` or `SetPenaltyPerUnitBelowTradeThreshold()`.
10. To enable grandfather rule for minimum holding level threshold, call `EnableGrandfatherRule()`. To set up the 5/10/40 rule, call `CConstraints::Init5_10_40Rule()` to get a `C5_10_40Rule` object. Then, call methods of the `C5_10_40Rule` object to populate the object with 5/10/40 rule-related parameters and issuer/branches data.
11. To set up constraint hierarchy, call `CConstraints::InitConstraintHierarchy()` to get a `CConstraintHierarchy` object. Then, call `CConstraintHierarchy::AddConstraintPriority()` to set priority for constraint categories or call

`CConstraintHierarchy::AddConstraintPriority4IndivCon()` to set priority for individual linear or factor constraints.

12. To add a general piecewise linear constraint, call `CConstraints::AddGeneralPWLinearConstraint()` to get a `CGeneralPWLinearConstraint` object. Then, call methods of `CGeneralPWLinearConstraint` to set the starting point, add break points, and slopes. Call `CGeneralPWLinearConstraint::SetConstraint()` to get a `CConstraintInfo` object and then call methods of the `CConstraintInfo` object to set up bounds, softness, etc.
13. To set total active weight constraint, call `CConstraints::SetActiveWeightConstraint()` to get `a CConstraintInfo` object and then call methods of the `CConstraintInfo` object to set up bounds, softness, etc.
14. To set an active weight constraint by group, call `CConstraints::SetActiveWeightConstraintByGroup()` to get `a CConstraintInfo` object and then call methods of the `CConstraintInfo` object to set up bounds, softness, etc.
15. To set diversification ratio constraint, call `CConstraints::SetDiversificationRatio()` to get a `CDiversificationRatio` object and then call methods of the `CDiversificationRatio` object to set up lower bound of the diversification ratio.
16. To set portfolio concentration constraint, call `CConstraints::SetPortConcentrationConstraint()` to get a `CPortConcentrationConstraint` object and then call methods of the `CPortConcentrationConstraint` object to set up upper bound, number of top holdings, etc.
17. To set the overlap constraint, call `CConstraints::SetOverlap()` to get a `CConstraintInfo` object and then call methods of the `CConstraintInfo` object to set up the upper bound on overlap.
18. To allow including futures in turnover, paring, roundlot and hedge constraints, call `CConstraints::SetIncludeFutures()`.
19. To set issuer constraints, `call CConstraints::InitIssuerConstraints` to get a `CIssuerConstraints` object. Then call `CIssuerConstraints::AddHoldingConstraint()` to get a `CConstraintInfo` object, then call methods of the `CConstraintInfo` object to set up bounds.

5.5.4 Set Up Efficient Frontier

To set up an efficient frontier, call `CCase::InitFrontier()` to get a `CFrontier` object.

Your application can pass appropriate parameters to specify one of the following types of the frontier:

- Risk-reward frontier varying return
- Risk-reward frontier varying risk
- Utility-turnover frontier varying turnover
- Utility-tax frontier varying tax costs
- Utility-constraint frontier varying the following constraints:
  - Factor constraint

- General linear constraint
- Transaction cost constraint
- Leverage (hedge) constraint

Then, call methods of the `CFrontier` object to populate the object with data, such as the number of frontier data points and frontier range. For the utility-constraint frontier, call `CFrontier::SetFrontierConstraintID()` to set the varying constraint. To account for transaction costs in the return for risk-reward frontier, call `CFrontier::SetIncludeTransactionCost()`. To set the type of bound (lower or upper) to vary for utility-constraint frontier, call `CFrontier::SetFrontierBoundType()`.

5.5.5 Set Up Tax-Aware Optimization Parameters

To set up tax-aware optimization parameters, call `CCase::InitTax()` to get a `CTax` object. Then, call methods of the `CTax` object to populate the object with tax related data such as:

- Tax rate
- One-rate or two-rate
- Cross netting gain/loss option
- Loss carry forward
- Wash sale rule
- Tax harvesting targets/penalties
- Selling order rule
- Tax unit (Default to dollar amount)

5.6 Run Optimization

To run an optimization, do the following:

- Call `CWorkSpace::CreateSolver()` with the populated `CCase` object as its parameter to get a `CSolver` object.
- If the interactive approach is used, call `CSolver::SetCallBack()` with a parameter of the pointer to an instance of a client application class that derives from the `CCallback` class. For information about the interactive approach, see the Interactive Approach section.
- Optionally, call `CSolver::SetOptimalityToleranceMultiplier()` to make trade-off of performance and optimality.
- Optionally, call `CSolver::SetCountCrossTradeAsOne()` to count a crossover trade as one or two trades.
- Optionally, call `CSolver::SetOption()` with the string “COMPATIBLE_MODE” to set the solver approach that is backward compatible.

- Optionally, call `CSolver::SetOption()` with the string "MAX_OPTIMAL_TIME" to set the maximum time in seconds for an optimization to run.
- Optionally, call `CSolver::SetOption()` with the string " OPTIMAL_WEIGHT_TOLERANCE" or " OPTIMAL_TRADE_TOLERANCE" to set the tolerance for asset weights or trade sizes in the optimal portfolio. If the absolute value of weights/trades is below the tolerance level, the weights/trades will be dropped from the optimal portfolio and the dropped weights/trades are accumulated and allocated based on next options. When setting this option, make sure to check the constraint diagnostics as constraint violations might occur.
- Optionally, call `CSolver::SetOption()` with the string "OPTIMAL_WEIGHT_ALLOCATION_MODE" OR "OPTIMAL_TRADE_ALLOCATION_MODE" to set the allocation modes to redistribute the accumulated small weights/trades. A value of "0" (default) specifies to convert the accumulated weights/trades to cash or cash flow; A value of "1" specifies to equally allocate the accumulated weights/trades to non-zero holdings or trades; A value of "2" specifies to allocate the accumulated weights/trades proportional to the asset's holding/trade size.
- Optionally, call `CWorkSpace::Serialize()` to save the input data to a file with extension name .wsp. The wsp file is useful for troubleshooting and is required when sending support requests to Barra Optimizer developers. You may utilize openopt_exe (included in the installation package) to review the input data and then run an optimization based on the wsp file. We recommend that users call `CWorkSpace::Serialize()` after calling `CWorkSpace::CreateSolver()`.
- Optionally, call `CSolver::SetOption`("REPORT_UPPERBOUND_ON_UTILITY", 1) to direct the optimizer to run heuristics to obtain solution for the problem with cardinality and threshold constraints and to report the estimated upper bound on the utility.
- Optionally, call `CSolver::SetOption`("RUN_BRANCH_AND_BOUND", 1) to direct the optimizer to run branch-and-bound procedure to obtain the solution for the problem with cardinality and threshold constraints; and report the estimated upper bound on the utility.
- Call `CSolver::Optimize()` to run the optimization.

5.7 Run Multiple-Period Optimization

To run multiple-period optimization, you need to first set up asset data and constraints for each period. Barra Optimizer allows up to 5 periods to be added to a case. To specify a specific period, call `CWorkSpace::SwitchPeriod()`.The subsequent (applicable) data being set in this case will be for the particular period only. Call `CWorkSpace::GetPeriod()` to retrieve the current period ID.

If certain asset data or constraints apply to all of the periods, simply call `CWorkSpace::SwitchPeriod()` and pass in the constant ALL_PERIOD. Then the subsequent data being set will be applicable for all of the periods. Note that setting data for ALL_PERIOD will invalidate the per-asset or constraint that was previously set for a particular period. To set a cross-period turnover constraint, call `CTurnoverConstraints::SetCrossPeriodNetConstraint()`, and to set a cross-period transaction cost constraint, call `CConstraints::SetCrossPeriodTransactionCostConstraint()`.

The typical steps to set up a multiple-period optimization case are as follows:

1. Create the workspace and add assets to the workspace.
2. Define primary risk model and construct portfolios.
3. To set asset alphas for a period, for example:
    a. Call `CWorkSpace::SwitchPeriod()` to specify a given period as the current period

APIs to Set Up Multiple Period Data	Multiple Period	Notes
`CAsset::SetAlpha`	Y	
`CAsset::AddPWLinearBuyCost`	Y	
`CAsset::AddPWLinearSellCost`	Y	
`CUtility::SetAlphaTerm`		
`CUtility::SetPrimaryRiskTerm`	Y	
`CUtility::SetPenaltyTerm`	Y	
`CUtility::SetTranxCostTerm`	Y	
`CCase::SetCashFlowWeight`	Y	
`CLinearConstraint::SetBetaConstraint`	Y	
`CLinearConstraint::AddGeneralConstraint`	Y	
`CLinearConstraint::AddGroupConstraint`	Y	
`CLinearConstraint::SetAssetRange`	Y	
`CLinearConstraint::SetFactorRange`	Y	
`CTurnoverConstraints::SetNetConstraint`	Y	
`CTurnoverConstraints::SetBuySideConstraint`	Y	
`CTurnoverConstraints::SetSellSideConstraint`	Y	
`CTurnoverConstraints::SetCrossPeriodNetConstraint`	Y	
`CConstraints::SetTransactionCostConstraint`	Y	
`CConstraints::SetCrossPeriodTransactionCostConstraint`	Y	
`CParingConstraints::AddAssetTradeParing`	Y	
`CParingConstraints::AddAssetTradeParingByGroup`	Y	
`CParingConstraints::AddLevelParing`	Y*	Minimum holding threshold supported for multiple periods. Minimum transaction size threshold supported only for first period.

APIs to Set Up Multiple Period Data	Multiple Period	Notes
CParingConstraints::AddLevelParingByAsset	Y*	See AddLevelParing
CParingConstraints::AddLevelParingByGroup	Y*	See AddLevelParing
CParingConstraints:: SetAllowCloseOut	N	A constraint applies to all periods once set; If set more than once, the latter setting overwrites the previous setting
CParingConstraints:: EnableGrandfatherRule	N	A constraint applies to all periods once set; If set more than once, the latter setting overwrites the previous setting

   b. Call `CWorkSpace::GetAsset()` to obtain a pointer to the `CAsset` object

   c. Call `CAsset::SetAlpha()` to set alphas for the current period

4. Create the case and set up constraints for each period. For example,

   a. Call `CCase::InitConstraints()` or `CCase::GetConstraints()` to obtain a pointer to the `CConstraints` object

   b. Call `CConstraints::InitLinearConstraints()` or `CConstraints::GetLinearConstraints()` to obtain a pointer to the `CLinearConstraints` object

   c. Call `CWorkSpace::SwitchPeriod()`

   d. Call `CLinearConstraints::SetAssetRange()` to set asset ranges for the current period

5. Set up the objective and set risk aversions or multipliers for each period. For example:

   a. Call `CCase::InitUtility()or CCase::GetUtility()` to obtain a pointer to the `CUtility` object

   b. Call `CWorkSpace::SwitchPeriod()`

   c. Call `CUtility::SetAlphaTerm()` to set alpha term for the current period

   d. Call `CUtility::SetPrimaryRiskTerm()` to set benchmark and risk aversions for the current period

6. Create the solver and call `CSolver::AddPeriod()` for each period to be included in multiple- period optimization.

5.8 Run Multiple-Account Optimization

To run Multiple-Account Optimization, you need to first set up constraints and case settings for each account. Before setting data for a specific account, call `CWorkSpace::SwitchAccount()`.The subsequent (applicable) data being set in this case will be for the particular account only. Call `CWorkSpace::GetAccountID()` to retrieve the current account ID.

If a certain constraint or utility term applies to all of the accounts, call `CWorkSpace::SwitchAccount()` and pass in the constant ALL_ACCOUNT. Then the subsequent data being set will be applicable for all of the accounts.

Note that setting data for ALL_ACCOUNT will invalidate the constraint that was previously set for a particular account.

The typical steps to set up a Multiple-Account Optimization case are as follows:

1. Create the workspace and add assets to the workspace.
2. Define primary risk model and construct portfolios.
3. Set asset alphas or transaction cost which will be applicable for all accounts.
4. Create the case and set up initial portfolio and base value for each account. For example,
    a. Call `CWorkSpace::CreateCase()` to obtain a pointer to a CCase object. Initial portfolio and portfolio base value is not required at this point.
    b. Call `CWorkSpace::SwitchAccount()` to switch to an account
    c. Call `CCase::SetTradeUniverse()` to set the trade universe for the current account, if different from the trade universe for ALL_ACCOUNT
    d. Call `CCase::SetIntialPort()` to set the initial portfolio for the current account
    e. Call `CCase::SetPortBaseValue()` to set the portfolio base value for the current account
5. Call `CCase::InitConstraints()` or `CCase::GetConstraints()` to obtain a pointer to the `CConstraints` object
    a. Call `CConstraints::InitLinearConstraints()` or `CConstraints::GetLinearConstraints()` to obtain a pointer to the `CLinearConstraints` object
    b. Call `CWorkSpace::SwitchAccount()`
    c. Call `CLinearConstraints::SetAssetRange()` to set asset ranges for the current account
6. To set cross-account constraints, first call `CConstraints::InitCrossAccountConstraints` to obtain a pointer to the `CCrossAccountConstraints` object. And then call the following APIs to set the corresponding cross-account constraints:
    a. Call `CCrossAccountConstraints :: SetNetTurnoverConstraint()` (and the member functions of the `CConstraintInfo` class) to set cross-account turnover constraint
    b. Call `CCrossAccountConstraints :: SetAssetBoundOnTotalPostion()` (and the member functions of the `CConstraintInfo` class) to set bound on total position across all accounts for an asset
    c. Call `CCrossAccountConstraints :: SetMaxAssetTotalTrades()` (and the member functions of the `CConstraintInfo` class) to set the maximum total trade size for an asset over all accounts
    d. Call `CCrossAccountConstraints :: SetMaxAssetTotalBuys`()(and the member functions of the `CConstraintInfo` class) to set the maximum total buys for an asset over all accounts

e. Call `CCrossAccountConstraints :: SetMaxAssetTotalSells()` (and the member functions of the `CConstraintInfo` class) to set the maximum total sells for an asset over all accounts

7. Set up the objective and set risk aversions or multipliers for each account. For example:
   a. Call `CCase::InitUtility()or CCase::GetUtility()` to obtain a pointer to the `CUtility` object
   b. Call `CWorkSpace::SwitchAccount()`
   c. Call `CUtility::SetAlphaTerm()` to set alpha term for the current account
   d. Call `CUtility::SetPrimaryRiskTerm()` to set benchmark and risk aversions for the current account
   e. Call `CUtility::SetUtilityScalar` to set the scalar for utility function for the current account
   f. Call `CUtility::SetJointMarketImpactTerm()` to set the cross-account joint market impact term
8. Create the solver and call `CSolver::AddAccount()` for each account to be included in Multiple-Account Optimization.
9. Optionally, call `CSolver::SetOption`("MAO_APPROACH", 1) to choose Nash Equilibrium approach.

APIs to Set Up Multiple Account Data	Individual Account	Notes
`CUtility::SetAlphaTerm`	Y	
`CUtility::SetPrimaryRiskTerm`	Y	
`CUtility::SetPenaltyTerm`	Y	
`CUtility::SetTranxCostTerm`	Y	
`CCase::SetTradeUniverse`	Y	
`CCase::SetInitialPort`	Y	
`CCase::SetPortBaseValue`	Y	
`CCase::SetCashFlowWeight`	Y	
`CUtility::SetJointMarketImpactTerm`	N	Cross-account term
`CCase::SetUtilityScalar`	Y	
`CLinearConstraint::SetBetaConstraint`	Y	
`CLinearConstraint::AddGeneralConstraint`	Y	
`CLinearConstraint::AddGroupConstraint`	Y	
`CLinearConstraint::SetAssetRange`	Y	

APIs to Set Up Multiple Account Data	Individual Account	Notes
CLinearConstraint::SetAssetTradeSize	Y	
CLinearConstraint::SetFactorRange	Y	
CTurnoverConstraints::SetNetConstraint	Y	
CTurnoverConstraints::SetBuySideConstraint	Y	
CTurnoverConstraints::SetSellSideConstraint	Y	
CConstraints::SetTransactionCostConstraint	Y	
CParingConstraints::AddAssetTradeParing	Y	
CParingConstraints::AddAssetTradeParingByGroup	Y	
CParingConstraints:: SetAllowCloseOut	N	A constraint applies to all accounts once set; If set more than once, the latter setting overwrites the previous setting
CParingConstraints:: EnableGrandfatherRule	N	A constraint applies to all accounts once set; If set more than once, the latter setting overwrites the previous setting
CCrossAccountConstraints::SetAssetBoundOnTotalPosition	N	Cross-account constraint
CCrossAccountConstraints::SetMaxAssetTotalTrades	N	Cross-account constraint
CCrossAccountConstraints::SetMaxAssetTotalBuys	N	Cross-account constraint
CCrossAccountConstraints::SetMaxAssetTotalSells	N	Cross-account constraint
CCrossAccountConstraints::SetNetTurnoverConstraint	N	Cross-account constraint

5.9 Inspect Output

5.9.1 Non-Interactive Approach

When using the non-interactive approach, your application obtains control after the `CSolver::Optimize()` method returns. `CSolver::Optimize()` returns a `CStatus` object that can be used for error handling.

For a portfolio optimization problem, call `CSolver::GetPortfolioOutput()` to get a `CPortfolioOutput` object if the optimization is successful.

Then, call methods of the respective objects to get output information such as the optimal portfolio, objective, risk, turnover, and so forth.

For an efficient frontier problem, call `CSolver::GetFrontierOutput()` to get a `CFrontierOutput` object. Then, call methods of the respective objects to get `CDataPoint` objects, and call `CDataPoint` methods to get portfolios and risk/return values of the frontier data points.

For multiple-period optimization, call `CSolver::GetMultiPeriodOutput()` to get a `CMultiPeriodOutput` object. Call `CMultiPeriodOutput::GetPeriodOutput()` to get a pointer to the `CPortfolioOutput` object which can be used to retrieve per period results. Call `CMultiPeriodOutput::GetCrossPeriodOutput()` to get a pointer to the `CPortfolioOutput` object which can be used to retrieve cross period results.

For Multiple-Account Optimization, call `CSolver::GetMultiAccountOutput()` to get a pointer to the `CMultiAccountOutput` object. Call `CMultiAccountOutput::GetAccountOutput()` to get a pointer to the `CPortfolioOutput` object that can be used to retrieve per account results. Call `CMultiAccountOutput::GetCrossAccountOutput()` to get a pointer to the `CPortfolioOutput` object which can be used to retrieve cross-account results. Call `CMultiAccountOutput::GetCrossAccountTradeInfo()` to retrieve total buys or sells across all accounts per asset. Call `CMultiAccountOutput::GetJointMarketImpactBuyCost(),GetJointMarketImpactSellCost()` to retrieve joint market impact costs of the optimal portfolio.

5.9.2 Interactive Approach

The interactive approach allows your application to process frontier data points/messages generated by Barra Optimizer during the optimization process, before the `CSolver::Optimize()` method returns.

To use the interactive approach, your application needs to derive a class from the `CCallBack` class and override its virtual functions `CCallback::OnDataPoint()` and `CCallback::OnMessage()`. Your application uses the derived class to handle frontier data points and messages sent by Barra Optimizer. Compared to the non-interactive approach, the interactive approach offers more flexibility, but is more difficult to implement.

This feature is not available for the COM API.

5.9.3 Basic Output Information

On successful optimization, Barra Optimizer produces the following basic output information:

- Optimal portfolio weights h*
- Objective value
- Return (in %): $100\mathbf{r}^{\mathrm{T}}\mathbf{h}^{*}$
- Active risk (in %): $100\sqrt{(\mathbf{h}^{*}-\mathbf{h}_{B})^{\mathrm{T}}(\mathbf{D}+\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})(\mathbf{h}^{*}-\mathbf{h}_{B})}$ if a benchmark is added for the primary risk model in the objective
- Total risk (in %): $100\sqrt{\mathbf{h}^{*\mathrm{T}}(\mathbf{D}+\mathbf{X}\mathbf{F}\mathbf{X}^{\mathrm{T}})\mathbf{h}^{*}}$, if no benchmark is added for the primary risk model in the objective
- Total penalty costs
- Beta of the portfolio with respect to the benchmark for the primary risk model used in the objective function

- Portfolio turnover (in %)
- Transaction costs (in %)
- Total short rebate costs
- Implied risk aversion (only available for efficient frontier and return/risk target cases)
- Constraint (turnover or tax cost) slack value in the constraint efficient frontier

For portfolio optimization (non-frontier optimization), Barra Optimizer provides the following additional output information via methods of the `CPortfolioOutput`:

- Constraint slack information
- For holding/trade cardinality information such as values of discrete variables (number of assets/trades, etc.), call `CPortfolioOutput::GetAssetTradeParingInfo()` to get a `CAssetTradeParingInfo` object. For holding/trade cardinality information about a group of assets, call `CPortfolioOutput::GetAssetTradeParingGroupInfo()` to get a `CAssetTradeParingInfo` object.
- For threshold violation information (also known as level paring), call `CPortfolioOutput::GetLevelParingViolationInfo()` to get a `CLevelParingInfo` object
- For post-optimization roundlotted portfolio, use `CPortfolioOutput::GetRoundlottedPortfolio()`.
- For a tax-aware optimization case, call `CPortfolioOutput::GetTaxOutput()` to get a `CTaxOutput` object. Then, call methods of the `CTaxOutput` object to get more detailed information on tax related outputs.
- For determining if the output portfolio is heuristic or optimal, use `CPortfolioOutput::IsHeuristic()`. For more information, see the Indication of the Output Portfolio as Heuristic or Optimal section.
- For trade list information, call `CPortfolioOutput::GetAssetTradeListInfo()` to get a `CAssetTradeListInfo` object. Alternatively, call `CSolver::GetTradeList()` to get a `CTradeListInfo` object that contains trade list information between any two portfolios.
- For the estimated upper bound on the utility for paring cases, call `CPortfolioOutput::GetUpperBoundOnUtility()`. To compute the heuristic gap, call `CPortfolioOutput::GetUpperBoundOnUtility() - CDataPoint::GetUtility()`.
- KKT table items

Note: You can log output information to a file by calling `CPortfolioOutput::WriteToFile()`.

5.9.4 Additional Statistics for Optimal Portfolio or Initial Portfolio

Barra Optimizer allows you to add the following additional statistics for a given portfolio, including the optimal/initial portfolio using the `CSolver::Evaluate()`:

- Portfolio return
- Factor risk
- Specific risk

- Information ratio/Sharpe ratio

For primary risk model, you may obtain the contribution to risk (or tracking error) from a set of assets or factors for a given portfolio, including the optimal/initial portfolio using `CSolver::EvaluateRiskContribution()`.

5.9.5 Constraint Slack Information

With the output, Barra Optimizer also provides information about constraint slacks such as a list of binding constraints, impact to utility, soft bound violations, etc.

- For a given constraint slack, one may call `CPortfolioOutput::GetSlackInfo()` to get `CSlackInfo` and then call methods of the `CSlackInfo` to get more details of the slack.
- Call `CPortfolioOutput::GetBindingSlackIDs()` to get a list of binding constraints. A constraint $l \le g(\mathbf{h}) \le u$ is called binding at the optimal portfolio h* if $g(\mathbf{h}^*) = l$ or $g(\mathbf{h}^*) = u$ and is called non-binding if $l < g(\mathbf{h}^*) < u$.
- Impact to utility:
  - Call `CSlackInfo::GetDownImpactToUtility()` to get the value of down-side impact to utility.
  - Call `CSlackInfo::GetUpImpactToUtility()` to get the value of up-side impact to utility.
  - Up-side impact to utility usually has the same value as down-side impact to utility. However, they may be different at the break points of the piecewise linear functions (for example, transaction cost function, penalty functions).
  - Information on impact to utility is not available for some non-standard optimization cases, for example, paring cases, efficient frontier cases, risk/return target cases, and roundlotting cases.
  - Value of impact to utility is usually zero for non-binding constraints.
- Call `CPortfolioOutput::GetSoftBoundSlackIDs()` to get a list of constraints that have soft bound violations. Then, call methods of the `CSlackInfo` objects to get detailed information on how much a soft bound is violated.

5.9.6 Error Handling

Call `CStatus::GetStatusCode()` to get the status code of EStatusCode type (see the following table for the various error messages). Call `CStatus::GetMessage()` to get a default English message string for the current status code.

EStatusCode	Message
eOK	Optimization terminated normally
eDATA_ERROR	Data Error
eNOT_ENOUGH_MEMORY	Memory error
eUSER_CANCELED	Optimization process canceled by

EStatusCode	Message
	the user
eLICENSE_ERROR	License check failed
eOTHER_ERROR	Other Error (%s)
eTIME_LIMIT_EXCEEDED	Optimization exceeded time limit set by the user. The default time limit is 1 hour.
eINTERNAL_ERROR	Internal Error
eINFEASIBLE	Infeasible Case
eINFEASIBLE_DUETO_51040	Infeasible due to 5-10-40
eINFEASIBLE_DUETO_ROUNDLOTTING	Infeasible due to roundlotting
eINFEASIBLE_DUETO_RISK_CONSTRAINTS	At least one of the risk constraint bounds is too tight
eINFEASIBLE_DUETO_PARING	A paring constraint such as threshold or cardinality is too restrictive
eINFEASIBLE_DUETO_PARING_AND_ROUNDLOTTING	The problem is infeasible due to paring and roundlotting constraints
eINFEASIBLE DUETO_OVERLAP	The problem is infeasible due to overlap constraint
eINFEASIBLE_DUETO_ISSUER_CONS	The problem is infeasible due to issuer constraint(s)
eMAYBE_INFEASIBLE_DUETO_OVERLAP	The problem may be infeasible due to overlap constraint
eMAYBE_INFEASIBLE_DUETO_RISK_CONSTRAINTS	At least one of the risk constraint bounds may be too tight
eMAYBE_INFEASIBLE_DUETO_HEDGE_AND_PARING	The problem may be infeasible due to hedge and paring constraints
eMAYBE_INFEASIBLE_DUETO_TAX_AND_PARING	The problem may be infeasible due to tax and paring constraints
eMAYBE_INFEASIBLE_DUETO_TURNOVERBYSIDE	The problem may be infeasible due to turnover-by-side constraints
eMAYBE_INFEASIBLE_DUETO_TAX_CONS	Tax constraints may be too tight

EStatusCode	Message
eMAYBE_INFEASIBLE_DUETO_51040	The problem may be infeasible due to 5/10/40 rule.
eMAYBE_INFEASIBLE_DUETO_RISKPARITY	The problem may be infeasible due to risk parity constraint.
eMAYBE_INFEASIBLE_DUETO_PARING_AND_ROUNDLOTTING	The problem may be infeasible due to paring and roundlotting constraints.
eMAYBE_INFEASIBLE_DUETO_GENERAL_PWLI	The problem may be infeasible due to general piecewise linear constraints
eMAYBE_INFEASIBLE_DUETO_HEDGE_OR_GENERAL_PWLI	The problem may be infeasible due to hedge constraints or general piecewise linear constraints
eMAYBE_INFEASIBLE_DUETO_ISSUER_CONS	The problem may be infeasible due to issuer constraint(s)
eRISK_TARGET_OR_RANGE_UNATTAINABLE	Risk target/range is unattainable. The closest risk we can get is x%
eRETURN_TARGET_OR_RANGE_UNATTAINABLE	Return target/range is unattainable. The closest return we can get is x%
eCONSTRAINT_RANGE_UNATTAINABLE	Constraint range is unattainable. The closest one we can get is x%
eTARGET_OR_RANGE_UNATTAINABLE_DUETO_PARING	Target/range is unattainable because of paring constraints. The best guess is x%
eTARGET_OR_RANGE_UNATTAINABLE_DUETO_NONCONVEXITY	Failed to reach the target/range because of non-convex features other than paring constraints. The best guess is x%
eRISK_TARGET_OR_RANGE_NOT_ON_FRONTIER	Risk target/range is outside efficient frontier range. Better utility with a smaller risk (x%) achieved.
eRETURN_TARGET_OR_RANGE_NOT_ON_FRONTIER	Return target/range is outside efficient frontier range. Better utility with a higher return (x%) achieved.
eCONSTRAINT_RANGE_NOT_ON_FRONTIER	Constraint range is outside efficient frontier range. Better

EStatusCode	Message
	utility with a tighter constraint (x) achieved.
eRISK_TARGET_FAILED	Risk target optimization failed.
ePARING_HEURISTIC_FAILED	Paring failed. The threshold or cardinality constraint may be too restrictive.
eHEDGE_HEURISTIC_FAILED	Hedge heuristic approach failed
eROUNDLOT_HEURISTIC_FAILED	Roundlotting heuristic approach failed
eMAO_HEURISTIC_FAILED	Multi-Account Optimization heuristic approach failed

With the status code, your application knows whether the optimization is successful or unsuccessful and can act accordingly. Your application may use the status code along with the additional information (obtained by calling `CStatus::GetAdditionalInfo()`) to display a customized message string that may contain more details about the case including timing and solver information.

5.9.7 Information for Infeasible Cases

If an optimization case proves infeasible, Barra Optimizer usually provides enough information to determine which constraints have been violated, and by how much. This information allows you to loosen constraints and make the problem feasible.

Some Restrictions:

- If the infeasibility was caused by overly restrictive asset bounds, Barra Optimizer may not be able to indicate which asset bounds were violated, and may only indicate that the portfolio balance constraint was violated.
- Barra Optimizer may not be able to provide information about infeasibility caused by some special constraints, such as threshold or holding constraints.

When an optimization case is determined to be infeasible, you may call: `CPortfolioOutput::GetInfeasibleSlackIDs()` to review slack IDs and determine which constraints may cause the infeasibility. After obtaining the slack IDs, call `CPortfolioOutput::GetSlackInfo()` to produce more detailed information on how much to relax bounds to make the case feasible.

For infeasible cases with threshold or holding (also known as paring) constraints, it may be useful to call the following for hints on relaxing the constraints:

- `CPortfolioOutput::GetAssetTradeParingInfo()`
- `CPortfolioOutput::GetLevelParingViolationInfo()`

5.9.8 KKT Table Items

Call the following functions to retrieve KKT table items for the objective terms:

- `CPortfolioOutput::GetPrimaryRiskModelKKTTerm()`
- `CPortfolioOutput::GetSecondaryRiskModelKKTTerm()`
- `CPortfolioOutput::GetResidualAlphaKKTTerm()`
- `CPortfolioOutput::GetTransactioncostKKTTerm()`

Call the following functions to retrieve KKT table items for the constraint slacks

- `CSlackInfo::GetKKTTerm()`
- `CSlackInfo::GetPenaltyKKTTerm()`

5.9.9 Asset-level Transaction Costs

Call the following functions to retrieve asset-level transaction costs from `CAssetTradeListInfo` object:

- `CAssetTradeListInfo::GetFixedTransactionCost()` - For more information, see the Fixed Transaction Costs section.
- `CAssetTradeListInfo::GetNonLinearTransactionCost()` - For more information, see the Non-Linear Transaction Costs section.
- `CAssetTradeListInfo::GetPiecewiseLinearTransactionCost()` - For more information, see the Piecewise Linear Transaction Costs section.
- `CAssetTradeListInfo::GetTotalTransactionCost()`

5.10 Data Serialization

Barra Open Optimizer supports serialization of the workspace, which includes any data loaded into the workspace. Prior to version 2.0, data of a `CWorkSpace` object could be serialized into the binary format with the wsp file extension. Starting with version 2.0, additional binary (protobuf) and XML formats are available via the OpenOpt APIs. The OpenOpt APIs take advantage of Protocol Buffers technology developed by Google, which encodes the workspace data in an efficient yet extensible format. The Protocol Buffer (abbreviated as protobuf) is platform-neutral and is portable across Windows and Linux platforms. The Protobuf format is fully backward compatible.

To run optimization with the new formats, we recommend users to load data into the workspace using the OpenOpt APIs for a more streamlined process and user-friendly experience. Alternatively, the existing `CworkSpace` data can be converted into the following formats by calling `CWorkSpace::Serialize()`.

EFileFormat	Description
PB_XML, eXML	XML format created by OpenOpt APIs
ePB	Binary format created by OpenOpt APIs
eWSP	Wsp file containing binary data of a `CWorkSpace` object

For more information, see the Using the Barra Optimizer XML and Using the Command Line Executable chapters.

5.11 Release Resources

Finally, your C++ or Java application must call the `CWorkSpace::Release()` method to release memory resources when the workspace is no longer needed. For C# API, call the `CWorkSpace::Dispose()` method to release the memory resources.

Once `CWorkSpace::Release()` or `CWorkSpace::Dispose()` is called, memory resources associated with the `CWorkSpace` object and all other objects (for example, `CAsset`, `CPortfolio`, `CSolver`, `CCase` objects) will be automatically released.

Applications using the COM API do not need to release resources.

5.12 Thread Safety

Barra Open Optimizer APIs are thread-safe in the context of a multi-threaded environment in which each thread creates a separate instance of the `CWorkSpace` object without the need for external synchronization. Each thread should run to completion and produces consistent result as if each optimization is invoked by a separate process.

5.13 Reference and Tutorial Sample Codes

API reference guides:

- C++ API Reference Guide
- Java API Reference Guide
- C# API Reference Guide
- COM API Reference Guide
- Python API Reference Guide

Tutorial sample codes:

- C++ API tutorials
- Java API tutorials
- C# API tutorials
- MATLAB tutorials

After installing Barra Optimizer, you can see the complete documentation set by navigating to:

<installation directory>/doc/index.html

You can also find the latest versions of some of the documents on the MSCI Client Support site.

6 Using MATLAB, R, and SAS Interface

Barra Open Optimizer provides an Application Programming Interface (API) to MATLAB, R Language, and SAS statistics systems, in addition to the standard C++, Java, C#, and COM APIs.

MATLAB, R Language, and SAS are powerful tools for data visualization, data analysis, and numeric computation. Through the APIs described here, these tools can fully utilize the portfolio optimization features of Barra Optimizer.

This section explains the setup steps for MATLAB, R Language, and SAS. Their interface to Barra Optimizer is derived from the existing Java and C++ APIs. Developers can use this information with the C++ or Java API information to derive the interface format. Some tutorial samples are provided to demonstrate the interface framework. Developers should be able to build their application by applying similar changes to all the other interface classes.

6.1 Working with MATLAB Interface

The Barra Optimizer functions can be accessed from MATLAB through their Java interface mechanism. MATLAB includes a Java Virtual Machine (JVM) enabling developers to use the Java interpreter via MATLAB commands, create, and run programs that create and access Java objects. You can invoke all the functions available in the Barra Optimizer Java API.

One small difference (coding with MATLAB vs. coding with Java) is in the method of passing a null object to the API functions. MATLAB does not have a null object. In these cases, you may pass in '' to denote the null object. For example:

```
% do not want to set benchmark portfolio
% benchmark portfolio object is null, use ''
ut.SetPrimaryRiskTerm('');
```

For detailed information, see the Barra Optimizer Java API Reference Guide.

6.1.1 Setting Up MATLAB

To use the Barra Optimizer functions, you need to make the Barra Optimizer Java classes available in the system. For MATLAB to correctly find and load the Barra Optimizer Java jar file OptJAVAAPI.jar, perform the following setup steps.

6.1.1.1 Verify Your Java Version

Barra Optimizer supports Java 1.4.2 or higher. To verify the JVM version installed in your MATLAB system, type *version –java* in the command window.

6.1.1.2 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux)

For Barra Optimizer to find the 32-bit (or 64-bit) run time library on the Windows platform, the environment variable PATH should include the path to the *bin\ia32* or *bin\intel64* subfolder.
For example:

```
C:\Program Files\MSCI Barra\Barra Optimizer\bin\ia32
```

For Barra Optimizer to find the run time library on a Linux platform, the environment variable LD_LIBRARY_PATH should include the path to the *lib/32* (32-bit) or *lib/intel64* (64-bit) subfolder.
For example:

```
/BarraOptimizer/lib/32
```

If using MATLAB R2013 **or later on Linux,** MATLAB has its own version of Intel MKL BLAS library that conflicts Barra Open Optimizer. You will need to set the environment variable as follows:

export BLAS_VERSION=libblas.so

For earlier versions of MATLAB on Windows, if certain operations such as matrix multiplication are not working, add the environment variable:

BLAS_VERSION=refblas.dll

6.1.1.3 Quick Start to Set Up the Java Class Path and Library Path

On Windows, Barra Optimizer provides a batch file, in the tutorials\matlab directory, to set up the necessary paths to import Java API classes into MATLAB.

To run the batch script, do the following:

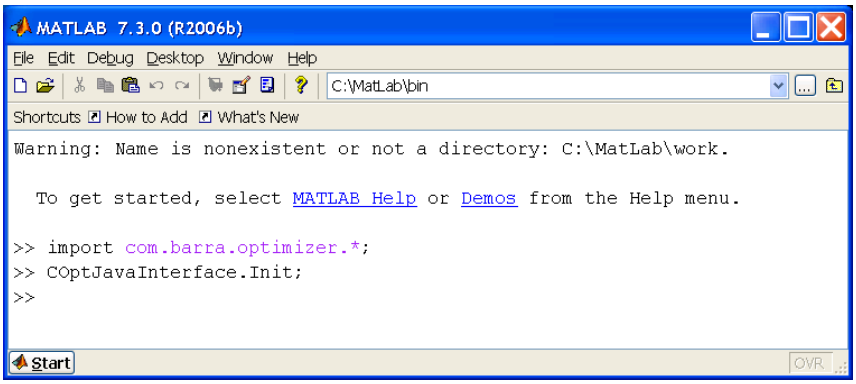
1. Open a command prompt window and navigate to tutorials\matlab.
2. Type setup.bat <32 or 64> and then press Enter.

Where, provide 32 or 64 based on your MATLAB installation architecture.

Note: To know about the MATLAB architecture (32-bit or 64-bit), see Help > About MATLAB.

The script sets up the static class and library paths, which are loaded by MATLAB at startup. Specifically, it appends the Barra Optimizer JAR file path to javaclasspath.txt and the Barra Optimizer executable path to javalibrarypath.txt. Both the text files are under the user folder on MATLAB's search path. (In MATLAB, you can type `userpath` to locate the mentioned user folder.) The script applies the same for classpath.txt and librarypath.txt configuration files, which are no longer supported after MATLAB version R2013.

Then, to verify that the Barra Optimizer interface is working properly, type the following instructions in the command window:

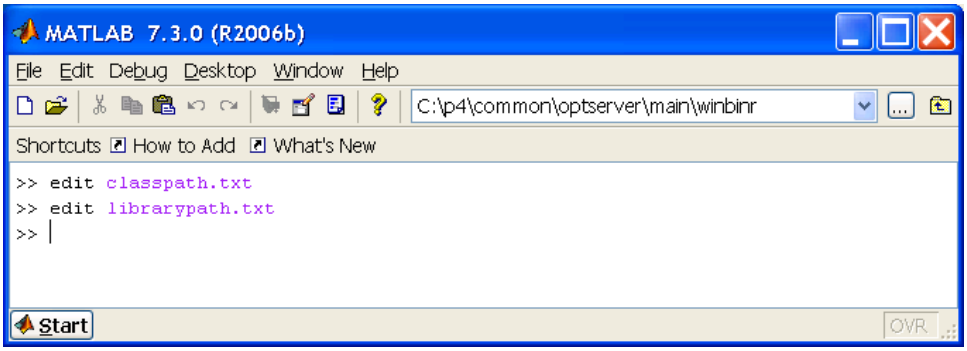


These instructions initialize the Barra Optimizer Java interface classes. If the system returns no errors, then the interface is ready.

6.1.1.4 Manually Setting Up Java Class Path and Library Path

If using a version earlier than MATLAB R2013, to make Barra Optimizer Java API classes available in MATLAB, place OptJAVAAPI.jar on the MATLAB class path, adding locations to the file `classpath`.txt.

To make Barra Optimizer C++ library files available in MATLAB, change the librarypath.txt to include the location of the sub-directory where the C++ library files are located. On the Windows platform, it is the *bin\ia32* (32-bit) or *bin\intel64* (64-bit) sub-directory. On Linux, it is the *lib/32* (32-bit) or *lib/intel64* (64-bit) sub-directory. You can do this directly from the command window (only the Windows paths are shown here):




```
##
## FILE: classpath.txt
##
## Entries:
##    o path_to_jarfile
##    o [glnxa64,glnx86,sol2,unix,win32,win64,mac]=path_to_jarfile
##    o $matlabroot/path_to_jarfile
##    o $jre_home/path_to_jarfile
##
```

```
C:\Program Files\MSCI Barra\Barra Optimizer\bin\optJAVAAPI.jar
```

On a 64-bit Windows PC, the above path may need to be replaced by:

```
C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin\optJAVAAPI.jar


##
## FILE: librarypath.txt
##
## Entries:
##    o path_to_jnifile
##    o [alpha,glnx86,sol2,unix,win32,mac]=path_to_jnifile
##    o $matlabroot/path_to_jnifile
##    o $jre_home/path_to_jnifile
##
C:\Program Files\MSCI Barra\Barra Optimizer\bin\ia32
$matlabroot/bin/$arch
```

On a 64-bit Windows PC, the above path may need to be replaced by:

```
C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin\intel64
```

If you are unable to modify the shared files classpath.txt and librarypath.txt, you may copy them to the current MATLAB directory for editing. After modifying the files classpath.txt and librarypath.txt, restart the MATLAB application. If you are not using the shared classpath.txt and librarypath.txt, you need to start MATLAB from the directory where classpath.txt and librarypath.txt are located.

If using MATLAB R2013 or later and MATLAB cannot find the Barra Open Optimizer Java classes, you will need to rename `classpath.txt` to `javaclasspath.txt`, and `librarypath.txt` to `javalibrarypath.txt`. For more information, refer to the MATLAB documentation.

6.1.2 MATLAB Tutorials

The set of MATLAB tutorials are located in the *tutorials/matlab* directory. They consist of three class definitions—TutorialData, TutorialBase, and TutorialApp—and the TutorialDriver script to drive through each individual tutorial.

Tutorial Files	Description
TutorialData.m	Defines a TutorialData class to handle simulation data used for all tutorials, including reading models from files
TutorialBase.m	Defines a TutorialBase class that contains shared routines for all tutorials
TutorialApp.m	Defines a TutorialApp class that implements individual tutorials

Tutorial Files	Description
TutorialDriver.m	User access point/interface to set up simulation data and run through each individual tutorial

As the user interface for the tutorials, you may want to run the TutorialDrive function at the MATLAB prompt to run all the tutorials. The runtime results will appear on your MATLAB terminal as the TutorialDriver function completes individual tutorials. The example tutorial 1a's result is shown below:

```
>> TutorialDriver
======== Running Tutorial 1a  ========
Minimize Total Risk
Optimization terminated normally.
Initializing Barra Optimizer 8.2.0
CPU by COptCore::Optimize() = 0.003 seconds
Quadratic Solver

Optimized Portfolio:
Risk(%)      = 19.3642
Return(%)    = 0.0000
Utility      = -0.0281
Turnover(%) = 7.7663
Penalty      = 0.0000
TranxCost(%)= 0.0000
Beta         = 0.0000

Asset Holdings:
USA11I1: 0.4829
USA13Y1: 0.5171

Balance constraint KKT term
USA11I1: -0.0562
USA13Y1: -0.0562
```

Alternatively, a runtutorial.bat (for Windows) or a runtutorial.sh (for Linux) is provided to run the tutorials at your operating system's command prompt.

6.1.3 Java APIs to for setting up workspace data

To improve performance, Barra Optimizer provides a number of Java interfaces to accept arrays of data in order to minimize the number of API calls in MATLAB. The Java APIs allow you to set factor and specific covariance

matrices, asset exposures, portfolio weights, and asset attributes. For example, instead of calling SetFactorExposure() in nested loops to set the asset exposure per asset and per factor:

```
for i = 1 : size(data.id,1)
    for j = 1 : size(data.expData,2)
        rm.SetFactorExposure(data.id(i),data.factor(j),data.expData(i,j));
    end
end
```

You can set the asset exposures with one API call by passing arrays of data:

```
rm.SetFactorExposures(assetIDs, factorIDs, exposures);
```

The following Java APIs set the workspace data with the specified arrays of values:

Java API	Description
`CRiskModel:: SetFactorCovariances()`	Sets the covariance matrix.
`CRiskModel:: SetFactorExposures()`	Sets the exposures for a list of assets. The i_{th} asset is exposed to the i_{th} factor with ith exposure.
`CRiskModel:: SetFactorExposuresForAsset()`	Sets the exposures of an asset.
`CRiskModel:: SetSpecificCovariances()`	Sets the specific covariances for list of assets. The ith covariance is the covariance between ith assetIDs1 and ith assetIDs2.
`CRiskModel:: SetSpecificVariances()`	Sets the specific variances for list of assets.
`CPortfolio::AddAssets()`	Adds a list of assets and their weights to the portfolio.
`CWorkSpace::CreateAttributeSet()`	Creates a CAttributeSet object with the arrays of keys and values.

6.1.4 MATLAB Toolbox

For the MATLAB users, Barra Open Optimizer has an additional support toolbox, which is provided with the installation package in the \docs\matlabref directory. With the MATLAB toolbox, all the documentation and tutorials are easily available while using Barra Open Optimizer.

To use the toolbox, you need to set \docs\matlabref as the current folder in MATLAB. Once the current folder is set, navigate to the home page in the MATLAB Help browser, and then click Supplemental Software at the bottom of the page.

6.2 Working with R Language Interface

R Language is a free software environment for statistical computing and graphics. It provides a wide variety of graphical and statistical techniques (for example, linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering, and so on). R Language is a GNU project available from http://www.r-project.org.

The Barra Optimizer API can be accessed from R via the external C/C++ DLL interface mechanism. We provide the API functions in a set of C wrapper classes packaged in barraopt_wrap shareable library. This library can be dynamically loaded in R together with an R macro package barraopt_wrap.R. You can invoke all the functions available in the Barra Optimizer C++ API.

For detailed information, see the Barra Optimizer C++ API Reference Guide.

The main differences between the R wrapper interface and the C++ API are:

- All the methods in the Barra Optimizer C++ API are available as function calls: `<class name>_<method name> ( object variable, method arguments )`
- For instance, in C++:

```
workspace->CreatePortfolio( "Portfolio Name" );
```

- In R:

```
CWorkSpace_CreatePortfolio( workspace, "PortfolioName" );
```

- All enumeration values are passed in as a String:
- For instance, in C++:

```
CRiskModel *riskmodel = workspace->CreateRiskModel( "BIM", eEQUITY );
```

- In R:

```
riskmodel = CWorkSpace_CreateRiskModel( workspace, "BIM", "eEQUITY" );
```

6.2.1 Setting Up the R Language Environment

To set up the R Language environment, perform the following steps.

6.2.1.1 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux)

For Barra Optimizer to find the run time library on the Windows platform, the environment variable PATH should include the path to the *bin\ia32* (32-bit) or *bin\intel64* (64-bit) subfolder. For example:

```
C:\Program Files\MSCI Barra\Barra Optimizer\bin\ia32
```

For Barra Optimizer to find the run time libraries on a Linux platform, the environment variable LD_LIBRARY_PATH should include the path to the *lib/32* (32-bit) or *lib/intel64* (64-bit) subfolder. For example:

```
/BarraOptimizer/lib/32
```

6.2.1.2 Initialize Barra Optimizer Access

Before calling your R program, do the following:

- Set the environment variable BARRAOPT_WRAP to point to the path of the library file barraopt_wrap.dll (Windows) or barraopt_wrap.so (Linux).
- Set the environment variable BARRAOPT_WRAP_R with the value being the path of the barraopt_wrap.R file.

At the beginning of your R program, load the barraopt_wrap library and the barraopt_wrap.R program:

```
dyn.load(Sys.getenv("BARRAOPT_WRAP"))
source(Sys.getenv("BARRAOPT_WRAP_R"))
```

6.2.2 R Tutorials and Sample Code

The set of R tutorials is located in the *tutorials/R* directory. They consist of three class definitions—TutorialData, TutorialBase, and TutorialApp—and the TutorialDriver script to drive through each individual tutorial. Classes with Reference Semantics are used in the R tutorials and R6 package is therefore required to run tutorials[6].

Tutorial Files	Description
TutorialData.r	Defines a TutorialData class to handle simulation data used for all tutorials, including reading models from files
TutorialBase.r	Defines a TutorialBase class that contains shared routines for all tutorials
TutorialApp.r	Defines a TutorialApp class that implements individual tutorials
TutorialDriver.r	User access point/interface to set up simulation data and run through each individual tutorial

A runtutorial.bat (for Windows) or a runtutorial.sh (for Linux) is provided to run the tutorials at your operating system's command prompt. The results will appear on the terminal and will be saved in TutorialDriver.r.Rout file.

In the same directory, the R Language sample code sample.r is also provided to illustrate the access of R interface without Classes with Reference Semantics. The provision of the additional command line parameter "sample" to the batch/shell command will run the sample script instead of the tutorials:

On Windows,

```
> runtutorial.bat sample
```

Or on Linux,

```
> runtutorial.sh sample
```

[6] As of R version 3.1.2 and R6 version 2.2.1, the class reference semantics may not work with some of the Barraopt R APIs. However, the semantics described in the Working with R Language Interface section always work. Some of these APIs are also illustrated in the R tutorials.

6.3 Working with SAS Interface

While you can invoke the Barra Open Optimizer JAVA interface through SAS, note that the Barra Optimizer SAS interface is experimental and we advise not to use this interface in the production environment. We recommend either creating an XML file that matches the Barra Open Optimizer's XML format, or using a different statistical package, such as the Matlab, R, or Java interface, directly.

From SAS, you can access any Java class library via the JavaObj interface. JavaObj is a component object dynamically allocated at DATA step runtime and exists until the step ends, or the object is explicitly deleted. To declare a Java component object:

```
DECLARE JavaObj variable;
variable = _new_ JavaObj ( class, constructor arguments );
```

or

```
DECLARE JavaObj variable ( class, constructor arguments );
```

Variable	A SAS Name
class	Required. Character (variable or literal) specifying a class name that should be found in the JVM's classpath. Package or hierarchy is indicated with a slash (/), not with a dot (.).
arguments	Depends on class. Constructor arguments are Character, Numeric, and JavaObj. The pattern of the argument types is called the signature.

The Barra Optimizer Java class library for SAS is packaged in optSAS.jar. The name space is: com/barra/optimizerSAS.

Before using the Barra Optimizer Java classes, you need to first initialize the interface by calling:

```
DECLARE JavaObj optInit('com/barra/optimizer/COptJavaInterface');
```

We recommend to create a separate SAS file, and specify the file in the –autoexec option in the SAS config file, to prevent re-initialization of the Java interface. The steps are:

1. Create initjavaopt.sas file with the following code:

```
data _null_;
dcl javaobj optInit('com/barra/optimizer/COptJavaInterface');
run;
```

2. Edit the SAS configuration file (.cfg) to include the following option:

```
-autoexec initjavaopt.sas
```

3. Launch SAS with the above configuration file and the Barra Optimizer Java interface is ready for use.

To initialize a `CWorkSpaceSAS` javaObj, the syntax is:

```
DECLARE JavaObj workspace ('com/barra/optimizerSAS/CWorkSpaceSAS');
```

Once a JavaObj is initiated, you can invoke the Java object's method:

```
rc = javaobj . call{modifier}{type}Method (method, argument(s), return)
rc return code - 0 if successful, not 0 otherwise.
```

modifier	Optional. If present, it should be Static.
type	A Java primitive type or String.
method	Required. Character. Name of Java class method.
argument(s	Depends on method. Character, Numeric or JavaObj. Signature must match that of method.
return	Character or Numeric variable that receives value returned by method. Not present if method is void. The return cannot be a JavaObj.

The following table shows the various combinations of *type* and *modifier* call-method:

Type	Modifier	
Void	callVoidMethod	callStaticVoidMethod
Double	callDoubleMethod	callStaticDoubleMethod
String	callStringMethod	callStaticStringMethod
Boolean	callBooleanMethod	callStaticBooleanMethod
Integer	callIntMethod	callStaticIntMethod
Short	callShortMethod	callStaticShortMethod
Byte	callByteMethod	callStaticByteMethod
Long	callLongMethod	callStaticLongMethod
Float	callFloatMethod	callStaticFloatMethod

For instance, to invoke void `SetRiskTarget`(double target) for a `CCase` JavaObj object case, the syntax is:

```
case.callVoidMethod('SetRiskTarget', target);
```

One limitation of the JavaObj interface is that returns from these methods cannot be a JavaObj. For instance, to create a portfolio object, `Cportfolio::CreatePortfolio` (const String & portfolioID) is available in the `CWorkSpace` class. However, there is no call method in the JavaObj interface to facilitate the return as a Java object.

To address this, the regular Barra Optimizer Java classes are wrapped with the corresponding SAS Java classes. The SAS Java is derived from the regular Barra Optimizer Java class. For example, `CWorkSpaceSAS` class represents the

`CWorkSpace` class. For the complete list of available Java classes, refer to the Barra Optimizer Java API Reference Guide.

Most signatures for all methods are the same except for those returning a Java class object. Instead of returning the object, the SAS Java class will return a String ID. This String ID can be used to create the reference object using a DECLARE statement later.

The following example creates a portfolio JavaObj in the `CWorkSpace` Java class:

```
length objID $20;


DECLARE JavaObj workspace('com/barra/optimizerSAS/CWorkSpaceSAS');
workspace.callStringMethod('CreatePortfolio', objID);


DECLARE JavaObj portfolio('com/barra/optimizerSAS/CPortfolioSAS', objID);
```

By calling the constructor in the DECLARE statement, it can look up the Java object and allow JavaObj to reference it in call methods.

For detailed information, see the Barra Optimizer Java API Reference Guide.

6.3.1 Setting Up SAS Java

For SAS to correctly find and load the Barra Optimizer OptSAS.jar, perform the following steps.

6.3.1.1 Verify PATH (Windows) or LD_LIBRARY_PATH (Linux)

For Barra Optimizer to find the run time library on the Windows platform, the environment variable PATH should include the path to the *bin/ia32* (32-bit) or *bin/intel64* (64-bit) subfolder. For example:

```
C:\Program Files\MSCI Barra\Barra Optimizer\bin\ia32
```

For Barra Optimizer to find the run time library on a Linux platform, the environment variable LD_LIBRARY_PATH should include the path to the *lib/32 (32-bit)* or *lib/intel64 (64-bit)* subfolder. For example:

```
/BarraOptimizer/lib/32
```

To make the LD_LIBRARY_PATH visible to SAS, you must define an environment variable SASENV before calling the *sas* script. The value of the SASENV environment variable should be the path of a file that contains definition of LD_LIBBRARY_PATH. For an example, see the files under the folder *tutorials/sas/*.

6.3.1.2 Set the Java Class Path

The Java class path must be set to include full paths of the following two files:

- OptSAS.jar
- OptJAVAAPI.jar

For Windows:

```
> set CLASSPATH=C:\Program Files\MSCI Barra\Barra Optimizer\bin\OptJAVAAPI.jar;C:\Program Files\MSCI Barra\Barra Optimizer\bin\OptSAS.jar
```

For Linux:

```
% setenv CLASSPATH /BarraOptimizer/Lib/OptJAVAAPI.jar:/BarraOptimizer/Lib/OptSAS.jar
```

To make the Java class path visible to SAS, if SAS version is prior to 9.4, create one of the following entries in the SAS configuration file (depending on SAS version):

```
-jreoptions (-Djava.class.path=$CLASSPATH)
```

or

```
-jreoptions (-Dsas.app.class.path=$CLASSPATH)
```

6.3.2 SAS Sample

The SAS sample sample.sas is located in the *tutorials/sas* directory. The sample.r can be executed using the runsample.sh script. On completion, it will create a file sample.log with this output:

```
…
Perform Optimization
Optimize result = Optimization terminated normally.
Risk = 33.240948014
Return = 0
Turnover = 11.176000205
Utility = -8.284293999
# of assets in the optimal portfolio = 4
GE  = 0.3
IBM  = 0.2117600021
IBM_LINK  = 0.1882399979
JAVA  = 0.3
…
```

The differences between Barra Optimizer SAS Java API and the Barra Optimizer Java API can be summarized as:

- All C*Java classes have a corresponding C*SAS.java class (except COptJavaInterface.java). All method names are the same as the regular Java interface.
- For any method that returns a Java object, the SAS Java class will return a String ID that can be later used to create the JavaObj.

For Java:

```
CIDSet idSet = workspace.CreateIDSet();
```

For SAS:

```
workspace.callVoidMethod('CreateIDSet', objID);

Declare JavaObj idSet('com/barra/optimizerSAS/CIDSet', objID);
```

If the method returns null, the objID will be empty.

- Initialize CWorkSpaceSAS first. Only one instance of CWorkSpaceSAS is allowed.
- Unlike Java program, COptJavaInterface.init() is not required. The Java interface is automatically initialized in CWorkSpaceSAS constructor.
- There are small differences in initializing a CWorkSpace object:

For Java:

```
CWorkSpace::CreateInstance();
```

For SAS:

```
dcl javaObj('com/barra/optimizerSAS/CWorkSpaceSAS')
```

- Any Release() method does not actually release the object references until CWorkSpace obj.Release() is called.
- Because SAS JavaObj supports only double, string or Java objects as input parameters, the signature of the SAS Java interface differs slightly from the regular Java API.
  - Replace boolean, integer input parameters with double value.
  - Replace any Enum input parameter with double value.

For example, enum type for asset type:

```
enum EAssetType {
eCASH = 0,
eFUTURES = 1,
eCURRENCY = 2,
eREGULAR = 3,
eCOMPOSITE = 4,
eCOMPOSITE_FUTURES = 5

}
```

In SAS:

```
/* 3 is eREGULAR */

ws.callStringMethod("CreateAsset", "IBM", 3, objIDt);
```

7 Using Python Interface

Barra Optimizer provides pre-compiled extension module that can be loaded into the Python interpreter version 2.7 and 3.4. On Windows, the DLL is called _barraopt.pyd available under the bin\ia32 (32-bit) and bin\intel64 (64-bit) folders.

To use the Barra Optimizer API in Python, perform the following steps:

1. Set the environment variable PYTHONPATH to where the extension dll (_barraopt.pyd) and python interface (barraopt.py) are located. For example:
   - If you are running 32-bit Python, then set `PYTHONPATH= C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin; C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin\ia32`
   - If you are running 64-bit Python, then set `PYTHONPATH= C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin; C:\Program Files (x86)\MSCI Barra\Barra Optimizer\bin\intel64`
2. Launch Python.
3. Import the Barra Optimizer module into Python: `>>> import barraopt`
4. If no errors are displayed, then the module has been imported successfully.
   - If you get the “ImportError: No module named barraopt” error, then check if the paths set in PYTHONPATH environment variable contain the _barraopt.pyd and barraopt.py files.
   - If you get the “ImportError: Module use of python27.dll conflicts with this version of Python”, you might be using a Python version other than 2.7. Install Python 2.7 in this case.

The interface is ready for use:

```
>>> ws = barraopt.CWorkSpace.CreateInstance()
>>> asset1 = ws.CreateAsset("A1")
>>> asset1.SetPrice(100)
>>> asset1.GetPrice()
100.0
```

Note that on Linux, the default Python extension .so file is linked with Python 2.7. To use Python 3.4, create a symbolic link “_barraopt.so” to barraopt.so.3.4.3.

For detailed information about the Python APIs, refer to Python API Reference Guide.

8 Using the Barra Optimizer XML

Barra Optimizer provides the flexibility of serializing optimization inputs into a readable XML format, which can be edited and parsed by OpenOpt executable, and invoke optimization. The XML schema is designed with the goal of reusability, such that the user can update a particular input and rerun optimization without having to rebuild the entire optimization profile. The XML format also allows for the possibility of building individual repositories for risk model data, asset data, and portfolios. The XML interface is easy to set up, provides additional features such as attribute and grouping schemes, and can be converted via OpenOpt executable into a binary format for improved performance and smaller file size.

The major schema elements are listed in the following table. For more information about each element, click on the element name.

WorkSpace	Data	Attribute
		Grouping_Scheme
		Transaction_Costs
		Attribute_Assignment
		Data_File
	Portfolios	
	Risk_Models	
	Rebalance_Profiles	Utility
		Linear_Constraint
		Cash_Asset_Bound
		NonCash_Asset_Bound
		Asset_Bounds
		Asset_Bounds_By_Group
		Factor_Constraint
		Factor_Constraint_By_Group
		Custom_Constraint
		Constraint_By_Group

		Beta_Constraint
		NonCash_Asset_Penalty
		Asset_Penalty_By_Group
		Asset_Penalty
		Asset_Free_Range_Penalty
		Asset_Trade_Size
		Cardinality_Constraint
		Cardinality Constraint By Group
		Threshold_Constraint
		Threshold Constraint By Group
		Threshold Constraint By Asset
		Transaction_Cost_Constraint
		Roundlot_Constraint
		Turnover_Constraint
		Turnover_Constraint_By_Group
		Turnover_Definition
		Leverage_Constraint
		Weighted_Total_Leverage_Constraint
		Risk_Constraint
		Risk_Constraint_By_Group
		Risk_Parity_Constraint
		Five_Ten_Forty_Rule
		Constraint_Priority
		Penalty

		Free_Range_Penalty
		Portfolio_Concentration_Constraint
		Diversificiation_Ratio
		Total_Active_Weight_Constraint
		Total_Active_Weight_Constraint_By_Group
		Multiple-Period Optimization: Cross-Period Constraints
		General_PWLI_Constraint
		Overlap_Constraint
		Issuer Constraint
		Include_Futures
		Multiple-Account Optimization: Cross-Account Constraints
	Rebalance_Job	Rebalance_Profile_ID
		Initial_Portfolio_ID
		Universe_Portfolio_ID
		Reference_Portfolios
		Portfolio_Base_Value
		Cashflow_Weight
		Primary_Risk_Model_ID
		Secondary_Risk_Model_ID
		Attribute Assignment
		Optimization_Setting
	Rebalance_Result	

	Multiple-Period Optimization	
	Multiple-Account Optimization	

8.1 <WorkSpace>

WorkSpace is the root node of the XML schema and contains the <Data>, <Portfolios>, <Risk_Models>, <Rebalance_Profiles>, and <Rebalance_Job> elements.

- The <Data> element contains attribute and grouping scheme definitions, transaction costs, and additional asset-level data such as prices, alphas, roundlots, issuer IDs, and asset type assignments.
- The <Portfolios> element contains set of portfolios that represent the initial portfolio, multiple benchmark portfolios, and investment universe.
- The <Risk_Models> element contains a set of risk models. There is no limit on the number of risk model elements, but at most two risk models can be specified for an optimization job. The OpenOpt executable can be used to load Barra ModelsDirect flat files and convert them into XML format that represent the risk model element.
- The <Rebalance_Profiles> element contains a set of rebalance profiles. Each rebalance_profiles element consists of parameters in the objective function and user constraints. The attributes and grouping schemes can be used in conjunction with certain constraints for flexibility in specifying a group of assets or factors which the constraint applies to.
- The <Rebalance_Job> element ties the optimization inputs together and allows the user to pick specific pieces of data to run an optimization. The rebalance_job element also controls the type of optimization to run, such as utility maximization, risk or return target, or frontier optimization, and contains rebalance_result element, which organizes the optimization outputs into profile diagnostics, portfolio summary, asset details, and trade list information.

8.2 <Data>

8.2.1 <Attribute>

The <attribute> element contains the attribute_id and attribute_value_type attributes. The attribute_id is the unique identifier of the attribute element and attribute_value_type sets the data type of the attribute values, which can be text, integer, or double. The attribute element contains a list of elements, and each element is identified by the element_id and one of the text_value, double_value, or integer_value attribute. An attribute for asset prices is defined as shown:

```
<attribute attribute_id="##default##price" attribute_value_type="double">
      <element double_value="18.879" element_id="#BRAA221|BARRAID"/>
      <element double_value="15.971" element_id="#BRAA222|BARRAID"/>
```

```
</attribute>
```

The following table lists each attribute and its value type:

Attribute	Attribute Value Type	Element attribute tag	Element value description
Price	double	double_value	Asset price
Alpha	double	double_value	Asset alpha
Residual Alpha	double	double_value	Asset residual alpha
Round lot	integer	integer_value	Asset round lot size
Issuer ID	text	text_value	Asset issuer ID, used when 5/10/40 rule is set
Asset Type	integer	integer_value	Asset type: 1=Futures, 2=Currency, 3=Regular, 4=Composite, 5=Composite Futures
Risk Parity Exclusion	integer	integer_value	Whether to exclude asset from risk parity portfolio: 1=Exclude, 0=Include
Fixed Buy/Sell Cost	double	double_value	Asset fixed transaction cost
Piecewise Linear Buy/Sell Cost	double	double_value	Asset piecewise linear transaction cost
Net Short Cost	double	double_value	Asset net cost on short position
Absolute Asset Bounds or Relative Asset Bounds to one benchmark	double	double_value	Asset upper/lower bounds
Asset Relative Bounds to multiple benchmarks	text	text_value	Formatted expression for asset relative bounds
Covariance Term Weight	double	double_value	Diagonal weight matrix in the covariance term
Custom Constraint Attribute	double	double_value	Asset attribute values to control portfolio-level exposure
ConstraintByGroup Attribute	double	double_value	Asset coefficient values for group constraint
Weighted Total Leverage Constraint Long/Short Side Attribute	double	double_value	Asset coefficient values for long/short side constraint
Fitted Cost	double	double_value	Fitted cost for the asset nonlinear transaction cost function
Exponent	double	double_value	Exponent for the asset nonlinear

Attribute	Attribute Value Type	Element attribute tag	Element value description
			transaction cost function
Traded amount	double	double_value	Traded amount for the asset nonlinear transaction cost function
Upside/Downside Fixed Holding Cost	double	double_value	Upside/Downside fixed holding cost
Reference Weight for Fixed Holding Cost	double	double_value	Reference weight for fixed holding cost

8.2.2 <Grouping_Scheme>

The <grouping_scheme> element groups assets based on their alphanumeric attributes. In the grouping_scheme element, the grouping_scheme_id uniquely identifies the grouping scheme, and contains a list of group elements. Each group element is identified by the group_id attribute and contains a list of values that corresponds to the attribute values that have to be grouped.

For example, you can define an attribute that contains a list of assets and their local markets, and a grouping scheme to group multiple local markets into Europe, North America, and Asia. You can then control the weight of a specific group in the portfolio by setting a group constraint. The following attribute and grouping scheme definitions illustrate this.

Define an attribute for local markets:

```
<attribute attribute_id="LocalMarket" attribute_value_type="text">
   <element text_value="US" element_id="MSCI"/>
   <element text_value="UK" element_id="HSBA.L"/>
</attribute>
```

Define a grouping scheme for local markets:

```
<grouping_scheme grouping_scheme_id="LocalMarket_GroupingScheme">
      <group group_id="North America">
        <value>US</value>
        <value>Canada</value>
      </group>
      <group group_id="Europe">
        <value>UK</value>
        <value>Germany </value>
      </group>
</grouping_scheme>
```

For information about applying the grouping scheme to group constraint, see the <Constraint_By_Group> section.

Grouping scheme can also be used to group a list of factors, and applied to factor and risk constraints. For example, to group the USE3L oil services and oil refining factors into the same group:

```
<grouping_scheme grouping_scheme_id="Factor_GroupingScheme">
   <group group_id="Oil Industry Factors">
     <value>USE3L_OILREF</value>
     <value>USE3L_OILSVCS</value>
   </group>
</grouping_scheme>
```

For information about applying the factor grouping scheme to the factor constraint, see the <Factor_Constraint_By_Group> section.

8.2.3 <Transaction_Costs>

To apply the asset fixed transaction cost to an optimization, first define an attribute for buy and sell costs, and set the attribute ID in fixed_buy_cost_attribute_id and fixed_sell_cost_attribute_id:

```
<attribute attribute_id="FixedBuyCost" attribute_value_type="double">
      <element double_value="0.01" element_id="MSCI"/>
      <element double_value=”0.01" element_id="HSBA.L"/>
   </attribute>
   <transaction_costs>
      <fixed_buy_cost_attribute_id>FixedBuyCost</fixed_buy_cost_attribute_id>
   </transaction_costs>
```

The piecewise linear transaction cost is set in the piecewise_linear_buy_cost and piecewise_linear_sell_cost elements. Each element contains an asset_id attribute and a list of cost elements that have the slope and breakpoint attributes:

```
    <transaction_costs>
      <piecewise_linear_buy_cost asset_id="USA11I1">
        <cost break_point="10000" slope="0.00283368"/>
        <cost break_point="1e+010" slope="0.00383368"/>
      </piecewise_linear_buy_cost>
</transaction_costs>
```

The profile-level nonlinear transaction cost is set in nonlinear_transaction_cost element, which has fitted_cost, exponent, and traded_amount attributes:

```
    <transaction_costs>
      <nonlinear_transaction_cost exponent="1.1" fitted_cost="1e-005"
traded_amount="0.01"/>
    </transaction_costs>
```

To set the asset nonlinear transaction cost, first define the attribute for fitted cost, and optionally define the attributes for exponent and traded amount. The attribute IDs are set in asset_nonlinear_transaction_cost element, which has fitted_cost_attribute_id, exponent_attribute_id, and traded_amount_attribute_id attributes:

```
    <attribute attribute_id="FittedCost" attribute_value_type="double">
      <element double_value="5e-005" element_id="USA11I1"/>
      <element double_value="2.5e-005" element_id="USA13Y1"/>
    </attribute>
    <attribute attribute_id="Exponent" attribute_value_type="double">
      <element double_value="1.1" element_id="USA11I1"/>
      <element double_value="0.55" element_id="USA13Y1"/>
    </attribute>
    <attribute attribute_id="TradedAmount" attribute_value_type="double">
      <element double_value="0.01" element_id="USA11I1"/>
      <element double_value="0.005" element_id="USA13Y1"/>
    </attribute>
    <transaction_costs>
      <asset_nonlinear_transaction_cost exponent_attribute_id="Exponent"
fitted_cost_attribute_id=" FittedCost" traded_amount_attribute_id="TradedAmount"/>
    </transaction_costs>
```

8.2.4 <Attribute_Assignment>

The <attribute_assignment> element defines which attribute to use for the optimization job, and can be set under <Data> or <Rebalance_Job>. The attribute assignment set under <Rebalance_Job> is given higher priority. It has the following elements:

Attribute	Attribute_Assignment Element
Price	price_attribute_id
Alpha	alpha_attribute_id
Residual Alpha	residual_alpha_weight_attribute_id
Round lot	round_lot_size_attribute_id
Issuer ID	issuer_attribute_id
Asset Type	asset_type_attribute_id
Risk Parity Exclusion	risk_parity_exclusion_attribute_id
Upside Fixed Holding Cost	upside_fixedHC_attribute_id
Downside Fixed Holding Cost	downside_fixedHC_attribute_id
Reference Weight for Fixed Holding Cost	reference_weight_fixedHC_attribute_id

To identify a particular asset as cash, set the cash_asset_id attribute to the cash asset ID:

```
<attribute_assignment cash_asset_id="CASH"/>
```

8.2.5 <Data_File> Attribute

The <data_file> attribute can be used to reference external XML file containing the Data section. For example, if the XML file for the data section is stored in a separate directory from the WorkSpace XML:

```
<data data_file="C:\Optimizer\20140201\data.xml"/>
```

Alternatively, the data_file attribute can be used to reference a CSV-formatted file containing asset attribute data:

```
<data data_file="C:\Optimizer\20140201\asset_data.csv"/>
```

For information about the CSV-formatted file, see the Add Assets into Workspace section.

8.3 <Portfolios>

The <portfolios> element contains a list of portfolio elements, and each portfolio element has a portfolio_id attribute, which is a unique identifier for the portfolio, and a list of holding elements. Each holding element has asset_id and amount attribute, which defines the weight of each asset in the portfolio. For the investable universe portfolio, set the amount for all assets to zero.

```
<portfolios>
    <portfolio portfolio_id="Initial Portfolio">
      <holding amount="0.3" asset_id="USA11I1"/>
      <holding amount="0.7" asset_id="USA13Y1"/>
</portfolio>
</portfolios>
```

8.4 <Risk_Models>

The <risk_models> element contains the definition of each risk model element. A risk models element has a risk_model_id attribute, which specifies the name of the risk model and an optional risk_model_type attribute, which may improve the efficiency of optimization.

The factor covariance matrix is defined by factor_covariances element, each factor_covariance element has factor_id1, factor_id2, and value attributes to set the covariance between two factors. For improved performance and reduced file size, there is no need to set the covariance if it is zero.

The factor_exposures element has the factor exposures for a list of assets. Inside the factor_exposure element, the exposure elements specify the set of factor ID and exposure value for a given asset. For improved performance, if an asset has zero exposure to a factor, the exposure element does not need to be set.

The specific risk matrix is set in specific_variances element that has the diagonal values of the matrix, and specific_covariances element that contains the specific covariances between two assets.

To define the factor block structure of the factor covariance matrix, add the factor_block elements with unique factor_block_id and the group of factor_id elements associated with the factor block. The factor_blocks element is optional and speeds up optimization by taking advantage of block sparsity of the matrix.

```
<risk_models>
    <riskmodel risk_model_id="GEM" risk_model_type="equity">
       <factor_covariances>
          <covariances factor_id1="Factor_1A" factor_id2="Factor_1A" value="0.00294054"/>
          <covariances factor_id1="Factor_1A" factor_id2="Factor_1B" value="-
0.000211062"/>
       </factor_covariances>
       <factor_exposures>
          <asset_factor_exposure asset_id="USA11I1">
            <exposure factor_id="Factor_1A" value="0.144"/>
            <exposure factor_id="Factor_1B" value="-0.666"/>
          </asset_factor_exposure>
       </factor_exposures>
       <specific_variances>
          <specific_variance asset_id="USA11I1" value="0.032472"/>
          <specific_variance asset_id="USA13Y1" value="0.0347077"/>
       </specific_variances>
       <factor_blocks>
         <factor_block factor_block_id="A">
           <factor_id>Factor_1A</factor_id>
           <factor_id>Factor_2A</factor_id>
         </factor_block>
       </factor_blocks>
    </riskmodel>
</risk_models>
```

8.5 <Rebalance_Profiles>

Multiple rebalance profiles can be set up and the <rebalance_job> attribute dictates that rebalance profile to use for an optimization. A rebalance profile contains the parameters for the objective function and any constraints on the optimization.

A constraint has attributes such as constraint ID that uniquely identifies a constraint, lower and upper bound, relative bound mode, a benchmark portfolio ID, whether the constraint is a soft constraint, and a penalty element. Most constraint bounds can be set as relative to a benchmark portfolio.

8.5.1 <Utility>

The <utility> element has utility_type attribute that sets the type of objective function, and allows you to set the following parameters through its child elements:

Element	Description
alpha_term	Multiplier of the linear alpha term
residual_alpha_term	Multiplier of the quadratic residual alpha penalty term
penalty_term	Constraint penalty term multiplier
risk_free_rate	Risk-free rate; applicable only for the Sharpe ratio optimization
transaction_cost_term	Transaction cost term multiplier
short_rebate_term	Short rebate term
risk_term	Quadratic risk term associated with a risk model
covariance_term	Additional covariance term
fixed_holding_cost_term	Fixed holding cost term
utility_scalar	Scalar for utility function for the current account in multi-account optimization
joint_market_impact_term	Joint market impact term for multiple account optimization

A risk term has parameters for common factor and specific risk aversions, a benchmark portfolio ID, and whether the term is associated with the primary or secondary risk model.

At most two risk terms can be specified as two risk models are supported for an optimization job. The following code defines multiple risk terms for primary and secondary risk models without a benchmark:

```
     <utility utility_type="quadratic">
        <risk_term common_factor_risk_aversion="0.0075" specific_risk_aversion="0.0075"/>
        <risk_term common_factor_risk_aversion="0.0075" is_primary_risk_model="false"
specific_risk_aversion="0.0075"/>
     </utility>
```

Additional covariance terms can be added to the objective function. If the term has diagonal weight matrix, an attribute for the weights can be created using the <Attribute> element. For more information, see the <Attribute> section. The term's weight_matrix_attribute_id can be set to the attribute ID. The following defines the additional covariance term for the primary risk model with a benchmark:

```
  <data>
    <attribute attribute_id="CovTerm_Weight_Attribute_1" attribute_value_type="double">
      <element double_value="1" element_id="USA11I1"/>
      <element double_value="1" element_id="USA13Y1"/>
   </attribute>
  </data>
  <rebalance_profiles>
    <rebalance_profile rebalance_profile_id="Case 16a">
      <utility utility_type="quadratic">
```

```
        <covariance_term benchmark_portfolio_id=”Benchmark” covariance_term=”0.0075”
covariance_term_type=”wxfxw” weight_matrix_attribute_id=”CovTerm_Weight_Attribute_1”/>
      </utility>
    </rebalance_profile>
  </rebalance_profiles>
```

For multi-account optimization, the joint_market_impact_term element has joint_market_impact_multiplier:

```
  <joint_market_impact_term joint_market_impact_multiplier="1" />
```

8.5.2 <Linear_Constraint>

Linear constraints allow setting asset bounds, factor constraint, custom constraint, group constraint, and beta constraint. It has the transaction_type attribute that sets the type of transaction strategy to be used in the optimization, and the enable_crossover attribute that allows asset position to change from short to long side and from long to short side.

8.5.2.1 <Cash_Asset_Bound>

The cash asset bound sets a holding constraint on the cash asset in the portfolio. To define the cash asset weight to be in between 5% and 10% of portfolio:

```
        <cash_asset_bound lower_bound="0.05" lower_bound_mode="absolute" upper_bound="0.1"
upper_bound_mode="absolute"/>
```

8.5.2.2 <NonCash_Asset_Bound>

The noncash bound sets a general holding constraint on all noncash assets in the investable universe. For example, you might be tracking a model portfolio and want to be sure you hold positions in all its assets. You can specify upper and lower bounds for all non-cash assets in the optimal portfolio. If you set a lower bound, the optimizer must include these assets and weights, or it will not produce an optimal portfolio. The following limits the weight of any asset (excluding cash) in the optimal portfolio to 30%:

```
        <noncash_asset_bound upper_bound="0.3" upper_bound_mode="absolute"/>
```

The bounds can be set in addition to individual asset bounds or asset bounds by group. If conflicting bounds are defined for the same asset, a more restrictive bound is determined based on the asset_bound_priority attribute. The lower priority bound will be relaxed to where it is no longer conflicting with the higher priority bound.

For example, resolving the following bounds on the same asset results in the upper bound on the lower priority noncash assets being relaxed to the lower bound on the higher priority asset:

Constraint	asset_bound_priority	lower_bound	upper_bound
asset_bound	highest	0.1	0.2
assest_bound_by_group	medium	0.05	0.15
noncash_asset_bound	lowest	0	0.05
Final Asset Bound		0.1	0.1

8.5.2.3 <Asset_Bounds>

Asset bounds allow you to set asset-level bound for specific assets in the optimal portfolio. There are two ways to set the bounds depending on the number of benchmarks used if you have relative bounds defined.

If all bounds are absolute, or if all bounds are relative and only one benchmark is used, first create a double type attribute with the bound values:

```
<attribute attribute_id="asset_upper_bound_attribute" attribute_value_type="double">
  <element element_id="USA11I1" double_value="0.03"/>
</attribute>
```

To limit the weight of USA11I1 to no more than its weight in the benchmark plus 3%, set the bound mode, attribute ID and reference portfolio ID in asset_bound element:

```
<asset_bounds upper_bound_attribute_id="asset_upper_bound_attribute"
upper_bound_mode="plus" reference_portfolio_id="Benchmark"/>
```

To limit the weight of USA11I1 to no more than 3% in the optimal portfolio, set the attribute ID:

```
<asset_bounds upper_bound_attribute_id="asset_upper_bound_attribute"/>
```

If bounds consist of both absolute and relative bounds, or if bounds are relative to multiple benchmarks, first create a text type attribute with the bound expression in the following format

[prefix] [operator] [constant]

where,

[prefix] is p, b, b2, b3, b4, or b5

where,

- p = initial portfolio
- b = primary benchmark
- b2 = secondary benchmark
- [operator] is a standard math operator, such as +, -, *
- [constant] is a numeric value (enter percentages as decimal values)

You can also enter absolute numbers. For example, to limit the weight of USA11I1 to no more than its weight in the benchmark plus 3%:

```
<attribute attribute_id="asset_upper_bound_attribute" attribute_value_type="text">
  <element element_id="USA11I1" text_value="b + 0.03"/>
</attribute>
```

The reference_portfolios element under <RebalanceJob> allows you to set the reference portfolio ID based on the following mapping:

Bound Expression Prefix	Reference_Portfolios Attributes
b	primary_benchmark
b2	secondary_benchmark
b3	b3
b4	b4
b5	b5

To set the benchmark portfolio ID to "Benchmark" in the above example:

```
  <rebalance_job>
<reference_portfolios primary_benchmark="Benchmark"/>
  </rebalance_job>
```

Set the asset bound as:

```
  <asset_bounds upper_bound_attribute_id="asset_upper_bound_attribute"/>
```

8.5.2.4 <Asset_Bounds_By_Group>

Asset bounds by group constraint allows you to specify upper and lower bounds for groups of assets in the investable universe. If you set a lower bound, the optimal portfolio must include all assets within a group at the given weight. This can be a way to ensure that you do not sell particular assets. You can also force a sell on an asset during rebalance by setting an upper bound of zero. You can use this constraint more than once in order to specify multiple grouping schemes. To use this constraint, an attribute and grouping scheme must be created first. For more information, see the <Grouping_Scheme> section.

The following sets the upper bound for all the assets in the North America group to be 30% from the example in the <Grouping_Scheme> section:

```
    <asset_bounds_by_group grouping_attribute_id="LocalMarket"
grouping_scheme_id="LocalMarket_GroupingScheme" group_id="North America"
upper_bound="0.3"/>
```

8.5.2.5 <Factor_Constraint>

Factor constraint allows you to set bounds for portfolio exposure to specific factors in the risk model. To set the factor constraint for the secondary risk model, set the attribute is_primary_risk_model to false. The by_side attribute allows you to apply the constraint to long, short, or all positions. For example, to limit the exposure to primary risk model's factor FACTOR_1A in the optimal portfolio to 1%:

```
    <factor_constraint factor_id="Factor_1A" lower_bound="0" lower_bound_mode="absolute"
upper_bound="0.01" upper_bound_mode="absolute"/>
```

8.5.2.6 <Factor_Constraint_By_Group>

Factor constraint by group allows you to define bounds for group-level factor exposure from the primary risk model. You must define a factor grouping scheme first. For more information, see the <Grouping_Scheme> section.

The following limits the group exposure of "Oil Industry Factors" to no more than 10%:

```
    <factor_constraint_by_group grouping_scheme_id="Factor_GroupingScheme" group_id="Oil Industry Factors" upper_bound="0.1"/>
```

8.5.2.7 <Custom_Constraint >

Custom constraints allow you to define an optimal portfolio by setting minimum and maximum values for portfolio-level exposure to asset attributes. The by_side attribute sets if the bound should apply to long, short, or all positions in the optimal portfolio. For custom constraints, you must first import your asset attributes. For more information, see the <Attribute> section.

8.5.2.8 <Constraint_By_Group>

Group constraints enable you to set bounds for group-level exposures. The by_side attribute sets the bound for long, short, or all positions in the optimal portfolio.

You can define groups for an alphanumeric grouping schemes such as GICS or user-defined industry groups. A grouping scheme is required to be created first. For more information, see the <Grouping_Scheme> section.

An attribute for asset coefficients is also required. For more information, see the <Attribute> section.

8.5.2.9 <Beta_Constraint>

Beta constraint specifies the desired market exposure of the optimal portfolio with respect to the benchmark set in the primary risk term. For example, to set the beta to a range from 0.9 to 1.0:

```
    <beta_constraint lower_bound="0.9" upper_bound="1"/>
```

8.5.2.10 <Noncash_Asset_Penalty>

Noncash asset penalty sets the asset-level penalty for all assets in the universe except cash. The penalty targets, lower and upper, are absolute weights. For example:

```
    <noncash_asset_penalty target="0.1" lower="0.05" upper="0.15"/>
```

8.5.2.11 <Asset_Penalty_By_Group>

Asset penalty by group sets the asset-level penalty for a group of assets in the universe. If noncash asset penalty is also set, asset penalty by group has higher priority for the assets in the group.

An attribute and grouping scheme must be created first. For more information, see the <Grouping_Scheme> section.

For example, to set asset penalty for all assets in the North America group:

```
    <asset_penalty_by_group grouping_attribute_id="LocalMarket" grouping_scheme_id="LocalMarket_GroupingScheme" group_id="North America" target="0.2" upper="0.25" lower="0.15"/>
```

8.5.2.12 <Asset_Penalty>

Asset penalty sets asset-level penalty for specific assets. It has higher priority for the same assets contained in asset penalty by group and noncash asset penalty. The penalty targets, lower and upper, are absolute weights. For benchmark-relative asset penalty, the slack values need to be converted to absolute weights by shifting the benchmark weights by the relative penalty.

For example, to set asset penalty as absolute weights, with target weight of 10% and 1% disutility at weight of 5% or 15%:

```
<asset_penalty asset_id="USA11I1" target="0.1" lower="0.05" upper="0.15"/>
```

To set a penalty of ±5% relative to benchmark, first find out the weight of the asset in the benchmark (for example, 7%), then shift the benchmark weight by the relative value (lower=7% minus 5%, upper=7% plus 5%):

```
<asset_penalty asset_id="USA11I1" target="0.07" lower="0.02" upper="0.12"/>
```

8.5.2.13 <Asset_Free_Range_Penalty>

Asset free range linear penalty penalizes assets whose bounds are outside of the specified free range. The free range penalty bound is in units of absolute weights. For benchmark-relative asset free range penalty, the slack values need to be converted to absolute weights by shifting the benchmark weights by the relative penalty.

For example, to set asset penalty as absolute weights with free range of 0.05 to 0.1 and penalty rate of 0.01 on both sides:

```
<asset_free_range_penalty asset_id="USA11I1" target_low="0.05" target_high="0.1"
downside_slope="-0.01" upside_slope="0.01"/>
```

8.5.2.14 <Asset_Trade_Size>

For multi-account optimization, a maximum trade size in dollar amount can be specified on the asset-level. The constraint can be set for a specific account by including the constraint under that account's rebalance profile, or for all accounts by including it under the rebalance profile with account_id="-1".

To specify the maximum trade size values, first create a double type attribute under the <data> section. For example, the following attribute sets trade size of 10 million for asset USA11I1:

```
    <attribute attribute_id="asset_max_trade_size_attribute"
attribute_value_type="double">
      <element element_id="USA11I1" double_value="10000000"/>
    </attribute>
```

Then set the trade size constraint as:

```
    <asset_trade_size upper_bound_attribute_id="asset_max_trade_size_attribute"/>
```

8.5.3 <Cardinality_Constraint>

Cardinality constraints set the minimum and maximum number of long, short, or all asset or trades in the optimal portfolio. The constraint can be either mandatory or soft. Soft constraints can be relaxed if the problem is otherwise infeasible. The attributes penalty_per_extra_asset and penalty_per_extra_trade are used to set penalty on violations of paring constraints. The following limits the number of assets the optimal portfolio can hold to 6:

```
    <cardinality_constraint>
      <num_assets is_max_soft="false" maximum="6"/>
    </cardinality_constraint>
```

8.5.4 <Cardinality_Constraint_By_Group>

Cardinality constraints by group set the minimum and maximum number of long, short, or all asset or trades for a specific group of assets.

For this, first create a grouping scheme. For more information, see the <Grouping_Scheme> section.

For example, the following limits the number of assets in the "Information Technology" group to 20 assets and sets the constraint as soft:

```
    <cardinality_constraint_by_group>
        <num_assets_by_group group_id="Information Technology"
grouping_attribute_id="GICS_SECTOR" grouping_scheme_id="GICS_SECTOR_GroupingScheme"
is_max_soft="true" maximum="20"/>
    </cardinality_constraint_by_group>
```

8.5.5 <Threshold_Constraint>

Threshold constraints specify a minimum level for long and short side holding and transaction size, or a minimum level for buy and sell trades. By setting the "allow_closeout" attribute to true, optimizer may close out a position even if the transaction size is below the minimum threshold. The constraint can be either mandatory or soft. Soft constraints can be relaxed if the problem is otherwise infeasible. The attributes penalty_per_unit_below_holding_threshold and penalty_per_unit_below_trade_threshold are used to set penalty

on threshold violations. If the enable_grandfather_rule attribute is set to true, then the grandfather rule is enabled for minimum holding level threshold constraint. The following sets a minimum holding threshold of 4% for both long and short sides, and a minimum trade size of 2% of initial portfolio for both long and short sides:

```
<threshold_constraint>
  <long_side_holding_level is_min_soft="false" minimum="0.04"/>
  <short_side_holding_level is_min_soft="false" minimum="0.04"/>
  <long_side_tranx_level is_min_soft="false" minimum="0.02"/>
  <short_side_tranx_level is_min_soft="false" minimum="0.02"/>
</threshold_constraint>
```

8.5.6 <Threshold_Constraint_By_Group>

Threshold constraint by group specifies a minimum level for long and short side holding and transaction size, or a minimum level for buy and sell trades, for a group of assets. The following sets a minimum long side holding threshold of 1% for all assets in the "Information Technology" group:

```
<threshold_constraint_by_group>
  <long_side_holding_level_by_group group_id="Information Technology"
grouping_attribute_id="GICS_SECTOR" grouping_scheme_id="GICS_SECTOR_GroupingScheme"
is_min_soft="false" minimum="0.01"/>
</threshold_constraint_by_group>
```

8.5.7 <Threshold_Constraint_By_Asset>

Threshold constraint by asset specifies a minimum level for long and short side holding and transaction size for a particular asset. The following sets a minimum long side holding threshold of 10% for the asset "USA13Y1":

```
<threshold_constraint_by_asset>
  <long_side_holding_level_by_asset asset_id="USA13Y1" is_min_soft="false"
minimum="0.1"/>
</threshold_constraint_by_asset>
```

8.5.8 <Transaction_Cost_Constraint>

Transaction cost constraint sets a limit on the total transaction cost as a percentage of portfolio value. The constraint can be either mandatory or soft. For information about how to set asset transaction costs, see the <Transaction_Cost> section.

The following limits the transaction cost to be no more than 0.05% of the portfolio:

```
<transaction_cost_constraint is_soft="false" upper_bound="0.0005"/>
```

8.5.9 <Roundlot_Constraint>

Roundlot constraint ensures that the optimal portfolio consists of only roundlotted trades. You are required to set asset prices and roundlot sizes by creating an attribute. For more information, see the <Attribute> section.

The portfolio base value is also required. For more information, see the <Rebalance_Job> section.

The allow_odd_lot_closeout attribute allows reducing an odd lot position to zero, and the constraint may be mandatory or soft. The following enables the roundlot constraint disallowing oddlot close out:

```
<roundlot_constraint allow_odd_lot_closeout="false" is_soft="false"/>
```

8.5.10 <Turnover_Constraint>

Turnover constraint limits the maximum turnover levels for short, long, buys, sells, or all positions of the optimal portfolio. This considers all transactions, including buys, sells, and short sells. Covered buys are measured as a percentage of initial portfolio value adjusted for any cash infusions or withdrawals you have specified. The following sets a long side turnover soft constraint with maximum turnover of 20%:

```
<turnover_constraint by_side="long" is_soft="true" upper_bound="0.2" />
```

8.5.11 <Turnover_Constraint_By_Group>

Turnover constraint by group limits the maximum turnover levels for a group of assets and supports net, long, short positions, as well as buy and sell transactions. You will first need to create a grouping scheme. For more information, see the <Grouping_Scheme> section. The following sets maximum turnover of 3% for group of assets in the “Information Technology” group:

```
<turnover_constraint_by_group by_side="net" group_id="Information Technology"
grouping_attribute_id="GICS_SECTOR" grouping_scheme_id="GICS_SECTOR_GroupingScheme"
upper_bound="0.03"/>
```

8.5.12 <Turnover_Definition>

Turnover definition contains the attributes use_portfolio_base_value and exclude_cash that controls how turnover is calculated for the turnover constraint. If the use_portfolio_base_value attribute is set to true, turnover is measured as a percentage of the portfolio base value. If use_portfolio_base_value attribute is set to false, the default turnover base value is used, calculated as Long Equity + |Short Equity| + |Change in Cash Position| + Cash Flow. If exclude_cash is set to true, then change in cash position and cashflow are excluded from the numerator in the definition of overall turnover when turnover constraint is set. The following uses the default turnover base value and excludes cash from the turnover calculation:

```
<turnover_definition exclude_cash="true" use_portfolio_base_value="false"/>
```

8.5.13 <Leverage_Constraint>

Leverage constraints are for long-short rebalances. They allow you to limit the amount of debt used to acquire additional assets. With the by_side attribute, you can constrain the use of leverage on the long side, the short side, the total portfolio, and two ratios between short and long sides. The leverage ratios do not include cash. The constraint can be mandatory or soft, and if the no_change attribute is set to true, the lower and upper bound will be set to the leverage value of the reference portfolio during optimization.

The following table lists the by_side attribute and the respective type of leverage constraint:

By_side Attribute	Type of Leverage Constraint
long	Leverage constraint for the long side of the optimal portfolio
short	Leverage constraint for the short side of the optimal portfolio
total	Leverage constraint for the optimal portfolio as a whole
short_to_long	A ratio of leverage in short and long positions in the optimal portfolio
net_to_total	A ratio of net leverage to total leverage in the optimal portfolio

To set a leverage constraint for 130/30 strategy with long position up to 130% and short positions up to 30% of the portfolio base value:

```
   <leverage_constraint by_side="long" is_soft="false" lower_bound="1"
lower_bound_mode="absolute" no_change="false" upper_bound="1.3"
upper_bound_mode="absolute"/>
   <leverage_constraint by_side="short" is_soft="false" lower_bound="-0.3"
lower_bound_mode="absolute" no_change="false" upper_bound="0"
upper_bound_mode="absolute"/>
```

8.5.14 <Weighted_Total_Leverage_Constraint>

Weighted total leverage constraint provides more flexibility than the total leverage constraint and allows you to specify asset-level coefficients for the long and short sides.

Attributes for the asset long and short side coefficients are required. For more information, see the <Attribute> section.

The following sets the asset coefficient attributes for long and short sides:

```
   <attribute attribute_id="Hedge_weighted_total_leverage0_Long"
attribute_value_type="double">
      <element double_value="1" element_id="USA11I1"/>
      <element double_value="1" element_id="USA13Y1"/>
   </attribute>
   <attribute attribute_id="Hedge_weighted_total_leverage0_Short"
attribute_value_type="double">
      <element double_value="1" element_id="USA11I1"/>
      <element double_value="1" element_id="USA13Y1"/>
   </attribute>
```

The following sets a weighted total leverage range between 100% and 130% of portfolio base value:

```
   <weighted_total_leverage_constraint constraint_info_id="Hedge_weighted_total_leverage0"
long_side_attribute_id="Hedge_weighted_total_leverage0_Long" lower_bound="1"
lower_bound_mode="plus" short_side_attribute_id="Hedge_weighted_total_leverage0_Short"
upper_bound="1.3">
   </weighted_total_leverage_constraint>
```

8.5.15 <Risk_Constraint>

Risk constraint sets an upper bound on the level of total or active risk for the optimal portfolio. You can limit total risk, factor risk, or specific risk with the risk_source_type attribute, and the is_primary_risk_model attribute toggles which risk model to use. If the relative_risk attribute is set to true, the upper bound limits the contribution of a particular risk source to the portfolio's total risk. If the reference_portfolio_id is set, the risk constraint limits active risk of the optimal portfolio. The constraint can be either soft or mandatory.

The following limits the total active risk in the secondary risk model to 10% in the optimal portfolio:

```
      <risk_constraint is_primary_risk_model="false" is_relative_risk="false"
is_soft="false" reference_portfolio_id="Benchmark2" risk_source_type="totalRisk"
upper_bound="0.1"/>
```

The following specifies a risk constraint using additive definition:

```
      <risk_constraint is_primary_risk_model="true" is_additive_definition="true"
is_relative_risk="false" is_soft="false" reference_portfolio_id="Benchmark2"
risk_source_type="totalRisk" upper_bound="0.1"/>
```

8.5.16 <Risk_Constraint_By_Group>

Risk constraint by group sets an upper bound on the level of total or active risk for a group of assets or factors.

An attribute and grouping scheme is required. For more information, see the <Grouping_Scheme> section.

Using the example from the <Grouping_Scheme> section, the following limits the total risk to 10% in the primary risk model for assets in the "North America" group:

```
   <risk_constraint_by_group asset_grouping_attribute_id="LocalMarket"
asset_grouping_scheme_id="LocalMarket_GroupingScheme" asset_group_id="North America"
is_primary_risk_model="true" risk_source_type="totalRisk" upper_bound="0.1"/>
```

Using the example from the <Grouping_Scheme> section, the following limits the total risk to 10% in the primary risk model for factors in the Oil Industry Factors group:

```
   <risk_constraint_by_group grouping_scheme_id="Factor_GroupingScheme" group_id="Oil
Industry Factors" is_primary_risk_model="true" risk_source_type="totalRisk"
upper_bound="0.1"/>
```

The following example specifies the same constraint on total risk for the group using additive definition:

```
   <risk_constraint_by_group grouping_scheme_id="Factor_GroupingScheme" group_id="Oil
Industry Factors" is_additive_definition="true" is_primary_risk_model="true"
risk_source_type="totalRisk" upper_bound="0.1"/>
```

8.5.17 <Risk_Parity_Constraint>

Risk parity constraint enables the construction of a risk parity portfolio in which included assets have equal marginal contribution to portfolio total risk in the primary or secondary risk model.

You can exclude assets from the risk parity portfolio by creating an attribute. For more information, see the <Attribute> section.

The following attribute excludes asset USA11I1 from the risk parity portfolio:

```
   <attribute attribute_id="##default##exclude_risk_parity"
attribute_value_type="integer">
      <element element_id="USA11I1" integer_value="1"/>
   </attribute>
   <attribute_assignment
risk_parity_exclusion_attribute_id="##default##exclude_risk_parity"/>
```

To enable the risk parity constraint using the primary risk mode:

```
      <risk_parity_constraint is_primary_risk_model="true" enabled="true"/>
```

8.5.18 <Five_Ten_Forty_Rule>

The 5/10/40 rule sets a holding constraint that limits the total weight of all issuers that represent more than 5% of the optimal portfolio to 40%, and limit any single issuer weight to 10% of the optimal portfolio.

The issuer ID attribute is required. For more information, see the <Attribute> section.

The following attribute sets the issuer IDs:

```
   <attribute attribute_id="##default##issuer_id" attribute_value_type="text">
      <element element_id="USA11I1" text_value="EBAY"/>
      <element element_id="USA13Y1" text_value="NYSE"/>
   </attribute>
```

To enable the 5/10/40 rule:

```
   <five_ten_forty_rule five="5" forty="40" ten="10"/>
```

8.5.19 <Constraint_Priority>

Constraint priority allows you to build a constraint hierarchy. If the problem becomes infeasible, the optimization algorithm will relax the constraints in the specified order until a solution can be found or infeasibility will be reported. The constraint priority can also apply to certain linear constraints by setting the relax_order attribute within the constraint elements.

8.5.20 <Penalty>

Penalty inside the applicable constraint elements sets the penalty term in the objective function. The target attribute sets the desired value of the slack, lower/upper attribute sets the slack value below/above target, is_pure_linear attribute sets if the penalty function is pure linear, and the multiplier attribute adjusts the impact of the individual slack.

The following sets a penalty that helps restrict market exposure:

```
   <beta_constraint is_soft="false">
         <penalty is_pure_linear="true" lower="0.8" multiplier="1" target="0.95"
upper="1.2"/>
   </beta_constraint>
```

8.5.21 <Free_Range_Penalty>

Free-range penalty inside the applicable constraint elements sets the free range linear penalty term in the objective function. The target_low and target_high attributes set the lower and upper bounds of the free range where penalty is zero, and the downside_slope and upside_slope attributes set the penalty rate of each side when the slack variable is outside the free range.
The following sets a free-range penalty that helps restrict market exposure when beta is outside the range of 0.9 to 1.1:

```
   <beta_constraint is_soft="false">
          <free_range_penalty target_low="0.9" target_high="1.1" downside_slope="-0.01"
upside_slope="0.01"/>
   </beta_constraint>
```

8.5.22 <Portfolio_Concentration_Constraint>

Portfolio concentration constraint limits the sum of the weights of top X names in the optimal portfolio. The upper_bound attribute sets the limit on the sum of weights and the num_top_holdings attribute sets the number of top holdings whose total weight cannot exceed the upper bound. To exclude certain assets from the constraint, first create an attribute for the list of excluded assets, and then set the attribute excluded_assets_attribute_id. For more information about how to create an attribute, see the <Attribute> section.

The following sets the constraint limiting total weight of top 5 holdings to no more than 70% of optimal portfolio:

```
<portfolio_concentration_constraint num_top_holdings="5" upper_bound="0.7"/>
```

8.5.23 <Diversification_Ratio >

Diversification ratio contains a lower_bound attribute, which sets the lower bound on the ratio of optimal portfolio's weighted average asset volatility to its actual volatility. The following sets the diversification ratio with lower bound of 1:

```
<diversification_ratio lower_bound="1"/>
```

8.5.24 <Total_Active_Weight_Constraint>

Total active weight constraint sets the bound on the sum of the absolute active weights with respect to a benchmark. For example, the following constrains the sum of the absolute active weights in the optimal portfolio to less than 1%:

```
<total_active_weight_constraint reference_portfolio_id="Benchmark" upper_bound="0.01"/>
```

8.5.25 <Total_Active_Weight_Constraint_By_Group>

Total active weight constraint by group sets the total active weight constraint for a group of assets. An attribute and grouping scheme must be created first. For more information, see the <Grouping_Scheme> section.

8.5.26 <Issuer_Constraint>

Issuer constraints set lower and/or upper bounds on net or absolute total holdings of a specific issuer. Users can also set global bounds (that are applied to all issuers) using the wildcard "*" for the issuer ID.

The following limits the absolute holding of any issuer (global) not to exceed 95%, and net holding of issuer "1" between the range of 26.75% and 50.65% in the optimal portfolio:

```
    <issuer_constraint>
        <holding_constraint constraint_info_id="Issuer_Abs_1" constraint_type="abs"
issuer_attribute_id="*" upper_bound="0.95"/>
        <holding_constraint constraint_info_id="Issuer_Net_2" constraint_type="net"
issuer_attribute_id="1" lower_bound="0.2675" upper_bound="0.5065"/>
    </issuer_constraint>
```

8.5.27 Multiple-Period Optimization: Cross-Period Constraints

Two types of cross-period constraints are supported for multiple-period optimization: the cross_period_transaction_cost_constraint element limits the total transaction cost for all of the periods, and the cross_period_turnover_constraint limits the total turnover for all of the periods. The following sets the maximum cross-period turnover to 50%:

```
<cross_period_turnover_constraint by_side="net"
constraint_info_id="Turnover_CrossPeriod_net" is_soft="false" upper_bound="0.5"/>
```

8.5.28 <General_PWLI_Constraint>

General piecewise linear constraint allows you to define upside and downside segments of a piecewise linear function for each asset. For each upside or downside element, you can set the slope and break point for a particular asset. The optional starting_point element sets the weight of the asset at which point the piecewise linear function evaluates to zero.

8.5.29 <Overlap_Constraint>

Overlap constraint limits the sum of the benchmark weights corresponding to the non-zero portfolio weights. The benchmark is the primary_benchmark attribute defined in <reference_portfolios> element under <rebalance_job> section. The following limits the overlap to 1%:

```
<overlap_constraint upper_bound="0.01"/>
```

8.5.30 <Include_Futures>

By default, cardinality/threshold constraint, leverage constraint, turnover constraint, and roundlot constraint excludes futures-type asset. To include futures asset for a particular type of constraint, set the corresponding attribute to true. The following includes futures asset in cardinality/threshold constraint:

```
<include_futures cardinality_threshold_constraint="true"/>
```

8.5.31 Multiple-Account Optimization: Cross-Account Constraints

Multiple-Account Optimization supports the following cross-account constraints: net turnover, asset bound on total positions, and maximum total trades/buys/sells per asset. All cross-account constraint bounds are in units of dollar amount. Some examples of setting these constraints are:

```
      <cross_account_net_turnover_constraint
constraint_info_id="Turnover_CrossAccount_net" is_soft="false" upper_bound="1000000000"/>
```

```
      <cross_account_asset_bound_on_total_position asset_id="USA11I1" is_soft="false"
lower_bound="10000000" upper_bound="20000000"/>
      <cross_account_asset_max_total_trades asset_id="USA11I1" is_soft="false"
upper_bound="10000000"/>
      <cross_account_asset_max_total_buys asset_id="USA11I1" is_soft="false"
upper_bound="10000000"/>
      <cross_account_asset_max_total_sells asset_id="USA11I1" is_soft="false"
upper_bound="10000000"/>
```

8.6 <Rebalance_Job>

The <rebalance_job> element contains information on the optimization job. You can set the rebalance profile, risk model, the initial portfolio and investable universe, and the type of optimization to run.

8.6.1 <Rebalance_Profile_ID>

The rebalance profile ID for the optimization job, and it should match the rebalance_profile_id attribute defined in the <Rebalance_Profile> element. For more information, see the <Rebalance_Profile> section.

8.6.2 <Initial_Portfolio_ID>

The initial portfolio ID for the optimization job, and it should match the portfolio_id attribute defined in the <Portfolio> element. For more information, see the <Portfolios> section.

8.6.3 <Universe_Portfolio_ID>

The investable universe for the optimization job, and it should match the portfolio_id attribute defined in the <Portfolio> element. For more information, see the <Portfolios> section.

8.6.4 <Reference_Portfolios>

The reference portfolios from the asset bound constraints. For more information, see the <AssetBounds> section.

8.6.5 <Portfolio_Base_Value>

The initial portfolio base value used to compute weights.

8.6.6 <Cashflow_Weight>

Cash flow weight is the percentage of cash contribution to the initial portfolio, entered as decimals.

8.6.7 <Primary_Risk_Model_ID>

The primary risk model ID for the optimization job, and it should match the risk_model_id attribute defined in the <RiskModel> element. For more information, see the <Risk_Model> section.

8.6.8 <Secondary_Risk_Model_ID>

The secondary risk model ID for the optimization job, and should match the risk_model_id attribute defined in the <RiskModel> element. For more information, see the <Risk_Model> section.

8.6.9 <Attribute_Assignment>

Attribute_assignment element defines which attribute to use for the optimization job. For more information, see the <Attribute_Assignment> section under the <Data> section.

8.6.10 <Optimization_Setting>

The optimization setting allows you to set the type of optimization to run, which can be maximum utility (default), risk or return target, or frontier optimization. The tolerance_mutiplier attribute is used for adjusting the optimality tolerance, and count_cross_over_trade_as_one determines whether to count crossover trades as one or two transactions, considering fixed transaction costs and constraining number of trades. For risk target or risk-return frontier, the factor_risk_scalar and specific_risk_scalar elements set the weights for factor risk and specific risk respectively.

The following sets a risk target of 14%:

```
<optimization_setting optimization_type="riskTarget">
   <risk_target>0.14</risk_target>
 </optimization_setting>
```

The following runs a risk-return frontier optimization with 10 points and upper bound of 10%:

```
    <optimization_setting optimization_type="efficientFrontier">
      <frontier frontier_type="riskReturn" lower_bound="0" max_data_points="10"
upper_bound="0.1"/>
    </optimization_setting>
```

To run a utility-constraint frontier optimization, additionally set the attribute <is_utility_constraint_frontier> to "true" for the supported constraints, such as <Factor Constraint>, <Factor Constraint By Group>, <Custom Constraint>, or <Beta Constraint>. To set the type of bound to vary for utility-factor or utility-general linear constraint, set the attribute bound_type to upper_bound or lower_bound.

```
The option element allows you to specify a number of settings passed to the solver, such
as compatibility mode and maximum time for optimization. For example, the following sets
the solver approach to prior version 8.0 and maximum time allowed for optimization to 5
minutes:    <optimization_setting optimization>
      <option option_id="COMPATIBLE_MODE" value="1"/>
      <option option_id="MAX_OPTIMAL_TIME" value="300"/>
    </optimization_setting>
```

To run multiple-period optimization, set for each period the multi_period element that has attributes cashflow_weight and period:

```
    <optimization_setting optimization>
      <multi_period cashflow_weight="0" period="0"/>
      <multi_period cashflow_weight="0" period="1"/>
    </optimization_setting>
```

8.7 <Rebalance_Result>

The rebalance results contain the outputs from an optimization job, which include profile diagnostics, portfolio summary, asset details, and trade list. The status of a job can be found in the optimization_status element, which contains a detailed message of the status, any additional information on the status, and logging from the optimization solver. The validation and data errors in the contents of the XML file are logged in the input_data_error element.

The profile_diagnostics element contains slack information on the constraints, threshold violation information on the assets, and characteristics of the discrete variables in the asset cardinality information.

The portfolio_summary element contains information on the optimal portfolio, such as its risk, return, objective value, turnover, and holdings. For frontier optimization, the portfolio_summary contains a set of optimal portfolios corresponding to each frontier point.

The asset_details element contains asset-level information such as weights in the initial and optimal portfolio, its residual alpha, and roundlotted weight if roundlotting is enabled.

The trade_list element contains a set of transaction information, such as trade type, traded number of shares, market values, and transaction costs, for each asset in the optimal portfolio. In addition to the total transaction cost, the trade list reports fixed transaction cost, nonlinear transaction cost, and piecewise linear transaction cost for each asset. The trade list is based on post-optimization roundlotted portfolio, which rounds the trades to their nearest roundlots after the optimization process. For more information, see Roundlotting—Optimal and Post-Optimization.

8.8 Multiple-Period Optimization Using XML

For multiple-period optimization, asset-level and profile-level data can be per period, which is controlled by the “period” attribute under <data> and <rebalance_profile> elements. The default value for the period attribute is -1, which denotes that the data is applicable for all of the periods. The following example illustrates setting alphas and asset-level bounds for different periods:

```
<data period="1">
    <attribute attribute_id="##default##alpha" attribute_value_type="double">
      <element double_value="0.01576034" element_id="USA11I1"/>
      <element double_value="0.002919658" element_id="USA13Y1"/>
    </attribute>
    <attribute attribute_id="asset_lower_bound_attribute" attribute_value_type="text">
      <element element_id="USA11I1" text_value="0.1"/>
    </attribute>
    <attribute_assignment alpha_attribute_id="##default##alpha"/>
</data>
<data period="2">
    <attribute attribute_id="##default##alpha" attribute_value_type="double">
      <element double_value="0.001459658" element_id="USA11I1"/>
      <element double_value="0.01849966" element_id="USA13Y1"/>
    </attribute>
    <attribute attribute_id="asset_lower_bound_attribute" attribute_value_type="text">
```

```
      <element element_id="USA13Y1" text_value="0.2"/>
    </attribute>
    <attribute_assignment alpha_attribute_id="##default##alpha"/>
</data>

  <rebalance_profiles>
    <rebalance_profile period="-1" rebalance_profile_id="Case 22">
      <utility utility_type="quadratic">
        <risk_term benchmark_portfolio_id="Benchmark"/>
      </utility>
      <linear_constraint enable_crossover="true" transaction_type="allowAll"/>
    </rebalance_profile>
    <rebalance_profile period="1" rebalance_profile_id="Case 22">
      <utility>
        <alpha_term>1</alpha_term>
        <risk_term common_factor_risk_aversion="0.0075" specific_risk_aversion="0.0075"/>
      </utility>
      <linear_constraint>
        <asset_bounds lower_bound_attribute_id="asset_lower_bound_attribute"/>
      </linear_constraint>
    </rebalance_profile>
    <rebalance_profile period="2" rebalance_profile_id="Case 22">
      <utility>
        <alpha_term>1.5</alpha_term>
        <risk_term common_factor_risk_aversion="0.0075" specific_risk_aversion="0.0075"/>
      </utility>
      <linear_constraint>
        <asset_bounds lower_bound_attribute_id="asset_lower_bound_attribute"/>
      </linear_constraint>
    </rebalance_profile>
  </rebalance_profiles>
```

To include the desired periods to run the multiple-period optimization, add the element <multi_period> under <optimization_setting> section. The <multi_period> element contains cashflow_weight attribute that allows you to set different cashflow weight for each period, and the period attribute. For example, to add period 0 and period 1:

```
  <optimization_setting>
    <multi_period cashflow_weight="0" period="1"/>
    <multi_period cashflow_weight="0" period="2"/>
  </optimization_setting>
```

The output will be saved under <rebalance_result>, which will contain a <optimal_portfolio> section for each period and will have information on optimal portfolio's statistics and holdings. The cross-period result will have period equal to -1. The <profile_diagnostics> element will also contain constraint slack information for each

period. Multiple-period optimization does not report post-optimization roundlotted portfolio and asset-level transaction costs, therefore <asset_details> element will not be available.

8.9 Multiple-Account Optimization Using XML

For Multiple-Account Optimization, profile-level data can be per account, which is controlled by the "account_id" attribute under <data> and <rebalance_profile> elements. The default value for the account_id attribute is -1, which denotes that the data is applicable for all of the accounts. The following example illustrates setting asset-level bounds and case inputs for different accounts:

```
<data account_id="1">
    <attribute attribute_id="asset_lower_bound_attribute" attribute_value_type="text">
      <element element_id="USA11I1" text_value="0.1"/>
    </attribute>
</data>
<data account_id="2">
    <attribute attribute_id="asset_lower_bound_attribute" attribute_value_type="text">
      <element element_id="USA13Y1" text_value="0.2"/>
    </attribute>
</data>

<rebalance_profiles>
    <rebalance_profile account_id="-1" rebalance_profile_id="Case 25">
      <utility utility_type="quadratic">
        <alpha_term>1.5</alpha_term>
        <risk_term/>
        <joint_market_impact_term joint_market_impact_multiplier="0.1"/>
      </utility>
      <linear_constraint enable_crossover="true"
enable_portfolio_balance_constraint="true" transaction_type="allowAll"/>
      <cross_account_net_turnover_constraint
constraint_info_id="Turnover_CrossAccount_net" is_soft="false" upper_bound="200000"/>
    </rebalance_profile>
    <rebalance_profile account_id="1" rebalance_profile_id="Case 25">
      <utility>
        <risk_term benchmark_portfolio_id="Benchmark" common_factor_risk_aversion="0.0075"
specific_risk_aversion="0.0075"/>
        <utility_scalar>0.25</utility_scalar>
      </utility>
      <linear_constraint>
        <asset_bounds lower_bound_attribute_id="asset_lower_bound_attribute"/>
      </linear_constraint>
    </rebalance_profile>
    <rebalance_profile account_id="2" rebalance_profile_id="Case 25">
      <utility>
        <risk_term benchmark_portfolio_id="Benchmark2"
common_factor_risk_aversion="0.0075" specific_risk_aversion="0.0075"/>
        <utility_scalar>0.75</utility_scalar>
      </utility>
      <linear_constraint>
        <asset_bounds lower_bound_attribute_id="asset_lower_bound_attribute"/>
      </linear_constraint>
    </rebalance_profile>
  </rebalance_profiles>
```

To include the desired accounts to run the Multiple-Account Optimization, add the element <multi_account> under <optimization_setting> section. Each <multi_account> element contains universe_portfolio_id, account_id,

initial_portfolio_id, portfolio_base_value, and cashflow_weight attributes that allow you to set them for each account. For example, to add account_id 1 and account_id 2:

```
  <optimization_setting>
      <multi_account account_id="1" cashflow_weight="0" initial_portfolio_id="Initial
Portfolio" portfolio_base_value="100000" universe_portfolio_id="Trade Universe"/>
      <multi_account account_id="2" cashflow_weight="0" initial_portfolio_id="Initial
Portfolio2" portfolio_base_value="300000" universe_portfolio_id="Trade Universe2”/>
  </optimization_setting>
```

The universe_portfolio_id attribute may be omitted if the account's trade universe is the same as that of ALL_ACCOUNT.

The output will be saved under <rebalance_result>, which will contain a <optimal_portfolio> section for each account and will have information on optimal portfolio's statistics and holdings. The cross-account result will have account_id equal to -1. The <profile_diagnostics> element will also contain constraint slack information for each account. Multiple-Account Optimization does not report post-optimization roundlotted portfolio and asset-level transaction costs, therefore <asset_details> element will not be available.

8.10 Using XML APIs

All the optimizer XML data can be created via Java and C# APIs. The Java library can be imported into MATLAB. Each XML element is represented by a message class and a corresponding builder class for creating message class instances. The message object is immutable, and you should use the builder for any modification to the message object. After the optimizer message objects are constructed, they can be serialized and passed to optimizer, which creates a RebalanceResult message that stores the results from an optimization.

To create a Message object in Java, using RebalanceProfile as an example, do the following:

1. Create a Builder object for RebalanceProfile:

```
RebalanceProfile.Builder profileBuilder = RebalanceProfile.newBuilder();
```

2. Set the rebalance profile data through the Builder. The set methods return back a reference to the modified Builder object, allowing multiple set method calls in one line:

```
profileBuilder .setRebalanceProfileId("Case 1a");
profileBuilder.setUtility(Utility.newBuilder()
            .setUtilityType(EUtilityType.QUADRATIC)
            .addRiskTerm(RiskTerm.newBuilder()
                        .setIsPrimaryRiskModel(true)
                        .setCommonFactorRiskAversion(0.0075)
                        .setSpecificRiskAversion(0.0075)));
```

3. After all the necessary data has been set, call the Build method to create RebalanceProfile object and add it to the WorkSpace message:

```
wsBuilder.setRebalanceProfiles(RebalanceProfiles.newBuilder()
                    .addRebalanceProfile(profileBuilder.build()));
```

4. Serialize the WorkSpace message to byte array and run optimization:

```
byte[] out = OpenOptimizer.Run(wsBuilder.build().toByteArray());
```

5. Convert the byte array to RebalanceResult message:

```
RebalanceResult result = RebalanceResult.parseFrom(out);
```

Below are some of the commonly used methods for Message and Builder classes:

Description	Java	C#
Create a builder object	<Message>.newBuilder()	<Message>.CreateBuilder()
Build a message object	<Builder>.build()	<Builder>.Build()
Create a builder from message	<Message>.toBuilder()	<Message>.ToBuilder()
Write to output stream	<Message>.writeTo()	<Message>.WriteTo()
Read from input stream	<Message>.parseFrom()	<Message>.ParseFrom()
Package to import	com.barra.openopt.protobuf.*	optimizer.proto, Google.ProtocolBuffers

For more information about the Message interfaces, refer to:

- Java - https://developers.google.com/protocol-buffers/docs/reference/java/index
- C# - http://code.google.com/p/protobuf-csharp-port/downloads/list

Additional reference:

- XML Schema Reference Guide
- XML/JAVA API Reference Guide
- Protocol Buffers Interface File (<Installation Folder>doc/optimizer.proto)
- Google Protocol Buffers API Reference

The OpenOptimizer class enables you to pass data to the optimizer and creates the output. The following methods are provided within the class:

Parameters:

data_file	Serialized binary file from Data message
portfolio_file	Serialized binary file from Portfolios message
risk_model_file	Serialized binary file from RiskModels message
rebalance_profile_file	Serialized binary file from RebalanceProfiles message
rebalance_job_file	Serialized binary file from RebalanceJob message

rebalance_result_file Result returned from Optimizer, in serialized binary file from RebalanceResult message

Returns:

Status code of optimization

```
static EStatusCode com.barra.openopt.OpenOptimizer.Run  ( String  data_file,
  String  portfolio_file,
  String  risk_model_file,
  String  rebalance_profile_file,
  String  rebalance_job_file,
  String  rebalance_result_file
 )
```

Parameters:

workspace_file Serialized binary file from WorkSpace message

rebalance_result_file Result returned from Optimizer, in serialized binary file from RebalanceResult message

Returns:

Status code of optimization

```
static EStatusCode com.barra.openopt.OpenOptimizer.Run  ( String  workspace_file,
  String  rebalance_result_file
 )
```

Parameters:

workspaceByteArray Serialized byte array from WorkSpace message

Returns:

Serialized byte array from RebalanceResult message

```
static byte [] com.barra.openopt.OpenOptimizer.Run  ( byte[]  workspaceByteArray   )
```

8.10.1 Setting Up in C#

To use the APIs in C#, you will need to add the following references to your C# project:

- opsproto_cs.dll
- Google.ProtocolBuffers.dll
- Google.ProtocolBuffers.Serialization.dll

The interfaces to import for Optimizer APIs are:

```
using optimizer.proto;
using Google.ProtocolBuffers;
```

8.10.2 Setting Up in MATLAB

Follow the steps in the Working with MATLAB Interface section using Java jar file OpenOptJAVAAPI.jar instead of OptJAVAAPI.jar. Then, import and initialize the interfaces by calling:

```
import com.barra.openopt.*;
COptJavaInterface.Init;
```

The message and builder methods are identical to the Java interface. Because schema objects are structured in Java as inner classes, the MATLAB function javaMethod should be called first to obtain the inner class object. For example, to create a RebalanceProfile builder:

```
profileBuilder = javaMethod('newBuilder', 'com.barra.openopt.protobuf$RebalanceProfile');
```

Alternatively, you can import a helper function file newBuilder.m (under tutorials\matlab_proto), to shorten the method call by passing just the class name:

```
profileBuilder = newBuilder('RiskModel');
```

Then, you can directly call the setter and getter methods and set data through the builder object:

```
profileBuilder.setRebalanceProfileId ('Case 1');
```

To obtain the value of an enumeration type, for AttributeValueType.DOUBLE:

```
doubleAttrType =
javaMethod('valueOf','com.barra.openopt.protobuf$AttributeValueType','DOUBLE');
```

For faster performance by reducing number of method calls to Java from MATLAB, COptJavaInterface class provides several convenience methods that can take arrays of native MATLAB data type and populate the Builder object.

Data Type	Method	Description
static boolean	SetDoubleAttributeElements(protobuf.Attribute.Builder attribute, java.lang.String attributeId, java.lang.String[] assetIds, double[] values)	Sets the double attribute elements given an Attribute builder.
static boolean	SetFactorCovariances(protobuf.FactorCovariances.Builder factorCovariances, java.lang.String[] factorIds1, java.lang.String[] factorIds2, double[] covariances)	Sets the factor covariances given a FactorCovariances builder.
static boolean	SetFactorExposureForAsset(protobuf.FactorExposure.Builder factorExposure, java.lang.String assetId, java.lang.String[] factorIds, double[] exposures)	Sets the factor exposure array for an asset given a FactorExposure builder.
static boolean	SetIntegerAttributeElements(protobuf.Attribute.Builder attribute, java.lang.String attributeId, java.lang.String[] assetIds, int[] values)	Sets the integer attribute elements given an Attribute builder.

Data Type	Method	Description
static boolean	SetPortfolioHoldings(protobuf.Portfolio.Builder portfolio, java.lang.String portfolioId, java.lang.String[] assetIds, double[] amounts)	Sets the portfolio holdings given a Portfolio builder.
static boolean	SetSpecificCovariances(protobuf.SpecificCovariances.Builder specificCovariances, java.lang.String[] assetIds1, java.lang.String[] assetIds2, double[] covariances)	Sets the specific covariances given a SpecificCovariances builder.
static boolean	SetSpecificVariances(protobuf.SpecificVariances.Builder specificVariances, java.lang.String[] assetIds, double[] variances)	Sets the specific variances given a SpecificVariances builder.
static boolean	SetTextAttributeElements(protobuf.Attribute.Builder attribute, java.lang.String attributeId, java.lang.String[] assetIds, java.lang.String[] values)	Sets the text attribute elements given an Attribute builder.

9 Using the Command Line Executable

You can use the OpenOpt.exe program to perform the following operations:

1. Load Barra ModelsDirect data and convert it into binary/XML format.
2. Convert serialized workspace data from one format to another.
3. Run optimization and output result in the binary/XML format.

The executable determines the file type by its extension. If the file name ends with .xml, then it is determined as XML format, otherwise binary format. To load ModelsDirect data, an assetID universe file should always be provided to avoid loading asset risk data that is not part of the optimization universe.

9.1 Loading Barra Models Direct Data

Option	Parameter	Description
-md	See below	Load ModelsDirect files and convert to protobuf/xml format
-path	<ModelsDirect path>	Location of ModelsDirect files
-m	<model name>	Name of the risk model
-d	<analysis date>	Date starting from which, ModelsDirect files will be loaded
-id	<assetID file>	Text file containing list of asset IDs to load exposures and specific risk for, with new line starting with each BARRAID
-o	<output file>	Output risk model filename

9.2 Running Optimization

You can run an optimization either with each element of the workspace or with a single workspace. The following sections describe both the methods of running optimization.

9.2.1 Optimize With Each Element of the Workspace

Option	Parameter	Description
-run	See below	Run optimization given input protobuf/xml files
-data	<data file>	Filename containing the data section
-pf	<portfolios file>	Filename containing the portfolios section
-rm	<risk models file>	Filename containing the risk_models section
-rp	< rebalance profile file>	Filename containing the rebalance_profile section
-rj	< rebalance job file>	Filename containing the rebalance_job section

Option	Parameter	Description
-csv	<asset attribute file>	CSV file containing asset attribute data
-o	<output file>	Filename containing the rebalance_result section

9.2.2 Optimize With a Single Workspace

Option	Parameter	Description
-run	See below	Run optimization given input protobuf/xml files
-ws	<workspace file>	Filename containing the workspace section
-csv	<asset attribute file>	CSV file containing asset attribute data
-o	<output file>	Filename containing the rebalance_result section

Alternatively, optimization can be run in conjunction with ModelsDirect loader. The ModelsDirect risk model will be the primary risk model and overrides any existing risk model specified in the workspace XML. To run optimization with ModelsDirect loader, specify the following additional switches:

Option	Parameter	Description
-path	<ModelsDirect path>	Location of ModelsDirect files
-m	<model name>	Name of the risk model
-d	<analysis date>	Date starting from which, the ModelsDirect files will be loaded
-id	<assetID file>	Text file containing list of asset IDs to load exposures and specific risk for, with new line starting with each BARRAID

Examples:

- Run optimization with each element of workspace and USE4L ModelsDirect data:

  openopt.exe –run –data data.xml –pf portfolios.xml –rp profile.xml –rj job.xml –path \ModelsDirect –m USE4L –d 20140201 –id Universe.txt –o result.xml

- Run optimization with a single WorkSpace and USE4L ModelsDirect data:

  openopt.exe –run –ws workspace.xml –path \ModelsDirect –m USE4L –d 20140201 –id Universe.txt –o result.xml

9.3 Converting Protobuf ⇔ XML

Option	Parameter	Description
-convert	See below	Convert a protobuf binary file to XML file and vice versa

Option	Parameter	Description
-msg	<message name>	Full name of protobuf message including namespace, for ex. optimizer.proto.RiskModel
-i	<input file to convert>	Input protobuf/xml filename
-o	<converted output file>	Output protobuf/xml filename

9.4 Converting WSP → Protobuf/XML

Option	Parameter	Description
-convert	See below	Convert a WSP file to protobuf/XML file
-i	<input file to convert>	Input WSP filename
-o	<converted output file>	Output protobuf/xml filename

9.5 Converting WorkSpace Protobuf/XML → WSP

Option	Parameter	Description
-convert	See below	Convert a WorkSpace Protobuf/XML file to WSP file
-i	<input file to convert>	Input WorkSpace Protobuf/XML filename
-o	<converted WSP file>	Output WSP filename

9.6 Converting WSP => Legacy XML format

Option	Parameter	Description
-convert	See below	Convert a WSP file to legacy XML file
-i	<input file to convert>	Input WSP filename
-oxml	<converted output file>	Output xml filename

9.7 Additional Options

Option	Parameter	Description
-v	<verbose level for logging>	0=Error, 1=Warning, 2=Info, 3=Debug
-l	<log file>	Full path and filename of log file
-debug	<debug file>	WSP file for troubleshooting in optimization mode

10 Frequently Asked Questions

10.1 Why do I get different portfolios from different Optimizer versions?

This is because Barra Optimizer's goal is to maximize the objective function. It pays no special attention to the optimal weights in the resulting portfolio unless there is a constraint, penalty, or transaction cost to limit the turnover or any deviation. Two feasible portfolios are considered equivalent if their objective value difference is within tolerance, even if they differ greatly in weight.

For convex problems, multiple optimal solutions may exist due to numerical precision and the nature of the problem. Barra Optimizer does not guarantee to return a particular solution because all these optimal solutions are considered equivalent and are globally optimal. Differences in computing architecture, risk models, or optimality tolerances between versions may lead to a different optimal solution being returned. The objective values should be the same or slightly different.

For non-convex, nonlinear, or discrete problems, path-dependent heuristics are employed. Barra Optimizer may get stuck with a different local optimal portfolio when it tunes or enhances the heuristics from one version to another. These heuristic portfolios may be drastically different from one version to another. Even though the overall heuristic quality should improve in a newer version, it is possible to see degenerating quality in a particular heuristic solution.

10.2 Why do I get a suboptimal or counterintuitive solution?

For non-convex, nonlinear, or discrete problems, it is impossible or impractical to obtain a global optimal solution. This is due to the complex nature of the problem and the limits of computing power. Barra Optimizer employs heuristics to tackle these problems.

10.3 Which features may lead to suboptimal or counterintuitive solutions?

Features that return a heuristic portfolio may lead to suboptimal or counterintuitive solutions from time to time. For general guidelines, refer to Appendix A.

10.4 Which feature combinations are supported in Barra Open Optimizer?

For information about the supported features, refer to Appendix B.

10.5 Is there any limit on the number of risk terms in the objective function?

No, as long as all risk terms come from not more than two risk models.

10.6 Is there any limit on the number of risk constraints?

No. However, the number of risk constraints may negatively impact the speed.

10.7 Is there any limit on the number of risk models?

Yes. The number of risk models underlying a given problem, either in the objective or in the constraints, is limited to two. Currently, we are investigating applications with more than two risk models.

10.8 Is there any limit on the number of benchmarks?

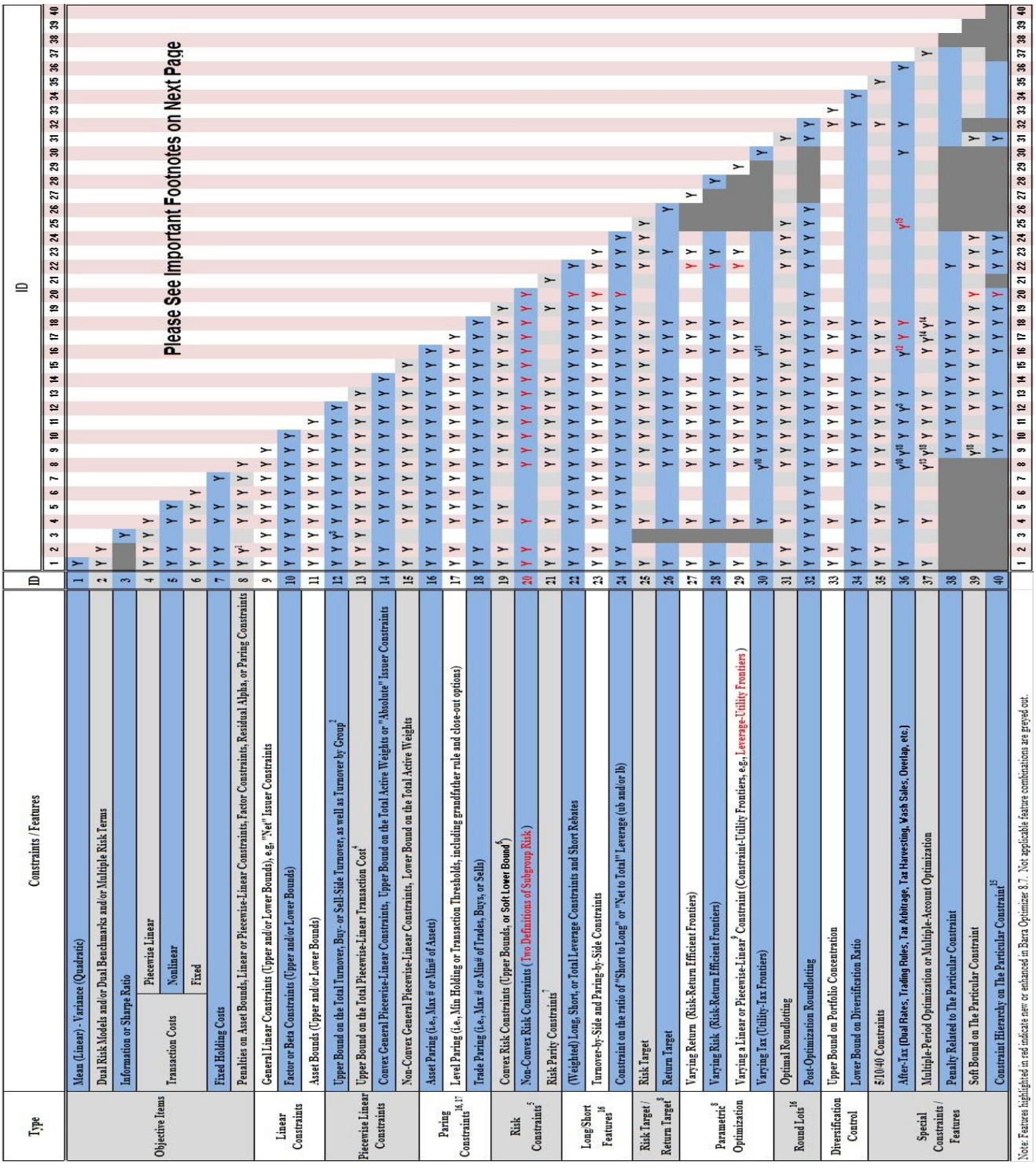
No. Multiple benchmarks are allowed in either the objective or constraints.

Appendix A: Explanation of Outputs

Problem Type	Output Portfolio(s)		Notes
	Optimal	Heuristic	
Standard Portfolio Optimization (Single- or Multiple-Period)	√		If a problem is infeasible, Barra Optimizer will return the best portfolio it is able to find. It will also provide information on which constraints to relax in order for the problem to be feasible.
General Linear or Convex Quadratic Programming	√		
Optimization with Soft Constraints • Soft Standard Constraints • Soft Special Constraints	√	 √	Barra Optimizer will provide information on which, if any, soft constraints are violated.
Paring Optimization		√	1. If no solution satisfying all *standard* constraints can be found, the problem will be declared as infeasible. 2. If Barra Optimizer can detect that these special constraints are too tight, it will report so and return the best available portfolio. 3. If it cannot be sure whether these special constraints are too tight, yet still cannot find a solution satisfying these constraints as well as the other standard constraints, it will return the best available portfolio and report the optimization as failed.
Round Lot Optimization		√	
5/10/40 Optimization		√	
Risk Constrained Optimization • Convex Risk Constraints • Non-Convex Risk Constraints • Risk Parity Constraint	 √ √	 √	Barra Optimizer will report a risk upper bound as being too tight, or a desired lower bound as being unreasonable if it can detect so for sure.
Diversification Control • Concentration Limit Constraint • Diversification Ratio Limit Constraint	 √ √		Barra Optimizer will provide a solution satisfying the concentration or diversification ratio limit constraint if the problem is feasible.
Long/Short (Hedge) Optimization • Convex Long/Short Constraints • Non-Convex Long/short Constraints	 √	 √	Barra Optimizer may report hedge optimization as failed if it cannot find a feasible solution satisfying the leverage constraints.
Tax-Aware Optimization		√	Barra Optimizer may report tax constraints as "maybe too tight", if numerical problems occur.

Problem Type	Output Portfolio(s)		Notes
	Optimal	Heuristic	
Parametric Optimization • With Standard Constraints • With Special Constraints	√	√	1. If the range is too high for the return or too low for the risk, Barra Optimizer will return the portfolio closest to the range. It will also output a message indicating the range is unattainable, and suggest the best possible range. 2. If a feasible portfolio with lower risk (or higher return) and a better objective value exists outside the given range, Barra Optimizer will return the better portfolio and indicate that the given range is not on the efficient frontier.
Optimization with Ratio Objectives • Non-Negative Numerator • Negative Numerator	√	√	If the numerator is non-negative and a solution exists, the problem is convex and Barra Optimizer will be able to return an optimal solution. Otherwise, a heuristic solution will be returned.

Appendix B. Availability of Features and Functions



Type	Constraints / Features	ID
Objective Items	Mean (Linear) - Variance (Quadratic)	1
	Dual Risk Models and/or Dual Benchmarks and/or Multiple Risk Terms	2
	Information or Sharpe Ratio	3
	Transaction Costs: Piecewise Linear	4
	Transaction Costs: Nonlinear	5
	Transaction Costs: Fixed	6
	Fixed Holding Costs	7
	Penalties on Asset Bounds, Linear or Piecewise-Linear Constraints, Factor Constraints, Residual Alpha, or Paring Constraints	8
Linear Constraints	General Linear Constraints (Upper and/or Lower Bounds), e.g., "Net" Issuer Constraints	9
	Factor or Beta Constraints (Upper and/or Lower Bounds)	10
	Asset Bounds (Upper and/or Lower Bounds)	11
Piecewise Linear Constraints	Upper Bound on the Total Turnover, Buy- or Sell-Side Turnover, as well as Turnover by Group[2]	12
	Upper Bound on the Total Piecewise-Linear Transaction Cost[4]	13
	Convex General Piecewise-Linear Constraints, Upper Bound on the Total Active Weights or "Absolute" Issuer Constraints	14
	Non-Convex General Piecewise-Linear Constraints, Lower Bound on the Total Active Weights	15
Paring Constraints[16,17]	Asset Paring (i.e., Max # or Min# of Assets)	16
	Level Paring (i.e., Min Holding or Transaction Thresholds, including grandfather rule and close-out options)	17
	Trade Paring (i.e., Max # or Min# of Trades, Buys, or Sells)	18
Risk Constraints[5]	Convex Risk Constraints (Upper Bounds, or Soft Lower Bound[6])	19
	Non-Convex Risk Constraints (Two Definitions of Subgroup Risk)	20
	Risk Parity Constraints[7]	21
Long/Short Features[16]	(Weighted) Long, Short, or Total Leverage Constraints and Short Rebates	22
	Turnover-by-Side and Paring-by-Side Constraints	23
	Constraint on the ratio of "Short to Long" or "Net to Total" Leverage (ub and/or lb)	24
Risk Target / Return Target[8]	Risk Target	25
	Return Target	26
Parametric[3] Optimization	Varying Return (Risk-Return Efficient Frontiers)	27
	Varying Risk (Risk-Return Efficient Frontiers)	28
	Varying a Linear or Piecewise-Linear[8] Constraint (Constraint-Utility Frontiers, e.g., Leverage-Utility Frontiers)	29
	Varying Tax (Utility-Tax Frontiers)	30
Round Lots[16]	Optimal Roundlotting	31
	Post-Optimization Roundlotting	32
Diversification Control	Upper Bound on Portfolio Concentration	33
	Lower Bound on Diversification Ratio	34
Special Constraints / Features	5/10/40 Constraints	35
	After-Tax (Dual Rates, Trading Rules, Tax Arbitrage, Tax Harvesting, Wash Sales, Overlap, etc.)	36
	Multiple-Period Optimization or Multiple-Account Optimization	37
	Penalty Related to The Particular Constraint	38
	Soft Bound on The Particular Constraint	39
	Constraint Hierarchy on The Particular Constraint[15]	40

Note: Features highlighted in red indicate new or enhanced in Barra Optimizer 8.7. Not applicable feature combinations are greyed out.

Important Notes:

[1] Residual alpha only applies to the primary risk model.

[2] There is an option to exclude cash from turnover definitions now. A separate option to count future assets in turnover constraints is also available.

[3] Buy- or sell-side turnover cannot be combined with ratio objective or after-tax features.

[4] Nonlinear or fixed transaction costs are not supported in transaction cost limit constraints.

[5] Dual risk models and/or multiple benchmarks are supported in risk constraints

[6] Soft lower bound on only one risk constrarint defined by the primary risk model is supported.

[7] Currently, soft bounds are not supported in risk parity cases.

[8] Soft bounds on linear or convex piecewise-linear constraints are now supported in parametric cases.

[9] "Piecewise-Linear" only refers to turnover, transaction costs, or leverage constraints here.

[10] Penalties on asset bounds and paring constraints are not supported with after-tax features.

[11] The only paring constraint that is supported in utility-tax frontiers is the "Max # of Assets" constraint.

[12] Combinations with the new 8.0 paring features (e.g., group-level paring constraints, grandfather rule, close-out small trades) are not supported.

[13] Peanlties on residual alpha and paring constraints are not supported in Multiple-Period Optimization and Multiple Account Optimization

[14] Trade paring and transaction level paring constraints are supported for all accounts in Multiple-Account Optimization, but only supported for the first period in Multiple-Period Optimization.

[15] No constraint hierarchy features can be combined with risk parity, parametric or multiple-period optimization cases.

[16] By default, future assets are excluded from paring, hedge and roundlot constraints. There are separate options to include future in these constraints now.

[17] Users have the option to obtain an upper bound on utility for certain paring cases.

[18] Combinations with Issuer Constraints are not supported

[19] The "trade-off wash sales" option is currently not supported with tax-aware risk target optimization.

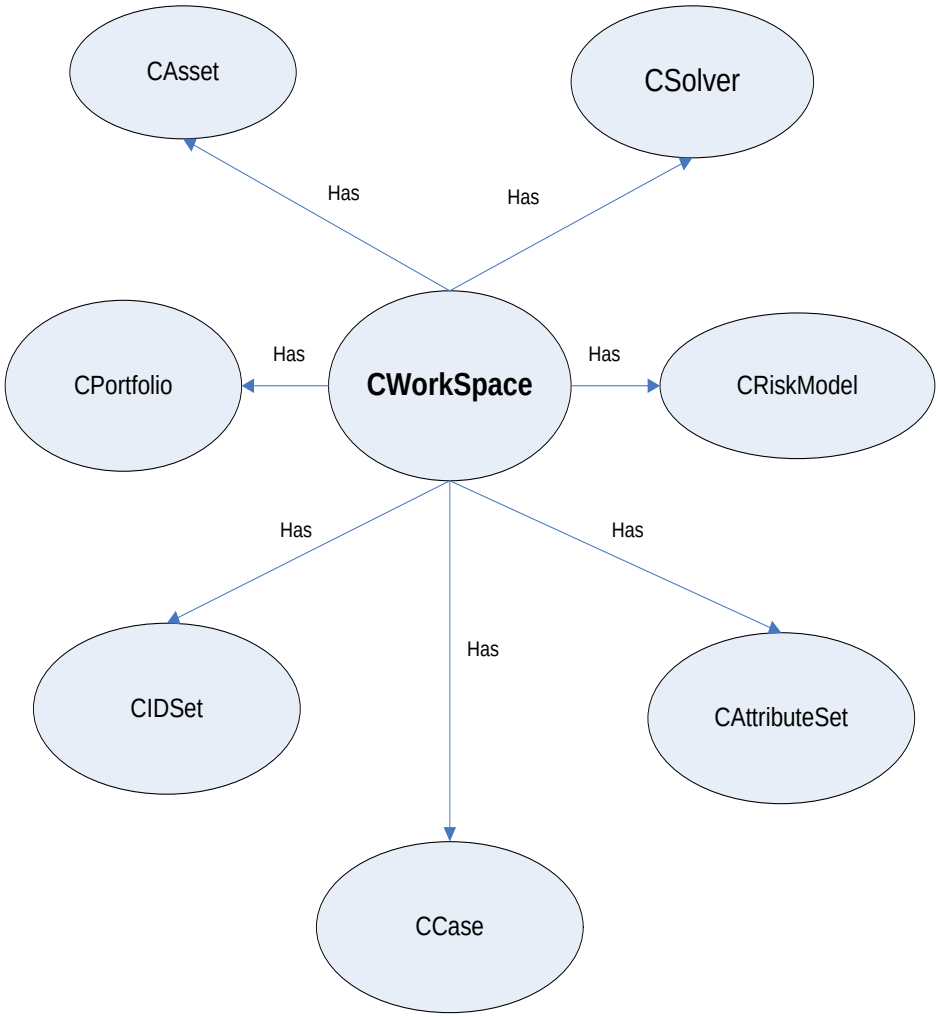
Appendix C: Glossary

Term	Definition
Trade universe	A list of assets that can be considered for inclusion in the optimal portfolio..
Initial portfolio	The portfolio to be optimized. Assets in the initial portfolio that are not in the trade universe can be considered for inclusion in the optimal portfolio. When there is no trade universe, only assets in the initial portfolio can be considered for inclusion in the optimal portfolio.
Investment universe	The union of the initial portfolio and the trade universe; a list of all assets eligible for inclusion in the optimal portfolio.
Benchmark	A portfolio of assets that the optimal portfolio will be compared against. Note: The assets in a benchmark may fall out of the investment universe.
Constraint slack	A constraint $l \le g(h) \le u$ can be rewritten as $g(h) - s = 0, \quad l \le s \le u$ where, s is called a constraint slack variable.
Asset weight	The position of an asset value compared with the base value: the asset value is divided by the base value and represented as a decimal number.
Base value	A reference number used in computing asset weights. Typically, you can choose the initial portfolio value, or the initial portfolio value plus the cash flow value, as the base value. You can also choose an arbitrary “assigned value” as the base value.
Active risk	Tracking error; a measurement of portfolio risk with respect to a benchmark.
Total risk	The standard deviation of the portfolio returns.

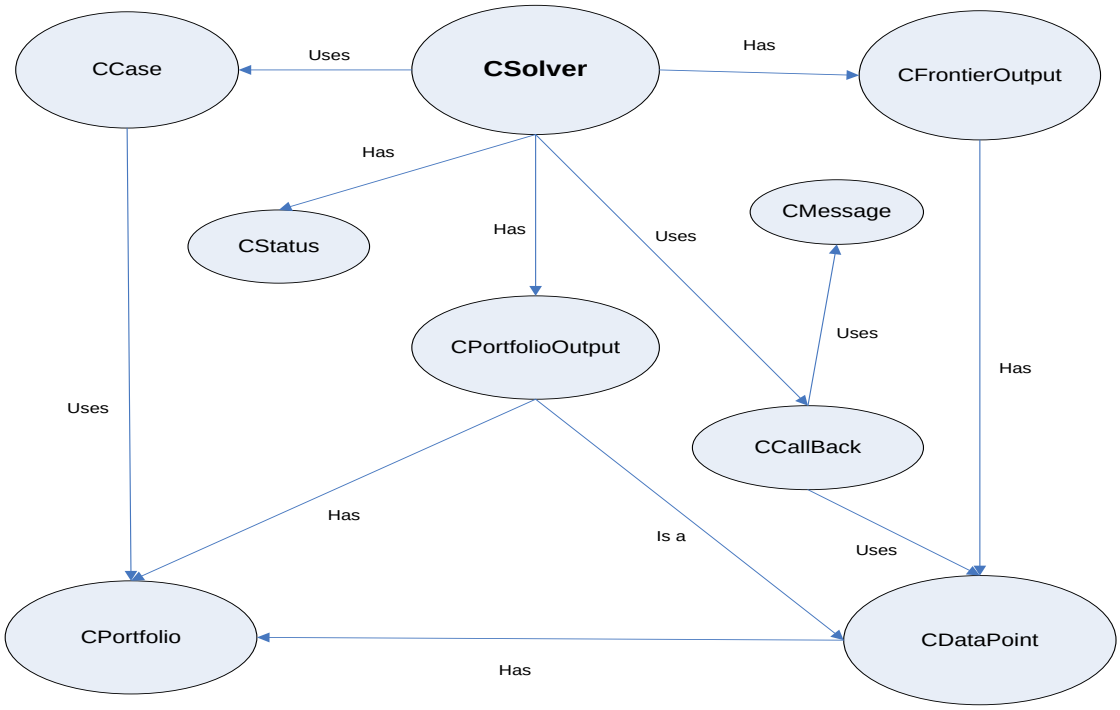
Appendix D: Object Model

The Barra Optimizer C++ API provides a set of classes representing entities used in portfolio optimization. By instantiating these classes and calling their methods, your application can define and solve a portfolio optimization case.

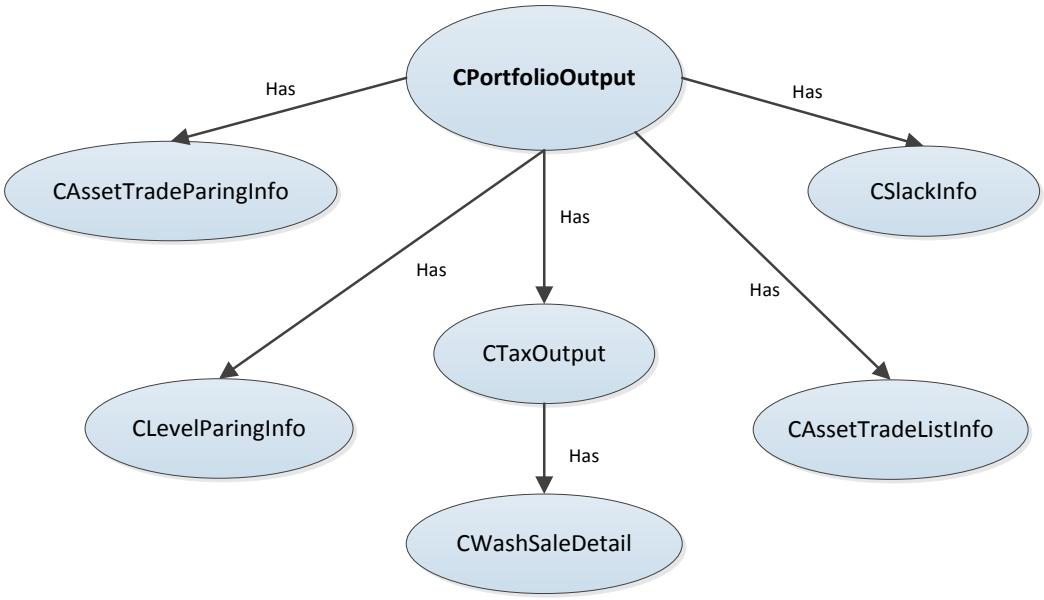
The following figure gives an overview of the major classes in the API:



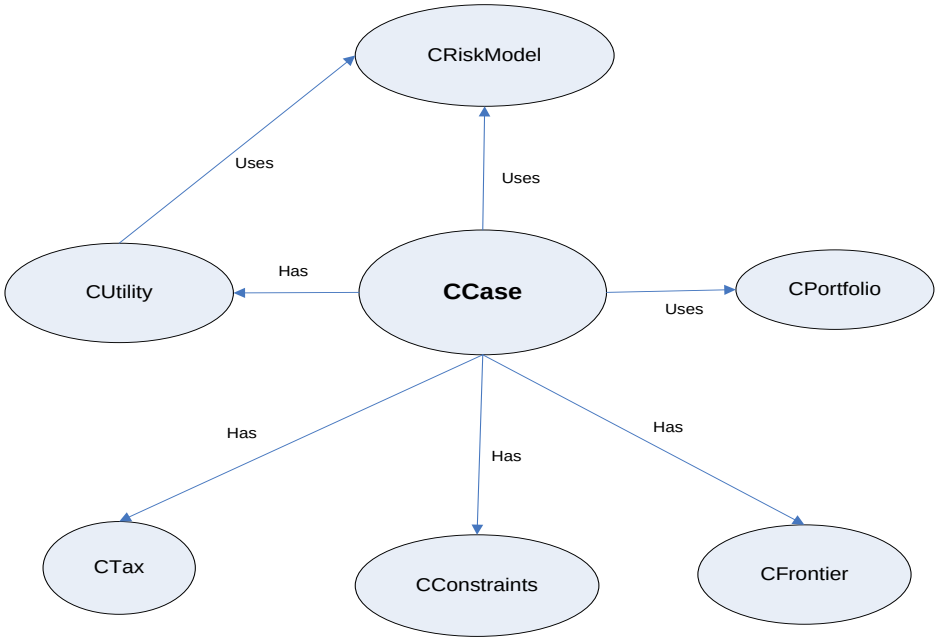
The following figure shows the relationships among the solver-related classes:



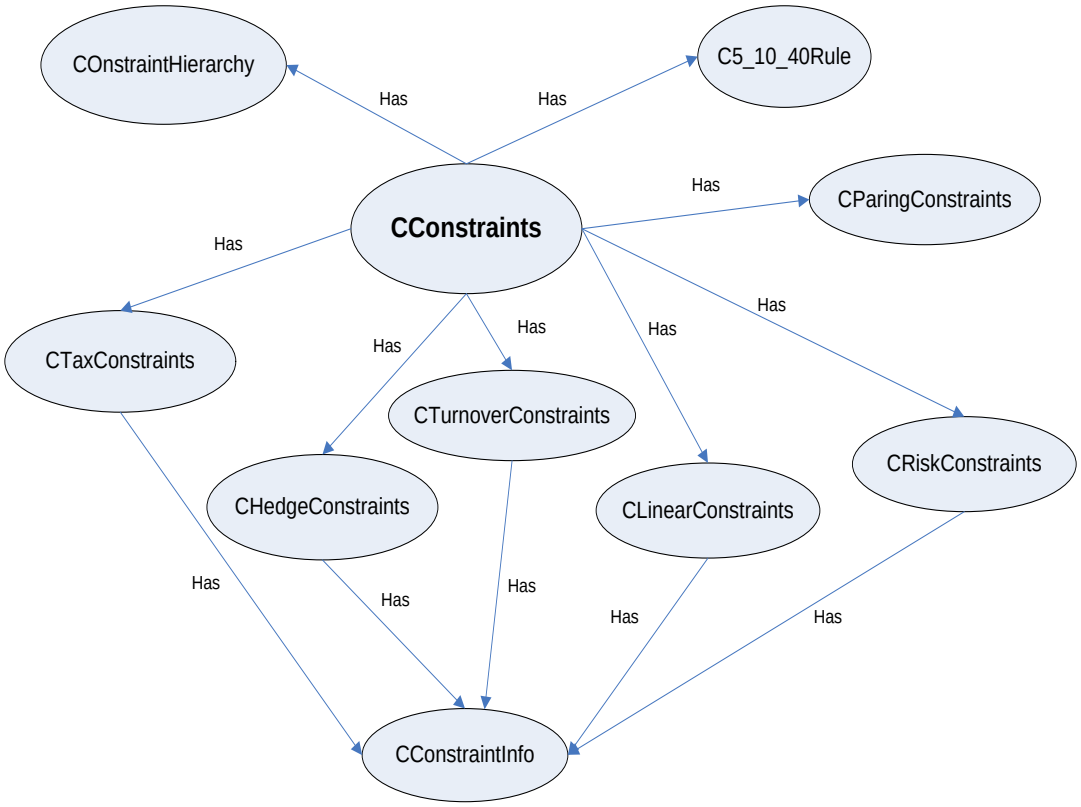
The following figure shows the relationships among the portfolio output-related classes:



The following figure shows the relationships among the case-related classes:



The following figure shows the relationships among the constraints-related classes:

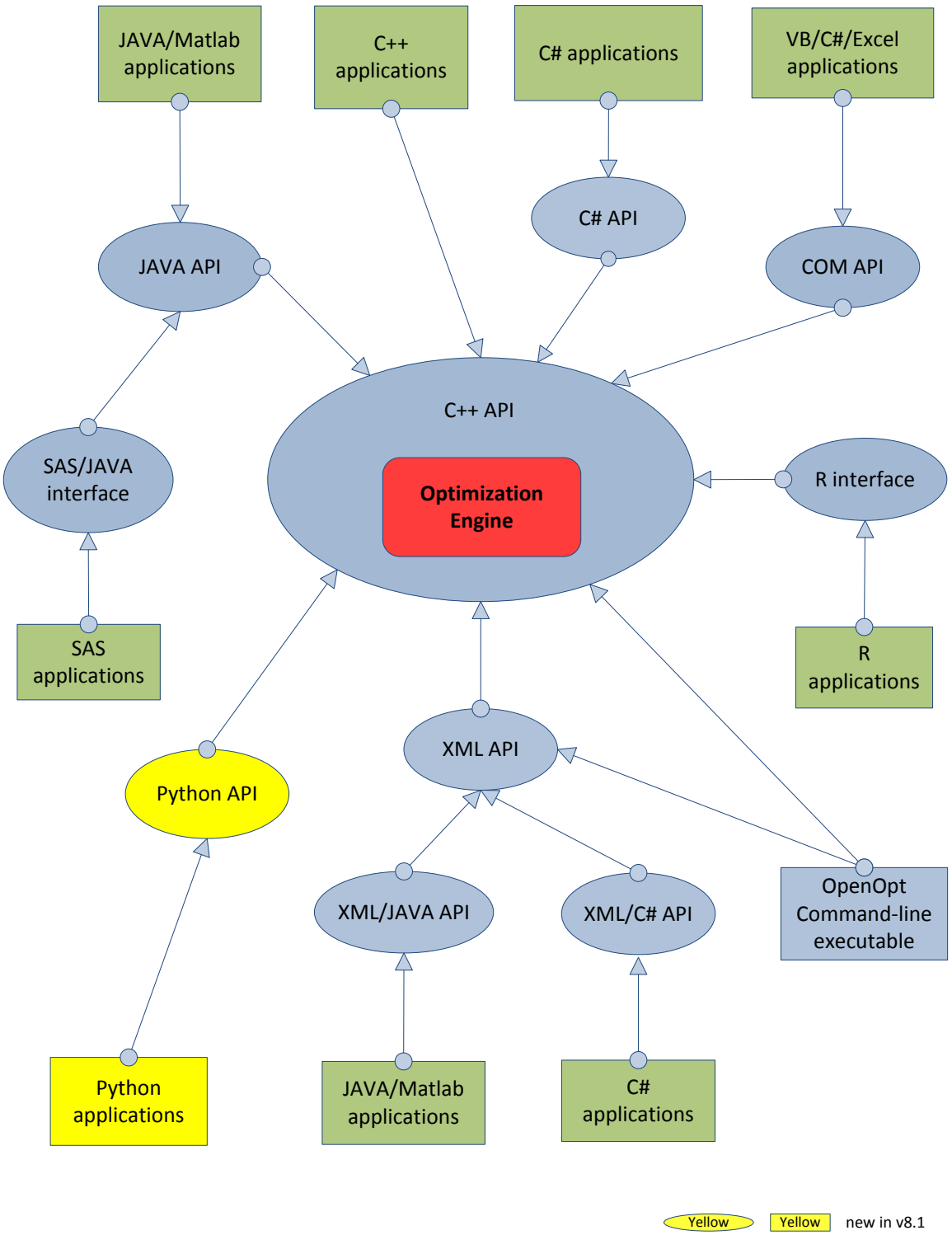


Appendix E: API Class References

Class	Description	Class References			
C5_10_40Rule	Represents 5/10/40 rule	C++	Java	C#	COM
CAsset	Contains details of an asset	C++	Java	C#	COM
CAssetTradeListInfo	Contains trade list information	C++	Java	C#	COM
CAssetTradeParingInfo	Describes asset/trade paring information of the optimal portfolio	C++	Java	C#	COM
CAttributeSet	Represents a set of key-value pairs	C++	Java	C#	COM
CCallBack	Used for setting callback functions	C++	Java	C#	N/A
CCase	Represents an optimization case	C++	Java	C#	COM
CConstraintHierarchy	Represents constraint hierarchy	C++	Java	C#	COM
CConstraintInfo	Contains constraint settings	C++	Java	C#	COM
CConstraints	Represents a collection of constraints	C++	Java	C#	COM
CCrossAccountConstraints	Represents cross-account constraints for Multiple-Account Optimization	C++	Java	C#	N/A
CDataPoint	Represents an optimal point on the efficient frontier	C++	Java	C#	COM
CDiversificationRatio	Represents diversification ratio	C++	Java	C#	COM
CFrontier	Represents an efficient frontier	C++	Java	C#	COM
CFrontierOutput	Contains output data for an efficient frontier	C++	Java	C#	COM
CHedgeConstraints	Represents long/short (hedge) constraints	C++	Java	C#	COM
CIDSet	Represents a set of IDs	C++	Java	C#	COM
CLevelParingInfo	Describes threshold-violation information	C++	Java	C#	COM
CLinearConstraints	Represents linear constraints	C++	Java	C#	COM
CIssuerConstraints	Represents issuer constraints	C++	Java	C#	COM
CMessage	Represents intermediate messages	C++	Java	C#	N/A
CMultiPeriodOutput	Represents output of multiple period optimization	C++	Java	C#	N/A
CMultiAccountOutput	Represents output of multi-account optimization.	C++	Java	C#	N/A
CParingConstraints	Represents paring (threshold and cardinality)	C++	Java	C#	COM

Class	Description	Class References			
	constraints				
CParingRange	Represents constraint settings for asset/trade paring constraint	C++	Java	C#	COM
CPortfolio	Represents a portfolio	C++	Java	C#	COM
CPortfolioOutput	Represents output of the optimal portfolio	C++	Java	C#	COM
CRiskConstraints	Represents risk constraints	C++	Java	C#	COM
CPortConcentrationConstraint	Represents portfolio concentration constraint	C++	Java	C#	COM
CRiskModel	Describes a factor risk model	C++	Java	C#	COM
CSlackInfo	Represents slack information for a constraint	C++	Java	C#	COM
CSolver	Executes the optimization and retrieves the results	C++	Java	C#	COM
CStatus	Describes the status of an optimization	C++	Java	C#	COM
CTax	Represents settings for tax-aware optimization	C++	Java	C#	COM
CTaxConstraints	Represents tax constraints	C++	Java	C#	COM
CTaxOutput	Contains tax-related results	C++	Java	C#	COM
CTurnoverConstraints	Represents turnover constraints	C++	Java	C#	COM
CUtility	Represents the utility function	C++	Java	C#	COM
CWashSaleDetail	Contains wash sale details	C++	Java	C#	COM
CWorkSpace	Represents a work space	C++	Java	C#	COM

Appendix F: API Diagram



References

Bender, J, J. Lee, and D. Stefek (March 2009) "Refining Portfolio Construction When Alphas and Risk Factors are Misaligned" MSCI Barra Research Insights.

Bender, J, J. Lee, and D. Stefek (June 2009) "Refining Portfolio Construction by Penalizing Residual Alpha—Empirical Examples" MSCI Barra Research Insights.

Bender, J, J. Lee, and D. Stefek (April 2010) "Constraining Shortfall" MSCI Barra Research Insights.

Bertsimas, D., G. Lauprete, and A. Samarov (2004) "Shortfall As a Risk Measure: Properties, Optimization and Applications" *Journal of Economic Dynamics & Control* 28, pp.1353-1381.

Fletcher, R. (2000) *Practical Methods of Optimization*, John Wiley & Sons, New York; 2nd edition.

Gill, P.E., W. Murray and M.H. Wright (1981) *Practical Optimization,* Academic Press, London-New York.

Grinold, R.C., and R.N. Kahn (1995) *Active Portfolio Management*, Probus Publishing Company.

Kopman, L., S. Liu, and D. Shaw (2009) "Using Lagrangian Relaxation to Obtain Small Portfolios" *The Journal of Portfolio Management*, Winter.

Kopman, L., and S. Liu (2009) "Risk Target Optimization." *Barra White Paper*.

Liu, S. (2004) "Practical Convex Quadratic Programming—Barra Optimizer for Portfolio Optimization." *Barra White Paper*.

Liu, S. and R. Xu (2017) "Introducing Multiple-Period Optimization—Breaking Through the Myopic Limitation of Traditional Mean-Variance Portfolio Optimization" *MSCI Product Insight*.

Liu, S. and R. Xu (2016) "When You Cannot Trade the Universe—Using Cardinality and Threshold Constraints in the Barra Optimizer" *MSCI Product Insight*.

Liu, S. and R. Xu (2010) "The Effects of Risk Aversion on Optimization—Demystifying Risk Aversion Parameters in Barra Optimizer." *Barra White Paper*.

Liu, S. and R. Xu (2014) "Managing the Unique Risks of Leverage with the Barra Optimizer: Theory and Practice—Leverage and Volatility Trade-offs in Long-Short Portfolio Construction." *MSCI Research Insight*.

Liu, S., A. Sheikh, and D. Stefek (1998) "Optimal Indexing." *Indexing for Maximum Investment Results,* John Wiley & Sons, New York.

Nemhauser, G.L., A.H.G. Rinnooy Kan, and M.J. Todd (1989) *Handbooks in Operations Research and Management Science, Vol 1: Optimization*, Elsevier Science Publishers B. V., North-Holland, London-New York.

Nemhauser, G. L. and L.A. Wolsey (1988) *Integer and Combinatorial Optimization*, John Willey & Sons, New York.

O'Cinneide, C., B. Scherer, and X. Xu (2006) "Pooling Trades in a Quantitative Investment Process", *The Journal of Portfolio Management*, Vol. 32, No. 4, pp 33-43.

Wolsey, L. A. (1998) *Integer Programming*, 1st edition, Wiley-Interscience.

Xu, R. and S. Liu (2011) "Portfolio Optimization with Trade Paring Constraints—A New Feature in the Barra Optimizer." *MSCI Research Insight*.

Yang, Y., F. Rubio, G. Scutari, and D. Palomar (2013) “IEEE Transactions on Signal Processing”, Vol. 61, No. 22 . November 15, 2013.

Xu, R. and S. Liu (2013) “Managing Odd Lot Trades with the Barra Optimizer—Roundlotting Tradeoffs in Portfolio Construction.” *MSCI Research Insight*.

CONTACT US

AMERICAS

Americas	1 888 588 4567 *
Atlanta	+ 1 404 551 3212
Boston	+ 1 617 532 0920
Chicago	+ 1 312 675 0545
Monterrey	+ 52 81 1253 4020
New York	+ 1 212 804 3901
San Francisco	+ 1 415 836 8800
Sao Paulo	+ 55 11 3706 1360
Toronto	+ 1 416 628 1007

EUROPE, MIDDLE EAST & AFRICA

Cape Town	+ 27 21 673 0100
Frankfurt	+ 49 69 133 859 00
Geneva	+ 41 22 817 9777
London	+ 44 20 7618 2222
Milan	+ 39 02 5849 0415
Paris	0800 91 59 17 *

ASIA PACIFIC

China North	10800 852 1032 *
China South	10800 152 1032 *
Hong Kong	+ 852 2844 9333
Mumbai	+ 91 22 6784 9160
Seoul	00798 8521 3392 *
Singapore	800 852 3749 *
Sydney	+ 61 2 9033 9333
Taipei	008 0112 7513 *
Thailand	0018 0015 6207 7181 *
Tokyo	+ 81 3 5290 1555

* = toll free

ABOUT MSCI

For more than 40 years, MSCI's research-based indexes and analytics have helped the world's leading investors build and manage better portfolios. Clients rely on our offerings for deeper insights into the drivers of performance and risk in their portfolios, broad asset class coverage and innovative research.

Our line of products and services includes indexes, analytical models, data, real estate benchmarks and ESG research.

MSCI serves 98 of the top 100 largest money managers, according to the most recent P&I ranking.

For more information, visit us at www.msci.com.

NOTICE AND DISCLAIMER

This document and all of the information contained in it, including without limitation all text, data, graphs, charts (collectively, the "Information") is the property of MSCI Inc. or its subsidiaries (collectively, "MSCI"), or MSCI's licensors, direct or indirect suppliers or any third party involved in making or compiling any Information (collectively, with MSCI, the "Information Providers") and is provided for informational purposes only. The Information may not be modified, reverse-engineered, reproduced or redisseminated in whole or in part without prior written permission from MSCI.

The Information may not be used to create derivative works or to verify or correct other data or information. For example (but without limitation), the Information may not be used to create indexes, databases, risk models, analytics, software, or in connection with the issuing, offering, sponsoring, managing or marketing of any securities, portfolios, financial products or other investment vehicles utilizing or based on, linked to, tracking or otherwise derived from the Information or any other MSCI data, information, products or services.

The user of the Information assumes the entire risk of any use it may make or permit to be made of the Information. NONE OF THE INFORMATION PROVIDERS MAKES ANY EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS WITH RESPECT TO THE INFORMATION (OR THE RESULTS TO BE OBTAINED BY THE USE THEREOF), AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, EACH INFORMATION PROVIDER EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES (INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF ORIGINALITY, ACCURACY, TIMELINESS, NON-INFRINGEMENT, COMPLETENESS, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE) WITH RESPECT TO ANY OF THE INFORMATION.

Without limiting any of the foregoing and to the maximum extent permitted by applicable law, in no event shall any Information Provider have any liability regarding any of the Information for any direct, indirect, special, punitive, consequential (including lost profits) or any other damages even if notified of the possibility of such damages. The foregoing shall not exclude or limit any liability that may not by applicable law be excluded or limited, including without limitation (as applicable), any liability for death or personal injury to the extent that such injury results from the negligence or willful default of itself, its servants, agents or sub-contractors.

Information containing any historical information, data or analysis should not be taken as an indication or guarantee of any future performance, analysis, forecast or prediction. Past performance does not guarantee future results.

The Information should not be relied on and is not a substitute for the skill, judgment and experience of the user, its management, employees, advisors and/or clients when making investment and other business decisions. All Information is impersonal and not tailored to the needs of any person, entity or group of persons.

None of the Information constitutes an offer to sell (or a solicitation of an offer to buy), any security, financial product or other investment vehicle or any trading strategy.

It is not possible to invest directly in an index. Exposure to an asset class or trading strategy or other category represented by an index is only available through third party investable instruments (if any) based on that index. MSCI does not issue, sponsor, endorse, market, offer, review or otherwise express any opinion regarding any fund, ETF, derivative or other security, investment, financial product or trading strategy that is based on, linked to or seeks to provide an investment return related to the performance of any MSCI index (collectively, "Index Linked Investments"). MSCI makes no assurance that any Index Linked Investments will accurately track index performance or provide positive investment returns. MSCI Inc. is not an investment adviser or fiduciary and MSCI makes no representation regarding the advisability of investing in any Index Linked Investments.

Index returns do not represent the results of actual trading of investible assets/securities. MSCI maintains and calculates indexes, but does not manage actual assets. Index returns do not reflect payment of any sales charges or fees an investor may pay to purchase the securities underlying the index or Index Linked Investments. The imposition of these fees and charges would cause the performance of an Index Linked Investment to be different than the MSCI index performance.

The Information may contain back tested data. Back-tested performance is not actual performance, but is hypothetical. There are frequently material differences between back tested performance results and actual results subsequently achieved by any investment strategy.

Constituents of MSCI equity indexes are listed companies, which are included in or excluded from the indexes according to the application of the relevant index methodologies. Accordingly, constituents in MSCI equity indexes may include MSCI Inc., clients of MSCI or suppliers to MSCI. Inclusion of a security within an MSCI index is not a recommendation by MSCI to buy, sell, or hold such security, nor is it considered to be investment advice.

Data and information produced by various affiliates of MSCI Inc., including MSCI ESG Research Inc. and Barra LLC, may be used in calculating certain MSCI indexes. More information can be found in the relevant index methodologies on www.msci.com.

MSCI receives compensation in connection with licensing its indexes to third parties. MSCI Inc.'s revenue includes fees based on assets in Index Linked Investments. Information can be found in MSCI Inc.'s company filings on the Investor Relations section of www.msci.com.

MSCI ESG Research Inc. is a Registered Investment Adviser under the Investment Advisers Act of 1940 and a subsidiary of MSCI Inc. Except with respect to any applicable products or services from MSCI ESG Research, neither MSCI nor any of its products or services recommends, endorses, approves or otherwise expresses any opinion regarding any issuer, securities, financial products or instruments or trading strategies and MSCI's products or services are not intended to constitute investment advice or a recommendation to make (or refrain from making) any kind of investment decision and may not be relied on as such. Issuers mentioned or included in any MSCI ESG Research materials may include MSCI Inc., clients of MSCI or suppliers to MSCI, and may also purchase research or other products or services from MSCI ESG Research. MSCI ESG Research materials, including materials utilized in any MSCI ESG Indexes or other products, have not been submitted to, nor received approval from, the United States Securities and Exchange Commission or any other regulatory body.

Any use of or access to products, services or information of MSCI requires a license from MSCI. MSCI, Barra, RiskMetrics, IPD, FEA, InvestorForce, and other MSCI brands and product names are the trademarks, service marks, or registered trademarks of MSCI or its subsidiaries in the United States and other jurisdictions. The Global Industry Classification Standard (GICS) was developed by and is the exclusive property of MSCI and Standard & Poor's. "Global Industry Classification Standard (GICS)" is a service mark of MSCI and Standard & Poor's.