Constructing Sparse Replicating Portfolios for Insurance Liabilities by Regularized Optimization

Oleksandr Romanko

Quantitative Research, Algorithmics
Dept. of Computing and Software, McMaster University

Joint work with **Curt Burmeister** and **Helmut Mausser**

UTORG Seminar, November 24, 2010

Outline

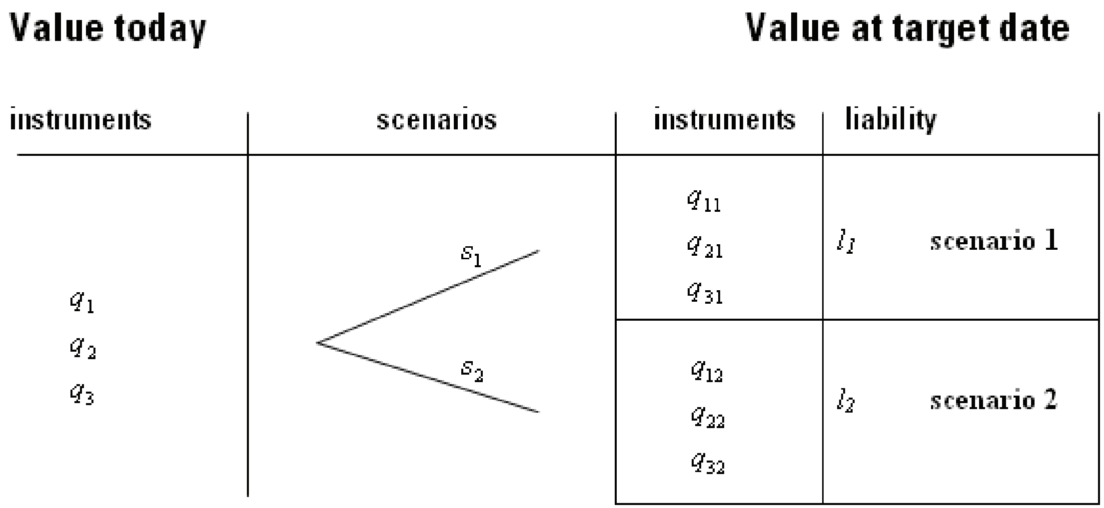
- **Portfolio replication** – selecting a computationally efficient proxy for a portfolio of liabilities (insurance policies)
  - portfolio replication background
  - stochastic scenarios, sparsity requirement
  - optimization formulations
  - trading costs vs. cardinality constraints
  - regularized optimization
  - alternative trading costs
- **Practical implementation issues**
  - computing efficient frontiers
  - selecting the final replicating portfolio
- **Case study** – constructing sparse replicating portfolios
  - replicating universe and optimization setup
  - weighted regularization illustration, comparison
- **Conclusions and future work**

Algorithmics

📖 Portfolio Replication

Replicating Portfolios

- "Find a portfolio of assets whose value is equal to the value of a liability portfolio under today's market conditions and future market conditions (scenarios)"



- Matching values vs. matching cash flows
- Applications include computationally intensive calculations:
  - Economic Capital
  - Regulatory Capital (Solvency II, RBC)
  - Risk Management (VaR, Tail-VaR)

Portfolio Replication

- Use optimization to find a portfolio of market securities whose future cash flows matches the future cash flows of insurance policies (liabilities)
- Pick asset holdings x_j to minimize:

$$\min_{x} \sum_{t=1}^{T} \sum_{i=1}^{S} p_i \varrho_i^t \left\| \sum_{j=1}^{N} c_{ij}^t x_j - c_{i0}^t \right\|$$

cash flow of instrument ***j*** in scenario ***i*** at time ***t***

liability is instrument **0**

Sum over time steps

Sum over scenarios

number of units of instrument ***j*** in the replicating portfolio

Deviation measure:

- Mean Absolute Error
- Mean Square Error
- …

- The replicating portfolio is used as a proxy for the liabilities

Algorithmics

Portfolio Replication

- Pick asset holdings x_j to minimize:

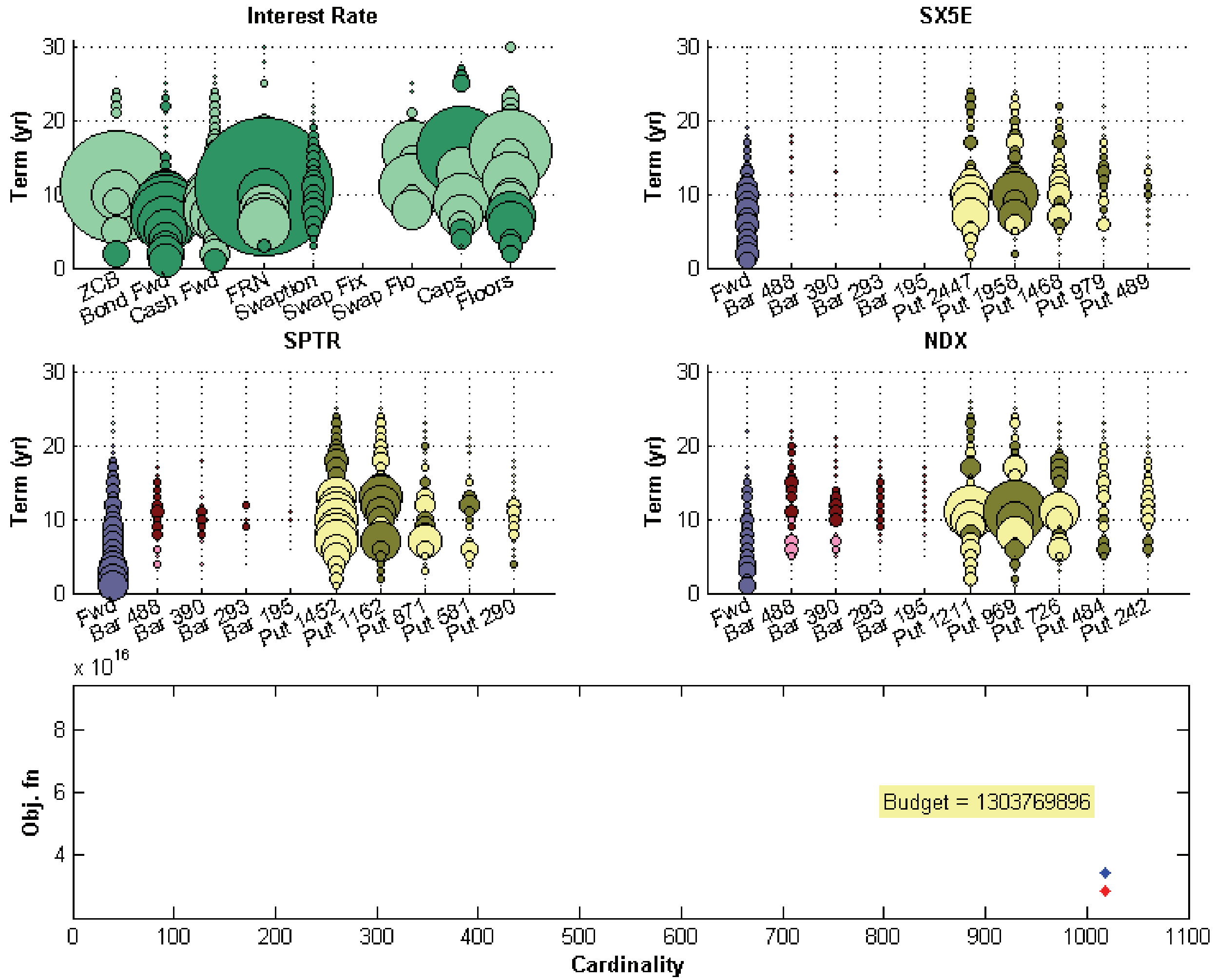
$$\min_x \sum_{t=1}^{T} \sum_{i=1}^{S} p_i \left\| \sum_{j=1}^{N} c_{ij}^t x_j - c_{i0}^t \right\|$$

	Bucketed PV Cash Flows	Annual PV Cash Flows
Mean Absolute Error (Match Linear)	$\min_x \frac{1}{S} \sum_{i=1}^{S} \left\| \sum_{j=1}^{N} c_{ij} x_j - c_{i0} \right\|$	$\min_x \frac{1}{S} \sum_{t=1}^{T} \sum_{i=1}^{S} \left\| \sum_{j=1}^{N} c_{ij}^t x_j - c_{i0}^t \right\|$
Mean Squared Error (Match Quadratic)	$\min_x \frac{1}{S} \sum_{i=1}^{S} \left(\sum_{j=1}^{N} c_{ij} x_j - c_{i0} \right)^2$	$\min_x \frac{1}{S} \sum_{t=1}^{T} \sum_{i=1}^{S} \left(\sum_{j=1}^{N} c_{ij}^t x_j - c_{i0}^t \right)^2$

$$\begin{aligned} \min_x \quad & \|Ax - b\| \\ \text{s.t.} \quad & x \in \Omega \end{aligned}$$

Algorithmics

Portfolio Composition and Trading Budget



Trading Budget Constraint

- The total number of units traded to construct the replicating portfolio cannot exceed budget ε:

unit-based trading costs $\sum_{j=1}^{N} |x_j| \leq \varepsilon \Rightarrow \|x\|_1 \leq \varepsilon$

$\Uparrow D = I$

***d*-based trading costs** $\sum_{j=1}^{N} d_j |x_j| \leq \varepsilon \Rightarrow \|Dx\|_1 \leq \varepsilon, \ \ D = \mathrm{diag}(d_j)$

- **Regularized optimization** – replace cardinality constraint with norm constraint

$$\begin{aligned} \min_{x} \quad & \|Ax - b\| \\ \text{s.t.} \quad & x \in \Omega \\ & \|Dx\|_1 \leq \varepsilon \end{aligned}$$

$$\mathrm{card}(x) = \textstyle\sum_i x_i^0 = \|x\|_0 \quad \text{with } 0^0 = 0$$

$$\|x\|_0 \leq K \Rightarrow \|x\|_1 \leq \varepsilon$$

Algorithmics

Alternative Trading Costs

Trading costs d_j:

- Unconstrained
  - solve unconstrained problem to obtain $\tilde{x}$
  - compute $d_j = 1/|\tilde{x}_j|$
- Marginal unconstrained
- Standard deviation scaling
  - compute standard deviation of the cash flows of each instrument σ_j
  - compute $d_j = \sigma_j/\sigma_0$
- Mean scaling
  - compute $d_j = |\mu_j|/|\mu_0|$
- Correlation scaling
  - compute $d_j = 1/|\rho_{j0}|$
- Beta scaling
  - compute $d_j = \sigma_j/|\rho_{j0}|\sigma_0$

$$\begin{aligned} \min_x \quad & \|Ax - b\| \\ \text{s.t.} \quad & x \in \Omega \\ & \|Dx\|_1 \leq \varepsilon \end{aligned}$$

$D = \operatorname{diag}(d_j)$

trading costs d_j are also called **regularization weights**

Portfolio Replication with Trading Budget Constraint

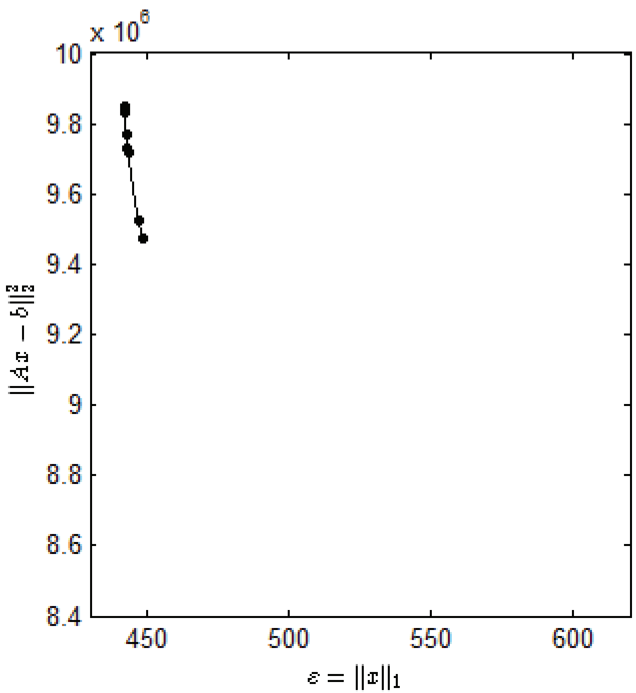
- Regularized optimization problem for portfolio replication:

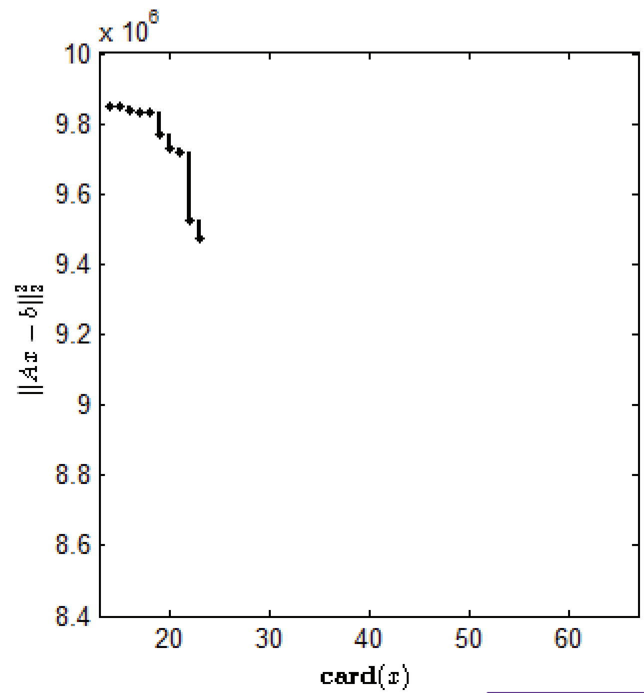
$$\begin{aligned} \min_{x} \quad & \|Ax - b\|_2^2 \\ \text{s.t.} \quad & x \in \Omega \\ & \|x\|_1 \leq \varepsilon \end{aligned}$$

liabilities, assets } cash flows

⬅ regularization constraint

- Regularized problem solution (exact efficient frontier):





Algorithmics

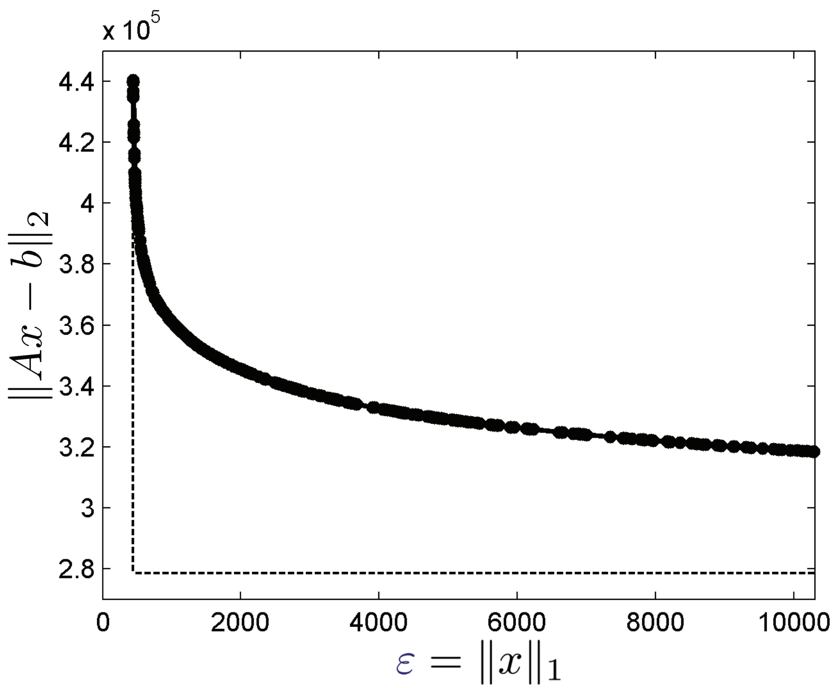
Portfolio Replication with Trading Budget Constraint

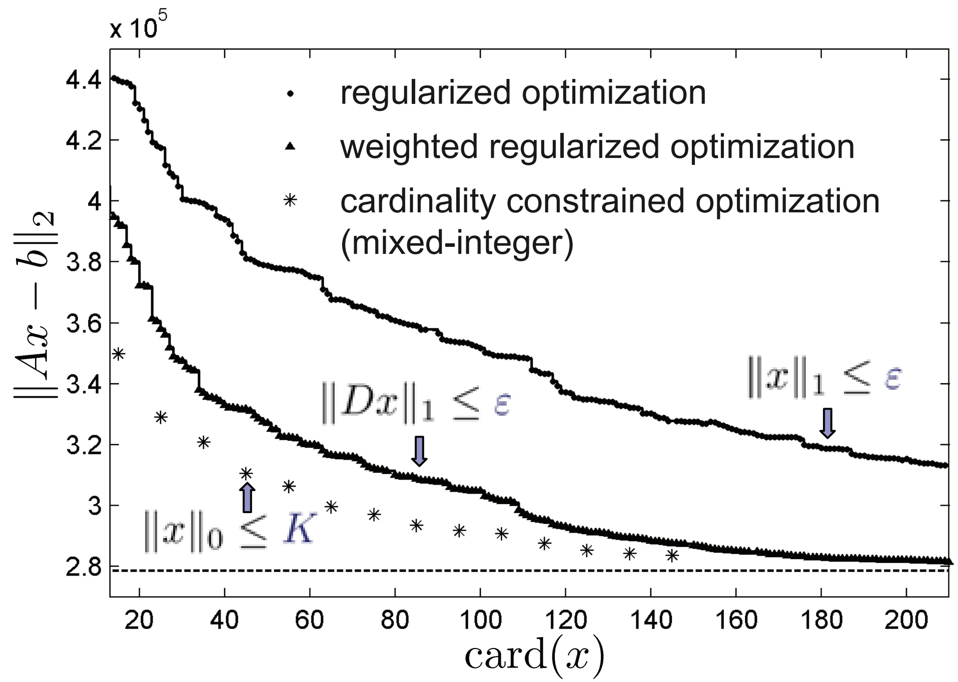
- Regularized optimization problem for portfolio replication:

$$
\begin{aligned}
\min_{x} \quad & \|Ax - b\|_2^2 \\
\text{s.t.} \quad & x \in \Omega \\
& \|Dx\|_1 \le \varepsilon
\end{aligned}
$$

⟸ regularization constraint

- Regularized problem solution (exact efficient frontier):





Algorithmics

📖 Practical Issues

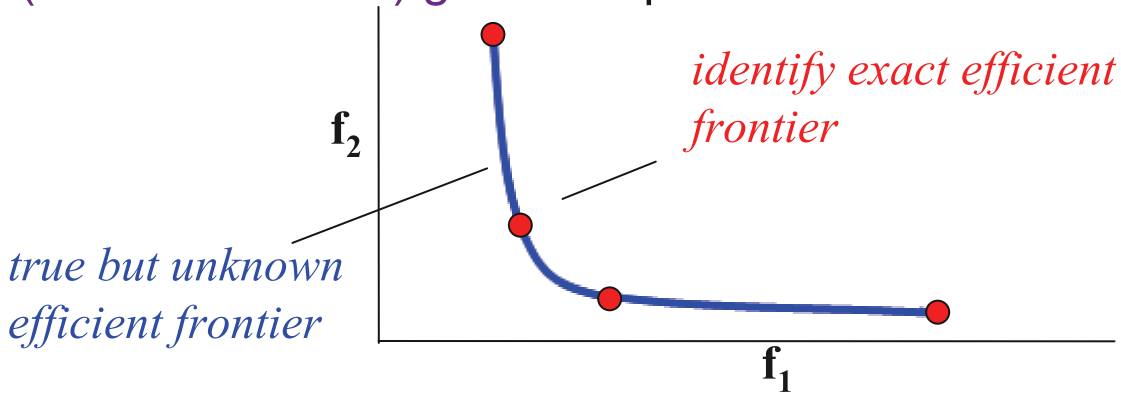
computing efficient frontiers

Computing Efficient Frontiers - Possibilities

Replicating portfolio analysis involves **computing the efficient frontier**, evaluating it out-of-sample and selecting the final replicating portfolio based on the minimal out-of-sample objective value

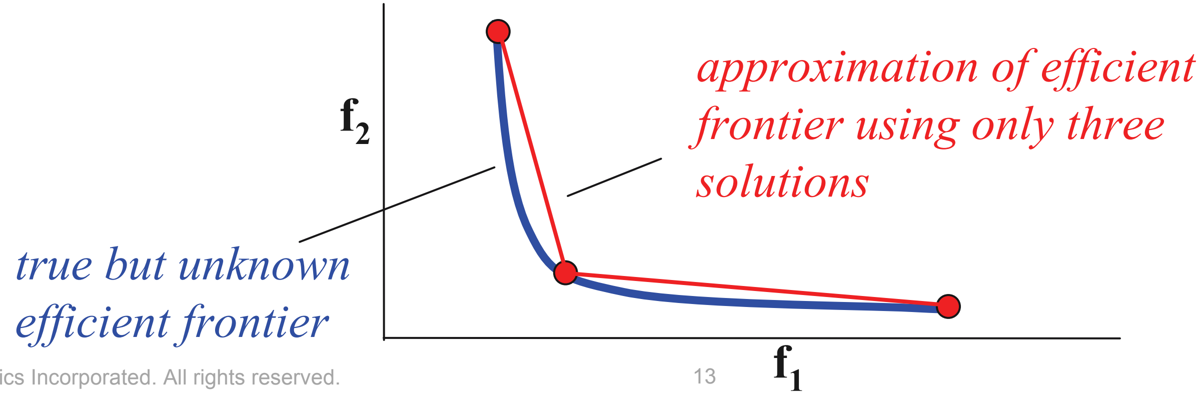
Computing efficient frontiers:

- Ideal (often unrealistic) goal: compute exact frontier



parametric optimization

- Typical (more realistic) goal: approximate the frontier



discretizing the frontier

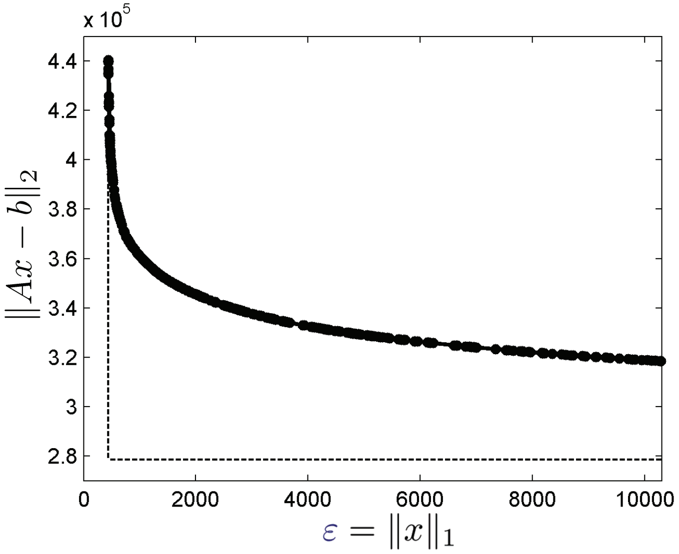
Algorithmics

Computing Efficient Frontiers - Possibilities

Replicating portfolio analysis involves **computing the efficient frontier**, evaluating it out-of-sample and selecting the final replicating portfolio based on the minimal out-of-sample objective value

Computing efficient frontiers:

- Exact frontier (parametric optimization):
  - produces exact frontier (in-sample)
  - may be slow as the number of intervals can be large
  - very sensitive to numerical difficulties



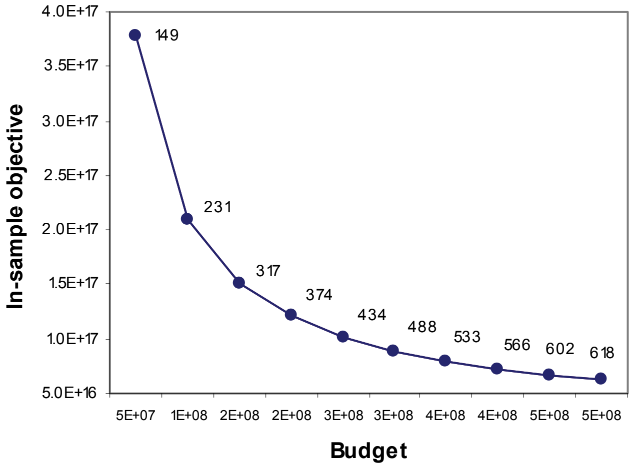
- Approximated frontier:
  - produces inexact frontier as we miss all information in-between the discretization points
  - can take advantage of the CPLEX warm-start

    Warm-start is based on the idea that some variables are already at their optimal values when solving a modified problem, so, an optimizer performs significantly fewer iterations when started from an initial solution

Algorithmics

Computing Efficient Frontiers – Examples

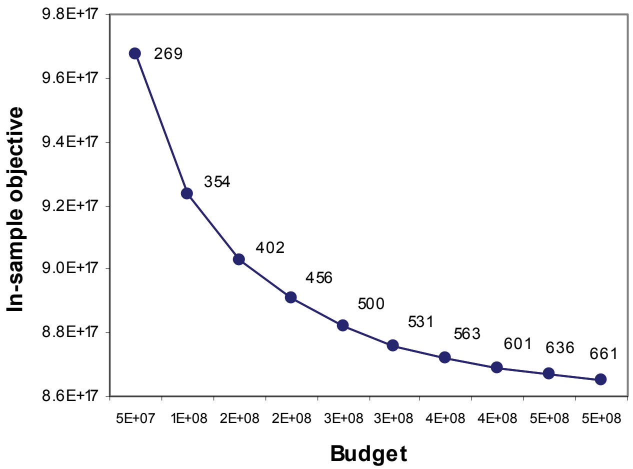
Quadratic matching, bucketed, reduced formulation:



- ✓ 1112 replicating instruments
- ✓ 2000 scenarios
- ✓ single 30-yr time bucket
- ✓ trading budget of 50 million units to 500 million units

Frontier	Solution		Frontier Point										Total Time
			1	2	3	4	5	6	7	8	9	10	
Increasing budget	Solution cardinality		149	231	317	374	434	488	533	566	602	618	
	Solve time	Warm start	10	5	3	3	3	3	4	4	4	4	**45**
		No start	10	19	36	72	115	163	223	287	357	382	**1663**
Decreasing budget	Solution cardinality		618	602	566	533	488	434	374	317	231	149	
	Solve time	Warm start	385	7	5	5	5	4	4	3	3	3	**422**
		No start	369	356	290	216	161	117	76	39	21	11	**1655**

Quadratic matching, annual, reduced formulation:



- ✓ 1112 replicating instruments
- ✓ 2000 scenarios
- ✓ 30 annual time steps
- ✓ trading budget of 50 million units to 500 million units

Frontier	Solution		Frontier Point										Total Time
			1	2	3	4	5	6	7	8	9	10	
Increasing budget	Solution cardinality		269	354	402	456	500	531	563	601	636	661	
	Solve time	Warm start	0.5	0.4	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	**3**
		No start	0.5	1.0	1.3	2.4	3.3	4.8	4.1	6.1	7.2	9.3	**40**
Decreasing budget	Solution cardinality		661	636	601	563	531	500	456	402	354	269	
	Solve time	Warm start	10.1	0.5	0.3	0.3	0.3	0.3	0.2	0.3	0.3	0.3	**13**
		No start	9.9	7.8	6.8	4.6	5.3	3.6	2.6	1.5	1.1	0.5	**44**

Algorithmics

Computing Efficient Frontiers – Examples

- Reduced formulation:

$$\begin{array}{ll}\min_x & \|Ax-b\|_2^2 \\ \text{s.t.} & x \in \Omega \\ & \|x\|_1 \le \varepsilon\end{array} \Rightarrow Q = \tfrac{1}{S}A^TA,\ u = \tfrac{2}{S}A^Tb,\ v = \tfrac{1}{S}b^Tb \Rightarrow \begin{array}{ll}\min_x & x^TQx - u^Tx + v \\ \text{s.t.} & x \in \Omega \\ & \|x\|_1 \le \varepsilon\end{array}$$

- Consider the following cash flows (S=3, N=2, T=2):

Time t	Scenarios	Instruments	
		$j=1$	$j=2$
Year 1	$i=1$	6	0
	$i=2$	9	0
	$i=3$	3	0
Year 2	$i=1$	0	1
	$i=2$	0	2
	$i=3$	0	4

- Bucketed cash flows:

$$A = \begin{bmatrix} 6 & 1 \\ 9 & 2 \\ 3 & 4 \end{bmatrix} \Rightarrow Q = \frac{1}{S}A^TA = \begin{bmatrix} 42 & 12 \\ 12 & 7 \end{bmatrix}$$

- Annual cash flows:

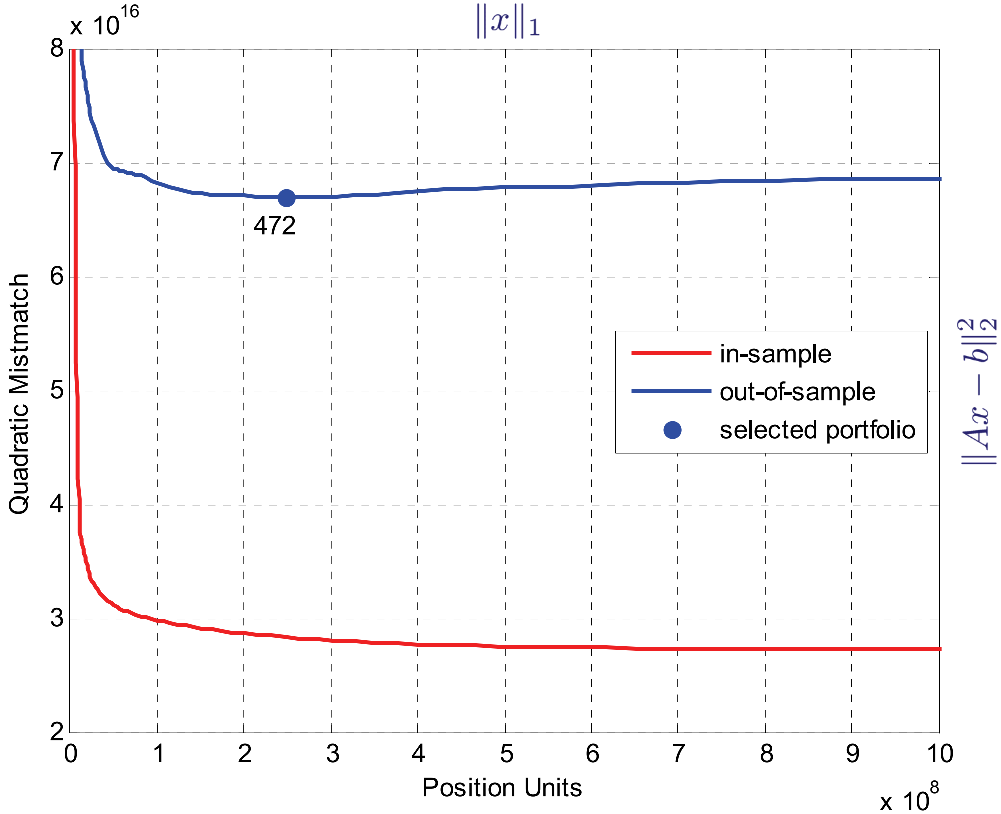
$$A = \begin{bmatrix} 6 & 0 \\ 9 & 0 \\ 3 & 0 \\ 0 & 1 \\ 0 & 2 \\ 0 & 4 \end{bmatrix} \Rightarrow Q = \frac{1}{S}A^TA = \begin{bmatrix} 42 & 0 \\ 0 & 7 \end{bmatrix}$$

📖 Practical Issues

selecting a final replicating portfolio

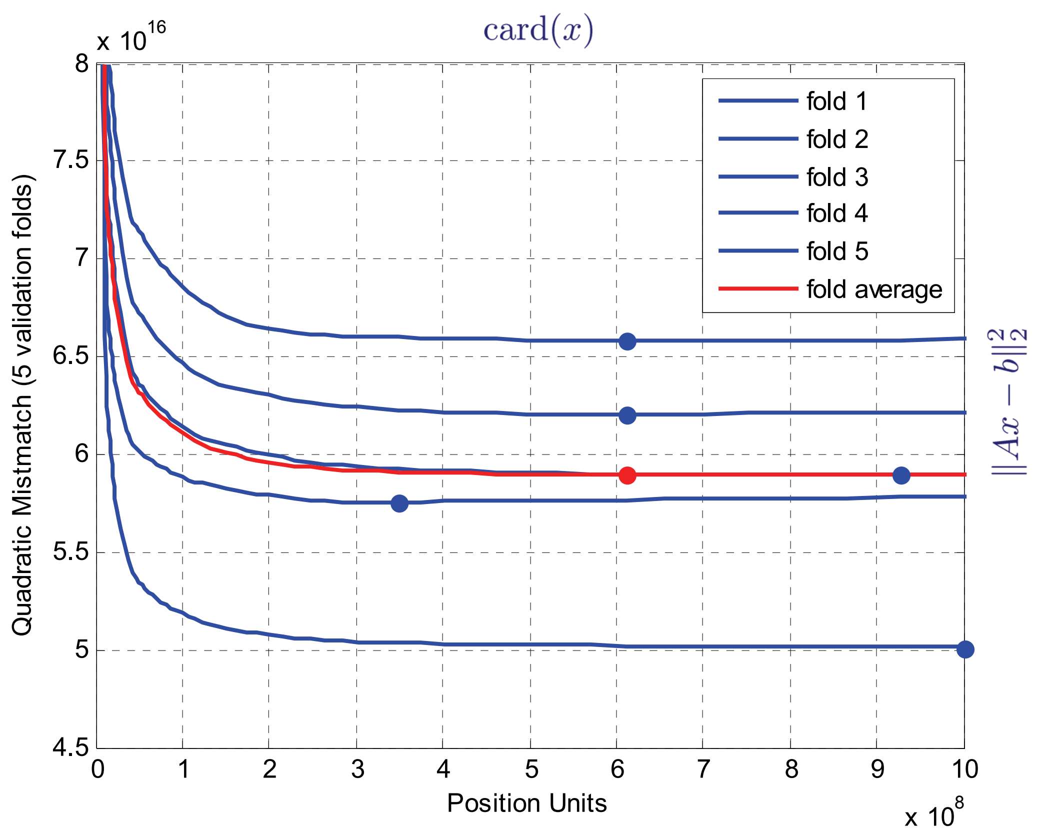
Selecting a Replicating Portfolio

- Quadratic matching, annual, reduced formulation:
  - compute in-sample frontier
  - evaluate the computed frontier out-of-sample
  - select a portfolio with the minimal objective on the out-of-sample frontier



Cross-Validation

- Quadratic matching, annual, reduced formulation:
  - 800 scenarios for cross-validation and 200 for out-of-sample testing
  - in *K*-fold cross-validation, the original sample is randomly partitioned into *K* subsamples
  - *K*-1 subsamples are used for testing and 1 subsample - for validation



Selecting a Replicating Portfolio

Replicating portfolio analysis involves computing the efficient frontier, evaluating it out-of-sample and **selecting a final replicating portfolio** based on the minimal out-of-sample objective value

Selecting a final replicating portfolio on the frontier:

- Out-of-sample performance:
  - 👍 computationally inexpensive
  - 👎 overly simplistic
- Cross-validation:
  - 👍 give good results if the number of folds is large (5 or 10)
  - 👎 computationally expensive
- Generalized cross-validation:
  - 👍 relatively inexpensive computationally
  - 👎 based on statistical assumptions
  - 👎 sensitive to numerical difficulties
- Other criteria? Ideas?

📖 Case Study

Algorithmics

Replicating Universe and Optimization Setup

- 883 replicating instruments
  - Zero Coupon Bonds
  - Swaptions (physical settlement)
  - Equity Forwards (on each index)
  - European Equity Options (on each index)

Market Indices
S&P 400 Midcap (MID)
Russell 1000 (RUX)
S&P 500 (SPX)
Nasdaq 100 (NDX)
MSCI EAFE (MSDUEAFE)
MSCI EM (MSEUEGF)
MSCI US REIT (RMS)
Lehman US Aggregate (USAGG)

- 500 scenarios
  - Set A = 250 (used for optimization)
  - Set B = 250 scenarios (used for out-of-sample testing)

Liability Cash Flows
guaranteed minimum death benefit
commissions expenses
mortality/expense charge
revenue sharing
surrender charges
per policy fees

- 20 years of annual cash flows

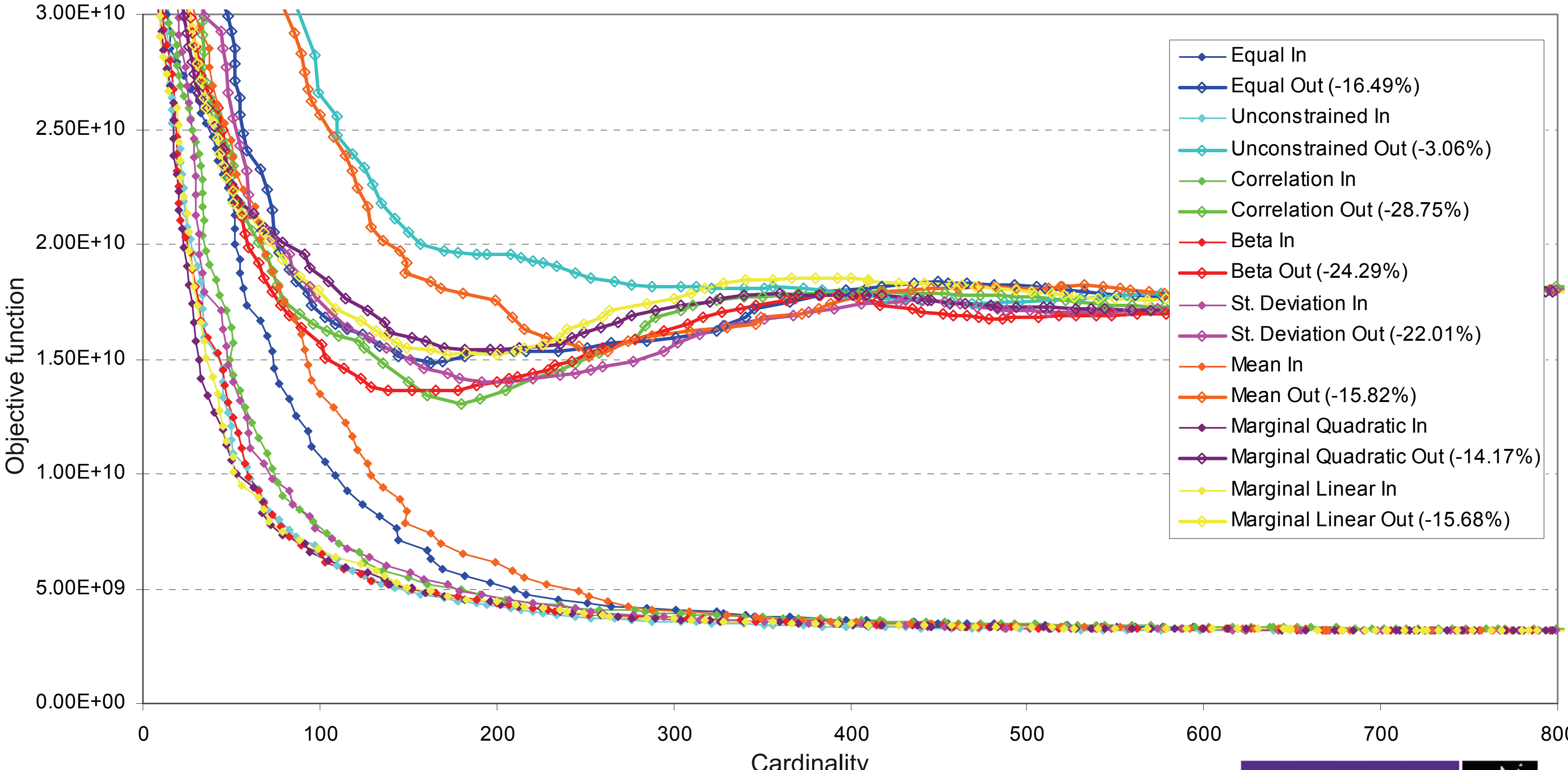
Algorithmics

Weighted Regularization

Comparison of different methods to compute regularization weights $\boldsymbol{D}$

$$\min_{x} \quad \|Ax - b\| \qquad \text{s.t.} \quad x \in \Omega, \quad \|Dx\|_1 \leq \varepsilon$$

minimizing quadratic mismatch
annual time bucket



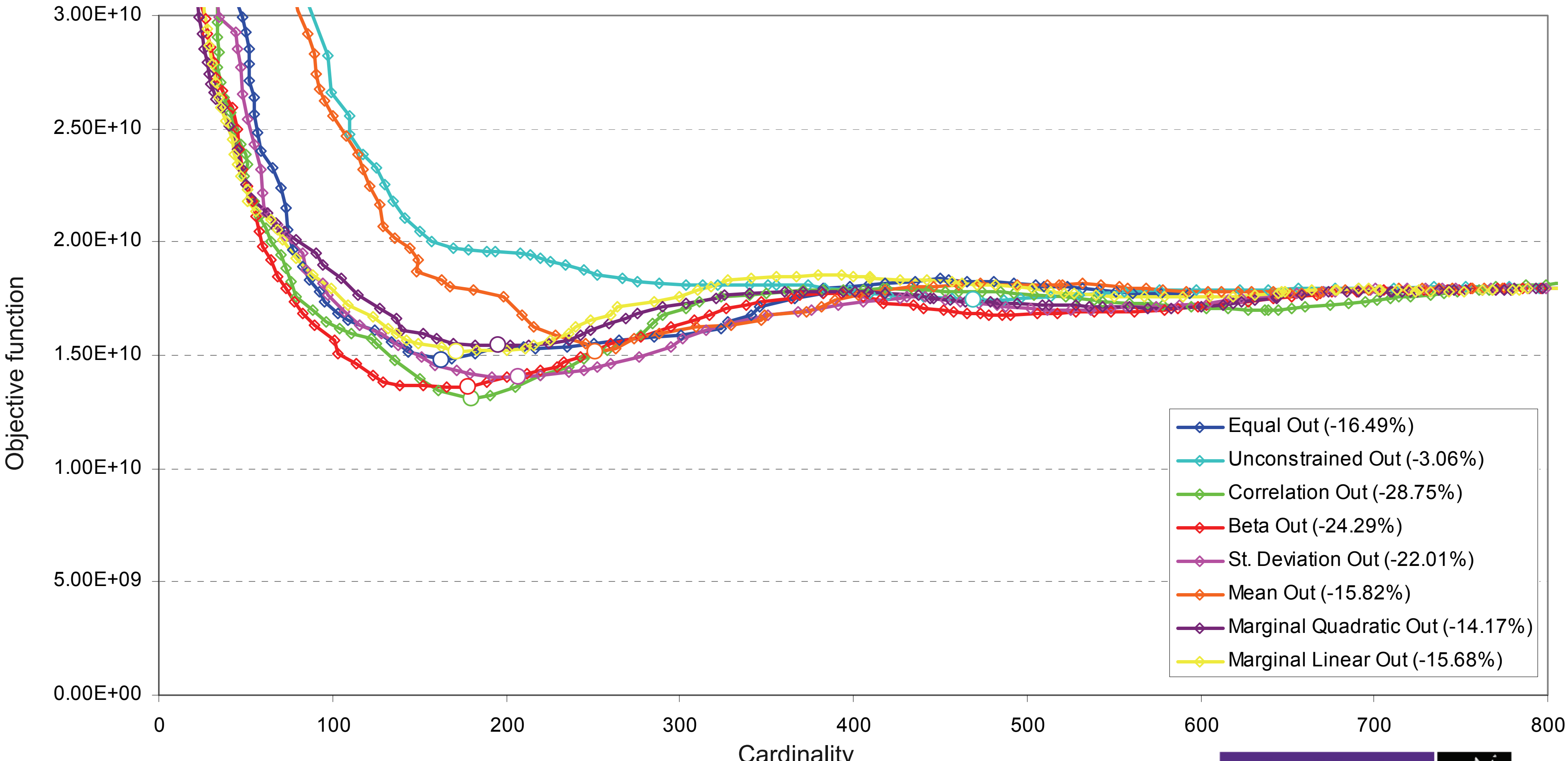
Algorithmics

Weighted Regularization

Comparison of different methods to compute regularization weights *D*

$$\min_x \quad \|Ax - b\| \\ \text{s.t.} \quad x \in \Omega \\ \|Dx\|_1 \le \varepsilon$$

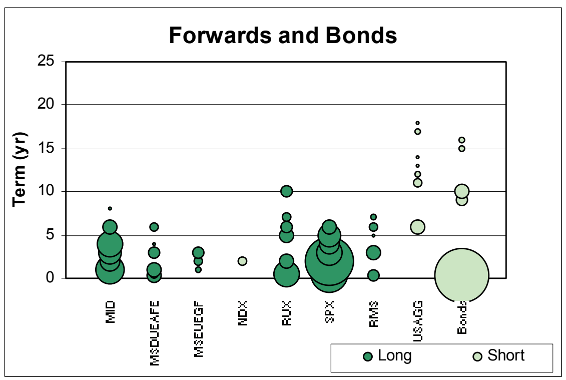
minimizing quadratic mismatch
annual time bucket

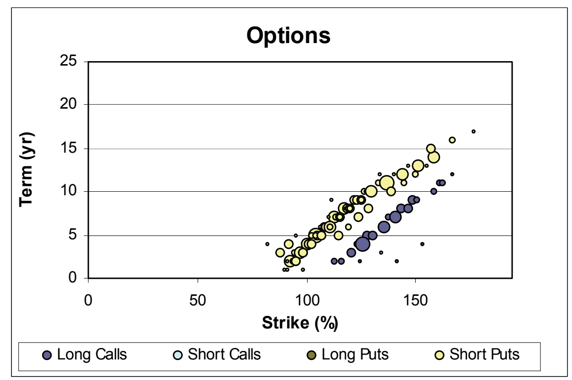


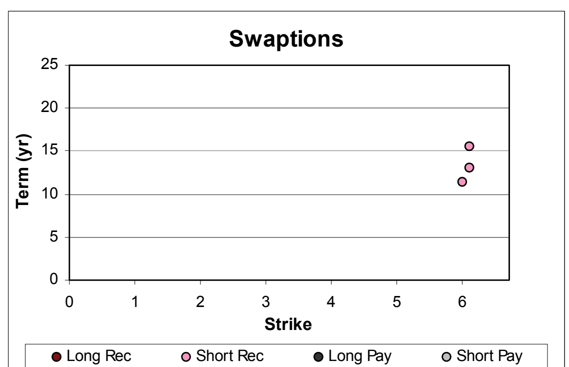
Algorithmics

Replicating Portfolio Analysis

Model	**Quadratic Annual** Cash Flow Matching
Present/current value	**Present Value** (PV)
Trading costs	**Beta**
Mean cash flow constraint	**No**
Cardinality	178
Budget	3870
MSE (in-sample)	4.6E+09
MSE(out-of-sample)	1.4E+10
Improvement (out-of-sample)	24.3%







📖 Conclusions

Conclusions

- Replicating portfolios are computational tools that facilitate the calculation of economic capital for complex insurance liability portfolios
- Replicating portfolio needs to closely match the liability value with a minimal number of standard financial instruments
- Trading constraints are an effective way of regularizing optimization problems so that they produce small replicating portfolios
- Making the trading costs in such restrictions instrument-dependent yields better results than using the same cost for all instruments:
  - investigated a number of different methods for obtaining effective trading costs
  - costs based on simple statistics provide good performance with minimal computational effort
- Dealt with a number of practical implementation issues

- Investigate alternative criteria for evaluating replicating portfolios
- Compare performance of different replicating portfolios when those are used for economic capital calculations

Algorithmics

Questions?