Journal of Global Optimization (2005) 32:207–219
DOI 10.1007/s10898-004-2703-x

Global Optimization Versus Integer Programming in Portfolio Optimization under Nonconvex Transaction Costs

HIROSHI KONNO and REI YAMAMOTO
Department of Industrial and Systems Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-Ku 112-855-1, Tokyo, Japan (e-mail: konno@indsys.chou-u.ac.jp)

(Received 1 April 2004; accepted in revised form 7 April 2004)

Abstract. This paper is concerned with a portfolio optimization problem under concave and piecewise constant transaction cost. We formulate the problem as nonconcave maximization problem under linear constraints using absolute deviation as a measure of risk and solve it by a branch and bound algorithm developed in the field of global optimization. Also, we compare it with a more standard 0–1 integer programming approach. We will show that a branch and bound method elaborating the special structure of the problem can solve the problem much faster than the state-of-the integer programming code.

Key words: Branch and bound algorithm, Global optimization, Nonconvex transaction cost, Portfolio optimization, 0–1 integer programming

1. Introduction

The purpose of this paper is to compare the relative advantage of global optimization approach and integer programming approach in portfolio optimization under nonconvex transaction cost.

One of the standard formulations of the portfolio optimization problem is to maximize the net return, i.e., the expected rate of return of the portfolio subtracted by the transaction cost subject to the constraint on the magnitude of risk [2, 11]. When the transaction cost is linear, then the problem can be solved by standard methods.

Unfortunately, however the transaction cost function is usually nonconvex. Transaction cost is usually relatively larger when the amount of transaction is smaller [7, 8] Typical transaction cost functions are piecewise linear concave and piecewise constant with several jumps. Therefore, the problem can no longer be solved by standard nonlinear programming methodologies.

One may argue that transaction cost can be ignored since it has been significantly reduced during the past decade. In fact, the amount of transaction cost is only a fraction of a decade ago. Nevertheless, it still bothers investors when the rate of return of a portfolio is small. Also, the cost will accumulate when one sells and buys assets frequently [9].

The standard approach for handling a concave or piecewise constant cost function is to introduce a number of 0–1 variables and solve the resulting 0–1 integer programming problem by branch and bound or branch and cut algorithms [13]. Among the renowned softwares for solving these problems is CPLEX, which can solve a large scale 0–1 linear integer programming problems when the number of integer variables is not too large. When, however the number of linear pieces (or number of jumps) is large, then the problem becomes more difficult since we need to introduce many integer variables. When the number of linear pieces is 7 and the number of assets is 1000, then we need to handle 7000 0–1 variables, which is still out of the scope of the state-of-the art integer programming software.

An alternative method for handling concave and piecewise constant functions is a branch and bound method recently developed in the field of global optimization [3, 15, 17]. It has been successfully applied to portfolio optimization problems under piecewise linear concave transaction cost [5–9]. The success depends upon the use absolute deviation instead of the standard deviation as the measure of risk [10, 15, 16], thereby formulating the problem as a concave minimization under linear constraints.

Linear formulation also plays a crucial role when using commercial softwares, which has been primarily designed for linear programming problems including integer variables. If the risk measure is standard deviation, it is almost impossible to solve the problem containing many integer variables.

In this paper, we will present a rigorous branch and bound algorithm for solving portfolio optimization problem under piecewise linear and piecewise constant cost functions. Also, we will develop a number of schemes to reduce the amount of computation and show that they work very well for problems under consideration.

In the next section, we will formulate the portfolio optimization problem under nonconvex transaction cost as a global optimization problem. Section 3 will be devoted to a branch and bound algorithm for solving the problem. Also we will propose schemes to reduce the amount of computation. In Section 4, we will compare the branch and bound algorithm with 0–1 integer programming approach. It will be shown that the global optimization approach is competitive to CPLEX and that it outperforms CPLEX when the number of linear pieces is beyond some bound.

2. Problem Formulation

In a series of the articles [5, 9], we discussed a portfolio optimization problem of the following type

$$\left|\begin{array}{ll} \text{maximize} & \displaystyle\sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} c(x_j) \\ \text{subject to} & \displaystyle W\left[\sum_{j=1}^{n} R_j x_j\right] \leqslant wM \\ & \displaystyle\sum_{j=1}^{n} x_j = M \\ & 0 \leqslant x_j \leqslant u_j, \quad j = 1, 2, \ldots, n, \end{array}\right. \tag{1}$$

where:

R_j: random variable representing the rate of return of jth asset
r_j: expected value of R_j
M: amount of total investment
u_j: constant specifying the upper bound of the amount of investment into jth asset
x_j: variable representing the amount of investment into jth asset
$R(\boldsymbol{x})$: rate of return of the portfolio $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
$W[R(\boldsymbol{x})]$: absolute deviation of $R(\boldsymbol{x})$, i.e., $E[|R(\boldsymbol{x}) - E[R(\boldsymbol{x})]|]$
w: constant specifying the allowable level of the risk
$c(x_j)$: transaction cost function associated with purchasing x_j

We will assume throughout that the cost function $c(\cdot)$ is either one of the types described in Figure 1.

We will further assume that $(R_1, R_2, \ldots, R_n)$ is distributed over a finite set of points $(r_{1t}, r_{2t}, \ldots, r_{nt}), t = 1, 2, \ldots, T$ and that

$$f_t = Pr\{(R_1, R_2, \ldots, R_n) = (r_{1t}, r_{2t}, \ldots, r_{nt})\}, \quad t = 1, 2, \ldots, T,$$

are known. In the following discussion, we will assume for simplicity that $f_t = 1/T$ for all t, so that

$$r_j = \sum_{t=1}^{T} r_{jt}/T \tag{2}$$

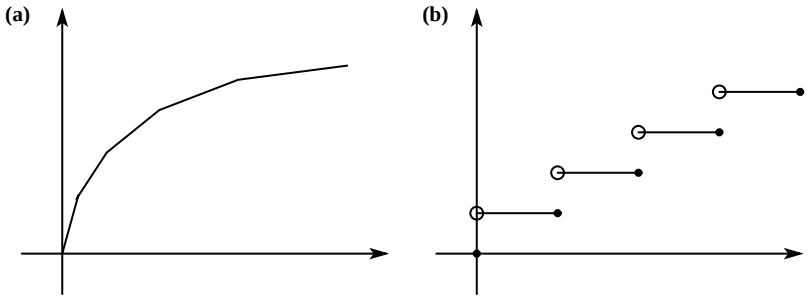

Figure 1. Transaction cost function.

$$W[R(\boldsymbol{x})] = \sum_{t=1}^{T} \left| \sum_{j=1}^{n} (r_{jt} - r_j) x_j \right| \Big/ T \tag{3}$$

and hence the problem (1) can be reformulated as a linearly constrained convex maximization problem with separable objective function (See [1, 7] for details)

$$\left|\begin{array}{ll} \text{maximize} & \displaystyle\sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} c(x_j) \\ \text{subject to} & \displaystyle\sum_{t=1}^{T} (\psi_t + \phi_t)/T \leqslant wM \\ & \psi_t - \phi_t = \displaystyle\sum_{j=1}^{n} (r_{jt} - r_j) x_j, \quad t = 1, 2, \ldots, T \\ & \psi_t \geqslant 0, \phi_t \geqslant 0, \quad t = 1, 2, \ldots, T \\ & \displaystyle\sum_{j=1}^{n} x_j = M \\ & o \leqslant x_j \leqslant u_j, \quad j = 1, 2, \ldots, n \end{array}\right. \tag{4}$$

3. Branch and Bound Algorithm

In this section, we will present a branch and bound algorithm for solving (4) using linear underestimating functions of the following two types of nonconvex transaction cost functions [5, 15].

Let $c_j(x_j)$ be a piecewise linear concave function defined in the interval $[\alpha, \beta]$. Then we approximate $c_j(x_j)$ by an affine function $\bar{c_j}(x_j)$ connecting two endpoints as depicted in Figure 2. Note that

$$\bar{c_j}(x_j) \leqslant c_j(x_j), \quad x_j \in [\alpha, \beta]$$

Let $c_j(x_j)$ be a piecewise constant function defined in the interval $[\alpha, \beta]$. We choose as $\bar{c_j}(x_j)$ the convex envelope of $c_j(x_j)$ over $[\alpha, \beta]$, which can be

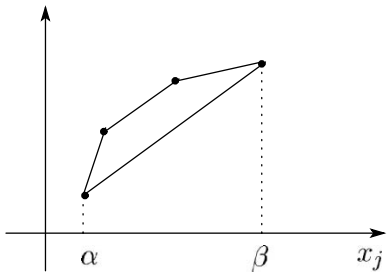

Figure 2. Piecewise linear concave function.

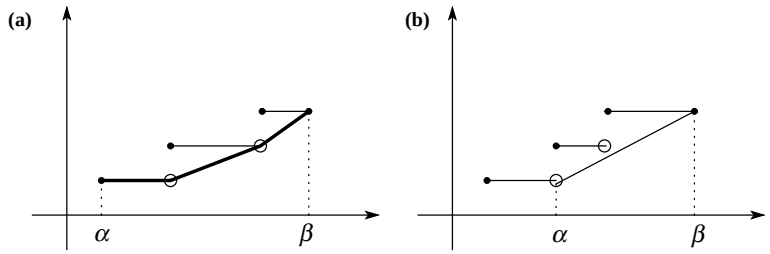

Figure 3. Convex envelope. Piecewise constant function.

obtained by connecting three endpoints (Figure 3-(a)) or two endpoints (Figure 3-(b)) when $c_j(\cdot)$ consists of three linear pieces.

Let us denote

$$F = \{(\boldsymbol{x}, \psi, \phi) | \sum_{t=1}^{T} (\psi_t + \phi_t)/T \leqslant wM, \sum_{j=1}^{n} x_j = M$$

$$\psi_t - \phi_t = \sum_{j=1}^{n} (r_{jt} - r_j) x_j, \psi_t \geqslant 0, \phi_t \geqslant 0, t = 1, 2, \ldots, T\} \tag{5}$$

and let us define the subproblems (P_k)

$$(P_k) \left| \begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} c_j(x_j) \\ \text{subject to} & (\boldsymbol{x}, \psi, \phi) \in F \\ & \alpha_j^k \leqslant x_j \leqslant \beta_j^k, \quad j = 1, 2, \ldots, n, \end{array} \right. \tag{6}$$

and

$$(\bar{P}_k) \left| \begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} \bar{c}_j^k(x_j) \\ \text{subject to} & (\boldsymbol{x}, \psi, \phi) \in F \\ & \alpha_j^k \leqslant x_j \leqslant \beta_j^k, \quad j = 1, 2, \ldots, n, \end{array} \right. \tag{7}$$

where $\bar{c}_j^k(x_j)$ is a convex underestimator of $c_j(x_j)$ in the interval $[\alpha_j^k, \beta_j^k]$.

Let $(\boldsymbol{x}^k, \psi^k, \phi^k)$ be an optimal solution of $(\bar{P}_k)$ and let

$$g_k = \sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} \bar{c}_j^k(x_j^k) \tag{8}$$

$$f_k = \sum_{j=1}^{n} r_j x_j - \sum_{j=1}^{n} c_j(x_j^k) \tag{9}$$

For details of branch and bound algorithm such as ω-subdivision strategy, breadth first branching and its convergence properties, the readers are referred to either [6, 7, 8] or [17].

Algorithm BB (Branch and Bound Algorithm)

0° $\boldsymbol{P} = \{P_0\}$, $\hat{f} = -\infty, k = 0, \varepsilon > 0$.
1° If $\boldsymbol{P} = \phi$ then go to 8°, Otherwise go to 2°.
2° Choose one subproblem P_l where $g_l = \max\{g_t | P_t \in \boldsymbol{P}\}$. $\boldsymbol{P} = \boldsymbol{P} \setminus \{P_l\}$.
3° If $|g_l - f_l| > \varepsilon$ then go to 6°. Otherwise go to 4°.
4° If $f_l < \hat{f}$ then go to 5°. Otherwise $\hat{f} = f_l$ and eliminate all the subproblems P_t for which $g_t \leqslant \hat{f} + \varepsilon$ and go to 1°.
5° If $g_l \leqslant \hat{f} + \varepsilon$ then go to 1°. Otherwise go to 6°.
6° Choose $c_s(x_s^l) - \bar{c}_s^l(x_s^l) = \max\{|c_j(x_j^l) - \bar{c}_j^l(x_j^l)| | j = 1, 2, \ldots, n\}$.
7° We generate two subproblems.

$$P_{T+1} = P_l \cap \{\boldsymbol{x} | \alpha_s^l \leqslant x_s \leqslant x_s^l\}$$

$$P_{T+2} = P_l \cap \{\boldsymbol{x} | x_s^l \leqslant x_s \leqslant \beta_s^l\}.$$

Solve $(\bar{P}_{T+1}), (\bar{P}_{T+2})$. If both $(\bar{P}_{T+1}), (\bar{P}_{T+2})$ are infeasible then go to 1°. If $(\bar{P}_{T+1}), (\bar{P}_{T+2})$ are feasible, let

$$f_{T+1} = \sum r_j x_j^{T+1} - \sum c(x_j^{T+1})$$
$$g_{T+1} = \sum r_j x_j^{T+1} - \sum \bar{c}_j^{T+1}(x_j^{T+1})$$
$$f_{T+2} = \sum r_j x_j^{T+2} - \sum c(x_j^{T+2})$$
$$g_{T+2} = \sum r_j x_j^{T+2} - \sum \bar{c}_j^{T+2}(x_j^{T+2}).$$

$\boldsymbol{P} = \boldsymbol{P} \cup \{P_{T+1}, P_{T+2}\}, k = k + 1$, and go to 1°.
8° stop.

THEOREM 1. The Algorithm BB generates an ε-optimal solution in finitely many steps.

Proof. See [5, 17].

Though convergent, the branch and bound algorithm may require excessive amount of computation time when n is large. To reduce the amount of computation, we employ two heuristics.

Heuristic Scheme 1. Problem Reduction

Let $\boldsymbol{x}^0$ be an optimal solution at the first iteration of the branch and bound algorithm and let $x_j^0 = 0, j \in J_0$ and $x_j^0 > 0, j \in J_+$. Then we eliminate all variables x_j's, $j \in J_0$ from the set of variables in the succeeding step.

The reasons behind this heuristic are

(i) The weight of assets j such that $x_j^0 = 0$ are not likely to have a large value in an optimal solution.

(ii) It is not profitable to purchase smaller amount of x_j since transaction cost per unit is relatively larger for smaller x_j.

Hence most x_j's $j \in J_0$ are expected to be zero in an optimal solution.

Heuristic Scheme 2. Skipping the Iteration Before Convergence

Let $(\tilde{\boldsymbol{x}}^t, \tilde{\psi}^t, \tilde{\phi}^t)$ be an optimal solution of (P_t). Let $(\bar{P}_t)$ be the relaxation of (P_t) by replacing the cost function by its convex underestimator. If the optimal solution $(\boldsymbol{x}^t, \psi^t, \phi^t)$ of $(\bar{P}_t)$ satisfies the condition

$$c(x_j^t) - \bar{c}_j^t(x_j^t) < \varepsilon, \forall j$$

then we are done. However, even when this condition is not satisfied, we would know after certain iterations that $\tilde{x}_j^t$ would not lie on certain linear pieces. For example, if x_j^t lies on the third linear piece (See Figure 4) for large enough t, then $\tilde{x}_j^t$ is not likely to lie on the first linear piece. Moreover, one may guess that $\tilde{x}_j^t$ lies on the third linear piece. If we know the piece on which $\tilde{x}_j$'s lies, then we can obtain it by solving an associated linear programming problem.

This leads us to fathom the subproblem (P_t) by solving the associated linear program when $c_j(x_j^t) - \bar{c}_j^t(x_j^t)$ become reasonably small.

4. Computational Experiments

We conducted numerical simulation using historical data collected for 900 stocks in the Tokyo Stock Exchange.

The branch and bound algorithm was coded in C++ using a PentiumIV 512 Mbyte 2.8 GHz personal computer. We used NUOPT V of Mathematical Systems, Inc. for solving linear programming subproblems. Also, we solved the same problems on the same computer by using CPLEX Version 7.1, where a

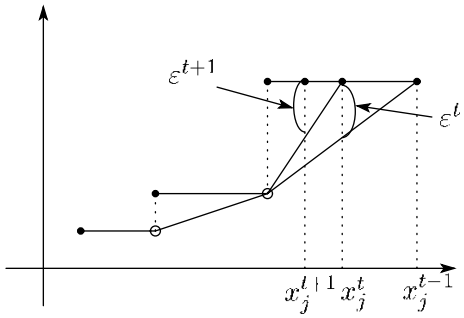

Figure 4. Iteration skipping.

Table 1. Transaction cost

Amount of transaction (10^4 yen)	Cost (10^4 yen)
(a) Piecewise linear concave cost	
0–50	1.40%
50–70	1.10% + 0.15
70–100	0.90% + 0.29
100–300	0.85% + 0.34
300–500	0.80% + 0.49
500–1000	0.68% + 1.09
1000–3000	0.55% + 2.39
(b) Piecewise constant cost	
0–300	0.2
300–600	0.4
600–900	0.6
900–1200	0.8
1200–1500	1.0
1500–1800	1.2
1800–2100	1.4

piecewise linear concave function and a piecewise constant function are represented as a linear function by introduction 0–1 integer variables [13].

Table 1-(a) shows the piecewise linear concave cost function of a leading security company of Japan. Table 1-(b) shows the piecewise constant cost table associated with a typical internet transaction.

Table 2 shows the computational result for BB algorithm and CPLEX when $M = 100$ million yen and $u_j = 0.02$ for all j. The maximal investable amount of money to each individual asset is 2 million yen, so that it suffices to consider piecewise constant cost function with only one linear piece. We see from Table 2 that CPLEX can solve the problem with less than 10 sec, even when n is 900. The BB algorithm with problem reduction strategy (Heuristic 1), on the other hand, requires 3-to 13-times more computa-

Table 2. Computational result ($w = 0.035$, piecewise constant cost)

n	CPLEX		BB		Objective function
	CPU(sec)	# of subproblem	CPU(sec)	# of subproblem	(CPLEX-BB)/ BB(%)
100	1	1	8	81	1.075
200	1	20	3	21	0.079
300	1	30	3	15	0.092
400	2	19	15	112	0.147
500	3	1	11	80	0.185
600	4	100	13	69	0.141
700	5	39	9	48	0.027
800	5	100	42	253	0.034
900	9	200	117	577	0.098

tion time. Also, CPLEX generates a better solution (in term of the value of the optimal net return) than BB algorithm.

Let us note, however that NUOPT solved each linear subproblem from scratch (using two phase simplex method), while CPLEX is supposed to use a more efficient dual simplex procedure based upon the information about the optimal basis of the master problem [1]. The computation time of BB algorithm should be significantly smaller if we use the dual simplex procedure.

Table 3 shows the computational result when $M = 100$ million and $u_j = 0.05$ for all j. The maximal amount of the fund to be invested into each asset is 5 million yen, so that we need to handle piecewise constant functions with 2 constant pieces.

We see from this that BB algorithm can solve problems faster. Also it is more stable. This is due to the fact that many variables are eliminated by problem reduction strategy since upper bound u_j is larger.

Tables 4 and 5 show the computational results of piecewise linear concave function with 7 and 6 linear pieces. We see from this that BB algorithm outperforms CPLEX. Let us note that BB algorithm sometimes

Table 3. Computational results ($w = 0.035$, piecewise constant cost)

n	CPLEX		BB		Objective function
	CPU (sec)	# of subproblem	CPU (sec)	# of subproblem	(CPLEX-BB)/BB(%)
100	1	7	3	46	0.195
200	1	100	8	107	0.125
300	1	48	8	109	0.183
400	5	145	4	38	0.075
500	5	100	11	63	0.053
600	5	40	4	18	0.002
700	6	84	12	71	0.065
800	8	50	7	22	0.064
900	8	20	33	152	0.062

Table 4. CPLEX versus BB ($w = 0.035$, $u_j = 0.05$, piesewise linear with 7 linear pieces)

n	CPLEX		BB		Objective function
	CPU (sec)	# of subproblem	CPU (sec)	# of subproblem	(CPLEX-BB)/BB(%)
100	2	10	1	25	0.726
200	32	2200	13	249	0.000
300	76	3400	72	1364	0.002
400	37	600	31	443	0.000
500	142	3000	102	972	0.030
600	194	3300	102	971	0.000
700	214	4000	121	1170	−0.001
800	397	6600	100	581	−0.003
900	279	3400	166	801	0.000

Table 5. CPLEX versus ($w = 0.035$, $u_j = 0.02$, piecewise linear with 6 linear pieces)

n	CPLEX		BB		Objective function
	CPU (sec)	# of subproblem	CPU (sec)	# of subproblem	(CPLEX-BB)/ BB(%) 16
100	9	239	3	26	0.000
200	19	1200	9	27	0.016
300	537	32400	10	100	0.012
400	296	12200	34	347	−0.003
500	462	12100	56	548	0.058
600	414	9699	83	535	−0.001
700	2559	57500	103	842	0.020
800	5669	144500	417	2704	0.000
900	1760	38900	92	572	−0.002

Table 6. CPLEX versus BB, with Heuristic 2

	BB ($\varepsilon = 10^{-4}$)	BB ($\varepsilon = 10^{-3}$)	BB ($\varepsilon = 10^{-2}$)	CPLEX
Net return(%)	1.087	1.087	1.084	1.087
Risk(%)	3.5	3.5	3.5	3.5
Cost (million)	2.159	2.159	2.169	2.159
CPU time(sec)	87	7	1	34
# of assets	23	22	24	22

generates a better solution than CPLEX in terms of the value of the objective function (See underlines in Table 4 and 5). When we employ dual simplex procedure, the difference should be more significant.

Next, we will show the remarkable effect of Heuristic 2 explained in Section 3. Table 6 shows the results for piecewise linear concave case with seven linear pieces when $M = 300$ million and $u_j = 0.05$. Optimal solutions of BB algorithm for $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-3}$ are exactly the same as those of

Table 7. Effect of Heuristic 2 (Piecewise constant cost)

n	BB ($\varepsilon = 10^{-3}$)		BB ($\varepsilon = 10^{-2}$)		CPLEX	
	CPU (sec)	Net return (%)	CPU (sec)	Net return (%)	CPU (sec)	Net return (%)
100	7	0.341	5	0.341	1	0.344
200	2	1.107	1	1.108	1	1.109
300	2	1.595	2	1.603	1	1.605
400	2	1.790	1	1.786	2	1.790
500	6	7.864	2	1.862	3	1.868
600	6	2.129	2	2.133	4	2.139
700	4	2.275	2	2.279	5	2.284
800	22	2.330	4	2.323	5	2.332
900	34	2.381	4	2.373	9	2.384

Table 8. Effect of Heuristic 2 (Piecewise linear cost)

n	BB ($\varepsilon = 10^{-3}$)		BB ($\varepsilon = 10^{-2}$)		CPLEX	
	CPU (sec)	net return (%)	CPU (sec)	net return (%)	CPU (sec)	net return (%)
100	1	0.376	0	0.376	2	0.379
200	7	1.087	1	1.084	32	1.087
300	26	1.717	1	1.715	76	1.717
400	6	1.894	1	1.889	37	1.894
500	26	1.981	1	1.981	142	1.982
600	15	2.264	2	2.264	194	2.264
700	26	2.354	2	2.348	212	2.354
800	26	2.358	2	2.355	397	2.358
900	37	2.376	3	2.371	279	2.376

CPLEX. Also, the computation time of BB algorithm for $\varepsilon = 10^{-3}$ is significantly smaller.

Table 7 shows the result for piecewise constant cost function with two linear pieces. We see that BB($\varepsilon = 10^{-2}$) is competitive to CPLEX.

Table 8 shows the result for piecewise linear transaction cost with seven linear pieces when $M = 300, u_j = 0.05$. Optimal solutions of BB ($\varepsilon = 10^{-3}$) are almost the same as those of CPLEX. Also, computation time of BB algorithm is much less than that of CPLEX. This shows that BB algorithm with Heuristic 1 and 2 is superior to CPLEX, particularly when the number of linear pieces is large.

5. Conclusions

We showed in this paper that a portfolio optimization problem under piecewise linear concave and piecewise constant cost functions can be solved in an efficient manner by a branch and bound algorithm.

We compared the performance of this algorithm with an alternative 0–1 integer programming approach and showed that branch and bound algorithm outperforms the state-of-the art integer programming code when the number of linear pieces is larger and hence 0–1 integer programming formulation requires a large number of 0–1 variables.

Let us note that this conclusion is valid only for the specific portfolio optimization problem treated in this paper, i.e., a problem with smaller number of linear constraints in addition to the lower and upper bound constraint on each variable. The reason is that heuristic procedures presented in Section 3, constraint are extremely powerful for this class of problems.

Many (heuristic) algorithms for portfolio optimization under nonconvex transaction costs have been proposed in the past. Most of these studies followed the standard mean–variance framework. However, these algorithms

need not generate a satisfactory result for quadratically constrained nonconcave maximization (or minimization of a convex quadratic function subject to nonconvex constraints). These problems can be solved in an efficient manner only when we formulate it within the linear framework, i.e., only when we use linear risk measure such as (lower-semi)-absolute deviation or conditional value at risk.

Acknowledgements

This research was conducted under the support of Grant-in-Aid for Scientific Research B(2) 15310122 and 15656025 of the Ministry of Education, Science and Culture of Japan.

References

1. Chvatál, V. (l983), *Linear programming*, Freeman and Co.
2. Elton, J.E. and Gruber, M.J. (1998), *Modern Portfolio Theory and Investment Analysis* (6th edition), John Wiley & Sons, New York.
3. Falk, J.E. and Soland, R.M. (1969), An Algorithm for separable nonconvex programming problem, *Management Science* 15, 550–569.
4. Konno, H. (1990), Piecewise linear risk functions and portfolio optimization, *Journal of the Operations Research Society of Japan* 33, 139–156.
5. Konno, H. and Wijayanayake, A. (1999), Mean–absolute deviation portfolio optimization model under transaction costs, *Journal of the Operations Research Society of Japan* 42, 422–435.
6. Konno, H. and Wijayanayake, A. (2001), Optimal rebalancing under concave transaction costs and minimal transaction units constraints, *Mathematical Programming* 89, 233–250.
7. Konno, H. and Wijayanayake, A. (2002), Portfolio optimization under D.C. transaction costs and minimal transaction unit constraints, *Journal of Global Optimization* 22, 137–154.
8. Konno, H. and Wijayanayake, A. (2001), Minimal cost index tracking under concave transaction costs, *International Journal of Theoretical and Applied Finance* 4, 939–957.
9. Konno, H. and Yamamoto, R. (2003), Minimal concave cost rebalance of a portfolio to the efficient frontier, to appear in *Mathematical Programming*.
10. Konno, H. and Yamazaki, H. (1991), Mean–absolute deviation portfolio optimization model and its applications to Tokyo stock market, *Management Science* 37, 519–531.
11. Markowitz, H. (1959), *Portfolio Selection; Efficient Diversification of Investment*, Wiley.
12. Mulvey, J.M. (1999), Incorporating transaction costs in models for asset allocation. In: Ziemba, W. et al., (eds.), *Financial Optimization*, Cambridge University Press, New York pp. 243–259.
13. Nemhauser, G.L. and Wolsey, L.A. (1998), *Integer and Combinational Optimization*, John Wiley & Sons.
14. Ogryczak, O. and Ruszczynski, A. (1999), From stockastic dominance to mean risk model, *European Journal of Operational Research* 116, 33–50.
15. Phong, T.Q., An, L.T.H. and Tao, P.D. (1995), Decomposition branch and bound method for globally solving linearly constraind indefinite quadratic minimization problems, *Operations Research Letters* 17, 215–220.

16. Simaan, Y. (1997), Estimation of risk in portfolio selection: the mean–variance model and the Mean–Absolute deviation model, *Management science* 43, 1437–1446.
17. Tuy, H. (1998), *Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht.