Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

AN EMPIRICAL COMPARISON BETWEEN NONLINEAR PROGRAMMING OPTIMIZATION AND SIMULATED ANNEALING (SA) ALGORITHM UNDER A HIGHER MOMENTS BAYESIAN PORTFOLIO SELECTION FRAMEWORK

Jingjing Lu
Merrill Liechty

101 North 33 St
LeBow College of Business, Drexel University
Philadelphia, P.A. 19104, U.S.A.

ABSTRACT

The optimal portfolio selection problem has long been of interest to both academics and practitioners. A higher moments Bayesian portfolio optimization model can overcome the shortcomings of the traditional Markowitz approach and take into consideration the skewness of asset returns and parameter uncertainty. This paper presents a comparison between the simulated annealing and the nonlinear programming methods of optimization for the Bayesian portfolio selection problem in which the objective function includes the portfolio mean, variance and skewness. We make the comparison for a utility function that is easily optimized using both methods. In particular we maximize a cubic utility function, and our results show that to achieve the same level of accuracy, the CPU time for the nonlinear programming optimization will be shorter than for the simulated annealing algorithm. Though it is slower, the simulated annealing algorithm is still a viable option for this utility function.

1 INTRODUCTION

The optimal portfolio selection problem has long been of interest to both academics and practitioners. The mean-variance paradigm introduced by Markowitz in 1952 is by far the most common formulation of portfolio optimization problems. The improvement achieved by Merton (1969,1971), Samuelson (1969), and Fama (1970) is also worth noting. There is a large amount of work using linear and quadratic programming, which considers with certainty the inputs from the first stage, *e.g.*, the exact means, variances and covariances, while ignoring the skewness and other higher moments of the asset returns.

The major theoretical work of Markowitz showed that maximizing the mean while minimizing the variance of a portfolio comes from the idea that the investor prefers higher expected returns and lower risk. Based on Markowitz's idea, Arrow (1971) argued that desirable utility functions should exhibit decreasing absolute risk aversion, implying that investors should have preference for positively skewed asset returns. In another words, investors would prefer a high probability of an extreme event in the positive direction over a high probability of an extreme event in the negative direction. This brings the concept of skewness into financial researcher's model. Harvey and Siddique (2000) introduced an asset pricing model that incorporates conditional skewness, and showed that an investor may be willing to accept a negative expected return in the presence of high positive skewness. Harvey et al. (2004) propose a method to address both parameter uncertainty and the inclusion of higher moments in the portfolio selection. This allows the investor to capture the asymmetry of the returns and include it in the portfolio selection task. They describe the asset return as a Skew Normal distribution under the Bayesian probability framework. In such a Bayesian framework the expected utilities are then maximized using predictive returns.

Harvey et al. (2004) suggests using the Bayesian framework to evaluate and optimize the expected utilities. The expected utility can be estimated via Markov chain Monte Carlo (MCMC) simulation by generating a set of samples from the posterior predictive distribution and then averaging the utility over the values obtained. MCMC methods have been frequently used in Bayesian estimation. The algorithm was introduced by Metropolis et al. (1953) and generalized by Hastings (1970). Later the Gibbs sampler method as proposed by Gelfand and Smith (1990) made another important contribution to these simulation methods. These methods are used to produce a Markov chain whose output corresponds to a sample from the joint posterior distribution. The expected utility can then be optimized via the Metropolis simulated annealing (SA) algorithm. In this way we can explore the expected utility as a function of the portfolio weights.

As a generic probabilistic meta-algorithm for the global optimization problem, simulated annealing has been widely used as a heuristic method to find approximate solutions to combinatorial optimization problems, such as network problems, scheduling, and transportation, etc. By moving both uphill and downhill, simulated annealing can escape from local optima and continue the searching process of global optimization.

However, for practitioners the simulated annealing algorithm is computationally expensive and often not easy to realize due to the complicated simulating process. The simulation time in this optimization methodology can not be easily ignored. In this paper, we make an empirical comparison between nonlinear programming optimization and the simulated annealing algorithm under the same higher moments Bayesian portfolio selection framework using the Skew Normal distribution. We show that the result of the nonlinear programming optimization can be close to the result of the simulated annealing algorithm. However, to achieve the same level of accuracy, the CPU time for the nonlinear programming optimization will be shorter than the Simulated Annealing algorithm in this portfolio selection problem.

2 BAYESIAN OPTIMIZATION MODEL AND SIMULATED ANNEALING

In order to capture the effect of skewness, a proper utility function must be considered. Following Harvey et al. (2004), let x^0 represent the historical observation of the asset price and let x represent future asset data. Suppose $m_p = \int xp(x \mid x^0)dx$ is the predictive mean of future return given x^0, $V_x = (x - m_p)(x - m_p)'$, and $S_x = V_x \otimes (x - m_p)'$, then $X = (x, V_x, S_x)$ can be used to represent the moments of future asset returns.

Using a third order polynomial to describe the utility of future returns, the investor's predicted utility can be written as

$$U(w, X) = w'x - \lambda[w'(x - m_p)]^2 + \gamma[w'(x - m_p)]^3,$$

where w represents the vector of weights, λ and γ determine the impact of predictive variance and skewness respectively. Then the expected utility can be expressed as

$$E(U(w))$$
$$= w'E[x \mid x^0] - \lambda w'E[V_x \mid x^0]w + \gamma w'E[S_x \mid x^0]w \otimes w$$
$$= w'm_p - \lambda w'V_p w + \gamma w'S_p w \otimes w$$

where $\theta_p = (m_p, V_p, S_p)$ are the predictive moments of x.

The allocation problem for the investor maximizing his expected utility can be stated as

$$\max_w E(U(w)) \text{ under the constraint } w'\mathbf{1}=1 \text{ and } w \geq \mathbf{0}.$$

For the above model, the Bayesian framework will be used to estimate the expected utility via Markov Chain Monte Carlo simulation by generating a set of random samples $x_{t+1} \sim p(x_{t+1} \mid x^0)$ from the posterior predictive distribution of the Skew normal distribution. Then by averaging the utility over the entire sample space, we can obtain the estimated expected utility, i.e.

$$E(u(w)) = w'\overline{m} - \lambda w'\overline{V}w + \gamma w'\overline{S}w \otimes w, \qquad (1)$$

where $\overline{m}$, $\overline{V}$ and $\overline{S}$ are the posterior predicted means of θ.

Then the expected utility can be optimized as a function of the weights with different numerical approaches. Harvey et al. (2004) chose the Metropolis-Hastings based simulated annealing algorithm to explore the maximum expected utility.

The simulated annealing method was originated from a Metropolis Monte Carlo simulation process used by Kirkpatrick et al. (1983) to find the lowest energy of a system. In the experiment the system was heated to a high temperature at the beginning and then cooled slowly so that at each temperature the atoms can move towards the most stable orientation. The slow cooling process is known as annealing, and so the stochastic simulation process for the determination of the atomic organization with minimum energy is named as simulated annealing.

Similarly, if we want to maximize an objective function value, our simulated annealing optimization will start with a Metropolis Monte Carlo simulation at a low value of a global parameter, which will be gradually increased during the following process. This means that at the beginning large random jumps which will result in a increase in the objective function value will be accepted. At each step, the SA algorithm searches in the neighborhood of the current state s and probabilistically replaces the current solution according to the difference between the objective function values and a probability parameter. The probability needs to be updated at each step so that the system gradually moves to a higher value. The Metropolis Monte Carlo simulation is continued until the final temperature is reached or until a given computation budget limit is reached.

Kirkpatrick et al. (1983) argued that, if a global maximum exists, the simulated annealing algorithm will eventually escape from any local maximum and terminate with the global optimal solution as long as the annealing schedule is extended. However, the method's effectiveness will largely depend on the choices of the initial state space, the neighbor selection method, the probability transition function, and the annealing. And unfortunately, there is no general way to find the best choices for these parameters. The annealing time required to ensure a significant probability of success will be significantly higher than the time re-

quired for a complete search of the solution space. Kripka (2004) pointed out that "Regarding the number of function evaluations needed by Simulated Annealing, it must be emphasized that this number is very high when compared to those required by mathematical programming. Therefore, this approach is specially indicated to problems where usual techniques are not efficient."

3 COMPARISON WITH NONLINEAR PROGRAMMING OPTIMIZATION

Considering the significant annealing time, we proposed using mathematical programming skills which can be much more efficient for this optimization problem. For nonlinear programming optimization, the expected utility optimization model (1) can be expressed in the following way:

Maximize:

$$\sum_{i=1}^{p} m_i w_i - \lambda \sum_{1\le i,j\le p} V_{ij} w_i w_j + \gamma \sum_{1\le i,j,k\le p} S_{ijk} w_i w_j w_k$$

Subject to:

$$\sum_{i=1}^{p} w_i = 1;$$

$$w_i \ge 0 \qquad i = 1,2,\cdots, p;$$

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i;$$

$$V = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - m)^T (x_i - m)$$

$$S_{ijk} = \frac{1}{n-1}\sum_{l=1}^{n} (x_{li} - m_i)(x_{lj} - m_j)(x_{lk} - m_k)$$

$$1 \le i, j, k \le p$$

Harvey et al. (2004) used 71 months' daily returns on four equity securities. Here we considered daily returns on eight stocks (Microsoft Corporation, General Electric Company, Proctor & Gambler Company, American Express Inc, IBM Corporation, CISCO Systems Inc., Eli Lilly & Company, and McDonald's Corporation) from different sectors of Standard & Poor's 500 Index from the time period January 02, 2001 to December 30, 2005.

First we predicted 100 groups' future stock returns for all the equities using Bayesian estimation and the Gibbs sampling method according to Harvey et al (2004). Table 1 describes the future powers of the asset returns.

Table 1: The future powers of asset return

Mean of the returns (m)
m=[0.0044356085, 0.0028075352, 0.0042204711, 0.0015598400, 0.0042772951, 0.0041233167, 0.0002538442, 0.0064195938];
Variance-Covariance matrix of the returns (V)
V=0.0001*
[0.7250 0.4018 0.0843 0.3390 0.3066 0.5398 0.2146 0.2310 0.4018 0.5939 0.1714 0.4526 0.3198 0.4864 0.2934 0.2593 0.0843 0.1714 0.4271 0.1868 0.0218 0.1048 0.2147 0.1298 0.3390 0.4526 0.1868 0.5755 0.2802 0.3816 0.2919 0.2388 0.3066 0.3198 0.0218 0.2802 0.3867 0.4562 0.1554 0.1241 0.5398 0.4864 0.1048 0.3816 0.4562 1.2906 0.3064 0.2171 0.2146 0.2934 0.2147 0.2919 0.1554 0.3064 0.4161 0.1546 0.2310 0.2593 0.1298 0.2388 0.1241 0.2171 0.1546 0.4183]
Skewness and Coskewness of the returns (S)
S(:,:,1) =0.0001 *
[0.1265 0.0947 0.0494 0.0977 0.0680 0.0746 0.0680 0.0637 0.0947 0.0664 0.0398 0.0638 0.0504 0.0767 0.0522 0.0536 0.0494 0.0398 0.0387 0.0387 0.0407 0.0346 0.0370 0.0375 0.0977 0.0638 0.0387 0.0572 0.0531 0.0835 0.0460 0.0497 0.0680 0.0504 0.0407 0.0531 0.0375 0.0521 0.0454 0.0381 0.0746 0.0767 0.0346 0.0835 0.0521 0.0572 0.0566 0.0371 0.0680 0.0522 0.0370 0.0460 0.0454 0.0566 0.0426 0.0336 0.0637 0.0536 0.0375 0.0497 0.0381 0.0371 0.0336 0.0691]
S(:,:,2) =0.00001 *
[0.9468 0.6644 0.3981 0.6381 0.5035 0.7665 0.5222 0.5365 0.6644 0.4328 0.3155 0.4377 0.3749 0.7630 0.4805 0.4113 0.3981 0.3155 0.3366 0.3679 0.2904 0.3397 0.3688 0.2750 0.6381 0.4377 0.3679 0.4367 0.3213 0.6150 0.4680 0.4478 0.5035 0.3749 0.2904 0.3213 0.1574 0.4025 0.3909 0.3647 0.7665 0.7630 0.3397 0.6150 0.4025 0.7023 0.6308 0.3503 0.5222 0.4805 0.3688 0.4680 0.3909 0.6308 0.4844 0.3823 0.5365 0.4113 0.2750 0.4478 0.3647 0.3503 0.3823 0.4858]
S(:,:,3) =0.00001 *
[0.4944 0.3981 0.3866 0.3873 0.4071 0.3460 0.3700 0.3748 0.3981 0.3155 0.3366 0.3679 0.2904 0.3397 0.3688 0.2750 0.3866 0.3366 0.0179 0.4467 0.3399 0.3168 0.1952 0.2281 0.3873 0.3679 0.4467 0.4309 0.2824 0.3613 0.4882 0.3213 0.4071 0.2904 0.3399 0.2824 0.1745 0.3083 0.3031 0.2501 0.3460 0.3397 0.3168 0.3613 0.3083 0.6595 0.2415 0.2758 0.3700 0.3688 0.1952 0.4882 0.3031 0.2415 0.3166 0.2640 0.3748 0.2750 0.2281 0.3213 0.2501 0.2758 0.2640 0.2919]
S(:,:,4) =0.00001 *
[0.9767 0.6381 0.3873 0.5720 0.5305 0.8346 0.4602 0.4971 0.6381 0.4377 0.3679 0.4367 0.3213 0.6150 0.4680 0.4478 0.3873 0.3679 0.4467 0.4309 0.2824 0.3613 0.4882 0.3213 0.5720 0.4367 0.4309 0.4464 0.2235 0.3632 0.6122 0.3980 0.5305 0.3213 0.2824 0.2235 0.1376 0.3177 0.3412 0.2764 0.8346 0.6150 0.3613 0.3632 0.3177 0.7199 0.5362 0.3055 0.4602 0.4680 0.4882 0.6122 0.3412 0.5362 0.5341 0.3828 0.4971 0.4478 0.3213 0.3980 0.2764 0.3055 0.3828 0.4679]
S(:,:,5) =0.00001 *
[0.6795 0.5035 0.4071 0.5305 0.3754 0.5213 0.4542 0.3808 0.5035 0.3749 0.2904 0.3213 0.1574 0.4025 0.3909 0.3647 0.4071 0.2904 0.3399 0.2824 0.1745 0.3083 0.3031 0.2501 0.5305 0.3213 0.2824 0.2235 0.1376 0.3177 0.3412 0.2764 0.3754 0.1574 0.1745 0.1376 0.0837 0.1545 0.2334 0.2078 0.5213 0.4025 0.3083 0.3177 0.1545 0.0210 0.4761 0.2005 0.4542 0.3909 0.3031 0.3412 0.2334 0.4761 0.3532 0.2412 0.3808 0.3647 0.2501 0.2764 0.2078 0.2005 0.2412 0.3908]
S(:,:,6) =0.00001 *

[0.7455 0.7665 0.3460 0.8346 0.5213 0.5720 0.5660 0.3705 0.7665 0.7630 0.3397 0.6150 0.4025 0.7023 0.6308 0.3503 0.3460 0.3397 0.3168 0.3613 0.3083 0.6595 0.2415 0.2758 0.8346 0.6150 0.3613 0.3632 0.3177 0.7199 0.5362 0.3055 0.5213 0.4025 0.3083 0.3177 0.1545 0.0210 0.4761 0.2005 0.5720 0.7023 0.6595 0.7199 0.0210 -0.5334 0.6453 0.4612 0.5660 0.6308 0.2415 0.5362 0.4761 0.6453 0.4698 0.3150 0.3705 0.3503 0.2758 0.3055 0.2005 0.4612 0.3150 0.3654]
S(:,:,7) =0.00001 * [0.6799 0.5222 0.3700 0.4602 0.4542 0.5660 0.4263 0.3364 0.5222 0.4805 0.3688 0.4680 0.3909 0.6308 0.4844 0.3823 0.3700 0.3688 0.1952 0.4882 0.3031 0.2415 0.3166 0.2640 0.4602 0.4680 0.4882 0.6122 0.3412 0.5362 0.5341 0.3828 0.4542 0.3909 0.3031 0.3412 0.2334 0.4761 0.3532 0.2412 0.5660 0.6308 0.2415 0.5362 0.4761 0.6453 0.4698 0.3150 0.4263 0.4844 0.3166 0.5341 0.3532 0.4698 0.4516 0.3456 0.3364 0.3823 0.2640 0.3828 0.2412 0.3150 0.3456 0.3536]
S(:,:,8) =0.00001 * [0.6375 0.5365 0.3748 0.4971 0.3808 0.3705 0.3364 0.6906 0.5365 0.4113 0.2750 0.4478 0.3647 0.3503 0.3823 0.4858 0.3748 0.2750 0.2281 0.3213 0.2501 0.2758 0.2640 0.2919 0.4971 0.4478 0.3213 0.3980 0.2764 0.3055 0.3828 0.4679 0.3808 0.3647 0.2501 0.2764 0.2078 0.2005 0.2412 0.3908 0.3705 0.3503 0.2758 0.3055 0.2005 0.4612 0.3150 0.3654 0.3364 0.3823 0.2640 0.3828 0.2412 0.3150 0.3456 0.3536 0.6906 0.4858 0.2919 0.4679 0.3908 0.3654 0.3536 0.6089]

Based on the predicted data, we optimize the expected utility by both the Simulated Annealing algorithm and nonlinear programming optimization using MatlabR2006b. For the purpose of nonlinear programming optimization, we use the Matlab function *fmincon* provided by Matlab optimization toolbox. For simulated annealing, we use the Matlab function *anneal* provided by Vandekerckhove (2006) based on the simulated annealing method proposed by Kirkpatrick et al. (1983).

We investigate the CPU time of both the simulated annealing algorithm and the nonlinear programming algorithm on a computer with Intel(R) Pentium(R) M processor 1.73 GHz with 0.99 GB of RAM. In the simulation study, we let λ and γ take values 0, 1, 2, 5, 10, 20, 30, 50, and 100. In each simulation, we run all 81 combinations and record the total CPU time. For the following results for both algorithms, we set the stopping tolerance to be 10^{-12}.

To keep the playing field level, we use two starting strategies for both algorithms. First, we use the equal weight portfolio as starting point and then we use the Markowitz mean-variance optimal portfolio returned by quadratic optimization as a starting point. We find that for both algorithms to achieve the same level of accuracy the total CPU time will be slightly reduced if we use the solution returned by starting with the Markowitz mean-variance optimal solution. We use this staring solution for the comparisons below, but note that similar results were obtained starting with the equal weight portfolio..

To account for differences in CPU time due to different random seeds, and also to reduce overall variation we run each of the 81 optimizations twenty different times for a total of 1620 separate optimizations. For the nonlinear optimization routine the total CPU time of these 20 simulations is 206.8750 seconds. The average CPU time of each simulation then is 10.34375 seconds for 81 optimizations (an average time for each optimization of 0.1277 seconds). Our standard for comparison is the average time to perform 20 simulations (10.34375 seconds). The results of the averaged portfolio weights and utility can be found in Table 2.

There are four major parameters for the SA algorithm: cooling schedule, maximum number of consecutive rejections (Maxconsrej), maximum number of tries within one temperature (Maxtries), and maximum number of successes within one temperature (Maxsucess). We also consider tolerance as a fifth parameter. We study the effect of each of them on optimization.

1. First, we set Maxconsrej=300, Maxtries=200, Maxsucess=600 and tolerance= 10^{-12}. Usually, practitioners choose a cooling schedule rate between 0.8 and 0.99. Here, to thoroughly study the effect of cooling schedule on the simulation time, we take three values for the rate: 0.01, 0.4, and 0.8. We record the average CPU time of 20 simulations which includes 1620 optimizations. Then we calculate the mean L_1 distance of the optimized utility returned by simulated annealing to the optimal solution (When the four parameters of simulated annealing are large enough, the output returned by simulated annealing will equal the output of nonlinear programming, contained in table 2. We consider this output to be the optimal solution.). The distance is an indicator of the accuracy of the algorithm. It can be seen from Table 3 that the CPU time can be dramatically decreased with the decrease of the cooling schedule rate, this is not done without penalty however. When the cooling rate is below 0.01, the SA algorithm returns a solution that is further from the exact solution. So there is a tradeoff between the accuracy and the speed of the algorithm.
2. Next we examine the effect of Maxsucess. We fix cooling rate=0.8, Maxconsrej=300, Maxtries=200 and tolerance=10^{-12}. Then we try four different Maxsucess rates: 20, 100, 600, and 1000. We record the average CPU time of 20 simulations and the mean L_1 distance of the simulated utility to the optimal solution. We see in Table 4 that with large values of Maxsucess, the SA algorithm is more accurate, but it is also slower.

Table 2: Optimized stock weights and portfolio utility

lambda (λ)	Gamma (γ)	MSFT	GE	PG	AXP	IBM	CSCO	LLY	MCD	Portfolio utility
0	0	0	0	0	0	0	0	0	1	0.00641959380000
0	1	0	0	0	0	0	0	0	1	0.00641959380000
0	2	0	0	0	0	0	0	0	1	0.00641959380000
0	5	0	0	0	0	0	0	0	1	0.00641959380000
0	10	0	0	0	0	0	0	0	1	0.00641959380000
0	20	0	0	0	0	0	0	0	1	0.00641959380000
0	30	0	0	0	0	0	0	0	1	0.00641959380000
0	50	0	0	0	0	0	0	0	1	0.00641959380000
0	100	0	0	0	0	0	0	0	1	0.00641959380000
1	0	0	0	0	0	0	0	0	1	0.00600741937100
1	1	0	0	0	0	0	0	0	1	0.00600741937100
1	2	0	0	0	0	0	0	0	1	0.00600741937100
1	5	0	0	0	0	0	0	0	1	0.00600741937100
1	10	0	0	0	0	0	0	0	1	0.00600741937100
1	20	0	0	0	0	0	0	0	1	0.00600741937100
1	30	0	0	0	0	0	0	0	1	0.00600741937100
1	50	0	0	0	0	0	0	0	1	0.00600741937100
1	100	0	0	0	0	0	0	0	1	0.00600741937100
2	0	0	0	0	0	0	0	0	1	0.00559524494200
2	1	0	0	0	0	0	0	0	1	0.00559524494200
2	2	0	0	0	0	0	0	0	1	0.00559524494200
2	5	0	0	0	0	0	0	0	1	0.00559524494200
2	10	0	0	0	0	0	0	0	1	0.00559524494200
2	20	0	0	0	0	0	0	0	1	0.00559524494200
2	30	0	0	0	0	0	0	0	1	0.00559524494200
2	50	0	0	0	0	0	0	0	1	0.00559524494200
2	100	0	0	0	0	0	0	0	1	0.00559524494200
5	0	0	0	0.0799	0	0.1168	0	0	0.8033	0.00442596501176
5	1	0	0	0.0789	0	0.1162	0	0	0.8049	0.00442601407581
5	2	0	0	0.0779	0	0.1155	0	0	0.8066	0.00442605248861
5	5	0	0	0.0748	0	0.1136	0	0	0.8117	0.00442610249901
5	10	0	0	0.0694	0	0.1103	0	0	0.8203	0.00442595768182
5	20	0	0	0.0582	0	0.1035	0	0	0.8383	0.00442472691379
5	30	0	0	0.0463	0	0.0965	0	0	0.8573	0.00442205705212
5	50	0	0	0.0197	0	0.0816	0	0	0.8987	0.00441129591672
5	100	0.0045	0	0	0	0.0235	0	0	0.972	0.00437424957962
10	0	0	0	0.2215	0	0.2618	0	0	0.5167	0.00315463371562
10	1	0	0	0.2212	0	0.2616	0	0	0.5172	0.00315465411217
10	2	0	0	0.221	0	0.2613	0	0	0.5177	0.00315467243508
10	5	0	0	0.2201	0	0.2605	0	0	0.5193	0.00315471464495
10	10	0	0	0.2187	0	0.2593	0	0	0.522	0.00315474198163
10	20	0	0	0.2158	0	0.2567	0	0	0.5275	0.00315462923056
10	30	0	0	0.2129	0	0.254	0	0	0.5331	0.00315428164018
10	50	0	0	0.2067	0	0.2487	0	0	0.5446	0.00315282070548
10	100	0	0	0.1896	0	0.2347	0	0	0.5757	0.00314384943707
20	0	0	0	0.2923	0	0.3343	0	0	0.3733	0.00108316982260
20	1	0	0	0.2922	0	0.3342	0	0	0.3735	0.00108318167289
20	2	0	0	0.2921	0	0.3341	0	0	0.3737	0.00108319292975
20	5	0	0	0.2919	0	0.3338	0	0	0.3743	0.00108322311810
20	10	0	0	0.2914	0	0.3332	0	0	0.3754	0.00108326143606
20	20	0	0	0.2905	0	0.3321	0	0	0.3774	0.00108329231962
20	30	0	0	0.2896	0	0.331	0	0	0.3794	0.00108326097139
20	50	0	0	0.2877	0	0.3288	0	0	0.3836	0.00108300485022
20	100	0	0	0.2827	0	0.3231	0	0	0.3942	0.00108115492080
30	0	0	0	0.3159	0	0.3585	0	0	0.3256	-0.00088368753236
30	1	0	0	0.3159	0	0.3584	0	0	0.3257	-0.00088367784293
30	2	0	0	0.3158	0	0.3584	0	0	0.3258	-0.00088366847486
30	5	0	0	0.3157	0	0.3582	0	0	0.3262	-0.00088364230271
30	10	0	0	0.3154	0	0.3578	0	0	0.3268	-0.00088360515308
30	20	0	0	0.3149	0	0.3571	0	0	0.328	-0.00088355534816
30	30	0	0	0.3144	0	0.3564	0	0	0.3292	-0.00088353863013
30	50	0	0	0.3134	0	0.355	0	0	0.3316	-0.00088360655607
30	100	0	0	0.3107	0	0.3515	0	0	0.3379	-0.00088439339315
50	0	0	0	0.3348	0	0.3779	0	0	0.2874	-0.00475466835892
50	1	0	0	0.3348	0	0.3778	0	0	0.2874	-0.00475466018011
50	2	0	0	0.3347	0	0.3778	0	0	0.2875	-0.00475465216325

50	5	0	0	0.3347	0	0.3777	0	0	0.2877	-0.00475462908294
50	10	0	0	0.3345	0	0.3775	0	0	0.288	-0.00475459386514
50	20	0	0	0.3343	0	0.377	0	0	0.2887	-0.00475453567424
50	30	0	0	0.334	0	0.3766	0	0	0.2893	-0.00475449392886
50	50	0	0	0.3335	0	0.3758	0	0	0.2907	-0.00475446035422
50	100	0	0	0.3322	0	0.3738	0	0	0.294	-0.00475467453487
100	0	0	0	0.3489	0	0.3924	0	0	0.2587	-0.01435894842110
100	1	0	0	0.3489	0	0.3923	0	0	0.2587	-0.01435894125600
100	2	0	0	0.3489	0	0.3923	0	0	0.2588	-0.01435893415980
100	5	0	0	0.3489	0	0.3923	0	0	0.2588	-0.01435891329831
100	10	0	0	0.3488	0	0.3922	0	0	0.259	-0.01435887994092
100	20	0	0	0.3487	0	0.392	0	0	0.2593	-0.01435881854446
100	30	0	0	0.3486	0	0.3918	0	0	0.2596	-0.01435876425831
100	50	0	0	0.3484	0	0.3914	0	0	0.2602	-0.01435867714053
100	100	0	0	0.3478	0	0.3904	0	0	0.2618	-0.01435858586998

Table 3: Simulation results for different cooling rate

Cooling rate	Average Total CPU time (20 simulations) in seconds	Average distance to the optimal utility
0.01	27.3281	0.0388
0.4	93.2031	0.0037
0.8	207.2906	0

Table 4: Simulation results for different Maxsuccess

Maxsuccess	Average Total CPU time (20 simulations) in seconds	Average distance to the optimal utility
20	59.5141	0.007412
100	128.5100	0.000796
600	209.3208	0
1000	277.9688	0

3. Next we study the effect of Maxtries. We fix cooling rate=0.8, Maxconsrej=300, Maxsucess=600 and tolerance=10^{-12}. Then we use four different Maxtries: 100, 200, 300 and 500. We see in Table 5 that with increasing Maxtries, the accuracy of the algorithm can be improved, but the CPU time will be dramatically increased also. So it is important for practitioners to choose an appropriate rate for Maxtires in real applications.

Table 5: Simulation results for different Maxtries

Maxtries	Average Total CPU time (20 simulations) in seconds	Average distance to the optimal utility
100	123.4727	0.000809
200	202.2656	0
300	298.8750	0
500	447.6561	0

4. At last, we study the effect of Maxconsrej. We set cooling rate=0.8, Maxtries=200, Maxsucess=600 and tolerance=10^{-12}. Four different Maxconsrej are used: 100, 300, 1000, and 2000. Table 6 shows that small Maxconsrej can lead to inaccurate solutions.

Table 6: Simulation results for different Maxconsrej

Maxconsrej	Average Total CPU time (20 simulations) in seconds	Average distance to the optimal utility
100	204.0938	0.000074
300	210.0901	0
1000	228.2301	0
2000	248.5635	0

5. To determine the effect of tolerance on simulation time we fix the parameters to be cooling rate=0.8, Maxconsrej=300, Maxsucess=600, and Maxtries = 200. Three different tolerances are used: 10^{-6}, 10^{-9}, and 10^{-12}. The results can be seen in Table 7. As expected the larger the tolerance the faster the algorithms, but these differences are not computationally expensive considering the average time per optimization (81 per simulation).

Table 7: Simulation results for different tolerance

Tolerance Level	Average Total CPU time for SA (20 simulations) in seconds	Average Total CPU time for NLP (20 simulations) in seconds
10^{-6}	178.0313	7.7898
10^{-9}	197.6797	8.7531
10^{-12}	207.2906	10.34375

4 CONCLUSIONS

We have presented an empirical comparison between simulated annealing and nonlinear programming optimization to a Bayesian portfolio selection problem. We have shown that to achieve the same level of accuracy, simulated annealing will take a longer time than nonlinear programming optimization. However, even though simulated annealing is slower, for the purpose of optimizing a portfolio with a particular lambda and gamma, this difference is not sufficiently large to warrant abandoning the simulated annealing algorithm. We have made this comparison by maximizing a cubic utility function for both methods, and have found that both algorithms can achieve the optimal expected utility within the same level of tolerance. By establishing the relationship between these two methods for a cubic objective function suggests that we may be able to obtain accurate and timely results when optimizing a function that nonlinear programming may not be suited for by using simulated annealing.

REFERENCES

Azzalini, A., and Dalla Valle, A. 1996. The Multivariate Skew-Normal Distribution. *Biometrika* 83:715-726.

Fama, E. F. 1970. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance* 25: 383-417.

Gelfand, A., and Smith, A. 1990. Sampling-based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association* 85: 398-409.

Harvey, C. R., J. C. Liechty, M. W. Liechty, and P. M¨uller. 2004. Portfolio Selection with Higher Moments: A Bayesian Decision Theoretic Approach. *Working Paper*, Duke University Durham.

Harvey, C. R., and Siddique, A. 2000. Time-Varying Conditional Skewness and the Market Risk Premium. *Research in Banking and Finance* 1: 27-60.

Hastings, W. K. 1970. Monte Calo Sampling Methods Using Markov Chains and their Applications. *Biometrika* 57: 97-109.

Kirkpatrick, S., C. D. Gelatt Jr. and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220: 671-680.

Kripka, M. 2004. Discrete Optimization of Trusses by Simulated Annealing. *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 26: 170-173.

Markowitz, H. 1952. Mean-variance analysis in portfolio choice and capital markets. *Journal of Finance* 7: 77-91.

Merton, R. 1969. Lifetime Portfolio Selection Under Uncertainty: the Continuous Time Case. *The Review of Economics and Statistics* 51: 847-857.

Merton, R. 1971. Optimum Consumption and Portfolio Rules in a Continuous Time Model. *Journal of Economic Theory* 3: 373-413.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E.Teller. 1953. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* 21: 1087-1091.

Samuelson, P. A. 1969. Lifetime Portfolio Selection by Dynamic Stochastic Programming. *The Review of Economics and Statistics* 51: 239-246.

Vandekerckhove 2006. General simulated annealing algorithm. Available via <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10548&objectType=file> [accessed June 20, 2007].

AUTHOR BIOGRAPHIES

MERRILL LIECHTY is an assistant professor in the Decision Sciences Department of LeBow College of Business at Drexel University. His research interest lies in the area of Bayesian statistics, portfolio selection and higher moment estimation. His web page can be found via <http://www.pages.drexel.edu/~mwl25>.

JINGJING LU is a Ph.D. student in the LeBow College of Business at Drexel University. Her research interest lies in the area of operation management; financial engineering; risk management and quantitative methods in management fields; time series analysis. Her web page can be found via <http://www.lebow.drexel.edu/Current/Doctorate/>