Large-Scale Portfolio Optimization

Andre F. Perold


Stable URL:
http://links.jstor.org/sici?sici=0025-1909%28198410%2930%3A10%3C1143%3ALPO%3E2.0.CO%3B2-V

Management Science is currently published by INFORMS.

Your use of the JSTOR archive indicates your acceptance of JSTOR’s Terms and Conditions of Use, available at http://www.jstor.org/about/terms.html. JSTOR’s Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at http://www.jstor.org/journals/informs.html.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is an independent not-for-profit organization dedicated to creating and preserving a digital archive of scholarly journals. For more information regarding JSTOR, please contact jstor-info@jstor.org.
LARGE-SCALE PORTFOLIO OPTIMIZATION*

ANDRE F. PEROLD
Graduate School of Business Administration, Harvard University,
Boston, Massachusetts 02163

This paper describes a practical algorithm for large-scale mean-variance portfolio optimization. The emphasis is on developing an efficient computational approach applicable to the broad range of portfolio models employed by the investment community. What distinguishes these from the "usual" quadratic program is (i) the form of the covariance matrix arising from the use of factor and scenario models of return, and (ii) the inclusion of transactions limits and costs. A third aspect is the question of whether the problem should be solved parametrically in the risk-reward trade off parameter, \( \lambda \), or separately for several discrete values of \( \lambda \). We show how the parametric algorithm can be made extremely efficient by "sparsifying" the covariance matrix with the introduction of a few additional variables and constraints, and by treating the transaction cost schedule as an essentially nonlinear nondifferentiable function. Then we show how these two seemingly unrelated approaches can be combined to yield good approximate solutions when minimum trading size restrictions ("buy or sell at least a certain amount, or not at all") are added. In combination, these approaches make possible the parametric solution of problems on a scale not heretofore possible on computers where CPU time and storage are the constraining factors.

(FINANCE; PORTFOLIO)

1. Introduction

This paper describes a practical algorithm for large-scale mean-variance portfolio optimization. The emphasis is on developing an efficient computational approach applicable to the broad range of portfolio models employed by the investment community. Stimulus for this work came from the observation that existing algorithms and implementations are (at best) efficient in processing only the particular problem formulation for which they were designed. While some of the ideas presented here were reported earlier in Markowitz and Perold (1981a), they are included here (and are considerably expanded upon) to make the paper self-contained.

The essence of this approach is the exploitation of the structural properties of current day portfolio models. What distinguishes these from the "usual" quadratic program is (i) the form of the covariance matrix arising from the use of factor and scenario models of return, and (ii) the inclusion of transactions limits and costs. A third aspect is the question of whether the problem should be solved parametrically in the risk-reward trade off parameter, \( \lambda \), or separately for several discrete values of \( \lambda \). We show how the parametric algorithm can be made extremely efficient by "sparsifying" the covariance matrix with the introduction of a few additional variables and constraints, and by treating the transaction cost schedule as an essentially nonlinear nondifferentiable function. Then we show how these two seemingly unrelated approaches can be combined to yield good approximate solutions when minimum trading size restrictions are added.

The idea of taking sparsity into account is by no means new, but its ability to yield dramatic computational savings has gone unappreciated and unexploited in the recent literature on portfolio optimization algorithms. The approach we take in the handling of transactions costs, while derived from separable programming, is new, especially in the way it is implemented. The extension to include minimum trading size restrictions appears altogether novel. In combination, these approaches make possible the parametric solution of problems on a scale not heretofore possible on computers where CPU time and storage are the constraining factors.

In the remainder of this introductory section, we state the mathematical program being solved, and then, in §§1.3, 1.4 and 1.5, motivate where this formulation and the coefficients come from. §1.6 comments briefly on the practical applications of mean variance optimization; §1.7 reviews existing computational approaches.

* Accepted by Marshall L. Fisher, former Departmental Editor; received December 1, 1981. This paper has been with the author 8 months for 2 revisions.
1.1. Statement of the Problem

For each \( \lambda > 0 \) solve

\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} x^T C x - d^T x - \lambda r(x) \\
\text{subject to:} & \quad Ax = b, \quad h(x) \leq \tau, \quad l \leq x \leq u,
\end{align*}
\]

where \( x \in \mathbb{R}^n \), \( C \) is symmetric positive semidefinite, \( r(\cdot) \) is an arbitrary concave separable piecewise linear function, i.e. \( r(x) = \sum_{i=1}^{n} r_i(x_i) \) where each \( r_i(\cdot) \) is concave and piecewise linear, \( h(\cdot) \) is of the form

\[
h(x) = \sum_{i \in \gamma} (x_i - x_i^0)^+ (y^+ \triangleq \max(y, 0))
\]

for some set \( \gamma \subset \{1, \ldots, n\} \), and \( A \) is an \( m \times n \) constraint matrix.

It will be desirable but nonessential that the matrices \( C \) and \( A \) exhibit as much sparsity as possible.

In addition, when the data are such that (1) can be formulated with \( C \) having nonzero off diagonal entries in a relatively small number of rows and columns, disjunctive constraints \( x_i \in [l_i, l_i^0] \) or \( x_i = x_i^0 \) or \( x_i \in [u_i^0, u_i] \) will be allowed (and approximately obeyed).

1.2. Interpretation

The mean-variance optimization problem posed originally by Markowitz (1959) in 1952 involves \( n \) securities that are jointly distributed with mean vector \( \mu \) and covariance matrix \( C \). A portfolio \( x \) is a vector of holdings in the securities whose components sum to unity, and its expected return and variance are given by \( \mu^T x \) and \( x^T C x \) respectively. Subject to a set of linear constraints, the efficient frontier is the set of portfolios that have maximal expected return given an upper bound on the variance, and minimal variance given a lower bound on the expected return. It can be shown (e.g. Markowitz 1959) that the efficient frontier is also the solution \( x(\lambda) \) that minimizes \( \frac{1}{2} x^T C x - \lambda \mu^T x \) over the constraint set, as \( \lambda \) ranges from zero to infinity.

Formulations employed in practice today sometimes include a linear term \( d^T x \) in the formula for the variance, and almost always adjust the expected return to take transaction costs into account. The function \( r(x) \) is thus the expected return \( \mu^T x \) less transaction costs (see 1.4). \( h(x) \) is the sum of purchases in the securities contained in \( \gamma \), \( x^0 \) being the vector of current holdings. When \( \gamma \) includes all securities, this becomes the portfolio turnover, a frequently constrained quantity.

The disjunctive constraints are the minimum trading size constraints: Stay at the current holding \( x_i^0 \), or if more is to be purchased, buy at least \( u_i^0 - x_i^0 \), or if some is to be sold, sell at least \( x_i^0 - l_i^0 \).

1.3. Models of Covariance

Estimating the \( n \) expected returns and \( n(n+1)/2 \) variances and covariances is by far the most difficult aspect of mean-variance analysis, especially with large numbers of securities. The estimation techniques in use today are nearly all based on the specification of a linear return generating process whose parameters, once estimated, can be used to infer the above quantities of interest. The most widely used models are the factor and/or index\(^1\) models (e.g. Sharpe 1970, Cohen and Pogue 1967, Rosenberg 1974). These require the selection of a few, say 20, common factors that impact the

\(^1\)The subtle distinction between factors and indexes is not of interest in this paper; we shall treat them here as being the same.
returns of all or a large group of securities simultaneously, and making the assumption that any remaining variability in the return on a security is specific to that security only, and (largely) independent of events that impact the returns on other securities. More specifically, if \( R_i \) denotes the return on the \( i \)th security, and \( F_k \) is the level of the \( k \)th factor, then the postulated return generating process is

\[
R_i = \alpha_i + \beta_{i1} F_1 + \cdots + \beta_{i\ell} F_\ell + \epsilon_i
\]  

(2)

for some constants \( \alpha_i, \beta_{ij}, \) and a random disturbance \( \epsilon_i \) that is uncorrelated with the \( F_k \) and \( \epsilon_j, j \neq i \). It is straightforward to deduce that the implied covariance matrix has the form

\[
C = D + G^T P G
\]  

(3)

where \( D \) is a diagonal matrix with \( i \)th diagonal entry \( \sigma_i^2 = \text{var}(\epsilon_i) \), \( G \) is the \( t \times n \) matrix of coefficients \( \beta_{ik} \), and \( P \) is the \( t \times t \) covariance matrix of the factors.

Other return generating models that, to our knowledge, have been used to date in practice all result in the same form (3) for the covariance matrix. These include the scenario model (Markowitz and Perold 1981b, Hobman 1975), and the multi-group model (Elton and Gruber 1973).

An important aspect of all these models is that \( t \) is small and fixed relative to \( n \), the latter implying that the estimation requirements (and, as we shall obtain, the computational effort) go up linearly in \( n \). Further, most models encountered in practice seem to have between 1 and 12 nonzeros in \( G \) per security, with \( t \) ranging from 1 to as high as 60 or 70. Thus \( G \) is typically a dense matrix for small \( t \) and a sparse matrix for larger \( t \).

In some applications, when working with a factor and or scenario model, the matrix \( D \) in (3) becomes block diagonal, i.e. certain small clusters of securities have significant within cluster residual correlations and insignificant between cluster residual correlations. For example, if the \( n \) securities consist of \( n/2 \) stocks and \( n/2 \) options, one per stock, then \( D \) will have \( n/2 \) blocks of order two on its diagonal.

1.4. Transactions Costs

Most applications of portfolio optimization involve the revision of an existing portfolio \( x^0 \), often because expectations of the future have changed and/or cash flows such as dividends and new subscriptions have to be (re)invested. This revision will

\[
\text{Figure 1. Transactions Cost.}
\]
usually entail both purchases and sales, the costs of which must enter into the formula for the expected return. These costs typically consist of brokerage commissions, taxes, illiquidity effects, and the like (see e.g. Pogue 1970), and will generally be a separable, nonlinear, nonconvex function of the difference in holdings, $x - x^0$, of the new and existing portfolios. Figure 1 is an example of a transactions cost function that may arise when brokerage commissions decrease and illiquidity costs increase with the size of the transaction.

The assumption usually made in practice is that the transactions cost for a given security is V-shaped, i.e. proportional in purchases and proportional in sales. In some cases the cost is assumed flat (zero) within a given interval, and proportional to either side, the particular interval and marginal costs being determined by certain long-term versus short-term considerations (see Perold 1981). With severe illiquidities and/or varying tax brackets, it becomes necessary to include several additional pieces of linearity.

1.5. Constraints

Constraints are imposed on the portfolio holdings for a variety of reasons. The budget constraint (all holdings summing to unity) is a normalization. Upper and lower bounds on the holdings are imposed partly because of institutional restrictions (e.g. nonnegativity restrictions to reflect no short sales) and partly because of deficiencies in the estimation of the returns and covariances. More general linear constraints are imposed usually to accomplish certain objectives, for example, limiting the exposure of a portfolio to fluctuations in one or more common factors. The portfolio turnover, defined to be the sum of purchases or sales, is also frequently constrained partly because of the inadequacy of proportional transactions costs, and partly because repeated use of a one-period model leads to greater turnover than one would obtain with a multi-period model.

Minimum trading size constraints are disjunctive constraints of the form “buy or sell at least a certain amount, or not at all.” Short of using an algorithm for mixed integer quadratic programming, it has not been possible, to date, to implement such constraints. However, such a capability is very desirable, especially with large numbers of securities, because optimal mean-variance portfolios often contain a large number of undesirable small trades or holdings. Small trades are undesirable because of the nonconvexity of the true transactions cost function (Figure 1). Portfolios with many small holdings are undesirable because of the large fixed costs involved in staying up-to-date with a security: securities not in the portfolio on average are paid much less attention.

1.6. Practical Applications of Mean-Variance Optimization

Mean-variance optimization has, under the banner of Modern Portfolio Theory (see e.g. Rudd and Clasing 1982), gained widespread acceptance as a practical tool for portfolio construction. This has occurred over the last decade primarily as a result of the advances made in estimating covariances of return. A great many investment advisory firms and pension plan sponsors (and/or their consultants) today routinely compute mean-variance efficient portfolios as part of the portfolio allocation process. Specific applications include asset allocation (allocation across the broad asset classes such as stocks and bonds), multiple money manager decisions (allocation across money managers with different strategies and objectives), index matching (finding a portfolio whose returns will closely track those of a predetermined index such as the S&P 500), and active portfolio management (optimizing risk-return tradeoffs assuming superior investment judgement). Asset allocation and multiple money manager decisions usually involve a small number of asset classes, typically from 4 to 10 and
sometimes as high as 50; index matching usually involves a very large number of holdings, typically from 200 to 500 but sometimes more than 1000; active portfolio management falls somewhere in between.

The above applications are the most common and, in the case of index matching and active portfolio management, involve common equities almost exclusively. Mean-variance optimization is being applied to portfolios of bonds and other types of fixed income securities, but only in more isolated instances. More recently an increasing number of applications have included options, stock index futures, commodity futures, and oil and gas prospects. The brokerage community in particular has begun using index matching to exploit apparent mispricing in stock index futures.

1.7. Existing Approaches

With the exception of Sharpe's work, discussed below, little or no attention has been paid to the inclusion of transaction costs other than by defining a separate variable for each piece of linearity, thus at least doubling or tripling the number of variables, which in turn has a comparable effect on the overall computational effort. As regards the turnover constraint, the problem disappears once one has already gone so far as to introduce the above-mentioned additional variables, since it then becomes linear. Schreiner (1980) has shown how the turnover constraint can be handled directly, but does not address the question of transactions costs.

One of the most widely used methods in current day practice is Von Hohenbalken's algorithm (Von Hohenbalken 1975, Rudd and Rosenberg 1979, and Wilshire Associates 1979), an iterative method that makes successive approximations of the constraint set using simplices of increasing dimension. It fairly quickly gives a good approximate solution but is extremely slow in attaining the optimum exactly. A similar method, due to Sharpe (1978), has the advantage of being extremely well suited for use out-of-core but can handle only upper and lower bounds on the variables, and a budget constraint. All other constraints have to be introduced with the use of penalty functions (fixed penalties). In these algorithms the covariance matrix enters only into the computation of the gradient of the objective function, and any special structure in C can thus be easily exploited. Both algorithms, however, can compute only one portfolio at a time on the efficient frontier, i.e. they are not parametric in λ.

There is essentially only one algorithm for computing the efficient frontier parametrically, namely the complementary pivot algorithm of Markowitz (1956) and Wolfe (1959). To see this consider the optimality conditions for (1), for now simplifying r(·) to be the linear function μTx and omitting the turnover constraint: x is optimal for a given λ if and only if there exist η and π such that

\[
\begin{bmatrix}
C & A^T & -I \\
A & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
π \\
η
\end{bmatrix}
= \begin{bmatrix}
d \\
b
\end{bmatrix} + \begin{bmatrix}
μ \\
0
\end{bmatrix} λ, \quad l \leq x \leq u, \quad π \text{ unrestricted, and}
\]

(5)

\[x_i = l_i \Rightarrow \eta_i \geq 0, \quad l_i < x_i < u_i \Rightarrow \eta_i = 0, \quad x_i = u_i \Rightarrow \eta_i \leq 0.\]  

(6)

The algorithm involves adjusting the value of λ while simultaneously maintaining the constraints (5) and, in particular, the complementarity conditions (6). Since, under a nondegeneracy assumption, these conditions uniquely determine x, π, and η as functions of λ, and since the conditions are necessary and sufficient, it follows that the path followed by the algorithm and the efficient frontier are one and the same.

---

2This turns out not to be a drawback in practice.
3This statement requires d to be in the range of C. We shall see later that when d is not in the range of C, uniqueness as a function of λ need not obtain.
Algorithms that solve the problem parametrically can thus differ only in the manner in which the basis matrix is factorized and updated from one iteration to the next (see §3), and in how the process is initiated. Existing implementations that exploit the covariance structure (3) are due to Pang (1980), IBM (1965) and Williamson and Downs (1970). However, these can handle only very special cases, Pang requiring $C$ to be positive definite, and IBM and Williamson and Downs requiring the matrix $G$ in (3) to have rank 1.

2. The Algorithm

Before deriving the algorithm, it will be convenient, both conceptually and computationally, to restate the problem formulation (1). Firstly, we incorporate the upper and lower bounds $l$ and $u$ into $r(\cdot)$ by letting $r(x_i) = -\infty$ if $x_i < l_i$ or $x_i > u_i$. Secondly, we define a slack variable for the turnover constraint, $h(x) \leq \tau$, view it as a fictitious security with a current holding and lower bound of zero, and augment the remaining vectors and matrices with zeros in the appropriate places in order that the problem can now be stated (equivalently) as:

For each $\lambda > 0$, solve

$$\text{minimize: } \frac{1}{2} x^T C x - d^T x - \lambda r(x) \quad \text{subject to: } A x = b, \quad h(x) = \tau.$$  

(7)

The approach here is exactly the same as indicated in 1.7: vary $\lambda$ subject to maintaining the optimality conditions. Because the functions $r$ and $h$ are non-differentiable, these conditions have to be stated in terms of their subgradients.

**Definition 1** (Rockafellar 1970, p. 214). Let $f: R^k \rightarrow R \cup \{-\infty\}$ be a concave function. The subgradient of $f$ at $x$, denoted by $\partial f(x)$, is the set of all $y \in R^k$ such that $f(z) \leq f(x) + y^T (x - z)$ $\forall z \in R^k$.

For $f$ a concave piecewise linear function of one variable with breakpoints $t_0$, $t_1, \ldots, t_p$, gradient $g_i$ on the interval $(t_{i-1}, t_i)$, $i = 1, \ldots, p$, and taking on the value $-\infty$ for $x < t_0$ and $x > t_p$, the subgradient is easily seen to be:

$$\partial f(x) = \left\{ g_i \right\}, \quad x \in (t_{i-1}, t_i), \quad i = 1, \ldots, p,$$

$$= \left[ g_{i-1}, g_i \right], \quad x = t_i, \quad i = 1, \ldots, p - 1,$$

$$= \left[ g_0, \infty \right], \quad x = t_0,$$

$$= (-\infty, g_p], \quad x = t_p.$$

If $f: R^n \rightarrow R$ is given by $f(x) = \sum f_i(x_i)$ where each $f_i$ is as above, then the subgradient of $f$ is the cross product of the subgradients of the $f_i$, i.e. $\partial f(x) = \prod_{i=1}^n \partial f_i(x_i)$.

**Theorem 1.** Optimality Conditions (Rockafellar 1970, p. 283). Let $\lambda > 0$ be given. Then $x$ is optimal for (7) if and only if there exist $\pi$, $\xi$, and $\rho > 0$ such that

$$Cx + A^T \pi - \xi = d, \quad Ax = b, \quad h(x) = \tau,$$

$$\xi \in \lambda \partial r(x) - \rho \partial h(x).$$  

(8)

(9)

The equations in (8) are simply a partial statement of feasibility and a definition of $\xi$. (9) is where the content lies, being a statement both of the inequalities that the variables have to satisfy and of the complementarity relationships between the primal and dual variables.

It will be useful at this stage to introduce the following notation: Let $\mathcal{P}_i$ denote the (finite) set of breakpoints in the interval $[l_i, u_i]$ at which either $r_i$ or $h_i$ is non-differenti-
able, $h_i$ being the $i$th term in the definition of $h$.\footnote{$h$ may without loss of generality be assumed an arbitrary separable convex piecewise linear function.} By the way we have defined $r$, $\mathcal{P}_i$ will always include at least $l_i, u_i$, these being allowed to take on the values $-\infty, +\infty$ respectively. For any feasible $x$, we shall let the disjoint index sets $\alpha$ and $\beta$, $\alpha \cup \beta = \{1, \ldots, n\}$, be defined generically as

$$\alpha = \{i : x_i \in \mathcal{P}_i \}, \quad \beta = \{i : x_i \notin \mathcal{P}_i \}. \quad (10)$$

Thus the functions $r_\beta$ and $h_\beta$ are differentiable at $x_\beta$, while $r_\alpha$ and $h_\alpha$ will in general be only subdifferentiable at $x_\alpha$. $r'_\alpha, r'_\beta$ and $h'_\alpha, h'_\beta$ will denote the gradients of $r$ and $h$, at $x_\alpha, i \in \beta$; $r'_{\alpha+}, r'_{\alpha-}, h'_{\alpha+}, h'_{\alpha-}$ will denote right and left derivatives of $r_\alpha$ and $h_\alpha$ at $x_\alpha, i \in \alpha$. The dependence on $x$ of $\alpha, \beta$, and these quantities, will be clear from the context.

We can now restate (9) as

$$i \in \beta \Rightarrow \xi_i = \lambda r'_\alpha - \rho h'_\beta, \quad (11)$$

$$i \in \alpha \Rightarrow \xi_i \in [\lambda r'_{\alpha+} - \rho h'_{\alpha+}, \lambda r'_{\alpha-} - \rho h'_{\alpha-}]. \quad (12)$$

It is easily checked that these conditions, together with (8), reduce to (5) and (6) when $h$ is absent and $r$ is linear.

By combining (11) and (8), and holding $x_\alpha$ fixed, it follows that $x_\beta, \xi_\alpha, \pi, \rho$ and $\lambda$ will satisfy

$$\begin{bmatrix}
C_{\beta \beta} & A_{\beta}^T & 0 & -I
C_{\alpha \beta} & A_{\beta}^T & h'_\beta & 0
A_{\beta} & h'_\beta & 0 & \pi
(h'_{\beta})^T & \xi_\alpha & \rho & \pi
\end{bmatrix}
\begin{bmatrix}
x_\beta \\
\pi \\
\rho \\
\xi_\alpha
\end{bmatrix} = \begin{bmatrix}
d_{\beta}^* \\
d_{\beta}^* \\
r'_\beta \\
\tau^*
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix} \lambda, \quad (13)$$

where $d'' = d - C_{\alpha} x_\alpha, b'' = b - A_{\alpha} x_\alpha$, and $\tau''$ is equal to $t - h_\alpha(x_\alpha)$ less the constant terms appearing in the formulae for $h_\alpha(\cdot), i \in \beta$. The coefficient matrix in (13) will be denoted by $B$, and termed the basis. $B$ will always be kept nonsingular (as we shall see) and will uniquely define the basic variables $(x_\beta, \pi, \rho, \xi_\alpha)$ as linear functions of $\lambda$. As $\lambda$ varies, the $x_\beta$ variables will be constrained to lie within their current intervals, the $\xi_\alpha$ variables will be constrained to satisfy (12), and $(\pi, \rho)$ will be unrestricted.\footnote{While $\rho$ may be at its lower bound of zero, it follows from the way in which the slack variable was added to make the turnover constraint an equality that $\rho$ may always be kept in the basis.}

We can now state a generic iteration of the algorithm.

2.1. **A Generic Iteration**

Assume that we are given an initial solution satisfying (8) and (9) for some $\lambda = \lambda_0$. Let $\alpha$ and $\beta$ be given by (10), and assume that the corresponding basis $B$ is nonsingular. Holding $x_\alpha$ fixed and letting (11) determine $\xi_\beta$, express the basic variables as linear functions of $\lambda$, by solving the equations

$$B(\bar{x}_\beta, \bar{\pi}, \bar{\rho}, \bar{\xi}_\alpha) = (d''_{\alpha}, d''_{\beta}, b'', \tau''), \quad (14)$$

$$B(\bar{x}_\beta, \bar{\pi}, \bar{\rho}, \bar{\xi}_\alpha) = (0, r'_\beta, 0, 0), \quad (15)$$

so that

$$\begin{bmatrix}
x_\beta \\
\pi \\
\rho \\
\xi_\alpha
\end{bmatrix} = \begin{bmatrix}
\bar{x}_\beta \\
\bar{\pi} \\
\bar{\rho} \\
\bar{\xi}_\alpha
\end{bmatrix} + \begin{bmatrix}
\bar{x}_\beta \\
\bar{\pi} \\
\bar{\rho} \\
\bar{\xi}_\alpha
\end{bmatrix} \lambda, \quad (16)$$
Decrease \( \lambda \), starting at \( \lambda_0 \), to the first critical value \( \lambda^* \) with property that any further decrease in \( \lambda \) will cause an \( x_i \), \( i \in \varphi \), to move beyond a breakpoint defining its current interval, or cause a \( \xi_i \), \( i \in \alpha \), to move outside the parametric interval given in (12), or cause \( \lambda \) to become negative. The critical value of \( \lambda \) determined by \( x_\beta \) can be found by performing the usual minimum ratio test with \( \overline{x}_\beta, \overline{x}_\beta \), and the relevant breakpoints. The critical value determined by \( \xi_i \), \( i \in \alpha \), is found by substituting \( \rho = \overline{\rho} + \overline{\rho} \lambda \) into (12) to obtain
\[
\overline{\xi}_i + \overline{\xi}_i \lambda \in [\overline{\delta}_i + \overline{\delta}_i \lambda, \overline{\rho}_i + \overline{\rho}_i \lambda] \text{ for some } \overline{\delta}_i, \overline{\delta}_i, \overline{\rho}_i, \overline{\rho}_i.
\]
This translates into the two inequalities
\[
\begin{align*}
(\overline{\xi}_i - \overline{\delta}_i) \lambda + (\overline{\xi}_i - \overline{\xi}_i) & > 0, \\
(\overline{\xi}_i - \overline{\rho}_i) \lambda + (\overline{\xi}_i - \overline{\xin}_i) & < 0,
\end{align*}
\]
which can also be solved for the critical \( \lambda \) using a ratio test. \( \lambda^* \) is of course the largest of these two critical values and zero.

If \( \lambda^* = 0 \) terminate the procedure. Else, identify the critical index \( i^* \) that defines \( \lambda^* \). If \( i^* \in \beta \), define new index sets \( \tilde{\alpha} = \alpha \cup \{i^*\}, \tilde{\beta} = \beta \setminus \{i^*\} \) and label this an out-pivot. If \( i^* \in \alpha \), define new index sets \( \tilde{\alpha} = \alpha \setminus \{i^*\}, \tilde{\beta} = \beta \cup \{i^*\} \) and label this temporarily an in-pivot. Let \( \tilde{B} \) be the new basis determined by the pair \( (\tilde{\alpha}, \tilde{\beta}) \) where, for an in-pivot, the interval assigned to \( i^* \) is the interval to the right of the breakpoint if \( \lambda^* \) is determined by (17) (i.e. involving right derivatives), and the interval to the left of the breakpoint otherwise.

If \( \tilde{B} \) is nonsingular go to the start of this iteration and repeat. With out-pivots this is always the case. \( \tilde{B} \) will be singular (see below) if and only if there is a nonzero vector \( y \) satisfying
\[
y_{i^*} = 1, \left[ \begin{array}{c} C_{\tilde{\alpha}\tilde{\beta}} \\ C_{\tilde{\beta}\tilde{\beta}} \\ A \cdot \tilde{\beta} \\ (h_{i^*})^T \end{array} \right] [y_{i^*}] = 0, \text{ and } y_{i^*} = 0. \tag{19}
\]
In such a case let \( \theta^* \) be the largest possible \( \theta \) such that \( x_\beta + \theta y_{i^*} \) remains within the given intervals, where \( \theta \) is negative or positive depending on whether \( x_{i^*} \) is to be moved left or right, i.e. whether \( \lambda^* \) was determined by (18) or (17). \( \theta^* \) is determined by a minimum ratio test, and will be the result of some variable, \( x_{i^*}, j^* \in \tilde{\beta}, j^* \neq i^* \), being blocked by a breakpoint. Define new index sets \( \tilde{\alpha} = \tilde{\alpha} \cup \{j^*\}, \tilde{\beta} = \tilde{\beta} \setminus \{j^*\} \), let \( \tilde{B} \) be the new basis determined by \( \tilde{\alpha} \) and \( \tilde{\beta} \), and go to the start of the next iteration. This is called an exchange-pivot, and will always result in \( \tilde{B} \) being nonsingular.

### 2.2. Discussion

It remains to be shown (i) that this procedure is valid, i.e., the steps are well defined, (ii) that termination will occur after finitely many steps at \( \lambda = 0 \) given any starting solution satisfying the optimality conditions, and (iii) that an initial solution can be found (easily). The third aspect is addressed in 2.3.

We remark first that with \( h(\cdot) \) absent, \( r(\cdot) \) linear within the bounds \( l \) and \( u \), and \( d \) in the range of \( C \), this is precisely the algorithm of Markowitz (1956) and Wolfe (1959) and shown by them to require only in- and out-pivots. Under the same conditions but with \( d \) outside the range of \( C \) one may in general require exchange-pivots, this more general case being the parametric linear complementarity algorithm of Cottle (1972), which in turn is based on Graves' (1967) nonparametric version. These papers show that the algorithm is valid in these special cases, and that finite termination occurs when degeneracy is appropriately taken into account.
We next argue that with $r(\cdot)$ and $h(\cdot)$ piecewise linear the same results apply, since each pivot involves only the local linear segments defined by the primal basic variables indexed by $\beta \cup \hat{\beta}$. Thus, as far as any proofs are concerned, we may assume that $r(\cdot)$ and $h(\cdot)$ are indeed linear. The condition that $r(\cdot)$ be concave and $h(\cdot)$ be convex is required (from the point of view of the algorithm) only to ensure that the interval of feasibility for $\xi_i$, $i \in \alpha$, in (12) is nonempty.

2.3. Getting Started

When $r(\cdot)$ is bounded above on the constraint set, as it always is with portfolio problems, the most efficient way of getting started is to solve

$$\text{maximize } r(x) : Ax = b, \quad h(x) = \tau. \tag{20}$$

This problem is readily solved using a modified simplex algorithm. If (20) has a unique optimal solution (easily determined by inspection of the vector of "reduced costs") this is also an efficient point, and a starting solution is immediately at hand for some sufficiently large $\lambda$. Else, the minimum variance solution must be sought from among the optimal solutions to (20). This can be done by employing the complementary pivot algorithm outlined above, where the starting solution is obtained by perturbing $r$ to obtain uniqueness, and where nonoptimal $x$ variables are temporarily considered ineligible for pivoting. The advantages of starting at the "top" of the frontier, i.e., by letting $\lambda$ be sufficiently large and solving (20), are three-fold: (i) solving (20) takes relatively little effort, and can sometimes even be done by inspection; (ii) as will be seen in §5, the cost per iteration often goes up as the algorithm progresses down the frontier; and (iii) it is in any event the upper part of the efficient frontier that is most often desired.

When $r(\cdot)$ is not bounded above, the vastly more expensive procedure must be used of first solving the problem for $\lambda = 0$ (the minimum variance solution) with the use of an artificial $r(\cdot)$ that is bounded above, and then performing the complementary pivot operations backwards.

3. A Sparse Implementation

Let $z$, $\bar{z}$ and $\overline{z}$ respectively denote the three vectors appearing in (16), so that $z = \bar{z} + \overline{z} \lambda$. Since the algorithm involves little arithmetic once $\bar{z}$ and $\overline{z}$ have been determined, it is clear that updating them from one iteration to the next will, explicitly or implicitly, account for almost all the work. $\bar{z}$ and $\overline{z}$ are defined by (14) and (15) as solutions of equations in terms of $B$, and it thus follows that updating them will involve the factor by which we have to multiply $B$ to obtain $\bar{B}$ or $\overline{B}$.

Notice that the change in $B$ involves up to three column interchanges: the exchange of a unit column for $(C_{\cdot, \cdot}, A_{\cdot, \cdot}, h_{\cdot, \cdot})$ or vice versa, likewise for $j^*$ in the event of an exchange-pivot, and the change from $h_{\cdot j}$ to $h_{\cdot \rho}$ (or $h_{\cdot \rho}$) in the column corresponding to $\rho$. The effect of each column exchange is to multiply $B$ on the left by an elementary matrix, as in the product form of the simplex method for linear programming (e.g. Dantzig 1963). It is straightforward to show that the new $\bar{z}$ and $\overline{z}$ can be obtained by multiplying the old ones by the inverses of these elementary matrices and then adjusting the result to take into account the change in the right-hand sides of (14) and (15), or in the reverse order, depending on the type of pivot. Each elementary matrix is determined by solving a system of the form

$$Bv = w \tag{21}$$

An elementary matrix is one that differs from the identity in a single column.
where $w$ is the incoming column. All that is required, therefore, from the point of view of implementation, is to find a representation of $B^{-1}$ that can both solve (21) and be updated in an efficient and stable manner.

In this light we next argue that $B$ is probably best treated as a general sparse matrix. The main reason for this is that with the covariance matrix structure (3), the problem can be sparsified, i.e. reformulated so that the matrix $B$ contains a very small percentage of nonzeros. This is done by noticing that the quadratic form $x^T Dx + y^T Py$ can be written equivalently as

$$x^T Dx + y^T Py$$

provided that $y$ and $x$ jointly satisfy

$$Gx - y = 0.$$  \hspace{1cm} (23)

Thus by adding a relatively small number of variables $y$ and as many constraints (23), we obtain an $(n + t) \times (n + t)$ “covariance” matrix $[D_{ab} \, 0]$ that is sparse. (Recall that $D$ is usually diagonal and occasionally block diagonal.) More importantly, the resulting basis$^7$

$$B = \begin{bmatrix}
D_{a\alpha} & A_{\alpha}^T & G_{\alpha}^T & 0 & -I \\
D_{\beta\beta} & A_{\beta}^T & G_{\beta}^T & h_{\beta} & 0 \\
0 & P & 0 & -I & 0 \\
A_{\cdot \beta} & 0 & 0 & 0 \\
G_{\cdot \beta} & -I \\
(h_{\beta})^T & 0
\end{bmatrix}$$

is sparse, and now automatically takes into account any additional sparsity that may be present in $A$ and $G$ (cf. 1.3, 1.5). In addition to the sparsity benefits, this formulation becomes the natural one when upper and lower bounds are imposed on the factor loadings, $y$, as is often the case.

That $B$ will be very sparse is reinforced by the fact that it contains mostly unit columns at the top of the frontier. This follows from (20) which, being in essence a linear program, will result in a $\beta$ of size $m + 1$, where $A$ is $m \times n$. As the algorithm progresses down the frontier, $\beta$ usually grows in size, i.e. the number of unit columns in $B$ decreases, and may in fact contain all the securities by the time the bottom of the frontier is reached. This, however, is very dependent on the nature of the constraints, as discussed in §5. Since it is often only the upper part of the frontier that is desired, these remarks imply that even when $C$ is 100% dense, e.g. when it is estimated directly from historical data, there is little lost in treating $B$ as being sparse.

Once the decision has been made to treat $B$ as a general sparse matrix, we can without modification use the sparse matrix routines available as part of any large-scale linear programming package. These commonly consist of an LU factorization routine, a routine to solve equations with respect to the LU factors, and a routine to update the LU factors after each change of basis. The most recent and successful LU implementations are due to Saunders (1976) and Reid (1975), both employing (in different ways) the numerically stable updating approach of Bartels and Golub (1970).

For our purposes the method of Reid seems preferable since it places most of the nonzeros in $U$. This means from the way the update is performed that, whenever the

$^7$This particular $B$ has all the $y$ variables basic, which will be the case when the $y$’s are constrained only by (23).
change of basis is an out-pivot, i.e. the incoming column is a unit column (containing far fewer nonzeros than the outgoing column), an overall net reduction in the numbers of nonzeros in L and U can be obtained. With piecewise linear transactions costs and tightly set upper and lower bounds, a great many of the basis changes will be of this form—see the computational results in §5. Saunders' method places most of the nonzeros in L (which in both methods can only grow with each update) and cannot profit as much from such pivots.

In concluding this section, we remark that the idea of defining additional variables and constraints in order to sparsify the problem was mentioned first by Sharpe (1970) in the context of portfolio optimization with multi-factor models, and is a well-known approach in the area of linear least squares. See e.g. Paige and Saunders (1978).

4. Minimum Transaction Sizes

As indicated earlier, we would ideally like to impose the (disjunctive) minimum transaction size constraints

\[ x_i \in [l_i, l^0_i] \quad \text{or} \quad x_i = x_i^0 \quad \text{or} \quad x_i \in [u_i^0, u_i]. \] (24)

We now show how approximate solutions can be obtained with minor modifications to the algorithm outlined in §2, in the case that the covariance structure (3) is present with D diagonal.

The approach is to work with the sparse formulation involving (22) and (23), except that the weighted sum of squares term \( x^T D x = \sum d_i x_i^2 \) in (22) is replaced by the separable function \( g(x) = \sum_{i=1}^n g_i(x_i) \) where each \( g_i \) is given by

\[ g_i(x_i) = d_{ii} x_i^2, \quad x_i \in [l_i, l^0_i] \cup \{ x_i^0 \} \cup [u_i^0, u_i]. \]

\[ = d_{ii} \left[ x_i x_i^0 + l^0_i (x_i - x_i^0) \right], \quad x_i \in (l_i^0, x_i^0), \] (25)

\[ = d_{ii} \left[ x_i x_i^0 + u_i^0 (x_i - x_i^0) \right], \quad x_i \in (x_i^0, u_i^0). \]

That is, \( g_i \) is equal to \( d_{ii} x_i^2 \) at feasible points, and is linear over the intervals of infeasibility as depicted in Figure 2.

\[ \begin{align*}
&\text{linearization} \\
&d_{11} x_1^2 \\
&\cdots \\
&g_1 \\
&\ell_1 \quad l_1^0 \quad x_1^0 \quad u_1^0 \quad u_1 \quad x_1
\end{align*} \] (Figure 2)
We then solve, for each \( \lambda > 0 \), the problem

\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} g(x) + \frac{1}{2} y^T P y - d^T x - \lambda r(x) \\
\text{subject to:} & \quad G x - y = 0, \quad A x = b, \quad h(x) = \tau.
\end{align*}
\]

(26)

It will turn out from what follows that the solution path \( x(\lambda) \) to (26) will have at most \( m + t + 2 \) variables that do not satisfy (24). Thus with a covariance matrix generated by \( t = 6 \) factors or scenarios, a 100 security run with a budget constraint \( (m = 1) \) and a turnover constraint will have at least 91 securities satisfying (24).

Let \((x, y)\) satisfy the constraints in (26), and augment the set of breakpoints \( \mathcal{P} \) to include \( l_0^t \) and \( u_0^t \). Let \( \alpha \) (as before) denote the index set of nonbasic \( x \) variables (the variables being held fixed at a breakpoint), let \( \beta \) denote the index set of basic \( x \) variables that are feasible in that they lie in one of \((l_t^t, l_0^t)\) or \((u_t^t, u_0^t)\), and let \( \gamma \) denote the remaining \( x \) variables, i.e., those that lie in the infeasible intervals \((l_t^t, x^0_t)\) or \((x^0_t, u_t^t)\). For clarity of exposition we shall assume that the \( y \) variables are unrestricted (and therefore always in the basis).

To obtain an algorithm, we proceed as before by writing down the optimality conditions, the only difference being that the left and right derivatives of \( g \) need now be considered. These will be denoted by \( g_{i+}, g_{i-} \), and where \( g_i \) is differentiable, by \( g_i' \). In terms of the partition \((\alpha, \beta, \gamma)\), the conditions analogous to (12) and (13) are:

\[
\begin{bmatrix}
0 & 0 & 0 & A^T_{\alpha} & G^T_{\alpha} & 0 & -I \\
D_{\beta\beta} & 0 & 0 & A^T_{\beta} & G^T_{\beta} & k_{\beta} & 0 \\
0 & 0 & 0 & A^T_{\gamma} & G^T_{\gamma} & k'_{\gamma} & 0 \\
A_{\beta} & A_{\gamma} & 0 & 0 & 0 & -I & 0 \\
G_{\beta} & G_{\gamma} & -I & 0 & 0 & 0 & 0 \\
(h_{\beta})^T & (h_{\gamma})^T & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_{\beta} \\
x_{\gamma} \\
y \\
\pi_1 \\
\pi_2 \\
\xi_{\alpha} \\
\xi_{\beta} \\
\xi_{\gamma}
\end{bmatrix}
= 
\begin{bmatrix}
d_{\alpha} \\
d_{\beta} \\
r_{\beta} \\
\rho_{\alpha} \\
b'' \\
G_{\alpha} x_{\alpha} \\
0 \\
0
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\lambda
\end{bmatrix}
\text{and}
\tag{27}
\end{align*}
\]

Except for the additional bookkeeping introduced by the index set \( \gamma \) and the derivatives of \( g \), the procedure and the proof that it works are exactly the same as outlined in §2.

To obtain the upper bound of \( m + t + 2 \) on the number of variables that may violate the restrictions (24), notice that for in- and out-pivots, the nonsingularity of \( B \) (the coefficient matrix in (27)) implies the submatrix

\[
\begin{bmatrix}
m & A_{\gamma} \\
t & G_{\gamma} \\
1 & (h_{\gamma})^T
\end{bmatrix}
\]

has full column rank. This in turn implies that \(|\gamma| \leq m + t = 1\). For exchange pivots, it follows from this argument that \( \gamma \) begins and ends with at most \( m + t + 1 \) elements. However, from (19), it is possible that \( \gamma \) contains an additional element during the exchange pivot. Hence \(|\gamma| \leq m + t + 2 \), as asserted.

4.1. Remarks

(i) Note that the above partial linearization of the residual variance formula (i.e. the substitution of \( g_i(x_i) \) for \( d_{ii} x_i^2 \)) has the effect of giving the \( \gamma \) variables zero variances

\[|\gamma| = \text{cardinality of } \gamma.\]
and covariances, while simultaneously giving them a nonzero linear term $d_i - g_i$ (see (27)). This means that the linear term in the “variance” formula will, locally, not lie in the range of the “covariance matrix”. Thus, per our discussion in §2, the formulation (26) will in general require exchange-pivots to be performed even if, because of the choice of $d$, none were required with the formulation (1).

(ii) The fact that the minimum transaction size problem is being solved only approximately is of little concern as regards practical application: In most cases where minimum transactions sizes are important, the bound $m + t + 1$ is indeed very small relative to the number of securities. The perturbation of the objective function due to the linearization enters in only for this small number of variables that are in violation of their limits, and does not differ greatly from the original objective. For example, with a 500 security optimization, a typical minimum trading size is about 1/2% (i.e. $x_i^0 - t_i^0 = u_i^0 - x_i^0 = 0.005$), which yields a maximum error, for each $i \in \gamma$, of

$$E_i = \max_{x_i} |d_i x_i^2 - g_i(x_i)| = \max \left\{ \frac{d_{ii}^2}{4} (x_i^0 - t_i^0)^2, \frac{d_{ii}^2}{4} (x_i^0 - u_i^0)^2 \right\}$$

$$= 6.25 \times 10^{-6} \times d_{ii}.$$

(iii) More general disjunctive constraints of the form (24) can be included and obeyed approximately by linearizing $g_i$ over the desired intervals of infeasibility. In particular, this technique can be used to allow for trading in round lots of, say, 1000 shares.

5. Computational Results

The full algorithm was implemented as described above, with the LU factorization and updating routines employed being those of Reid (1975). The computational results presented here are intended to illustrate

(a) the effect on the computational effort of various formulations of the same problem;
(b) the effect on the computational effort of various types of constraints; and
(c) the effectiveness of the minimum trading size constraints.

All runs are made with the multi-factor model of the Boston Company (1980). This model has the form (3) where both $D$ and $P$ are diagonal. There are 6 factors, and data were supplied for 502 securities. Run times are in CPU seconds net of setup time, and were recorded on the DEC 10 computer at the Harvard Business School.

All runs have the following base case in common:
(i) Current portfolio holdings all equal, i.e. $x_i^0 = 1/n$;
(ii) Transactions costs assumed to be flat rates of 1% for both purchases and sales;
(iii) Lower bounds of pero for all securities;
(iv) Only constraint other than any indicated below is the budget constraint;
(v) Refactorization after every 50 pivots.

5.1. Various Formulations of the Same Problem

Here, we set the number of securities at $n = 50, 75, 100$, and for each computed the efficient frontier using four mathematically equivalent formulations. The problem specifications are precisely those listed above.

Run S1 uses the canonical approach of this paper, i.e. treats the transactions costs as a piecewise linear function (cf. (1)) and sparsifies the covariance matrix (cf. (22), 23)).

Run S2 sparsifies the covariance matrix, but has two variables per security: one for purchases and one for sales ($2n$ “securities” in all), thus not exploiting the piecewise linear feature.

Run F1 uses the piecewise linear feature, but treats the covariance matrix explicitly by first multiplying out the right-hand side of (3).

Run F2 uses neither the sparsity nor the piecewise/linear feature.

The two (predictable) observations to be made from this table are:
TABLE 1
Alternative Formulations of the Same Problem

<table>
<thead>
<tr>
<th># “securities”</th>
<th>n</th>
<th>S1</th>
<th>S2</th>
<th>F1</th>
<th>F2</th>
<th># Pivots</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n</td>
<td>2n</td>
<td>n</td>
<td>2n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU seconds</td>
<td>50</td>
<td>12</td>
<td>18</td>
<td>17</td>
<td>30</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>20</td>
<td>32</td>
<td>38</td>
<td>72</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>32</td>
<td>51</td>
<td>75</td>
<td>146</td>
<td>279</td>
</tr>
<tr>
<td>Maximum #</td>
<td>50</td>
<td>609</td>
<td>1,037</td>
<td>1,452</td>
<td>2,902</td>
<td></td>
</tr>
<tr>
<td>nonzeros in B</td>
<td>75</td>
<td>863</td>
<td>1,497</td>
<td>2,777</td>
<td>5,552</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>1,161</td>
<td>2,009</td>
<td>5,102</td>
<td>10,202</td>
<td></td>
</tr>
</tbody>
</table>

1The largest basis (nonzero wise) encountered in traversing the efficient frontier: # nonzeros = proxy for incremental storage requirements.
2The number of pivots is independent of the formulation (barring numerical ill-conditioning and degeneracy) because the efficient frontier is unique (cf. 1.6).

(i) The piecewise linear feature saves between approximately 40% and 50% in CPU time (S1 vs. S2, F1 vs. F2). The savings is reduced when the problem is sparsified because the additional linear constraints (23) result in $G_{SB}$ being present as a “fixed cost.” With more than two pieces of linearity per security, say $k$, this effect is reduced, and CPU time will be approximately $1/k$ of that required if the problem is formulated with $k$ variables per security.

(ii) The storage and CPU time requirements are vastly increased by not exploiting the covariance structure (3) (S1 vs. F1, S2 vs. F2). Indeed, run F2 required prohibitively much core with $n \geq 150$.

5.2. Varying the Nature of the Constraints
Here, we tabulate the results of 10 runs, each with $n = 502$ securities and formulated canonically as per S1. Run 1 is with the above-mentioned specifications. The remain-

TABLE 2
Variations in the Nature of the Constraints

<table>
<thead>
<tr>
<th>Run</th>
<th>Description</th>
<th># pivots</th>
<th>CPU secs</th>
<th>Average sec/pivot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Base case</td>
<td>945</td>
<td>349</td>
<td>0.369</td>
</tr>
<tr>
<td>2</td>
<td>Upper Bounds</td>
<td>15%</td>
<td>1118</td>
<td>417</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>10%</td>
<td>1216</td>
<td>451</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5%</td>
<td>1323</td>
<td>502</td>
</tr>
<tr>
<td>5</td>
<td>Turn-over &lt;</td>
<td>35%</td>
<td>939</td>
<td>395</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>25%</td>
<td>735</td>
<td>292</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>15%</td>
<td>459</td>
<td>183</td>
</tr>
<tr>
<td>8</td>
<td>Minimum trading</td>
<td>1/2%</td>
<td>1154</td>
<td>417</td>
</tr>
<tr>
<td>9</td>
<td>trading size</td>
<td>3%</td>
<td>1296</td>
<td>416</td>
</tr>
<tr>
<td>10</td>
<td>Upper bounds =</td>
<td>10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Turn-over &lt;</td>
<td>25%</td>
<td>1101</td>
<td>435</td>
</tr>
<tr>
<td></td>
<td>Minimum trading</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>size = 1%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ing runs except the last are with this base case and one other specification, e.g., upper bounds all equal to 10%. The last run, #10, is with the base case and the three additional specifications as indicated.

Notice that the number of pivots increases with respect to the base case as the upper bounds are tightened, decreases markedly with respect to the base case as the turnover constraint is tightened, and increases to a lesser extent with the introduction of minimum trading sizes. The combination of all three seems to have a neutralizing effect. These results make sense intuitively since we would expect a constraint set with more "corners" to result in a larger number of pivots.

The average time per pivot is fairly consistent within each category. Differences between categories are attributable to two effects: (i) the turnover constraint requires more column interchanges per iteration on the average than the others because of changes in the basis column corresponding to $p$; (ii) minimum trading sizes result in many more variables being nonbasic, i.e. being held at a breakpoint, which in turn results in the bases containing many more unit columns, thus making them sparser. For verification of this latter claim, see Figure 3 which gives the number of nonzeros in $L$ and $U$ immediately after refactorization for runs 1 and 10. In run 10 the size of $L$ is roughly constant, corresponding to the number of nonbasic $x$ variables being roughly constant, whereas in run 1, the size of $L$ grows as the algorithm moves down the frontier, corresponding to a growth in the number of basic $x$ variables.

![Nonzeros in L and U Immediately Following Refactorization.](image)

Figure 3. Nonzeros in $L$ and $U$ Immediately Following Refactorization.
Changing the refactoring frequency from 50 to 100 increased the overall time for run 1 from 349 secs to 369 secs, and decreased the time for run 10 from 435 secs to 394 secs. This effect is accounted for by the very different growth patterns in $L$ and $U$ for the two runs, graphed in Figure 4 for pivots 501 to 600 (a typical sequence). Note that in run 10 the number of nonzeros in $U$ stays roughly constant, and even decreases, the growth in the number of nonzeros in $L$ moreover being very slow and occurring at a roughly constant rate. In run 1, $L$ grows very quickly, and $U$ begins to grow only when $L$ reaches a certain critical size, about 750 (after pivot #525). Since both runs (coincidentally) have 49 in- and 51 out-pivots during this sequence and both have essentially the same density in $L$ and $U$ after refactoring after the 600th pivot as after refactoring at the 500th pivot, the size of $L$ being above or below some critical value does appear to be a key determinant of how much the factors will grow. This was confirmed by inspection of the other runs.

5.3. How Effective Are the Minimum Trading Size Constraints?

Figure 5 shows the effect of the minimum trading size constraints. Run 1, without these constraints, had up to 78 trades smaller than 3%, while run 9 which differs from run 1 only in the minimum trading size limit of 3%, had no more than 7, as expected. The difference in the mean and variance characteristics of the two frontiers was negligible for all practical purposes, the largest difference in true variances for any given value of the mean occurring in the third significant digit.
This count does not include variables that are at their lower bounds of zero which, since the current holdings are roughly 0.2%, technically constitute small trades. It is thus mainly a count of purchases less than 3%.

Bibliography


———, "The Optimization of a Quadratic Function Subject to Linear Constraints," *Naval Res. Logist. Quart.*, 3 (1956), 111–133.


