Comput Optim Appl
DOI 10.1007/s10589-012-9471-1

A local relaxation method for the cardinality constrained portfolio optimization problem

Walter Murray · Howard Shek

Received: 1 February 2011

Abstract The *NP*-hard nature of cardinality constrained mean-variance portfolio optimization problems has led to a number of different algorithms with varying degrees of success in reaching optimality given limited computational resources and under the presence of strict time constraints present in practice. The proposed *local relaxation* algorithm explores the inherent structure of the objective function. It solves a sequence of small, local, quadratic-programs by first projecting asset returns onto a reduced metric space, followed by clustering in this space to identify sub-groups of assets that best accentuate a suitable measure of similarity amongst different assets. The algorithm can either be cold started using a suitable heuristic method such as the centroids of initial clusters or be warm started based on the last output. Results, using a basket of up to 3,000 stocks and with different cardinality constraints, indicates that the proposed algorithm can lead to significant performance gain over popular branch-and-cut methods. One key application of this algorithm is in dealing with large scale cardinality constrained portfolio optimization under tight time constraint, such as for the purpose of index tracking or index arbitrage at high frequency.

Keywords Portfolio optimization · Local relaxation method · Nonlinear programming · Cardinality constrained optimization

1 Introduction

In the classical one-period Markowitz mean-variance optimization framework [15], asset returns are modeled either under the assumption of a joint Gaussian distribution,

W. Murray · H. Shek (✉)
Stanford University, Stanford, CA 94305, USA
e-mail: shek@stanford.edu

W. Murray
e-mail: walter@stanford.edu

Published online: 07 March 2012

or of a rational investor having a quadratic utility function. Given these assumptions, Markowitz has shown that the optimal portfolio for the investor is located on the *mean-variance efficient frontier*, in the sense that for all assets on this efficient frontier, for any given expected return, there is no other portfolio with lower variance; and for any given variance, there is no other portfolio with higher expected return. In the two dimensional mean-variance space, the efficient frontier is a parabola, whereas for mean-standard-deviation, it is a hyperbola. Formally, an agent has mean-variance preference if her utility function has the following property

$$U(R_p) = f\big(E[R_p], var(R_p)\big), \quad f_1 > 0, f_2 < 0,$$

where R_p is the portfolio return, and f_k is the partial derivative of the function f r.w.t. the kth coordinate. It assumes that even when the underlying distribution of the portfolio return is not Gaussian, the investor still only cares about the first two moments. While the original Markowitz model forms a quadratic-programming (QP) problem, a number of attempts have been made to linearize the portfolio risk measure, replacing the second order variance term, such that the portfolio optimization problem can be reduced to being linear-programming (LP) solvable with lower computational cost. Konno and Yamazaki [11] proposed using mean absolute deviation (MAD), such that for cases where the returns have zero mean, one uses $\mathbb{E}[|Rx|]$ instead of $x^\top \mathbb{E}[R^\top R]x$ for risk. Konno and Yamamoto [10] further showed that this alternative measure of risk can be successfully applied to portfolio problem with integer constraints. Yitzhaki [23] introduced the mean-risk model using Gini's mean (absolute) difference as the risk measure.

The original Markowitz problem, and a number of its subsequent extensions, and be solved as a simple QP using standard solvers that rely on well known techniques such as null-space method, trust-region method or sequential quadratic-programming, see for example [16] and [19]. However, computational issues can still arise if problems are large and solutions are needed quickly. Typical constraints present in practical problems include equality constraints such as budget constraints where the optimal assets weights need to sum to a predetermined value based on available capital or a specific investment mandate. Often used inequality constraints include a turn over bound that limits the maximum change in asset weight from a set of given initial holding weights, mainly to avoid excessive trading and smooth out the change in portfolio composition. Holdings constraint is also often used in order to impose an upper bound so as to limit excessive concentration to specific stocks, and to impose a lower bound so that the resulting portfolio would have meaningful weight on included assets. Typically, a transaction cost term is also part of the objective function that penalizes both excessive turnover and the inclusion of illiquid stocks in the portfolio. Most of these constraints and features do not change the convexity of the original problem and hence can be handled by standard optimization algorithms. However, for practical reasons, such as in presence of transaction costs, fees and exchange trading rules, one often faces the problem of requiring additional integer constraint to handle features such as the round lot rule, where for example the *round lot* trading size for most US stocks is 100 shares, or the minimum notional for the three-month Eurodollar interest rate futures contract is $1MM. Cardinality constraint is another often used feature that limits the number of assets in which one can

invest as part of a portfolio, out of a large universe of potential candidates. One classic example is the problem of stock index tracking, where an index with a large number of constituents needs to be tracked by a portfolio using a much smaller subset of underlying assets. This leads to the introduction of cardinality constraints, which can increase the complexity of the problem significantly. In fact, the problem is known to be *NP*-hard and hence optimality is not guaranteed in polynomial time. For these scenarios, even for modest sized problems, computationally effective algorithms do not exist and, up until recently, there has been relatively little work presented in the literature.

General Cardinality-Constrained Quadratic Program (CCQP) with linear constraints can be expressed as

$$\min_{x} \quad f(x) = -c^{\top}x + \lambda x^{\top}Hx \tag{1}$$

$$\text{s.t.} \quad Ax \geq b, \tag{2}$$

$$\sum_{i} \mathbf{1}_{\{x_i \neq 0\}} = K, \tag{3}$$

where $c, x \in \mathbb{R}^{N\times 1}$, $H \in \mathbb{R}^{N\times N}$ is positive definite, $A \in \mathbb{R}^{M\times N}$, $b \in \mathbb{R}^{M\times 1}$ and $M \leq N$. K is the cardinality constraint and the scalar λ is known as the *relative risk aversion* parameter. Let x^* be our solution set, the indicator function in (3) is given by

$$\mathbf{1}_{\{x_i \neq 0\}} = \begin{cases} 1 & x_i \in x^*, \\ 0 & \text{o.w.} \end{cases}$$

In other words, the constraint expressed in (3) forces the number of non-zero elements of the solution vector x be equal to a predetermined scalar, K. Note that there is no explicit non-negativity constraint of $x_i \geq 0\ \forall i$, hence short selling (i.e. selling what one does not own) is allowed. This together with the constraint $\sum_i x_i = 0$ gives what is known as a *dollar neutral* portfolio. It is called dollar neutral because the monetary value of the exposure on the *long* part (i.e. the part of the portfolio that consists of the stocks bought) equals to the exposure on the *short* part (i.e. the part of the portfolio which consists of the stocks that are sold short).

In the context of portfolio construction, x is the proportion of the capital allocation to each asset in the basket, c is the expected return of N candidate assets, i.e. a portfolio manager's subjective assessment of the one period ahead forecast of asset returns; H is the corresponding forecast for the variance-covariance matrix. The matrix A encapsulates M sets of linear constraints needed, for example, to impose holding limits, maximum leverage, etc. The scalar K limits the number of different assets the portfolio is allow to hold. The vector c is often modeled using time series analysis based on historical asset returns together with other covariates that could enhance the forecast signal. More sophisticated frameworks explore additional sources of information such as those embedded in the limit-order-book, see for example [21]. H is often modeled based on historical returns, $H = \mathbb{E}[(R - \bar{R})^{\top}(R - \bar{R})]$, where $R \in \mathbb{R}^{T\times N}$ is the return matrix for T observations of N assets and $\bar{R}$ is the temporal mean. More sophisticated frameworks explicitly explore known dynamics, such as

clustering effect of variances, and rely on using high-frequency intraday tick-data to enhance forecast power, see for example [6], and its multivariate extension in [22].

The cardinality constraint in (3) changes the complexity of the problem from that of an inequality constrained convex QP to that of a non-convex QP in which the feasible region is a mixed-integer set with potentially many local optima. [20] has reduced a three-partitioning problem to a CCQP, hence establishing the *NP*-hardness of the problem.

Although by construction, the covariance matrix H is positive-definite, it is usually ill conditioned. One common method to rectify the problem is by factor analysis, where one decomposes the return matrix into a rank k symmetric matrix, where $k \ll N$, and a diagonal matrix, known as the factor variance and specific variance matrix, respectively. In doing so, our implicit prior is that the aggregate market dynamics is spanned by a set of k orthogonal basis, with the remaining $N - k$ dimensions spanned by uncorrelated asset specific factors.

The type of problems that CCQP represents can be broadly categorized as a mixed-integer programming (MIP) problem with a quadratic objection function, resulting in a class of problem know as Mixed Integer Quadratic Programming (MIQP). A typical CCQP can be expressed, alternatively, as

$$\begin{aligned}
\min_{x,y} \quad & f(x) = -c^\top x + \frac{1}{2} x^\top H x \\
\text{s.t.} \quad & Ax \geq b, \\
& \sum_i y_i = K, \\
& y_i \geq x_i \geq -y_i \quad i = 1, \ldots, N, \\
& y_i \in \{0, 1\} \quad i = 1, \ldots, N,
\end{aligned}$$

where we have introduced a binary variable $y \in \{0, 1\}$ to enforce cardinality of the solution set. Solvers with mixed-integer capability are less readily available than standard convex QP solvers. If we are only interested in a reasonably good, feasible, solution without much concern about it's optimality, then one obvious method is via successive truncation, where a sequence of relaxed QP is solved and, at each iteration, a selection of assets with small or zero weights are truncated off. More sophisticated methods, require algorithms that are capable of globally optimize the objective function over the feasible parameter space.

There are two main approaches to solving a CCQP—exact methods and heuristic algorithms. Exact methods range from the most naive uniform grid search that exhaustively visits all node points of a grid mesh over the whole parameter space, to the widely adopted branch-and-bound or the closely related branch-and-cut methods. Branch-and-bound (B-B) algorithms are widely adopted tools in solving the types of *NP*-hard discrete optimization problems to optimality, by essentially searching the complete space of solutions. Since explicit enumeration is normally impossible due to an exponentially increasing number of potential solutions, B-B algorithms use bounds for the objective function combined with the value of the current best solution, which enable the algorithm to search parts of the solution space more efficiently.

The algorithm was introduced by [12], and generalized to non-linear functions by [4]. For a detailed overview of some typical B-B algorithms, see [3]. For application of B-B in solving CCQP, see for example [1, 13, 20]. Branch-and-cut methods, used by leading commercial grade MIQP solvers such as CPLEX, are a hybrid of branch-and-bound and cutting plane methods, where cutting planes are added to the original set of constraints so that the search space is effectively reduced at each step. For a comprehensive coverage of exact methods for solving global optimization problems, see [8]. Although the objective function of CCQP is convex, given the non-convex nature of the constraints and hence the overall problem, a number of heuristic methods have been proposed to deal with such problems. These methods generally fall under the class of adaptive optimization algorithms, which include Genetic Algorithm, Tabu Search and Simulated Annealing, and have been used extensively to solve global optimization problems with arbitrary objective function and constraints. Essentially, these methods cast the CCQP as a generic global optimization problem, oblivious to the underlying structure of the objective function. See [14] for application of genetic algorithm in portfolio optimization problems. Refer to [2] for an application of Tabu Search in portfolio optimization.

Both branch-and-bound and heuristic methods are generic method in the sense that the algorithm used does not explicitly explore any specific characteristics of the underlying problem. As a result, these methods are often not the most efficient way to solve CCQP in a portfolio optimization setting. In the following section, we propose a method that explores the structure of the problem, such that we are able to improve the algorithmic efficiency significantly.

This paper is organized into two main sections. Section 2 introduces the local relaxation method together with proposed projection and clustering algorithms. Section 3 compares the proposed algorithm with a leading commercial mixed-integer optimizer in solving CCQP, using a basket of up to 3,000 actively traded assets, over a range of different constraints and risk aversion parameters.

2 Local relaxation

One way of improving the efficiency of an algorithm is to put in more information about the problem. In [17, 18] an algorithm was proposed to solve nonlinear facility location problems (FLP). The authors also describe the algorithm applied to the location of substations in an electrical network, which in mathematical terms is very similar to the cardinality constrained portfolio problem. The essence of the method proposed is to exploit structure in the FLP that a given facility has neighbors in the sense that some locations are closer to a given location than others. A graph can be generated in which the locations are nodes and arcs are the links to neighboring nodes. Given a set of nodes (locations) the algorithm proceeds by moving from the current set of nodes to a set contained within the set of neighbors to these locations. Which set is chosen is determined by solving a locally relaxed problem of the original problem in just this set of nodes. The relaxed problem solved is much smaller than the original problem typically by at least a factor of 10. The solution of this relaxed problem distributes the value of the *center* node to itself and its neighbors. By observing these values a change from the original node may be identified (for example by

finding the node nearest the center of gravity). There are other issues, a key one being the need for the replacement to the "old" set to be an improvement. In FLP typically a facility comes in a fixed size. Consequently, in solving the relaxed problem the sum of the values associated with a neighborhood sums to one. The algorithm mimics successful algorithms for continuous variables by making incremental changes based upon local information.

The cardinality constrained portfolio problem differs from the FLP in two important ways. Firstly there is no obvious graph and secondly the weight of a specific node is typically not fixed. The latter is not of great significance and the issues that arise can be addressed. However, a suitable graph is essential. Typically for the FLP the graphs are planar although occasionally they are in three dimensions. It is hard to see how one could find a similar graph for the cardinality constrained portfolio problem. It may be questioned that such a graph exists. However, it is clear that in some sense a given asset is more similar to some assets than others. For example, one may expect that Target and Wal-mart behave more similarities than say Target and Genentech. The issue we need to address is how to quantify the similarity. It seems unlikely differences in assets can be characterized by two or three variables. What we propose is an algorithm based upon a graph in a much higher dimension, but one that is still much smaller than the number of variables. The set of neighbors to a given node is then determined simply by choosing the set of nodes that are closest (the number of neighbors can be varied). It should be noted that the modest variations in the length of the arcs are not critical to the solution obtained. They are used to define the set of variables over which the relaxation is made. Adding more nodes or subtracting some does not have a critical impact. These distances also play a role in defining the *center of the relaxed solution*, but that choice is tested and rejected if an improved solution is not found.

Without loss of generality, we replace the generic constraint $Ax \geq b$ in (2) by an equality constraint $\sum_i x_i = 0$ so that it mimics the case of a *dollar neutral* portfolio, and an inequality constraint $\underline{x} \leq x_i \leq \overline{x}$, commonly known as the *minimum and maximum holdings* constraint that imposes lower and upper bounds on the magnitude of the optimal portfolio weights. The resulting CCQP problem is then given by

$$\begin{aligned} \min_x \quad & f(x) = -c^\top x + \lambda x^\top H x \\ \text{s.t.} \quad & \sum_i x_i = 0, \\ & \underline{x} \leq x_i \leq \overline{x} \quad i = 1, \ldots, K, \\ & \sum_i \mathbf{1}_{\{x_i \neq 0\}} = K. \end{aligned}$$

We could, if we had wished, add additional constraints both equalities and inequalities. Also the approach is not critically dependent on the objective being a quadratic function. The proposed algorithm is similar to that proposed for FLPs but naturally differs in critical ways. By necessity we require a much higher dimensional space and we replace physical distance by a distance metric that is a function of factor loading,

risk aversion and expected return. Other possibilities exist, but we demonstrate that our proposal works well. We also introduce a mechanism that allows neighborhoods to overlap. These features are formally described in the following sections.

2.1 Clustering

Given a return matrix, $R \in \mathbb{R}^{T\times N}$, we first project our universe of N asset returns onto an orthogonal k-dimensional space, where often $k \ll N$. We can write

$$R = P_k V_k^\top + U,$$

where $P_k \in \mathbb{R}^{T\times k}$ is the return of our k factors, $V_k \in \mathbb{R}^{N\times k}$ is the factor loadings and $U \in \mathbb{R}^{T\times N}$ is the matrix of specific returns. This reduces the dimension of our problem from N to k, such that the ith column of the return matrix R, $r_i \in \mathbb{R}^T$, can be written as a linear combination of the k factors

$$r_i = v_{1,i}\, p_1 + v_{2,i}\, p_2 + \cdots + v_{k,i}\, p_k + u_i,$$

where $v_{i,j} \in \mathbb{R}$ is the (i, j) component of the matrix V_k, p_i and u_i are the ith column of the matrix P and U respectively. One method to identify the k factors is by Principal Component Analysis (PCA). An outline of the procedure can be expressed as follows:

- obtain spectral-decomposition:[1] $\frac{1}{T} R^\top R = V \Lambda V^\top$, where V and Λ are ordered by magnitude of the eigenvalues;
- form principal component matrix: $P = RV$;
- take first k columns of P and V to give P_k and V_k, known as the principle component and factor loading matrix, respectively.

Once we have identified the k-dimensional space, then the proposed clustering method works as follows:

- define a cluster distance metric $d(r_i, r_j) = \|v_i - v_j\|_2$, where $v_i \in \mathbb{R}^k$ and $v_j \in \mathbb{R}^k$ are the ith and jth row of the factor loading matrix V_k;
- filter outliers in this space by dropping assets with distance significantly greater than the median distance from the *center-of-gravity*[2] of the k-dimensional space;
- identify clusters (using e.g. *k-means clustering;* see [7] for details) in this metric space.

Figure 1 illustrates the proposed clustering method using returns of 50 actively traded US stocks. Here we have set the number of clusters to three and have dropped four outliers stocks (FE, CBS, CTL and PTV) based on the criteria listed above, and then projected the remaining 46 returns onto a three-dimensional space spanned by the three dominant principal components. Once the corresponding member assets have been identified for each cluster, the algorithm will propose a new set of K centroids at the end of each iteration.

[1] Here we assume columns of R have zero mean.

[2] The *center-of-gravity* is defined in the usual sense by assigning a weight of one to each asset on the projected space

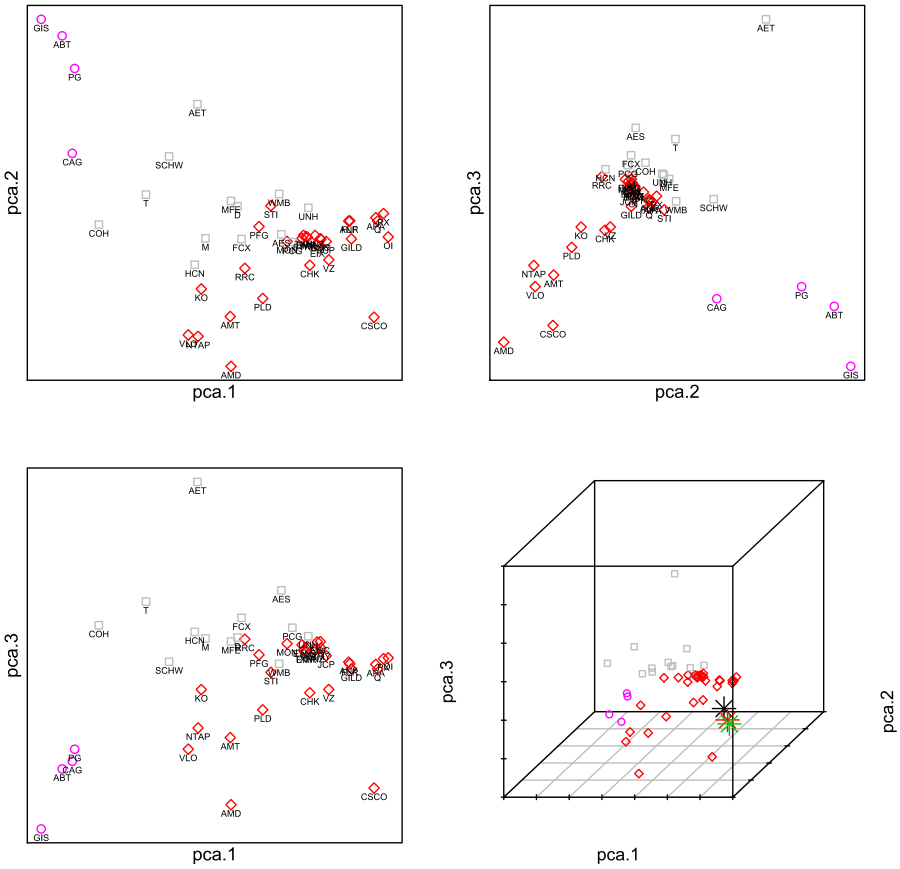

Fig. 1 *k-means clustering* on space spanned by the first three principal components

2.2 Algorithm

2.2.1 Algorithm for iterative local relaxation

The local relaxation algorithm, with pseudo-code shown in Algorithm 1, solves a number of small QPs for subsets of the original set of assets $\mathcal{S}$, such that $\mathcal{S}_o \subset \mathcal{S}_r \subseteq \mathcal{S}$, where $\mathcal{S}_o$ is the *centroid asset* set with K elements, and $\mathcal{S}_r$ is the *neighborhood asset set* which is a union of all the assets in the K clusters.

At each iteration, we solve the *centroid-asset QP* involving K number of centroid assets:

$$\begin{aligned} \min_{x} \quad & -c_o^\top x_o + \frac{1}{2} x_o^\top H_o x_o \\ \text{s.t.} \quad & e^\top x_o = 0, \\ & \underline{x} \le x_{o,i} \le \overline{x} \quad i = 1, \ldots, K, \\ & x \in \mathcal{S}_o, \end{aligned}$$

Algorithm 1 Local relaxation search in projected space

Require: Asset sets $\mathcal{S}$, $\mathcal{S}_o$; algorithm parameters α_s, σ_d, $M_{\min}$, $M_{\max}$, K, $\bar{\eta}$, ϵ_o.
Ensure: A series of solutions with increasing optimality.

while *max-iteration-not-reached* or *max-time-limit-not-reached* **do**
  [Solve centroid-assets QP]
  $x_o^* \leftarrow$ solution of *centroid-assets* QP based on $\mathcal{S}_o$
  [Generate *random-centroid-asset-order*]
  $I_{sampled} \leftarrow$ randomly sample $\{1, 2, \ldots, K\}$ w.p. given by (4) without replacement
  $\grave{\mathcal{S}} \leftarrow \mathcal{S} \setminus \mathcal{S}_o$
  [Identify neighbors of centroid assets]
  for $\{s_{o,i} : s_{o,i} \in \mathcal{S}_o, i \in I_{sampled}\}$ **do**
    ith cluster size: $m_i \leftarrow \max(\lfloor p_i M_{\max} \rfloor, M_{\min})$
    $N(s_{o,i}) \leftarrow \{s \in \grave{\mathcal{S}} : d(s; a_o) \le r\}$; for the $(m_i - 1)$th asset, $d(s_{(m_i-1)}; a_o) = r$
    $\grave{\mathcal{S}} \leftarrow \grave{\mathcal{S}} \setminus N(s_{o,i})$
  end for
  [Solve relaxed-neighborhood QP]
  $\mathcal{S}_r \leftarrow \bigcup_{i=1}^{K} N(s_{o,i})$
  $x_r^* \leftarrow$ solution of *relaxed-neighborhood* QP based on $\mathcal{S}_r$
  [Identify new centroids]
  $\mathcal{S}_0 \leftarrow$ `findNewCentroids(`$\mathcal{S}_r$`,` $\mathcal{S}_o$`)`
end while

where c_o and H_o are expected return and covariance matrix for the set of centroid assets, $\mathcal{S}_o$, only. This gives optimal weight vector x_o^*, where $x_{o,i}^*$ is the ith element of this vector corresponding to ith centroid asset, $s_{o,i}$. We then identify new *neighborhood* assets $N(s_{o,i})$ belonging to each centroid asset cluster, $s_{o,i} \in \mathcal{S}_o, i = 1, \ldots, K$ and $\mathcal{S}_r = \bigcup_{i=1}^{K} N(s_{o,i})$. The composition of each new cluster clearly depends on the order that the centroids are picked. For example, clusters A and B could be sufficiently close in the projected space, such that there exists a group of assets which are equally close to both clusters. Since assets cannot belong to more than one group, so once cluster A claims an asset, the asset drops out of the feasible set of assets for cluster B. We propose a randomization algorithm to address this ordering issue.

Let p_i be the probability of centroid asset i being picked for cluster assignment, we propose the following logit transform to parametrize this probability:

$$\log \frac{p_i}{1 - p_i} = \alpha_s \left(\frac{|x_{o,i}^*|}{\sum_i |x_{o,i}^*|} - \frac{1}{K} \right), \tag{4}$$

where $x_{o,i}^*$ corresponds to the optimal weight from the *centroid-asset QP* for the ith centroid asset, such that the higher the centroid weight the higher the probability that it will have priority over other centroids in claiming its member assets. Cluster components or *neighbors* are identified by the *center-of-gravity* method, defined for

the ith cluster as

$$g_i = \frac{V_k^\top x_r^{*(i)}}{\|x_r^{*(i)}\|}, \tag{5}$$

where the vector $x_r^{*(i)}$ corresponds to optimal weights for assets belonging to the ith cluster. The member assets are picked according to the degree of closeness to centroid i, defined by the distance measure $d(s; g_i)$ for asset s, in the projected space relative to an arbitrary reference point, a_0, by

$$d(s; a_o) = \sigma_d \|v_s - a_0\|_2 + (1 - \sigma_d)\big(\|c\|_\infty - |c_s|\big), \tag{6}$$

where c_s is the sth element of the vector of expected return, c. The economic interpretation is that we use the tuning parameter σ_d to control the trade off between picking stocks highly correlated and stocks that have higher expected return. Note that $\sigma_d = 1$ corresponds to the Euclidean distance measure.

Once we have identified the centroid assets and their cluster neighbors, we solve the *relaxed-neighborhood QP*

$$\begin{aligned} \min_w \quad & -c_r^\top x_r + \frac{1}{2} x_r^\top H_r x_r \\ \text{s.t.} \quad & e_i^\top x_r = x_{o,i}^* \quad i = 1, \ldots, K, \\ & x_r \in \mathcal{S}_r, \end{aligned}$$

where e_i is a column vector with 1's at positions that correspond to the ith centroid cluster assets and are zero everywhere else; c_r is the vector of expected returns and H_r is the corresponding variance-covariance matrix of our sub-universe of assets. We impose the constraint that the sum of weights of the neighbors of the ith centroid add up to the centroid weight, $x_{o,i}^*$.

Based on the result of the *relaxed-neighborhood QP*, the algorithm then looks for a new set of centroid assets that would improve the optimality of the previous solution. Let $\{m_k\}_{k=1,\ldots,K}$ be the number of members in each of the K clusters. We generate $\|m\|_\infty$ number of centroid proposals based on assets within the same cluster to give $\Phi_1, \ldots, \Phi_{\|m\|_\infty}$. $\Phi_1 \in \mathbb{R}^K$ is a vector of closest distances, defined by (6), to the centroid; Φ_2 is the vector of distances that are second closest, etc. For cluster group k with $m_k \leq \|m\|_\infty$, we pad the last $\|m\|_\infty - m_k$ entries with distance of the cluster member that is farthest away from the centroid in cluster k. Once we have generated the centroid proposals $\{\Phi_i\}_{i=1,\ldots,\|m\|_\infty}$, Algorithm 2 then loops through each of the proposals. During each loop, the algorithm replaces the current centroids with progressively less optimal[3] proposals, and breaks out as soon as it detects an improvement to the objective function.

If none of the proposals gives a lower objective function value, the algorithm then attempts to swap centroids with assets from other clusters (Algorithm 3). The choice of substitution is governed by the expected returns of the assets, weighted by the

[3] In the sense of larger distance measure from the cluster centroid assets.

Algorithm 2 `findNewCentroids`$(\mathcal{S}_r, \mathcal{S}_o)$

Require: Asset sets $\mathcal{S}_r$ and $\mathcal{S}_o$, cluster expansion parameters Υ and $\Upsilon_{\max}$.
Ensure: More optimal objective function value by identifying new centroids.

```
[Generate centroid proposal list]
for i = 1 to ‖m‖∞ do
    Φ_i ← proposal of K centroid assets with distance ranked ith close-set based on (6)
end for
j ← 1
while new-centroid-not-found do
    for i = 1 to K do
        s_{o,i} ← Φ_{j,i}
        x*_o ← solution of centroid-assets QP based on S_o
        break if centroid-assets QP is more optimal
    end for
    j ← j + 1
    if j > ‖m‖∞ then
        Υ ← Υ^η
        if Υ > max(⌊|S_0|/K/M_max⌋, Υ_max) then
            swapCentroids(S_r)
        else
            M_max ← ⌊Υ^η M_max⌋
            η ← η + 1; break current while-loop
        end if
    end if
end while
```

magnitude of the relaxed solution from the *relaxed-neighborhood* QP, to give a vector $\varpi \in \mathbb{R}^{\sum_i m_i}$,

$$\varpi = \left[\left|x^*_{o,1}\right| c_{s\in N(s_{o,1})}, \ldots, \left|x^*_{o,2}\right| c_{s\in N(s_{o,2})}, \ldots, \left|x^*_{o,K}\right| c_{s\in N(s_{o,K})}\right]^\top,$$

where $c_{s\in N(s_{o,k})}$ is a row vector consists of the expected returns for assets belonging to the kth cluster.

There are a number of methods to identify starting centroids, $\mathcal{S}_o$, such as using *k-means* (Algorithm 4) to identify K clusters, where K is the cardinality of the problem, in the projected factor space spanned by $\mathcal{R}(P_k)$. Then in this projected space, we identify the set of K *centroid-assets*, $\mathcal{S}_o \subset \mathcal{S}$, which are assets closest to the cluster centers.[4]

Another widely used heuristic method is based on *successive truncation* (Algorithm 5) where at each iteration the algorithm discards a fixed portion of assets which have small weights, until the algorithm terminates with the appropriate number of assets in our portfolio equal to the desire cardinality value.

[4]Note a cluster center is simply a point in the projected space, and does not necessarily coincide with any projected asset returns in this space.

Algorithm 3 `swapCentroids`($\mathcal{S}_r$)

Require: Cluster group assets set $\mathcal{S}_r$.
Ensure: More optimal objective function value by swapping centroid assets.

$j \leftarrow 1$
while *new-centroid-not-found* **do**
  $\mathcal{S}_o \leftarrow$ swap current centroid asset with asset in $\mathcal{S}_r$
  $x_o^* \leftarrow$ solution of *centroid-assets* QP based on $\mathcal{S}_o$
  break if $\mathcal{S}_o$ is more optimal
end while

Algorithm 4 K-means in factor space

Given: R, K
[Singular Value Decomposition] $R = U \Sigma V^\top$
$V_k \leftarrow$ first k columns of V
[Random initialization] $m_1^{(0)}, m_2^{(0)}, \ldots, m_K^{(0)}$
while not-converged **do**
  for $i = 1$ to K **do**
    E-Step: $\mathcal{C}_i^{(n)} = \{v_j : \|v_j - m_i^{(n)}\| \leq \|v_j - m_{\tilde{i}}^{(n)}\|, \forall \tilde{i} = 1, \ldots, K\}$
    M-Step: $m_i^{(n+1)} = \frac{1}{|\mathcal{C}_i^{(n)}|} \sum_{v_j \in \mathcal{C}_i^{(n)}} v_j$
  end for
end while

Algorithm 5 Successive truncation

Given: N, I
$j \leftarrow 0$
$\mathcal{S}_0 \leftarrow \mathcal{S}$; $n_0 \leftarrow N$; $\phi_0 \leftarrow 0$
repeat
  [Solve QP] for x^* where

$$x_j^* = \arg\min_x \ -c^\top x + \frac{1}{2} x^\top H x$$
$$\text{s.t.} \quad Ax \geq b,$$
$$x \in \mathcal{S}_j$$

  if arithmetic truncation **then**
    return $\phi_j \leftarrow \max(n_j - \lfloor \frac{N}{I} \rfloor, K)$
  else
    return $\phi_j \leftarrow \max(\lfloor (1 - \frac{K}{N}) n_j \rfloor, K)$
  end if
  $\mathcal{S}_{j+1} \leftarrow \{s_{(n)}, s_{(n-1)}, \ldots, s_{(\phi_j)}\}$
  $n_{j+1} = |\mathcal{S}_{j+1}| - \phi_j$
until $\phi_j = K$

3 Computational result

3.1 Preparation of dataset

Empirical results in this section are based on two key datasets. The first dataset consists of daily closing prices for 500 of the most actively traded stocks, as measured by mean daily transacted volume, on the main stock exchanges in the US, spanning the period between 2002-01-02 and 2010-01-22. The second dataset consists of 3,000 monthly closing prices for actively traded stocks in the US, spanning the period between 2005-05-31 and 2010-04-30. These two datasets test the performance of our algorithm under different market environments and for different scales of the problem. The results based on the proposed algorithm are compared with those obtained by the successive truncation algorithm in Algorithm 5 and by the branch-and-cut MIQP solver used in CPLEX.

Cleaning and filtering We follow a three step procedure to clean, filter and prepare the datasets for analysis:

- *Recorded price validation*. We discard assets that do not have a complete price history over the whole sample period.
- *NAICS sector matching*. We purge any asset whose ticker cannot be matched to the latest NAICS sector definition.
- *PCA outliers detection*. We project the returns, defined as the difference in log-prices, onto a four-dimensional space spanned by the dominant PCA factors. Any asset with a projected return that is more than two standard deviations away from the sample median is considered an outlier and dropped.

Factorization of covariance matrix For the dataset with 3,000 assets, since $N \gg T$ so the covariance matrix from raw returns is rank deficient. We take the necessary step of factorizing the raw covariance matrix by using the first four dominant PCA factors. This ensures that the resulting covariance matrix is both full rank and well conditioned. The first four factors together capture 35.07% of the total variance for the 500 stock case, and 38.15% for the 3,000 stocks case.

3.2 Algorithm benchmarking

In this section, we compare the performance of our proposed algorithm against the commonly used heuristic of successive truncation, followed by a more in depth comparison of our algorithm against a leading commercial solver, CPLEX. For the first part, we loosen the inequality holdings bounds by setting $\underline{x} = -\infty$ and $\overline{x} = \infty$, and retaining only the dollar neutral constraint, so that we can better highlight the essential features of the proposed algorithm. In latter parts of the section on *warm starting*, we tighten the holding bounds and show how the algorithm handles a typical inequality constraint and how it measures against CPLEX.

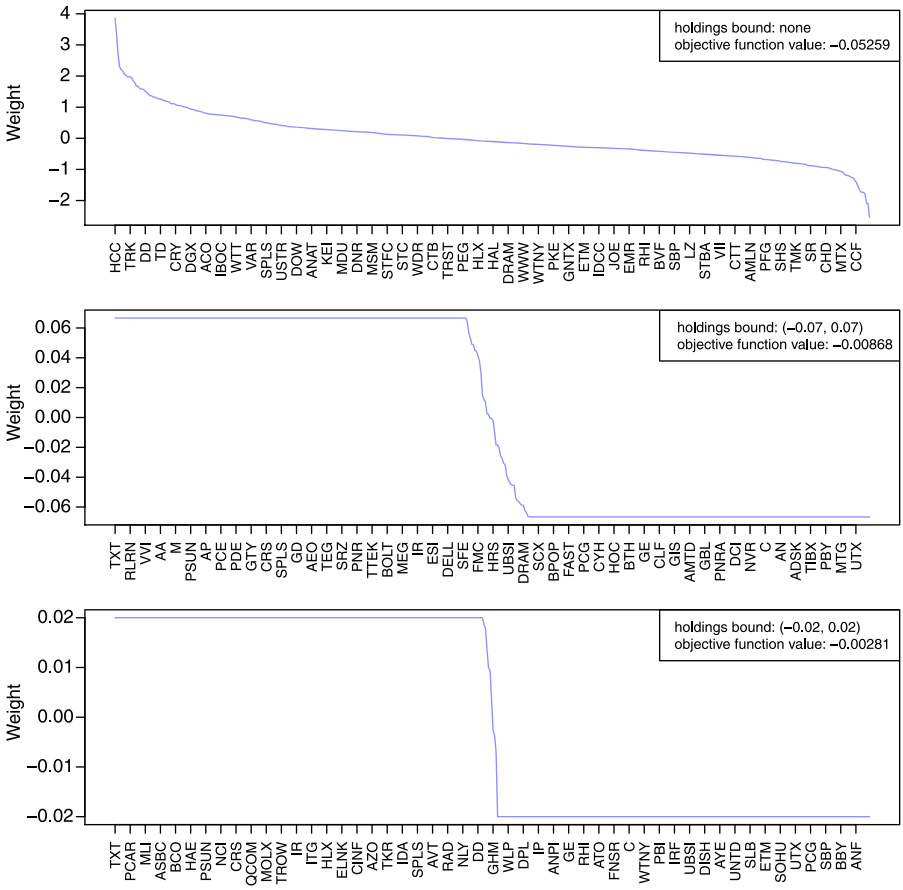

Fig. 2 Relaxed (i.e. without cardinality constraint) QP solution for 500 actively traded US stocks (not all tickers are shown), with dollar neutral constraint and holdings bounds as shown

3.2.1 Successive truncation

Figure 2 shows the result for the fully relaxed problem (i.e. without the cardinality constraint), ordered by value of the optimal weights, for different portfolio holdings bounds. Positive weights mean that we go *long* those assets and negative weights correspond to *shorting* those assets. Since we impose the constraint, $\sum_i x_i = 0$, therefore the net *dollar exposure* of the result portfolio is zero. This is one interpretation of a *market neutral* portfolio. An alternative definition of market neutrality requires neutralizing specific factors of the net portfolio exposure, by effectively taking into account correlation between different assets.

We apply the successive arithmetic truncation algorithm given in Algorithm 5 to CCQP. Table 1 gives the result for the 500-choose-15 (i.e. solving a 500 asset problem with cardinality equal to 15) case. It can be seen that if the number of iterations is set to one-tenth of the size of the universe, i.e. by discarding a large portion of stocks from the feasible set at each iteration, the method is able to obtain a result quickly.

Table 1 Successive arithmetic truncation method result for 500 of the most liquid US stocks, with dollar neutral constraint and cardinality, $K = 15$

Number of iterations	Objective value	Time (sec)	Optimal assets
10	−0.00615	0.246	AGII BIG CPO ETH FBP FICO HCC IGT LPNT PFG RYN SRT TAP UIS WPP
20	−0.00637	0.455	AGII BWA CDE CPO DYN ETH FICO HCC NWBI PFG RYN SRT TAP UIS WPP
50	−0.00629	0.832	AGII BWA CDE CPO EMC ETH FICO HCC IGT PFG RYN SRT TAP UIS WPP
100	−0.00635	1.677	AGII BWA CDE CPO DYN EMC ETH FICO HCC PFG RYN SRT TAP UIS WPP
500	−0.00638	8.073	AGII BWA CDE CPO ETH FICO HCC IGT PFG RYN SRT TAP TRK UIS WPP

Table 2 Successive arithmetic truncation method result for 3,000 of the most liquid US stocks, with dollar neutral constraint and cardinality, $K = 15$

Number of iterations	Objective value	Time (sec)	Optimal assets
10	−12.08450	33	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE EDMC MJN SEM TLCR VRSK
100	−12.13756	210	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE MJN SEM TLCR TMH VRSK
300	−12.23406	561	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK SEM TLCR VRSK
500	−12.23406	929	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK SEM TLCR VRSK
1,000	−11.72131	1,839	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK
3,000	−11.72131	5,536	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK

However, as we will show later, if one needs a higher degree of optimality, this simple method often fails to deliver. Note it is not guaranteed that the more iterations we use, the better the solution. Table 2 shows the result for the 3,000-choose-15 case, we see that the algorithm does not give monotone improvement of the objective function. For comparison, Table 3 shows the result for geometric truncation for the 3,000-choose-15 case. Again, we notice that increasing the number of iterations does not necessarily increase optimality.

Table 3 Successive geometric truncation method result for 3,000 of the most liquid US stocks with dollar neutral constraint and cardinality constraint, $K = 15$

Number of iterations	Objective value	Time (sec)	Optimal assets
10	−12.37960	19	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE LOPE MJN SEM TLCR VRSK
100	−11.72131	63	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK
300	−11.72131	161	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK
500	−11.72131	290	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK
1,000	−11.72131	507	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK
3,000	−11.72131	1,259	AOL ART CFL CFN CIE CIT CVE DGI DOLE MJN NFBK RA SEM TLCR VRSK

Both of these truncation methods are widely used in practice. Although there is clearly no dispute on the speed of this heuristic algorithm, we later show that it can be significantly inferior than algorithms that are able to explore the solution space more effectively, at only a small cost of execution time.

Throughout the rest of this paper, we refer to successive arithmetic truncation as simply successive truncation, unless otherwise stated.

3.2.2 Local relaxation and CPLEX MIQP

Each of the six panels in Fig. 3 represents a two dimensional view of the four dimensional PCA projection of the returns of our 500 assets. In the same figure, we also show the initial K clusters, indicated by different colors and symbols. The local relaxation method that solves a CCQP with cardinality equal to 15 may be initialized with the cluster centroids assets, iteratively solve for relaxed but smaller QPs by including a subset of the cluster members, then followed by moving to a new set of centroids with strictly increasing optimality.

A prototype of the local relaxation algorithm has been implemented in the language *R* running under a 64-Bit Linux kernel. The solution of the relaxed QPs required by the algorithm and the truncation method are obtained by using the built-in QP solver in *R*, which is based on [5]. To help assess the performance of our proposed algorithm, we present three solved problems, each with different asset universe sizes and cardinality values. The sets of parameters used in the algorithm for these three problems are given in Table 4.

For comparison, we have implemented CPLEX v12 under a 64-Bit Linux kernel using the CPLEX C++ API to access the built-in MIQP solver, which is based on the dynamic search branch-and-cut method [9]. The code is set to run in parallel using up

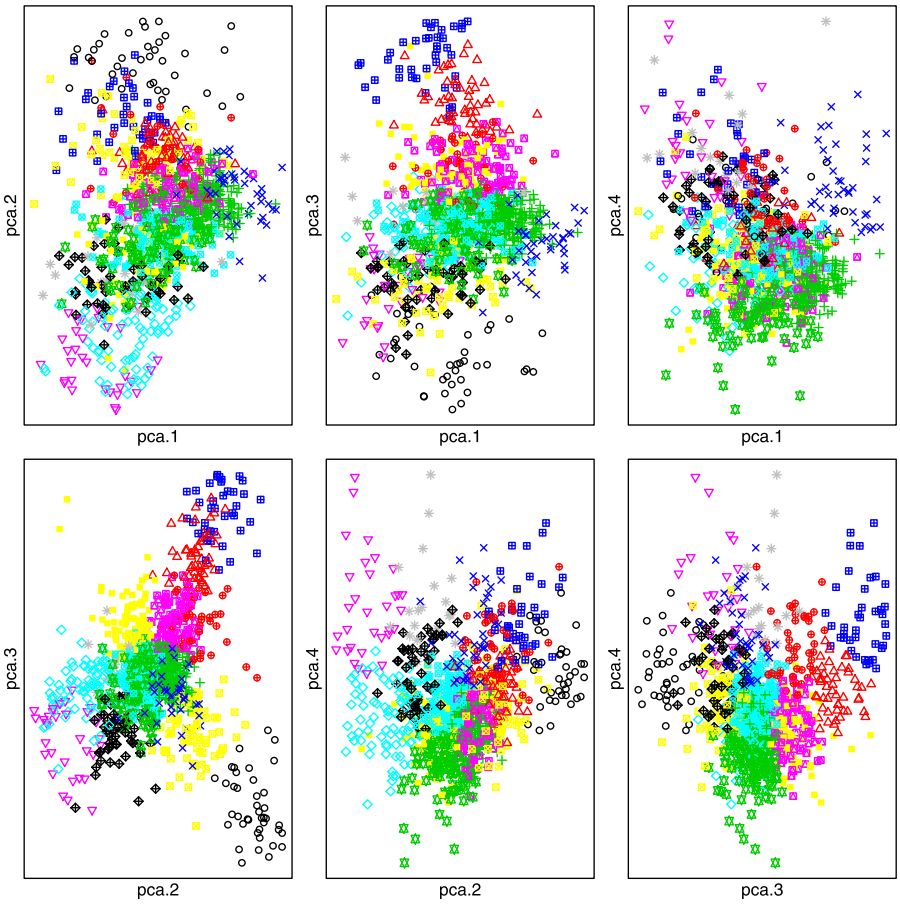

Fig. 3 Projection onto a four dimensional space spanned by the first four dominant PCA factors, based on returns for the most liquid 500 US traded stocks

Table 4 Parameter setup for local relaxation method. (500, 15) means solving a 500 universe problem with cardinality set to 15

	(500, 15)	(3000, 15)	(3000, 100)
$M_{\min}$	5	5	3
$M_{\max}$	30	30	10
σ_d	0.2	0.2	0.2
α_s	80	80	80
ϵ_0	1×10^{-5}	1×10^{-5}	1×10^{-5}
$\bar{\eta}$	10	10	10
Υ	1.1	1.2	1.8

to eight threads on a Quad Intel Core i7 2.8 GhZ CPU with access to a total of 16 GB of memory. Table 5 shows the parameter settings used. Table 6 shows the result, for the 500-choose-15 case, with a run time cut-off set to 1,200 sec. Figure 4 shows the

Table 5 CPLEX parameter settings (with the rest of the parameters set to default)

Parameter	Setting
Preprocessing	`true`
Heuristics	`RINS at root node only`
Algorithm	`branch-and-cut`
MIP emphasis	`balance optimality and feasibility`
MIP search method	`dynamic search`
Branch Variable Selection	`automatic`
Parallel mode	`deterministic, using up to 8 threads`

Table 6 Computational result for 500 asset case with cardinality, $K = 15$. For CPLEX and the local relaxation algorithm, we have imposed a maximum run time cutoff at 1,200 seconds

Method	Objective value	Time (sec)	Optimal assets
CPLEX	−0.00796	1,200	AOI AYI CCF CPO DD DST DY FBP HCC LRCX MRCL NST TMK UIS WPP
Local relaxation	−0.00870	1,200	AOI AVT AYI BIG BPOP CCF CPO CSCO DST HCC HMA IBM MCRL TAP ZNT

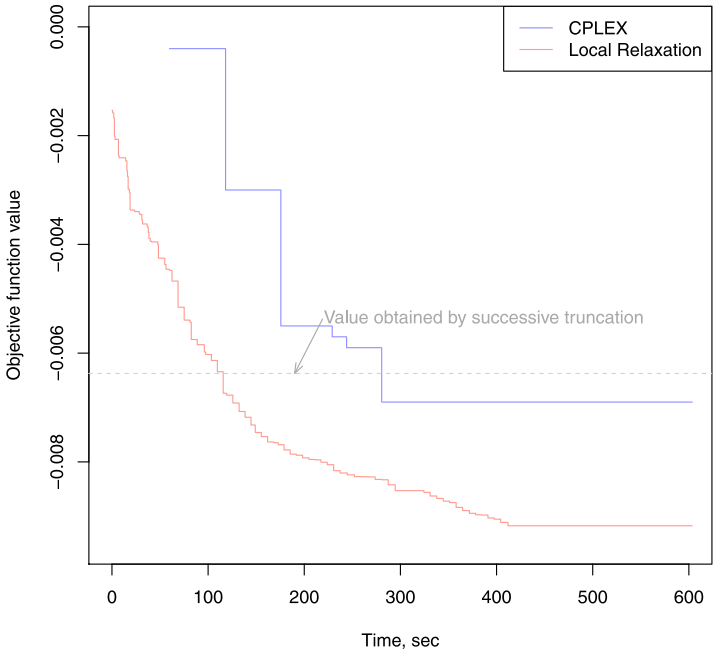

Fig. 4 CPLEX vs. local relaxation method performance comparison for 500 asset case, with cardinality, $K = 15$

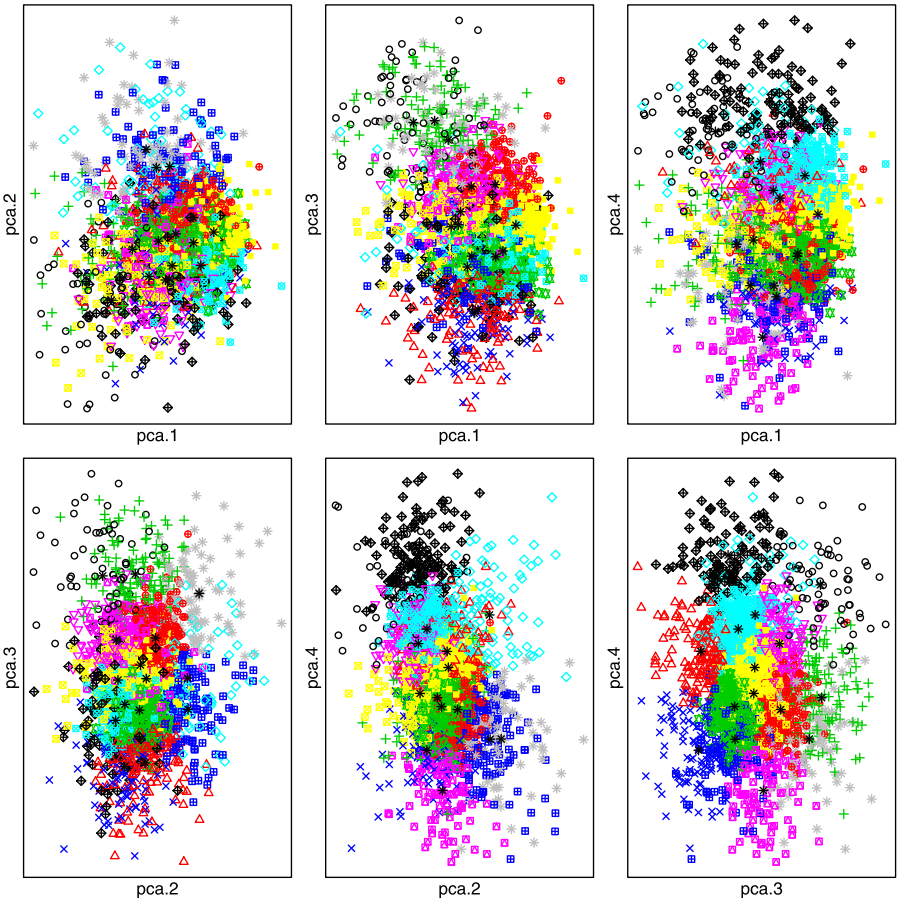

Fig. 5 Projection onto a four dimensional space spanned by the first four dominant PCA factors, based on returns for the most liquid 3,000 US traded stocks

performance comparison between CPLEX and the proposed algorithm as a function of time. We see that the local relaxation method is able to obtain a better solution than CPLEX right from the beginning, and it strictly dominates CPLEX throughout the run period.

Next, we assess the scalability of the new algorithm, by switching to the larger dataset which consists of 3,000 actively traded stocks. As before, we first project the factored covariance matrix onto a four dimensional PCA space and then identify the initial K clusters based on *k-means*, as shown in Fig. 5. Table 7 shows the computational result. We see that the local relaxation method is able to reach a significantly better result than CPLEX quickly. Figure 6 shows the performance comparison of the local relaxation method verses CPLEX for the 3,000 choose 15 case, which again indicates dominance of the proposed algorithm over CPLEX.

Table 7 Computational result for 3000 asset case, with cardinality, $K = 15$. Successive truncation is done over 10 iterations

Method	Objective value	Time (sec)	Optimal assets
Successive truncation	−12.084	36	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE EDMC MJN SEM TLCR VRSK
CPLEX	−12.304	25,200	AOL AONE ART CFL CFN CIE CIT CVE DGI DOLE MJN SEM TLCR VECO VRSK
Local relaxation	−12.765	7,200	AOL AONE ARST ART BPI CFL CFN CIE CIT CVE DGI DOLE MJN SEM TLCR

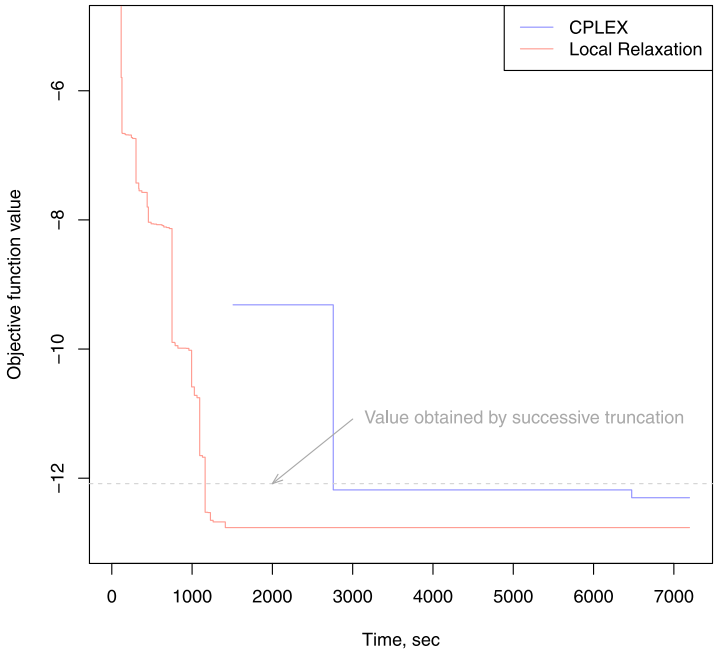

Fig. 6 CPLEX vs. local relaxation method performance comparison for 3,000 asset case, with cardinality constraint $K = 15$

3.2.3 Warm-starting CPLEX with local relaxation output

It is reasonable to conjecture that the strength of the local relaxation algorithm is in its ability to explore specific structures of the problem, and hence in identifying the sub-group of assets within which the optimal solution set lies. To see whether the branch-and-cut method in CPLEX is able to improve on the solution from the proposed algorithm once the algorithm has remained stationary on a sub-group of assets, we propose the following test:

- solve the CCQP with local relaxation algorithm;

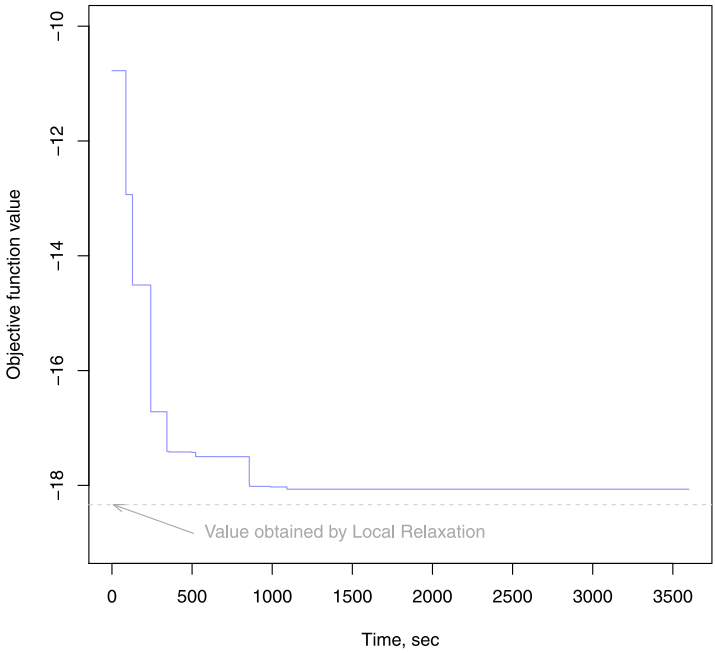

Fig. 7 Apply CPLEX MIQP to union of cluster groups identified by local relaxation algorithm upon convergence (the beginning of the flat line in Fig. 8). Maximum run-time is set to one hour

- once the algorithm has reach a stationary group of assets (here stationary refers to the observation that the cluster groups do not change over certain number of iterations), stop the algorithm and record the union of all identified cluster groups;
- solve the CCQP to optimality only for this group of assets using CPLEX.

To illustrate the idea, we apply the above procedure to the result for the 3,000-choose-100 case. In Fig. 6 we see that the algorithm appears to have converged after 15,916 seconds. We take a union of the assets in the 100 clusters from the local relaxation algorithm, then feed these assets into CPLEX. Figure 7 shows the output from CPLEX at each iteration. It can be seen that the CPLEX branch-and-cut takes almost 20 minutes to reach a level close to the result given by the local relaxation method, and never quite reach it within the run time limit of one hour.

3.2.4 Local relaxation with warm-start

One feature of the local relaxation algorithm is that it can be warm started with minimum effort using the result from another algorithm or from a similar problem directly. We have seen that the simple successive truncation algorithm could often lead to a good local minimum at little computational cost. Taking advantage of this, we propose warm starting the local relaxation algorithm using the result from successive truncation as the initial centroids assets. Since local relaxation method is strictly feasible and monotonically improving, we can set a time limit and take the result at the end of the period. This way, we are guaranteed a solution no worse than the one

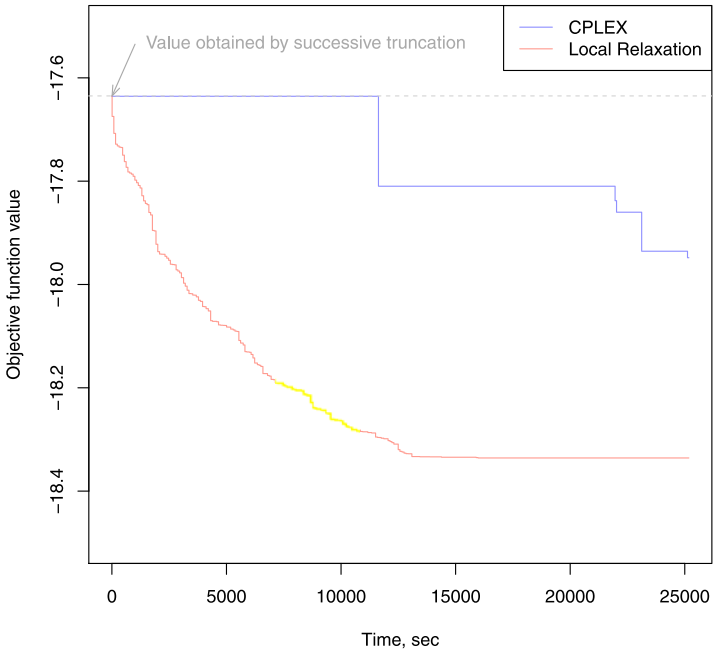

Fig. 8 CPLEX vs. local relaxation method performance comparison for 3,000 asset universe, with cardinality constraint, $K = 100$ and $\underline{x} = -\infty$ and $\overline{x} = \infty$; both methods are warm started using the solution of arithmetic successive truncation over 20 iterations. Maximum run-time is set to seven hours

obtained by successive truncation. This hybrid approach gives the best combination and has shown to be significantly more efficient than a warmed started CPLEX solve. Figure 8 shows the result, solving a 3,000 asset CCQP with $K = 100$ and $\underline{x} = -\infty$ and $\overline{x} = \infty$, by warm starting the local relaxation algorithm with the output from successive truncation. Note that at least for this instance, CPLEX has never managed to find a minimum that comes close to the one obtained by the proposed algorithm, even if it has been left running for 35 hours.

In order to assess the performance of our proposed algorithm in presence of inequality constraints, we impose a holdings bound $\underline{x} \leq x \leq \overline{x}$ and solve the portfolio problem for different cardinality constraints and relative risk tolerance parameters. Tables 8 and 9 show the computational results that compare the proposed algorithm with CPLEX based on two typical sets of holding constraints. The CPLEX MIQP solver setup is per Table 5 except for the number of threads is limited to one. We see that in most instances, the local relaxation method performs significantly better, both in term of finding a more optimal solution and of reducing the time taken to do so, especially when the holdings constraint is less tight relative to the final optimal solution and when the complexity of the problem is high.

Figure 9 illustrates typical progress plots for the two algorithms with $\lambda = 0.5$, $-0.5 \leq x \leq 0.5$, and for different cardinality constraints. We see that the successive truncation method is able to provide a good set of warm stating points for the local relaxation algorithm and, in some cases, these initial starting points are even better than those values obtained by CPLEX after ten hours of calculation. For cases where

Table 8 Result for the CPLEX vs. local relaxation method performance comparison for 3,000 asset universe, with cardinality constraint, $K = \{5, 15, 25, 50, 75, 100, 150, 300, 500, 1000\}$ and holdings constraint, $\underline{x} = -1$ and $\overline{x} = 1$. Elapsed times for local relaxation method include time spent in successive truncation, used to warm start the algorithm. CPLEX results, in brackets, are based on single thread execution. Time limit is set to 36,000 seconds for both algorithms. *Time to match CPLEX* is the time, in seconds, for the local relaxation algorithm to produce a result that matches the final optimal solution found by CPLEX

Cardinality, K:	5	15	25	50	75	100	150	300	500	1000
Panel A: $\lambda = 0.5$										
Objective value	−8.083 (−6.967)	−12.778 (−11.217)	−14.499 (−13.367)	−16.253 (−14.815)	−17.408 (−16.547)	−18.348 (−17.687)	−19.405 (−19.423)	−21.880 (−22.184)	−24.304 (−24.550)	−28,003 (−24,520)
Elapsed time, sec	460 (35,291)	1,345 (3,814)	1,698 (5,346)	15,822 (17,188)	34,001 (23,056)	32,956 (15,706)	35,707 (24,626)	36,000 (26,889)	35,685 (11,727)	33,547 (32,507)
Time to match CPLEX	144	195	120	185	103	1,002	†	†	†	209
Panel B: $\lambda = 1.0$										
Objective value	−4.041 (−3.271)	−6.382 (−6.026)	−7.249 (−6.142)	−8.127 (−7.721)	−8.704 (−8.388)	−9.173 (−8.981)	−9.714 (−9.808)	−10.933 (−11.156)	−12.140 (−10.676)	−14.01 (−12.046)
Elapsed time, sec	294 (1,019)	920 (5,851)	1,770 (25,813)	12,944 (29,477)	26,586 (11,947)	30,398 (10,862)	36,000 (22,872)	36,093 (31,808)	35,692 (33,807)	32,402 (31,266)
Time to match CPLEX	63	142	124	166	1,851	5,501	†	†	178	160
Panel B: $\lambda = 1.5$										
Objective value	−2.694 (−2.421)	−4.255 (−3.475)	−4.832 (−4.390)	−5.418 (−5.194)	−5.803 (−5.678)	−6.116 (−5.993)	−6.465 (−6.511)	−7.284 (−7.382)	−8.100 (−7.681)	−9.332 (−8.077)
Elapsed time, sec	544 (12,125)	1,283 (23,455)	1,486 (19,668)	14,972 (30,621)	26,673 (11,924)	33,597 (10,258)	36,000 (23,009)	36,000 (14,664)	33,592 (12,556)	35,595 (32,611)
Time to match CPLEX	195	102	122	131	5,859	5,934	†	†	180	192

†CPLEX has found a more optimal final solution than that by local relaxation

Table 9 Result for the CPLEX vs. local relaxation method performance comparison for 3,000 asset universe, with cardinality constraint, $K = \{5, 15, 25, 50, 75, 100, 150, 300, 500, 1000\}$ and holdings constraint, $\underline{x} = -0.5$ and $\overline{x} = 0.5$. Elapsed times for local relaxation method include time spent in successive truncation, used to warm start the algorithm. CPLEX results, in brackets, are based on single thread execution. Time limit is set to 36,000 seconds for both algorithms. *Time to match CPLEX* is the time, in seconds, for the local relaxation algorithm to produce a result that matches the final optimal solution found by CPLEX

Cardinality, K:	5	15	25	50	75	100	150	300	500	1000
Panel A: $\lambda = 0.5$										
Objective value	−7.241 (−7.285)	−12.121 (−10.505)	−13.870 (−12.840)	−15.618 (−14.310)	−16.775 (−15.015)	−17.714 (−16.379)	−18.792 (−18.335)	−21.221 (−21.407)	−23.654 (−23.821)	−27.371 (−24.999)
Elapsed time, sec	1,256 (24,618)	710 (5,010)	1,691 (3,902)	16,263 (7,707)	26,989 (32,694)	35,334 (25,979)	35,574 (12,694)	35,151 (33,019)	33,431 (28,646)	33,980 (25,956)
Time to match CPLEX	†	157	192	184	121	208	1,925	†	†	212
Panel B: $\lambda = 1.0$										
Objective value	−4.041 (−3.657)	−6.382 (−5.680)	−7.249 (−6.547)	−8.127 (−7.465)	−8.704 (−8.296)	−9.175 (−8.798)	−9.713 (−9.724)	−10.940 (−11.134)	−12.141 (−11.709)	−13.998 (−12.544)
Elapsed time, sec	308 (25,872)	727 (3,844)	2,65 (4,790)	14,926 (27,539)	26,758 (30,529)	29,470 (12,145)	36,000 (7,325)	34,767 (12,607)	34,291 (12,899)	36,000 (31,903)
Time to match CPLEX	308	184	183	173	120	208	†	†	196	222
Panel B: $\lambda = 1.5$										
Objective value	−2.694 (−2.145)	−4.255 (−4.029)	−4.833 (−4.063)	−5.418 (−5.115)	−5.803 (−5.635)	−6.116 (−5.791)	−6.472 (−6.514)	−7.291 (−7.410)	−8.104 (−7.980)	−9.337 (−8.170)
Elapsed time, sec	752 (1,962)	1,079 (5,879)	2,142 (18,343)	14,680 (28,502)	27,535 (12,959)	31,405 (25,581)	36,000 (27,568)	36,000 (34,767)	33,972 (25,365)	35,945 (27,598)
Time to match CPLEX	91	249	183	152	2,942	4,855	†	†	173	153

†CPLEX has found a more optimal final solution than that by local relaxation

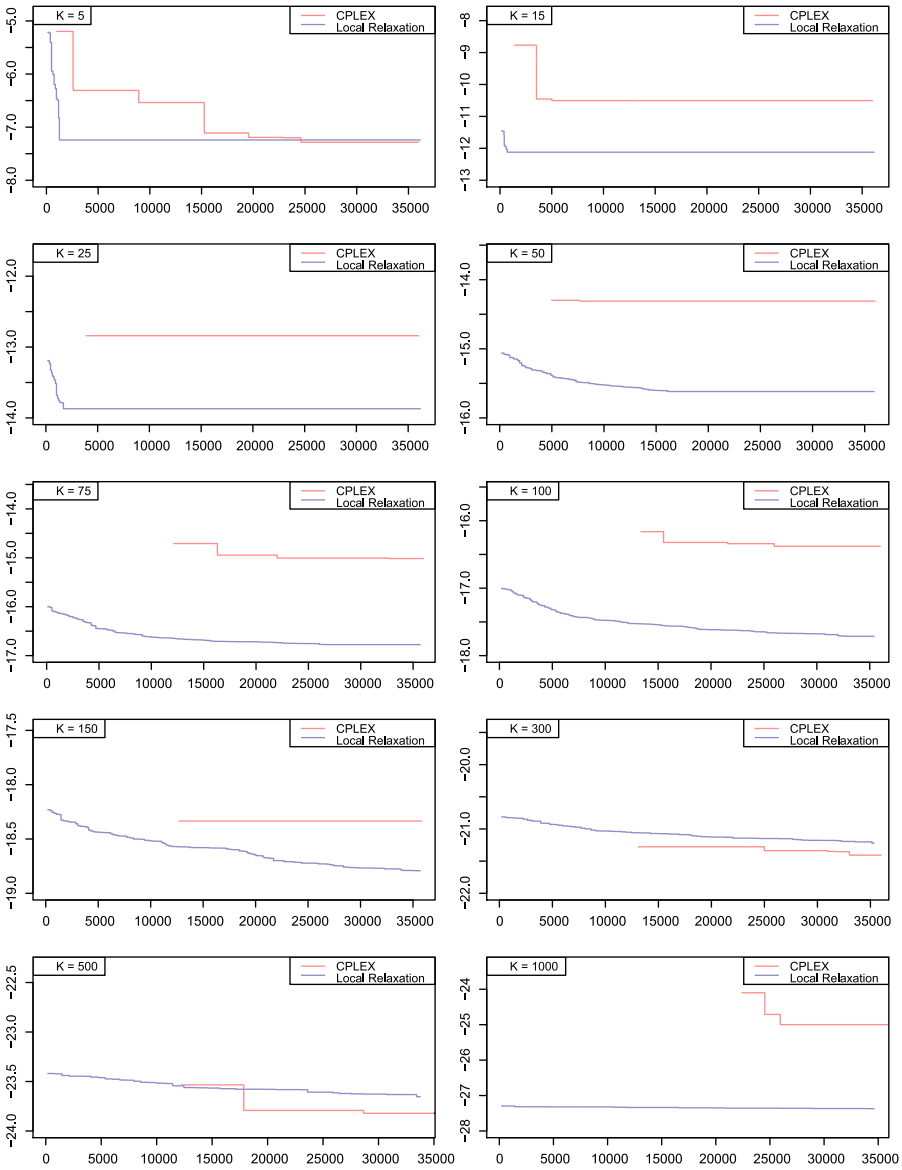

Fig. 9 CPLEX vs. local relaxation method performance comparison for 3,000 asset case, with $\lambda = 0.5$ and $\underline{x} = -0.5$ and $\overline{x} = 0.5$. Elapsed times for local relaxation method include time spent in successive truncation, used to warm start the algorithm. CPLEX results are based on single thread execution. Time limit is set to 36,000 seconds for both algorithms

the successive truncation is not able to suggest a good starting point, the local relaxation algorithm is still able to improve optimality monotonically at each iteration. For most cases, the local relaxation method is able to obtain a significantly better optimal solution at a fraction of the time that CPLEX takes.

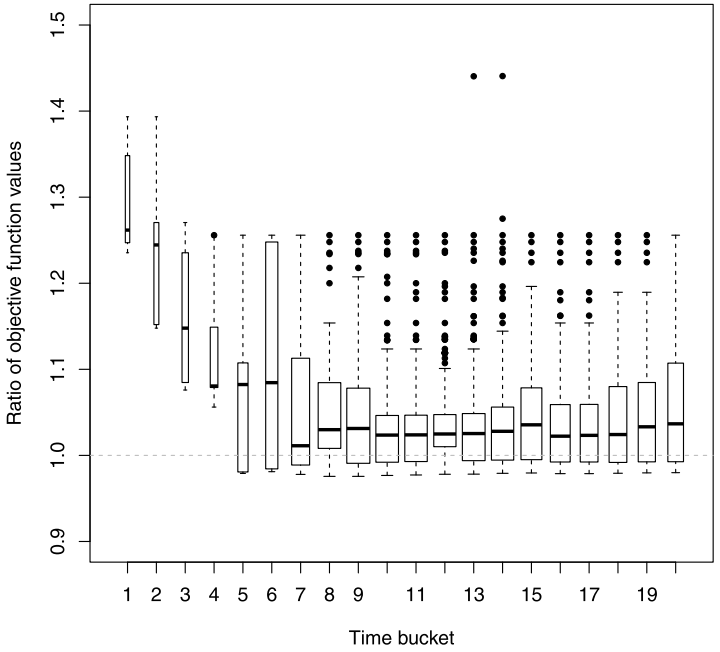

Fig. 10 Summary progress plot of the objective function value ratio, $\frac{f(x_t^*)_{local-relaxation}}{f(x_t^*)_{cplex}}$, where x_t^* is the optimal weights at time, $t \leq T$. *Each box unit* shows summary statistics over a 30 minute period, giving minimum and maximum, bounds of lower and upper quartiles and the median; width of the box gives relative sample size for each period. Result is based on 3,000 asset universe, with different values of λ, K, $\underline{x}$ and $\overline{x}$. Elapsed times for local relaxation method include time spent in successive truncation, used to warm start the algorithm. CPLEX results are based on single thread execution. Time limit is set to 36,000 seconds for both algorithms

Figure 10 summaries in results in Tables 9 and 5. Each box unit corresponds to a set of summary statistics of all unique objective function value ratios, local relaxation over that of CPLEX, over a series of non-overlapping thirty-minute periods. Figure 11 plots the ratio for final objective function values, for different λ, K, $\underline{x}$ and $\overline{x}$, at the end of the ten-hour time limit. We see that for a majority of cases, the local relaxation method gives significant performance gain over CPLEX.

3.3 Mean-variance efficient frontier

The set of optimal portfolios formed based on different risk tolerance parameters is known as *frontier portfolios*, in the sense that all portfolios on the frontier are optimal, given the specific risk tolerance of an investor. Therefore, a mean-variance optimizing agent will only choose frontier portfolios. Figure 12 shows this efficient frontier for our universe of 3,000 US stocks.We observe that, in-sample, the CCQP solutions are strictly dominated by QP solutions without the cardinality constraint—a consequence of the cardinality constraint cutting off parts of the feasible space. We observe that this domination increases as we decrease the cardinality of the problem, as larger parts of the feasible space becomes inaccessible to the CCQP. However,

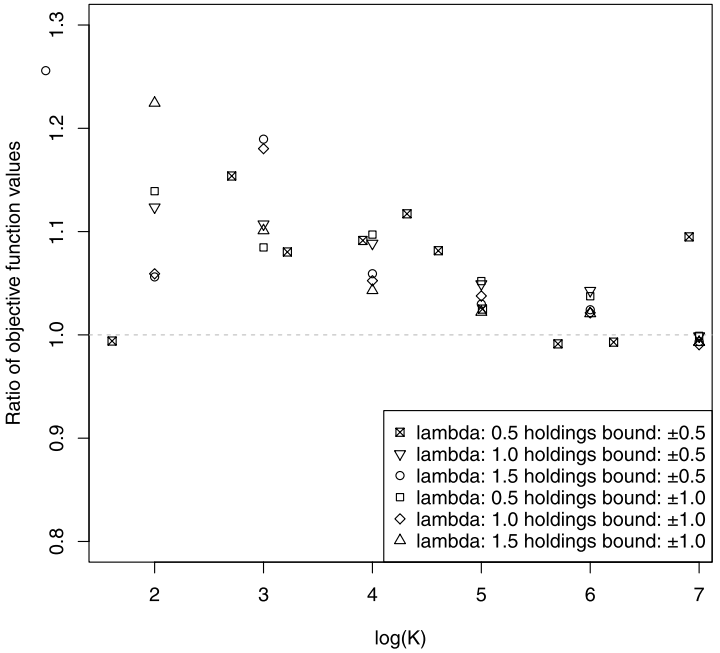

Fig. 11 Ratio of the local relaxation method final objective function value over that of CPLEX, $\frac{f(x_T^*)_{local-relaxation}}{f(x_T^*)_{cplex}}$, where x_T^* is the optimal weights at termination time, T. Result is based on 3,000 asset universe, with different values of λ, K, $\underline{x}$ and $\overline{x}$. Elapsed times for local relaxation method include time spent in successive truncation, used to warm start the algorithm. CPLEX results are based on single thread execution. Time limit is set to 36,000 seconds for both algorithms

in practice, the robustness of forming a portfolio with smaller subset of assets with higher expected returns can often lead to better out of sample performance.

4 Conclusion

The computational effort needed to be assured of solving problems of the size considered here is large and often beyond the performance of machines envisaged. However, solving a problem exactly is of questionable valued over that of obtaining a "near" solution. Here we have presented an algorithm that is both fast and which obtains an estimate of the solution that is better than those obtained by alternative methods.

It is possible to use other ways of defining clusters and we could also check the final solution by solving the reduced MIQP exactly in place of the center of gravity approached defined here. Indeed if computational resources, or the size of problem permits it, such an approach could be used at each iteration. The method has some versatility. For example, usually an improved solutions may be obtained by increasing cluster size at the cost of an increase in computational effort. A key quality of the method is that it can take full advantage of a known good solution as is the case when the problem is perturbed and a new solution sought. In particular it can obtain a good

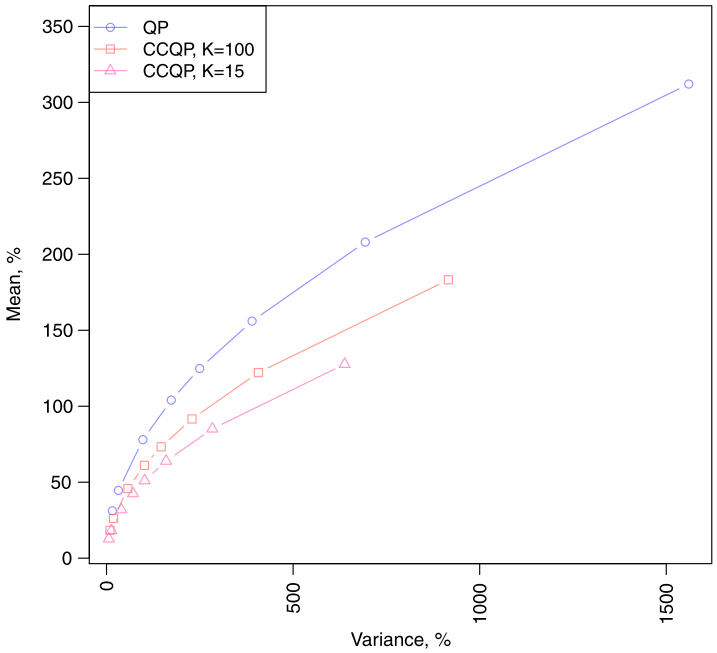

Fig. 12 Mean-variance efficient frontiers for a 3,000 asset universe. QP is the frontier without the cardinality constraint. CCQP is the frontier in presence of cardinality. To produce the frontier for CCQP, we warm-start the local relaxation algorithm, based on the successive truncation solution, and set a maximum run time limit of 252,000 seconds for the case where $K = 100$ and 3,600 second for the case where $K = 15$, and $\underline{x} = -\infty$ and $\overline{x} = \infty$

solution very fast if a current holding is eliminated or we reduce limits below current holdings.

Acknowledgements We thank the referees for their careful reading of the manuscript and for prompting us to improve the clarity of the presentation.

References

1. Bienstock, D.: Computational study of a family of mixed-integer quadratic programming problems. Math. Program. **74**, 121–140 (1995)
2. Chang, T.-J., Meade, N., Beasley, J.E., Sharaiha, Y.M.: Heuristics for cardinality constrained portfolio optimisation. Comput. Oper. Res. **27**(13), 1271–1302 (2000)
3. Clausen, J.: Branch and bound algorithms—principles and examples (2003)
4. Dakin, T.J.: A tree-search algorithm for mixed integer programming problems. Comput. J. **8**(3), 250–255 (1965)
5. Goldfarb, D., Idnani, A.: Dual and primal-dual methods for solving strictly convex quadratic programs. In: Numerical Analysis. Springer, Berlin (1982)
6. Hansen, P.R., Huang, Z., Shek, H.H.: Realized GARCH: a joint model for returns and realized measures of volatility. J. Appl. Econom. (2011). doi:10.1002/jae.1234. ISSN 1099-1255
7. Hastie, T., Tibshirani, R., Friedman, J. (eds.): The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, Berlin (2003)
8. Hosrt, R., Pardalos, P.M. (eds.): Handbook of Global Optimization. Kluwer Academic, Norwel (1995)

9. IBM: IBM ILOG CPLEX optimization studio V12.2 documentation. IBM ILOG (2010)
10. Konno, H, Yamamoto, R: Integer programming approaches in mean-risk models. Comput. Manag. Sci. **2**(4), 339–351 (2005). doi:10.1007/s10287-005-0038-9
11. Konno, H., Yamazaki, H.: Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market. Manag. Sci. **37**(5), 519–531 (1991)
12. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)
13. Leyffer, S.: Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Comput. Optim. Appl. **18**(3), 295–309 (2001)
14. Loraschi, A., Tettamanzi, A., Tomassini, M., Verda, P.: Distributed genetic algorithms with an application to portfolio selection. In: Artificial Neural Nets and Genetic, pp. 384–387. Springer, Berlin (1995)
15. Markowitz, H.M.: Portfolio selection. J. Finance **7**, 77–91 (1952)
16. Markowitz, H.M. (ed.): Mean-Variance Analysis in Portfolio Choice and Capital Markets. Blackwell, Oxford (1987)
17. Murray, W., Shanbhag, V.V.: A local relaxation method for nonlinear facility location problems. Multiscale Optim. Methods Appl. **82**, 173–204 (2006)
18. Murray, W., Shanbhag, U.V.: A local relaxation approach for the siting of electrical substation. Comput. Optim. Appl. **38**(3), 299–303 (2007)
19. Perold, A.F.: Large scale portfolio optimization. Manag. Sci. **30**(10), 1143–1160 (1984)
20. Shaw, D.X., Liu, S., Kopman, L.: Lagrangian relaxation procedure for cardinality-constrained portfolio optimization. Optim. Methods Softw. **23**(3), 411–420 (2008)
21. Shek, H.H.: Modeling high frequency market order dynamics using self-excited point process. Working paper, Stanford University (2010)
22. Shek, H.H.: Statistical and algorithm aspects of optimal portfolios. PhD thesis (2010)
23. Yitzhaki, S.: Stochastic dominance, mean variance, and Gini's mean difference. Am. Econ. Rev. **72**(1), 178–185 (1982)