Algorithmic Operations Research

Mean–Absolute Deviation Portfolio Models with Discrete Choice Constraints

Roy H. Kwon et Stephen J. Stoyan

Volume 6, numéro 2, fall 2011

URI : https://id.erudit.org/iderudit/aor6_2art05

Aller au sommaire du numéro

Éditeur(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (numérique)

Découvrir la revue

Citer cet article

Kwon, R. H. & Stoyan, S. J. (2011). Mean–Absolute Deviation Portfolio Models with Discrete Choice Constraints. *Algorithmic Operations Research*, *6*(2), 118–134.

Résumé de l'article

In this paper, we consider the problem of incorporating a wide set of real-world trading constraints to the mean-variance portfolio framework. Instead of using the mean-variance model directly, we use the equivalent Mean-Absolute Deviation (MAD) linear programming formulation. The addition of the trading constraints transforms the MAD model to a mixed-integer linear programming problem. We solve both the mean-variance and MAD models with the various trading constraints using a commercial solver and find that MAD model is substantially more tractable. In addition, a heuristic is developed for the extended MAD model to provide solutions for larger problem instances.

Algorithmic Operations Research Vol.6 (2011) 118–134

Mean–Absolute Deviation Portfolio Models with Discrete Choice Constraints

Roy H. Kwon

Department of Mechanical & Industrial Engineering, University of Toronto, 5 King's College Road, Toronto, Ontario, M5S 3G8, CANADA

Stephen J. Stoyan

Starbucks Coffee Company, 2401 Utah Ave. S., Seattle WA 98134, USA

Abstract

In this paper, we consider the problem of incorporating a wide set of real-world trading constraints to the mean-variance portfolio framework. Instead of using the mean-variance model directly, we use the equivalent Mean-Absolute Deviation (MAD) linear programming formulation. The addition of the trading constraints transforms the MAD model to a mixed-integer linear programming problem. We solve both the mean-variance and MAD models with the various trading constraints using a commercial solver and find that MAD model is substantially more tractable. In addition, a heuristic is developed for the extended MAD model to provide solutions for larger problem instances.

Key words: Efficiency frontier, heuristic, decomposition, cardinality constraint. Abbreviations: Quadratic Mixed Integer Programming (QMIP); Mean-Variance Optimization (MVO); Mean-Absolute Deviation (MAD).

1. Introduction

In 1952 Harry M. Markowitz designed a portfolio selection model that revolutionized finance and lead to numerous future developments in financial theory. The model in its purest conception omits a number of real-world portfolio characteristics/restrictions such as portfolio size, trading constraints, and buy-in thresholds. In its basic form the Mean-Variance Optimization (MVO) model is a quadratic program (QP), which becomes very challenging to solve if real-world characteristics are included since the addition of these features will involve adding discrete variables. Researchers have addressed practical and computational issues involved in the addition of these types of constraints. Chang *et al.* (2000) consider a cardinality constrained MVO model and characterize efficient frontiers with this constraint. Genetic algorithms are developed to compute solutions. Crama and Schyns (2003) add buying, selling, trading, portfolio size, and floor and ceiling constraints to the MVO model, which results in a quadratic mixed-integer program (QMIP). The QMIP is solved using simulated annealing (SA), where an approximate to the efficient frontier is produced when all constraints are included in the model versus only having one or two. Jobst *et al.* (2001) investigate MVO models where portfolio size, buy-in thresholds, and roundlot constraints are considered separately. The resulting NP-hard QMIP is solved using two different heuristics, of which they demonstrate that even the addition of one discrete choice portfolio constraint requires a heuristic to produce solutions. Chang *et al.* (2000) propose heuristics for cardinality constrained MVO models. In addition, [1,10,11] investigate Markowitz QP models while considering transaction costs. Most approaches for solving MVO with real-world trading constraints involve the construction of heuristics to obtain feasible solutions. The principle complication is the addition of discrete choice constraints to a non-linear (quadratic) optimization problem.

Konno and Yamazaki (1991) show that under multivariate normal security return distributions the MVO model has an equivalent linear form in a linear programming model they define as the Mean-Absolute Deviation (MAD). The MAD model incorporates a (mean) absolute deviation measure (L1 risk measure) as risk as opposed to portfolio variance as in the MVO framework. In practice, returns may not be exactly normal and so the MAD model will give an approximation

Email: Roy H. Kwon [rkwon@mie.utoronto.ca], Stephen J. Stoyan [stoyan@usc.edu].

to the MVO model. MAD may be further justified in that the MVO framework assumes that returns are normal as well. More importantly, if the L1 risk measure is sufficient for a measure of risk in a risk and return framework then the MAD model can be used without connection to MVO. In this paper, we investigate the addition of a comprehensive set of practical portfolio constraints to the MAD model and use a commercial solver CPLEX to compute optimal solutions as well as develop a heuristic to compute feasible solutions for larger instances. We develop portfolio selection models based on the constraints outlined in [4] and [6]. Hence, we add constraints on portfolio size, buying, selling, trading, and floor and ceiling bounds to the MAD model and compare the performance with MVO designs in the publications mentioned above. Mansini and Speranza (2005) consider a mean semi-deviation model that is equal to one-half of the MAD model, where transaction costs and roundlots are included in the design. The model is partitioned into subproblems and a heuristic is used to obtain solutions. The model is an extension of [8], where they investigate a similar mixed-integer program (MIP) with fixed costs and transaction lots. The approach in Mansini and Speranza is significant in that a linear model of risk and return was employed as the basis to which some discrete choice constraints were added, thereby avoiding the complexities associated with adding discrete choice constructs in a non-linear (quadratic) optimization framework. We follow in a similar manner, but the model we propose involves a larger array of practical constraints and we find that computational results are comparable to what is shown in [12] despite the presence of a larger set of constraints/variables in the model we consider. In addition, we use the MAD model as the basis to add discrete choice constraints as the connection to the MVO model is important for practitioners in the investment community due to its prevalence in practice. We also develop a heuristic inspired by those developed in Jobst *et al.* (2001) for (non-linear) MVO models with cardinality constraints to compute feasible solutions for instances outside the scope and scale of what has been reported in the literature. The heuristic generates solutions in reasonable time that are similar to solutions on the efficient frontier of the associated MVO model without the discrete choice constraints, which indicates the effectiveness of the heuristic. In summary, we find that a linear model of risk and return (MAD) is a much more tractable framework in which to express a wide range of discrete choice constraints compared to the non-linear MVO framework.

The paper is organized as follows: in Section 2. comparisons between the MAD and the MVO model are made both with and without the various set of discrete choice constraints including instances that contain all of the constraint types discussed above. In Section 3., we present a heuristic that generates feasible solutions for the MAD model with cardinality constraints and highlight the results of the heuristic on more challenging instances of the MAD model. In the last section, concluding remarks and directions for future research are given.

2. Model and Comparisons

In this section we consider MAD models with various subsets of discrete choice constraints added, including model instances with all of the constraint types mentioned in the introduction. The efficient frontiers are generated and compared. All instances in this section are solved to optimality using CPLEX MIP solver 9.1. Where appropriate, we compare running times reported for similar MVO models from the literature.

2.1. *MVO and MAD models*

We begin by defining the basic MVO problem in [13]. The MVO portfolio selection problem is the following:

$$\min \quad \sum_{i}^{n} \sum_{j}^{n} x_i \, Q_{ij} \, x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{i}^{n} \mu_i x_i \geq R \tag{2}$$

$$\sum_{i}^{n} x_i = 1 \tag{3}$$

$$x_i \geq 0 \qquad \forall \, i = 1, ..., n \tag{4}$$

where x_i is the decision variable for the amount (proportion of wealth) invested in security $i = 1, ..., n$, μ_i is the mean return of security i, n is the total number of securities, R is the expected portfolio return, and $Q_{ij} = \text{cov}(r_i, r_j) = (1/T) \sum_t^T (r_{it} - \overline{r}_i)(r_{jt} - \overline{r}_j)$ is the covariance matrix for the return rate r_i of security i over a total of T time-stages. If the rates of return (r_i) have a multivariate normal distribution, then Konno and Yamazaki show that (1)–(4) is equivalent to the following MAD model:

$$\min \quad \sum_{t=1}^{T} y_t + z_t \tag{5}$$

$$\text{s.t.} \quad y_t - z_t = \sum_{i=1}^{n} (r_{it} - \mu_i) x_i \quad \forall\, t = 1, ..., T \tag{6}$$

$$\sum_{i=1}^{n} \mu_i x_i \geq R \tag{7}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{8}$$

$$y_t \geq 0, z_t \geq 0 \qquad \forall\, t = 1, ..., T \tag{9}$$

where constraint (6) and y_t and z_t allow for the linear transformation. As mentioned, under practical situations the distribution of r_i may not be multivariate normally distributed. This can lead to sources of error when comparing MAD and MVO portfolio values; however, there are also imperfections of the MVO model under practical situations. We will discuss these issues below and show that the MAD model has greater advantages with respect to practical modeling issues and computational tractability.

2.2. *MVO and MAD Cardinality models*

As the general MVO model shown in (1)–(4) can often produce very dense portfolios i.e. investments in only a few securities or produce investments in many securities but with impractically small weights, various attempts to maintain diversification or reduce the portfolio size has been considered. A cardinality constraint or names-to-hold constraint that limits or enforces the number of security investments helps portfolio managers with handling the complexities of portfolio management. In [6], a cardinality constraint (as embodied in constraints (16) and (17) below) is added to (1)–(4) to get an MVO cardinality model. Crama and Schyns (2003) take the cardinality model further by adding buying, selling, and trading constraints. Below we have an MVO model that includes cardinality, buying, selling, and trading constraints. The problem is as follows:

$$\min \quad \sum_{i}^{n} \sum_{j}^{n} x_i Q_{ij} x_j \tag{10}$$

$$\text{s.t.} \quad \sum_{i}^{n} \mu_i x_i = R \tag{11}$$

$$\sum_{i}^{n} x_i = 1 \tag{12}$$

$$\max(x_i - x_i^0, 0) \leq \overline{B}_i \qquad \forall\, i = 1, ..., n \tag{13}$$

$$\max(x_i^0 - x_i, 0) \leq \overline{S}_i \qquad \forall\, i = 1, ..., n \tag{14}$$

$$x_i = x_i^0 \text{ or } x_i \geq (x_i^0 + \underline{B}_i) g_i \text{ or} \tag{15}$$
$$x_i \leq (x_i^0 - \underline{S}_i) g_i \quad \forall\, i = 1, ..., n$$

$$\sum_{i}^{n} g_i = G \tag{16}$$

$$l_i g_i \leq x_i \leq u_i g_i \qquad \forall\, i = 1, ..., n \tag{17}$$

$$g_i \in \mathbb{B} \qquad \forall\, i = 1, ..., n \tag{18}$$

where g_i is a binary variable that is equal to one if $x_i > 0$, and zero otherwise. Also, G is the number of different securities to be held in the portfolio, and l_i/u_i are lower/upper investment constraints (also referred to as floor/ceiling constraints), respectively. Finally, x_i^0 is the weight of security i in the initial portfolio, $\overline{B}_i$ and $\overline{S}_i$ denote the maximum purchase/sale of security i, and $\underline{B}_i$ and $\underline{S}_i$ denote the minimum purchase/sale of security i, respectively. In Section 2.3. we present the results of the MVO cardinality model, which refers to solving equations (10)–(12) and (16)–(18). The cardinality model is later taken further in Section 2.4. by adding buying, selling, and trading constraints, similar to what is done in [4]. In the MVO model of (10)–(18), constraint (13) bounds the number of securities that may be purchased, constraint (14) bounds the number of securities that may be sold, constraint (15) bounds the number of transactions, constraint (16) limits portfolio size, and constraint (17) bounds the buy-in minimum/maximum amount invested in any security. As we will present in Section 2.4., each of the additional constraints are important elements for portfolio managers, however, their inclusion pose solvability issues.

By adding the same constraint to (5)–(9) the MAD Linear Program (LP) becomes:

$$\min \quad \sum_{t=1}^{T} y_t + z_t \tag{19}$$

$$\text{s.t.} \quad y_t - z_t = \sum_{i=1}^{n} (r_{it} - \mu_i) x_i \qquad \forall\, t = 1, ..., T \tag{20}$$

$$\sum_{i=1}^{n} \mu_i x_i \geq R \tag{21}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{22}$$

$$\max(x_i - x_i^0, 0) \leq \overline{B}_i \quad \forall\, i = 1, ..., n \tag{23}$$

$$\max(x_i^0 - x_i, 0) \leq \overline{S}_i \quad \forall\, i = 1, ..., n \tag{24}$$

$$x_i = x_i^0 \text{ or } x_i \geq (x_i^0 + \underline{B}_i)g_i \text{ or } \tag{25}$$
$$x_i \leq (x_i^0 - \underline{S}_i)g_i \quad \forall\, i = 1, ..., n$$

$$\sum_{i=1}^{n} g_i = G \tag{26}$$

$$l_i g_i \leq x_i \leq u_i g_i \quad \forall\, i = 1, ..., n \tag{27}$$

$$y_t \geq 0, z_t \geq 0 \quad \forall\, t = 1, ..., T \tag{28}$$

$$g_i \in \mathbb{B} \quad \forall\, i = 1, ..., n, \tag{29}$$

where (23), (24), and (25) refer to buying, selling, and trading constraints, respectively. Also, (19)–(22) and (26)–(29) refers to the MAD cardinality model.

2.3. *MVO and MAD Cardinality Results*

Following the same data procedures in [6], we randomly selected 30 stocks from the S&P TSX 60 Composite Index for 60 monthly returns (June 2002 to May 2007). In Figure 1 we present the MAD and MVO efficiency frontier over this data set for problem (5)–(9) and (1)–(4), respectively. In Figure 1 the MAD efficient frontier is captured by dots, and the MVO efficient frontier is represented by stars. As one can see, the efficient frontiers are almost identical. The MAD efficient frontier has a slightly steeper slope and greater standard deviation as the expected return increases in comparison to the MVO results. The y-axes expected return values are kept consistent for all illustrations in this document, which were set to range from $3\% - 14\%$ and increased by 0.2% intervals. Another interesting property in Figure 1 is that both of the portfolio values become closer as the expected return decreases. The solution time (CPU time) was faster for the MAD model, however, more significate CPU time differences between the two models come into effect when cardinality constraints are added; shown in equations (16) and (26). In addition, commercial quadratic optimization solvers such as CPLEX require that the hessian is positive definite, which may pose a problem for the MVO model when using real data.

For the MVO cardinality model, Jobst *et al.* (2001) report that a heuristic is necessary to obtain a solution, which is what we find when trying to implement (10)–(18) using the TSX data set mentioned above. On the other hand, the MAD cardinality model in (19)–(29) was solved to optimality using CPLEX 9.1, where following [6] we set $G = 4$. The efficiency frontier for solving equations (19)–(22) and (26)–(29) is shown in Figure 2. The MAD cardinality model has similarities to the frontier illustrated in Figure 1, but the steep downward slope from 0–0.1 units resemble the MVO model of Figure 1. The subtle discontinuities in the efficient frontier of Figure 2 are due to the addition of the cardinality constraint, which were first observed in the MVO cardinality model of [2] and later shown in [4,6]. When using the integer restart heuristic in [6] on the same size problem, they report that it takes a CPU time of 57.55s (seconds) and when using their simple reoptimization heuristic it takes 10.00s. Without addressing the quality of the portfolios generated by the heuristics, the MAD cardinality model is solved to optimality using CPLEX directly (no heuristic necessary) with an average CPU time of 0.08s. Table 2 illustrates the dramatic differences in solvability between the MVO with cardinality constraints and MAD with constraints. The MAD models can be solved to optimality up to size instances for which the MVO model with cardinality constraints could not. We use a generous linear extrapolation of solution time for the MVO model as indicated by the $\sim$ for instances that could not be solved, even so, the MAD model is several orders of magnitude faster. Next, we pushed the MAD cardinality model to its computational limits by dramatically increasing the number of securities and time-stages (monthly returns). In order to account for over 120 time-stages and keep the covariance matrix Q dense, we found that a maximum of 853 TSX securities could be used over that time period. Figure 3 presents the efficiency frontier when 100 time-stages are used with the portfolio size $G = 35$. In Figure 4, 120 time-stages are used with $G = 75$; both results involved 853 securities. Note that an algorithm was not necessary for the results shown in Figures 3 or 4. It took CPLEX an average of 0.67s to compute the values for the 100 time-stage results and 1.05s to compute the values for the 120 time-stage results. Figure 5 is a plot of Figures 2–4 imposed on each other. The efficient frontiers depicted in Figure 5 follow a general path, where the differences lie in the total number of securities considered, number of securities used in the portfolio, and time stages. In comparison to the graphs of Figures 1 and 2, the slopes are similar, however the standard deviation is lower for higher expected return values, which is primarily due to the larger data set that is used for these results.

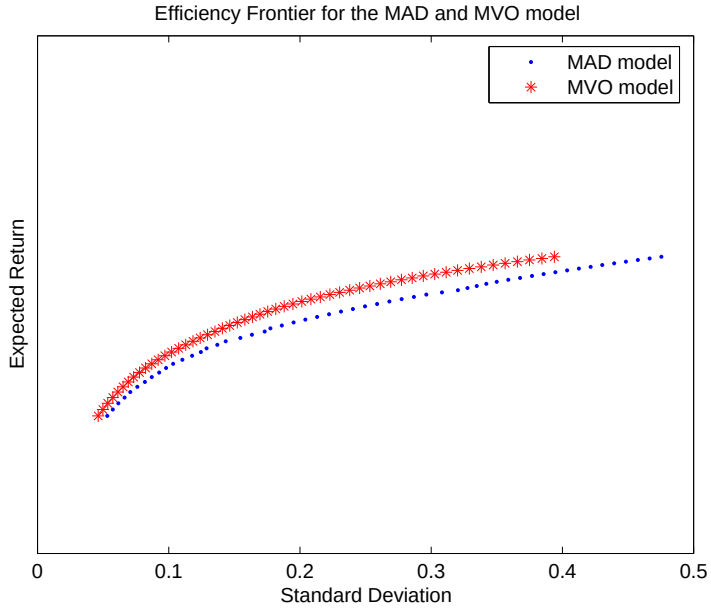

Fig. 1. Comparison of the efficient frontier for the MAD and MVO model.

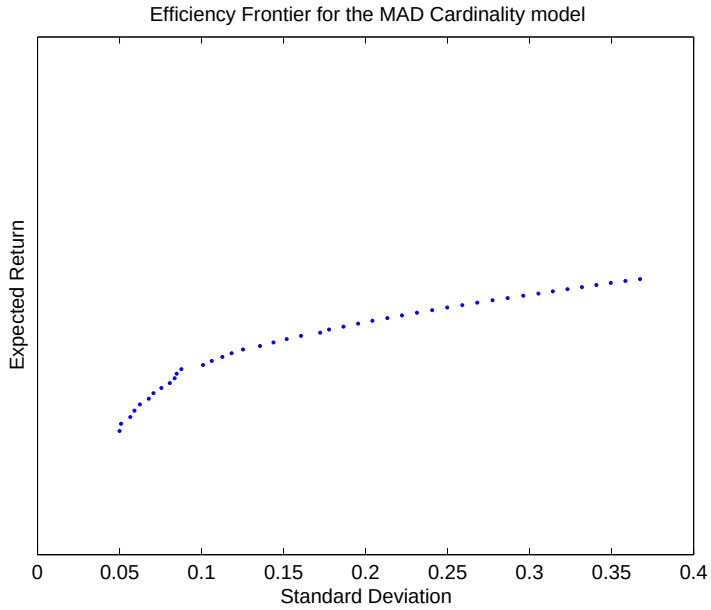

Fig. 2. Efficiency frontier for the MAD cardinality model with $G = 4$.

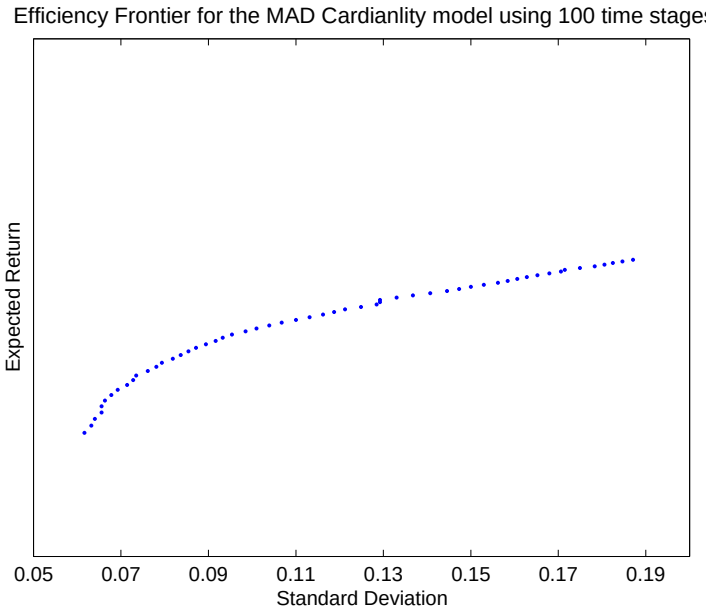

Fig. 3. Efficiency frontier for the MAD Cardinality model with $G = 35$ using 100 time-stages and 853 securities.

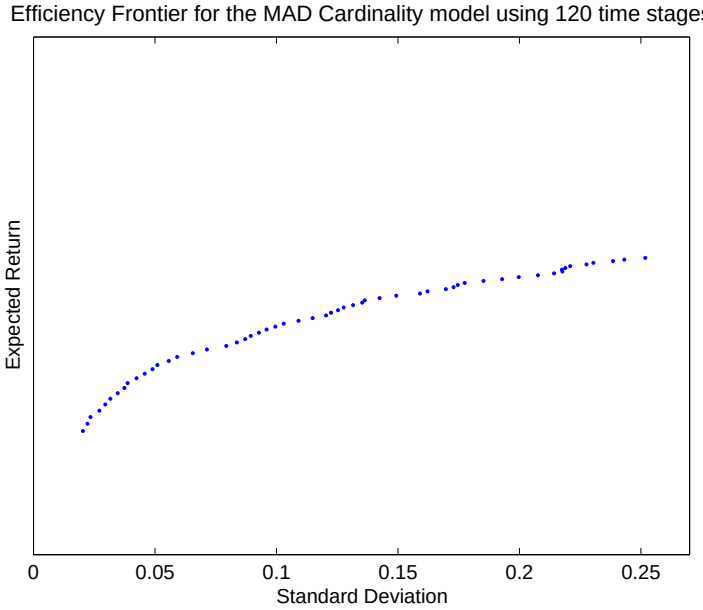

Fig. 4. Efficiency frontier for the MAD Cardinality model with $G = 75$ using 120 time-stages and 853 securities.

Table 1

Problem Size	MVO CPU time (s)	MAD CPU time (s)	% Speed up
30	10.00	0.08	99.22
100	44.93	0.28	99.37
300	$\sim$ 144.73	0.36	99.75
500	$\sim$ 244.53	0.48	99.80
853	$\sim$ 370.78	0.67	99.82

CPU speed up difference between the MAD and MVO model.

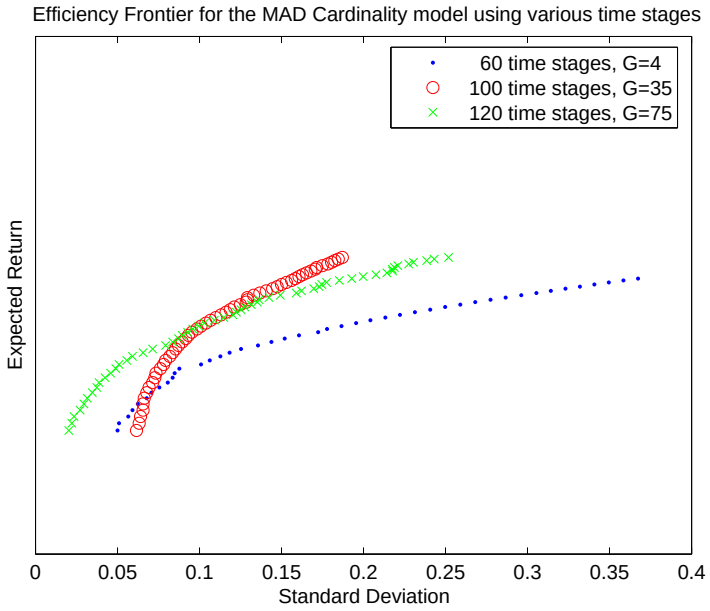

Fig. 5. Efficiency frontier for the MAD Cardinality model comparing Figures 2–4.

As in Figure 2, Figures 3 and 4 resemble more of the quadratic functions in the MVO of Figure 1. Hence, the MAD model under cardinality constraints produce efficiency frontiers closer to Markowitz MVO models than when the constraints are not included. In order to decrease the size of the portfolio (i.e. G value) further than what is presented in Figures 3 and 4 an algorithm is necessary, which we discuss in the next section.

2.4. *Additional Modeling Results*

We consider several MAD models each with the cardinality constraint and exactly one of the constraints from (23)–(25). As mentioned in Section 2.2., each of the additional constraints are important elements for portfolio managers and just as in [6], a heuristic is necessary to solve the QMIP presented in (10)–(18). Crama and Schyns (2003) provide a few techniques that may be used to solve the large model, but most of the results they present involve solving a constraint subset of (10)–(18). To solve the large QMIP a Simulated Annealing (SA) algorithm is implemented, however, they only provide a brief discussion on the results of the MVO efficient frontier when all constraints are considered. Adding the same constraints to the MAD model gives the constraint subset in (23)–(27), to which we also compare the model when all constraints are considered (21)–(29). Figures 6–9 show the efficient frontiers of these MAD models. We then consider the MAD model with all of the constraint types added including cardinality and the resulting efficient frontier is shown in Figure 9. To keep the graphs similar to what is shown earlier in Figures 1 and 2 and [6], we use 60 securities over 60 time periods (June 2002 to May 2007) from the TSX 60 Composite Index and set $G = 15$. Crama

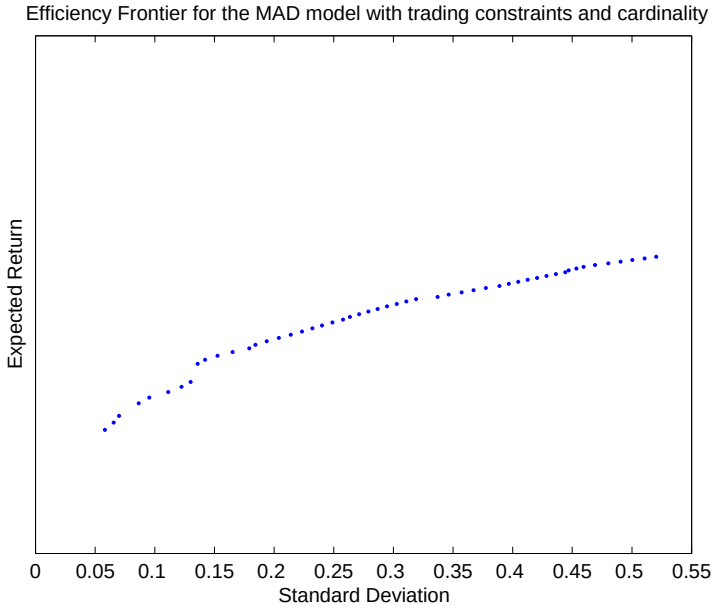

Fig. 6. Efficiency frontier for the MAD model with trading constraints.

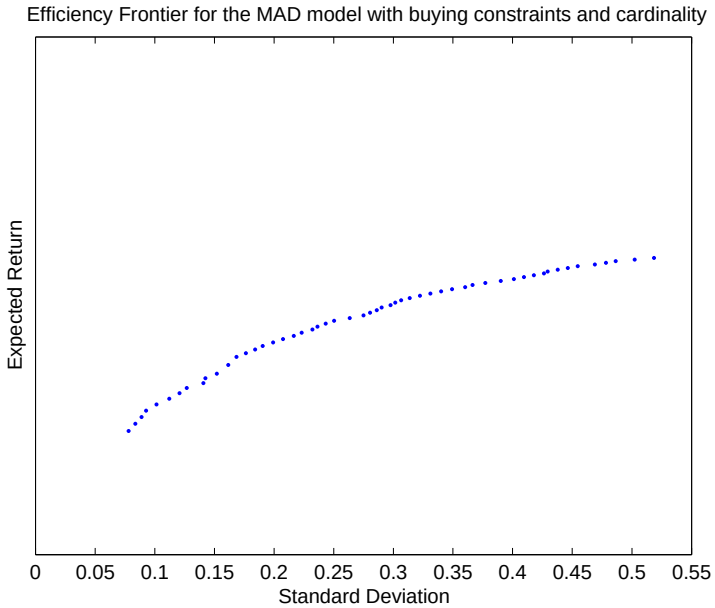

Fig. 7. Efficiency frontier for the MAD model with buying constraints.

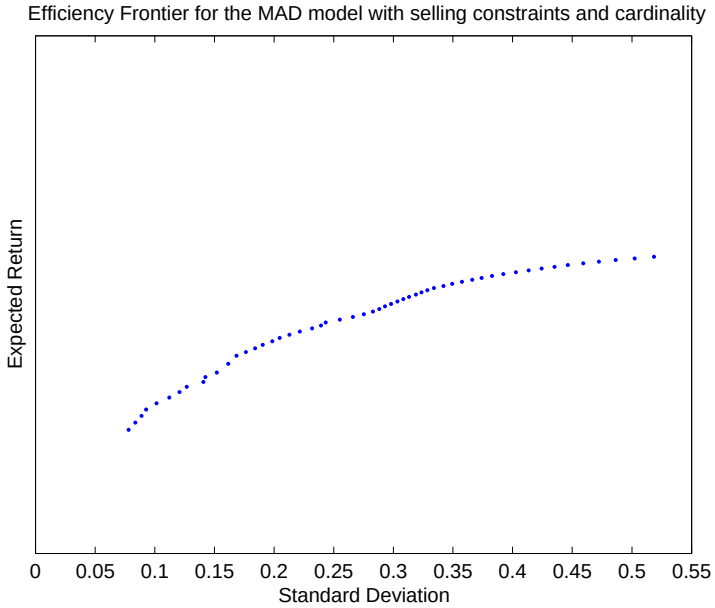

Fig. 8. Efficiency frontier for the MAD model with selling constraints.

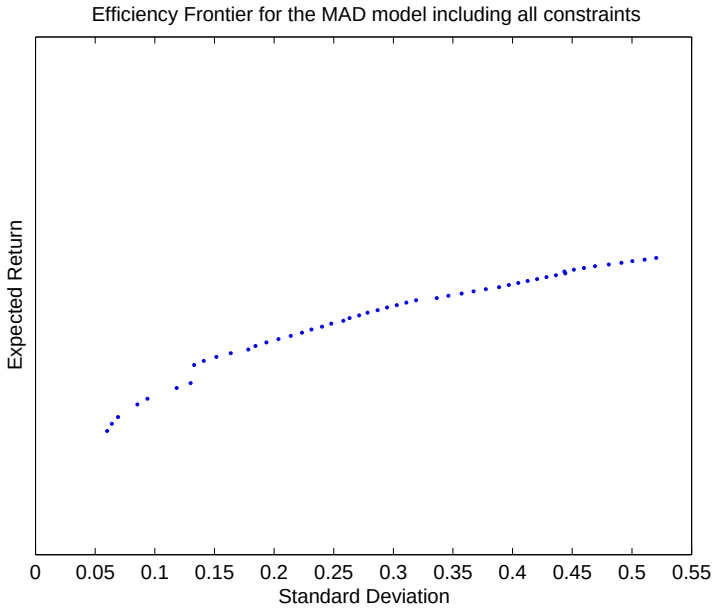

Fig. 9. Efficiency frontier for the MAD model with all constraints, (19)–(29).

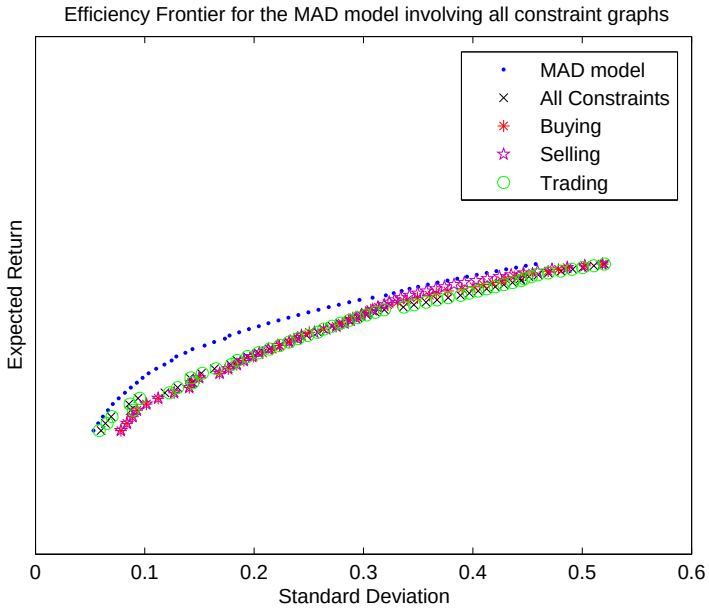

Fig. 10. Efficiency frontier for the MAD model with Figures 6–9 imposed on each other and the MAD model in Figure 1.

and Schyns (2003) have results with 3 – 151 securities, and for lack of space we chose a security data set that lies between this range and can be compared to [6]. Figure 6 is an illustration of the efficient frontier when the trading constraint (25) is included. As one may observe, the standard deviation has increased slightly in comparison to Figure 2, where only the cardinality constraint is used. In Figure 7, the buying constraint (23) was imposed on the model. Similar to Figure 6, constraining the amount of buying produces portfolio risk (standard deviation) issues as the expected return increases. Adding the selling constraint (24) produces the efficiency frontier shown in Figure 8. This portfolio has a comparable efficient frontier to Figure 7, which one would expect since it is performs a similar task. Finally, Figure 9 provides the efficient frontier when the MAD model was solved including all of the constraints presented in (19)–(29). Figure 9 is the product of running the constraints in Figures 6–8 collectively, and Figure 10 is an illustration of the plots imposed on each other. With respect to the MVO presented in Crama and Schyns (2003), the model that considered all constraints posed the greatest challenge to solvability and produced the worst efficiency frontier in their experimental results section. The full MAD model (19)–(29) was solved without any heuristic, where as a heuristic is necessary for the MVO model of [4]. The average solution time was 0.14s, Crama and Schyns (2003) do not provide solution times for their complete model. However, they do provide solutions times for their constraint subproblem runs using their SA algorithm, which is dependent on the number of moves defined and the number of securities used; nevertheless, the CPU time ranged from 0.15s – 7h (hours) and 58m (minutes).

From the results and comparisons above, we find that the various MAD models considered are much more tractable in terms of solving for the optimal solution than its quadratic counterpart (MVO). The MAD results are solved in seconds without using any heuristic, whereas the same can not be said about the QMIPs in [4,6]. The cardinality and additional portfolio constraints used in both papers do not pose the same computational difficulties when applied to the MAD model, even when variable counts are increased. In addition, the MAD model does not require a positive definite hessian matrix, which may be necessary depending on the solver being used. The portfolio size constraint is one of the most important characteristics, as [4,6] illustrate; however, it is also one of the most difficult to satisfy. In the next section we push the MAD variable count past the computational limits shown in Figures 3 and 4 with the introduction of a model specific algorithm.

3. MAD Implementation

In the previous section, it was shown that the MAD models with discrete choice constraints to be much more computationally tractable than the equivalent MVO versions with respect to computing optimal solutions. As shown in Figures 3 and 4, however, the MAD model is challenging to solve in a number of instances. In particular, instances with lower G values and with more variables were more challenging to solve. In order to lower the value of G and/or increase the number of decision variables a heuristic is necessary. In [6], two types of heuristics are designed and investigated, namely an integer restart and re-optimization heuristic. The integer restart heuristic is based on using a previous integer solution and relaxing the expected return constraint to obtain portfolio results. The other heuristic presented in [6], namely the re-optimization heuristic, is described as follows. First, the MVO model without the cardinality constraint and buy-in lower bound is solved. Then, only the G securities with the largest weights are used to solve the full MVO cardinality model with the buy-in lower bound. For the re-optimization heuristic, optimality issues are even greater than in the integer restart since a large amount of information is disregarded before solving the model the second time. Nevertheless, of the two heuristics proposed by Jobst *et al.* (2001) the re-optimization heuristic seems to be the most promising with respect to CPU time, and produces a complete efficiency frontier.

For the MAD cardinality problem, we design a heuristic that first solves a sub-problem of (19)–(22) and (26)–(29), and then uses an algorithm to enforce the cardinality constraint. The basics of the algorithm are as follows: first, (i) solve the MAD model without cardinality or buy-in lower bound constraints. Next, (ii) add the cardinality constraint and a relaxation to G for any $x_i > 0$ in (i); and minimize the relaxation in an iterative procedure. Then, (iii) if the cardinality number is still not satisfied but reduced in comparison to step (i), add the lower bound constraint and solve; otherwise impose the cardinality constraint using $x_i > 0$ in (ii) and an iteration to reduce the basis to the desired size. Finally, (iv) solve the MAD model with the buy-in lower bound thereby satisfying all constraints. In each of the steps (i)–(iv) mentioned above, a subproblem of (19)–(22) and (26)–(29) is solved until the last step where all constraints are considered. The first subproblem for the algorithm to solve is shown below

MAD–LP:

$$\min \sum_{t=1}^{T} y_t + z_t \tag{30}$$

$$\text{s.t. } y_t - z_t = \sum_{i=1}^{n} (r_{it} - \mu_i) x_i \quad \forall\, t = 1, ..., T \tag{31}$$

$$\sum_{i=1}^{n} \mu_i x_i \geq R \tag{32}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{33}$$

$$0 \leq x_i \leq u_i \qquad \forall\, i = 1, ..., n \tag{34}$$

$$y_t \geq 0, z_t \geq 0 \qquad \forall\, t = 1, ..., T \tag{35}$$

where there cardinality constraint and buy-in lower bound is removed. The second subproblem takes the optimal basis from MAD–LP and solves the same problem with a relaxation or penalty variable ξ added to the cardinality constraint and penalty parameter ς in the objective. If the optimal basis, any $x_i^\star > 0$, has been reduced in the solution to (30)–(35), we let x_i be the subset $\forall\ x_i^\star > 0$ in (30)–(35) for $i = 1, ..., n$ and solve the following problem

MAD–CP:

$$\min \sum_{t=1}^{T} y_t + z_t + \varsigma\, \xi \tag{36}$$

$$\text{s.t. } y_t - z_t = \sum_{i=1}^{n} (r_{it} - \mu_i) x_i \quad \forall\, t = 1, ..., T \tag{37}$$

$$\sum_{i=1}^{n} \mu_i x_i \geq R \tag{38}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{39}$$

$$\sum_{i=1}^{n} g_i = G + \xi \tag{40}$$

$$0 \leq x_i \leq u_i \qquad \forall\, i = 1, ..., n \tag{41}$$

$$y_t \geq 0, z_t \geq 0 \qquad \forall\, t = 1, ..., T \tag{42}$$

$$g_i \in \mathbb{B} \qquad \forall\, i = 1, ..., n. \tag{43}$$

If $\xi = 0$ then the cardinality constraint is met and we are satisfied, otherwise we increase the value of ς in an iterative procedure such that the algorithm tries to force $\xi = 0$. If the cardinality constraint is not met, but the number is reduced in comparison to MAD–LP, then using the basis in (36)–(43) we solve

MAD–LC:

$$\min\ (30) \tag{44}$$
$$\text{s.t. } (31)-(33),\ (35) \tag{45}$$
$$l_i g_i \leq x_i \leq u_i g_i \qquad \forall\ i=1,...,n \tag{46}$$

in anticipation of further reducing the cardinality number. If the cardinality number is reduced we repeat the last two steps - solving MAD–CP then MAD–LC; otherwise we enforce the cardinality constraint by solving MAD–LP using the current basis and removing the lowest weight x_i. After the cardinality number is met, then the solution to

MAD–LB:

$$\min\ (30) \tag{47}$$
$$\text{s.t. } (31)-(33),\ (35) \tag{48}$$
$$l_i \leq x_i \leq u_i \qquad \forall\ i=1,...,n \tag{49}$$

ensures all constraints in the MAD Cardinality model ((19)–(22) and (26)–(29)) are met. The pseudocode for the proposed heuristic is as follows:

Algorithm 1: The MAD Cardinality Algorithm.

1. Initialize:
   Set model parameters R, G, μ_i, l_i, $u_i\ \forall\ i=1,...,n$ and computational parameters $\varsigma=0, h=0,\ \tau>0,$ $\Delta>0,\ H>0$.
2. LP solution:
   Solve MAD–LP (30)–(35)
   (i) if $\sum g_i \leq G \quad \Rightarrow$ solve MAD–LB for $x_i^\star>0$ $\Rightarrow$ `Terminate.`
   (ii) otherwise, set $\varsigma=\tau$, $\widehat{G}=\sum g_i$ and $\widehat{x} \subseteq \{x^\star : x_i^\star>0\ \forall\ i=1,...,n\}$, `go to 3.`
3. Cardinality and Penalty:"
   Solve MAD–CP (36)–(43) using $\widehat{x}$ from 2 (ii)
   (i) if $\xi=0 \quad \Rightarrow$ solve MAD–LB for $x_i^\star>0$ $\Rightarrow$ `Terminate.`
   (ii) else if $\sum g_i \leq \widehat{G}$ and $h<H$, set $\overline{G}=\sum g_i$ and $\overline{x} \subseteq \{x : x_i>0\ \forall\ i=1,...,n\}$, `go to 4.`
   (iii) else if $h<H$, set $\tau=\tau+\Delta$ and $h=h+1$, `go to 3.`
   (iv) otherwise, set $\overline{G}=\sum g_i$ and $\overline{x} \subseteq \{x : x_i>0\ \forall\ i=1,...,n\}$, `go to 4.`
4. Re-solve:"
   (i) if $\overline{G}<\widehat{G}$, solve MAD–LC (44)–(46) using $\overline{x}$ from 3 (ii) or (iv)
      (a) if $\sum g_i \leq G \quad \Rightarrow$ `Terminate.`
      (b) else if $\sum g_i < \overline{G}$, set $\widehat{G}=\sum g_i$ and $\widehat{x} \subseteq \{x^\star : x_i^\star>0\ \forall\ i=1,...,n\}$, `go to 3.`
      (c) otherwise, `go to 5.`
   (ii) otherwise, `go to 5.`
5. Impose Cardinality:"
   Solve MAD–LP (30)–(35) using $\overline{x}$ from 3 (ii) or (iv)
   (i) set $\widetilde{x}=\{x-\min(x_i,\ \forall\ i=1,...,n)\}$ and $\widetilde{G}=\sum g_i$
      (a) if $\widetilde{G} \leq G$, `go to 5` (ii).
      (b) otherwise, `go to 5` using $\widetilde{x}$ from 5 (i)
   (ii) solve MAD–LB (47)–(49) using $\widetilde{x} \quad \Rightarrow$ `Terminate.`

In the algorithm above, τ is the initial value of the penalty parameter when the iterative penalty adjustment procedure is initiated and Δ is the amount τ is increased every time a step is repeated. H defines the number of times the algorithm will undergo the penalty adjustment procedure, or try to push the cardinality penalty variable ξ to be equal to zero, where h is the counter. In step 2, the algorithm solves MAD–LP and keeps the optimal basis to be the starting point for step 3, where the cardinality constraint, penalty variables, and penalty parameters are added. Given the cardinality number is not satisfied, the iterative penalty procedure tries to force the cardinality constraint to be met by making the penalty variable (which relaxes this constraint) very expensive. This is achieved by increasing the value ς in the objective function. After repeating this H times, if $\xi \neq 0$ and the cardinality number is reduced in comparison to step 2, then step 4 requires that the current optimal basis is re–solved using MAD–LC, where the lower bound is imposed. If MAD–LC further reduces the cardinality number then the steps are repeated. Otherwise, in the worst case scenario, step 5 imposes the cardinality constraint by solving MAD–LP using the current basis and subtracting the minimum x_i investment weight from the portfolio until equations (19)–(22) and (26)–(29) are satisfied. An overview of the algorithm is shown in Figure 11. Step 4 was added to the implementation because under some test runs the basis was reduced using the lower bound buy-in constraint (46), and this problem solved in milliseconds; which further justified its inclusion. In step 5, MAD–LB is solved last since the lower bound constraint l_i may cause more than one portfolio weight x_i to have the prescribed lower bound value; as was the case with our test runs. In such a case, deciding which basic variable to remove from the set of lower bounds becomes a problem. Solving MAD–LP gives

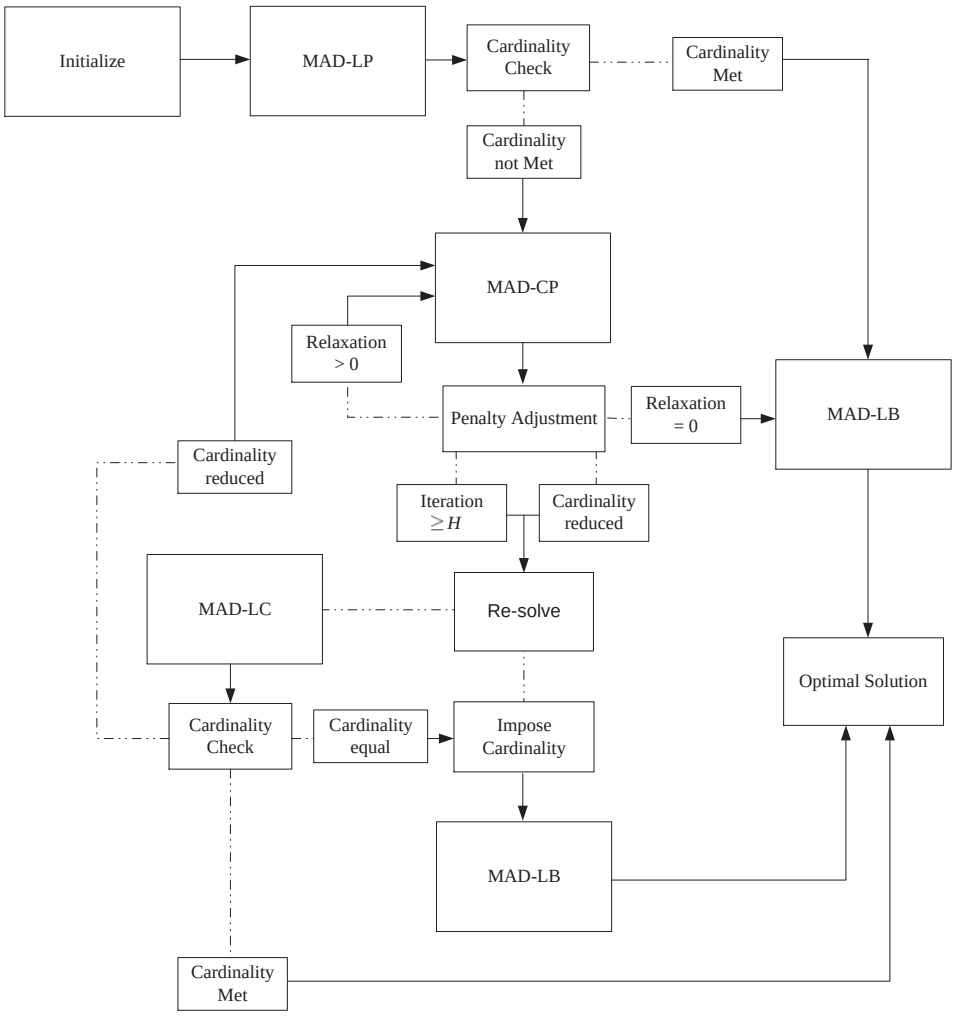

Fig. 11. MAD Cardinality algorithm overview.

one (or possibly a few) minimum x_i weights, which is (are) less than or equal to the corresponding basis values in MAD–LB. Thus, after each iterative solution of MAD–LP, one variable in the optimal basis can be removed until $n \leq G$. This argument is shown in lemma 1, of which the proof is trivial.

Lemma 1 Given the following problems have a unique solution:

$$\begin{array}{lll} \min & c^\top x & \\ & Ax \geq b & (P) \\ & x \geq 0 & \end{array} \quad \text{and} \quad \begin{array}{lll} \min & c^\top \overline{x} & \\ & A\overline{x} \geq b & (\overline{P}) \\ & \overline{x} \geq l & \end{array}$$

if $l > 0$, then the values of the optimal basis satisfy $x^\star \leq \overline{x}^\star$.

The algorithm in Figure 11 tries to satisfy the MAD cardinality model in (19)–(29) without sacrificing optimality. Each subproblem aims to reduce the size of the basis until the cardinality constraint is met. In the worst case, the algorithm enforces the cardinality constraint by iteratively removing the smallest weight in the basis. This is performed in step 5, where removing one weight at a time (or a few, given there is equivalent optimal basis values) the algorithm terminates when the cardinality number is reached. If the values of l_i and/or G are small, this step can be sped up; otherwise, it may give no improvement with respect to CPU time and not be necessary. In such a case, step 5 is replaced with the following two steps:

Algorithm 2: The MAD Cardinality Algorithm under the *Fast Step*.

5. Impose Cardinality *Large Step*:
   Solve MAD–LC (44)–(46) using $\overline{x}$ from `4` (i)(c) or (ii) (via `3` (ii) or (iv))
   (i) set $\widetilde{x} = \{x - (x_i = l_i \cup x_i = 0, \forall\, i = 1, ..., n)\}$ and $\widetilde{G}^i = \sum g_i$
      (a) if $\widetilde{G}^i = G \quad \Rightarrow$ solve MAD–LB for $x_i^\star > 0 \Rightarrow$ `Terminate.`
      (b) else if $\widetilde{G}^i = \widetilde{G}^{i-1}$ or $\widetilde{G}^i = \overline{G}$, `go to 6.`
      (c) else if $\widetilde{G} > G$, `go to 5` using $\widetilde{x}$ from `5` (i).
      (d) otherwise, `go to 6.`
6. Impose Cardinality *Small Step*:
   Solve MAD–LP (30)–(35) using $\{\widetilde{x} + (x_i = l_i \cup x_i = 0,\ \forall\, i = 1, ..., n)\}$ from `5` (i)
   (i) set $\breve{x} = \{x - \min(x_i,\ \forall\, i = 1, ..., n)\}$ and $\breve{G} = \sum g_i$
      (a) if, $\breve{G} \leq G$, `go to 6` (ii).
      (b) otherwise, `go to 6` using $\breve{x}$ from `6` (i)
   (ii) solve MAD–LB (47)–(49) using $\breve{x} \quad \Rightarrow$ `Terminate.`

The *Large Step* removes all the lower bound weights as long as this set of weights does not make the cardinality number less than G. If so, the *Small Step* takes over and removes one weight at a time until the cardinality number is reached. Hence, if l_i and/or G are fairly small, then this speeds up step 5 (`Impose Cardinality`) of the initial algorithm. Otherwise, large l_i values will take too many weights x_i away and/or large G values will only need to remove a small number of weights, making *Large Step* 5 above ineffective. In Table 4, we provide the results with respect to CPU time when using the initial heuristic (or normal step) and the fast step. From Table 4, the average CPU solution time was 6.94s and 12.47s for the fast and normal step algorithm, respectively. In [6], a maximum of 225 securities were solved using 60 time-stages, which took a time of 18345.56 and 280.92 using their integer restart and reoptimization heuristic, respectively. Figure 12 depicts the efficiency frontier using 120 time-stages and 853 securities. In the previous section, MAD Cardinality solved the problem to optimality using $k = 75$ securities. Using the proposed heuristic we have reduced this number to $k = 60$ and the results are remarkably similar to Figure 4 with respect to the shape of the efficiency frontier and the standard deviation values. Another interesting result from the computational runs is that if one solves MAD–LP and then takes the lowest values out to meet the cardinality constraint, as done in the reoptimization heuristic of [6], then the optimal basis ($x_i^\star$) is not the same as our proposed method. This is because the heuristic in Jobst *et al.* (2001) heavily reduces the size of the basis in one step, whereas we iteratively reduce the problem size in an attempt to minimize such occurrences. In any case, from the results of Figure 4 and Table 4, we have constructed a heuristic that satisfies the cardinality constraint, has fast CPU time, and performs well with in that the efficient frontier is close to the efficient frontier of the unconstrained problem.

4. Conclusion

We have considered a linear model MAD as the basis for which various discrete choice constraints relevant to practice is added. Previous considerations have looked at adding such constraints to the MVO framework, but the resulting models are generally very difficult to solve optimally and find feasible solutions for. The main finding is that the linear model approach is substantially more computationally tractable. A commercial solver was able to solve to optimality in reasonable CPU time instances whose corresponding MVO models needed a heuristic to find a feasible solution. In particular, we show that the MAD model can incorporate the same portfolio constraints considered in [4,6] without the use of a model specific algorithm. In fact, the graphs of 2–9 produce better efficient frontiers (lower risk for a given return goal), are solved in less time, and can handle problems with more variables than what is shown in [4,6].

Although the MAD model does not require an algorithm to solve the types of instances seen in the literature for corresponding MVO instances, we constructed a heuristic inspired by the designs in Jobst *et al.* (2001) to solve much larger and challenging instances of the basic MAD with cardinality constraint model e.g. over 65 times larger. From Table 4, the algorithm proved to solve the large set of portfolio decisions in under 2/3 of a minute. The algorithm uses a decomposition strategy that involves solving a sequence of subproblems to generate optimal solutions that minimize instances where decision variables are negated. The algorithm performance results with respect to the efficient frontier are very good for lower portfolio return values, es-

Table 3

Index	Number of Securities / Time-Stages	Cardinality Number	Solution Time (s): *Fast Step*	*Normal Step*
S&P TSX	853/100	25	8.45	8.45
	853/120	56	3.37	4.31
		58	4.02	13.52
		59	9.48	12.27
		59	3.55	13.69
		60	13.39	23.14
			5.00	8.06
			6.36	17.05
			5.42	6.61
			4.44	5.97
			6.03	8.89
			4.16	14.00
			5.50	6.20
			4.31	6.80
	853/138	70	20.59	38.03

CPU time for the initial heuristic (normal step) and fast step.

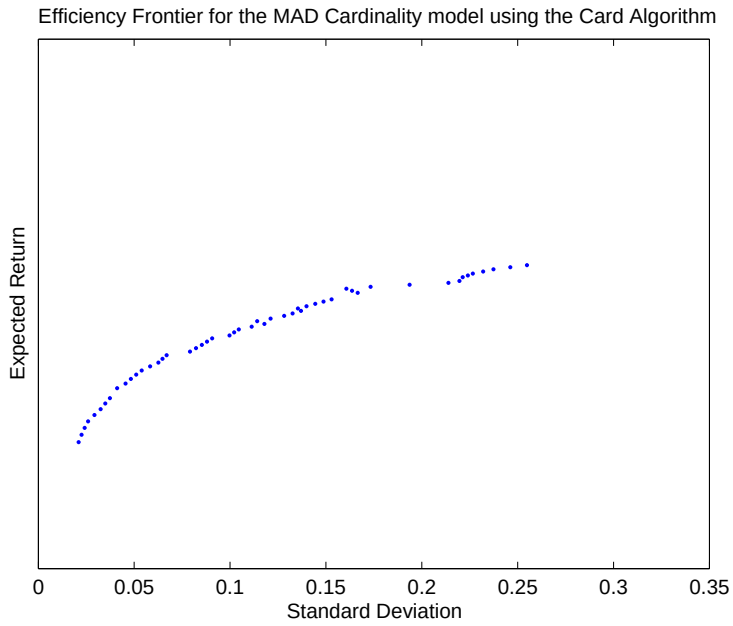

Fig. 12. Efficiency frontier for the MAD Cardinality model using 120 time-stages and 853 securities, where $G = 60$.

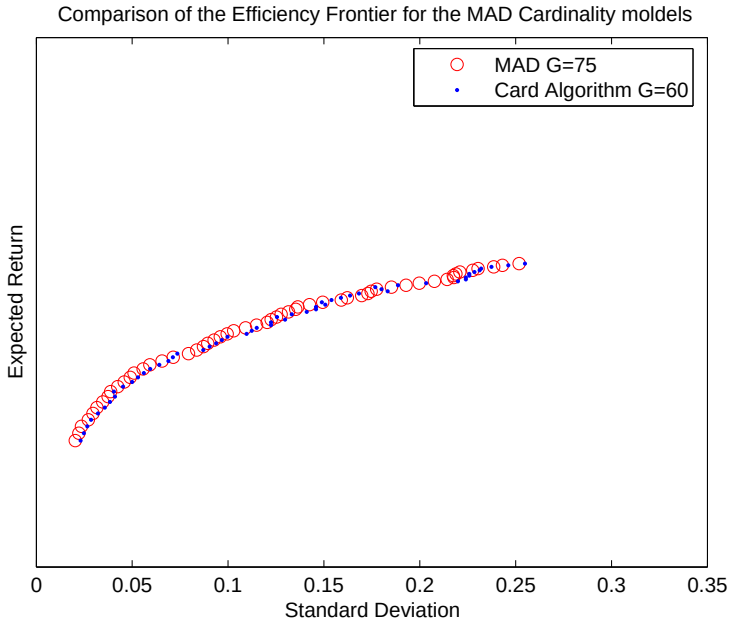

Fig. 13. Efficiency frontier comparison of the results presented in Figure 4 ($G = 75$) of Section 2. and the MAD Card Algorithm shown in Figure 12 ($G = 60$).

pecially when considering the number of variables and constraints used in the problem. In fact, only for high expected return values does the graph in Figure 12 become somewhat distorted, which is most probably due to the cardinality constraint and not the algorithm; as this characteristic is also prevalent in the graph of Figure 4 where no algorithm was used.

Future research can involve adding further constraints and considerations in the MAD portfolio design, as well as exploring different types of algorithmic approaches. As shown in [14], there exists a number of portfolio goals, risk measures, and managing characteristics such as transaction costs, that can be added to the model. Another area of interest consists of exploring different types of algorithmic approaches, i.e. the simulated annealing method used in [4]. There exists a number of heuristics that have been applied to various MIP problems. The combination of an exact algorithmic approaches and additional heuristics may improve CPU time and improve the quality of solutions.

References

[1] Adcock, C.J. and Meade, N.: A simple algorithm to incorporate transactions costs in quadratic optimisation, *European Journal of Operational Research*, 79 (1994) 85-94.

[2] Chang, T.-J., Meade, N., Beasley, J.E. and Sharaia, Y.M.: Heuristics for cardinality constrained portfolio optimisation, *Computers & Operations Research*, 27 (2000) 1271-1302.

[3] Chopra, V.K. and Ziemba, W.T.: The effect of errors in means, variances, and covariances on optimal portfolio choice, *The Journal of Portfolio Management*, 19 (1993) 6-11.

[4] Crama, Y. and Schyns, M.: Simulated annealing for complex portfolio selection problems, *European Journal of Operational Research*, 150 (2003) 546-571.

[5] Hanoch, G. and Levy, H.: The efficiency analysis of choices involving risk, *The Review of Economic Studies*, 36 (1969) 335-346.

[6] Jobst, N.J., Horniman, M.D., Lucas, C.A. and Mitra, G.: Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints, *Quantitative Finance*, 1 (2001) 489-501.

[7] Kallberg, J.G. and Ziemba, W.T.: Comparison of alternative utility functions in portfolio selection problems, *Management Science*, 29 (1983) 1257-1276.

[8] Kellerer, H., Mansini, R. and Speranza, M.G.: Selecting portfolios with fixed costs and minimum transaction lots, *Annals of Operations Research*, 99 (2000) 287-304.

[9] Konno, H. and Yamazaki, H.: Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market, *Management Science*, 37 (1991) 519-531.

[10] Lin, D. and Wang, S.: A genetic algorithm for portfolio selection problems, *Advanced Modeling and Optimization*, 4 (2002) 13-27.

[11] Liu, S., Wang, S.Y. and Qiu, W.: Mean-variance-skewness model for portfolio selection with transaction costs, *International Journal of Systems Science*, 34 (2003) 255-262.

[12] Mansini, R. and Speranza, M.G.: An exact approach for portfolio selection with transaction costs and rounds, *IIE Transactions*, 37 (2005) 919-929.

[13] Markowitz, H.M.: Portfolio selection, *The Journal of Finance*, 7 (1952) 77-91.

[14] Stoyan, S.J. and Kwon, R.H.: A two-stage stochastic mixed-integer programming approach to the index tracking problem, *Optimization Engineering*, 11 (2010) 247-275.

[15] Xia, Y., Liu, B., Wang, S. and Lai, K.K.: A model for portfolio selection with order of expected returns, *Computers & Operations Research*, 27(2000) 409-422.

Received 21-6-2011; revised 5-9-2011; accepted 6-9-2011