Transaction Cost Function Minimization Using Simulated Annealing and Smoothing

by

Yichen Zhang

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Thomas F. Coleman

Waterloo, Ontario, Canada, 2014

Abstract

Transaction cost function minimization is an important problem in finance. In this essay, we develop and apply a simulated annealing and smoothing method to this particular problem. We illustrate that this method is an improvement over using the trust-region method or simulated annealing algorithm alone. We will provide examples in different dimensions and using different parameter settings to illustrate the advantage of using a new approach that combines simulated annealing and smoothing.

Acknowledgements

I would like to thank my supervisor, Professor Thomas F. Coleman, for his support and guidance. I would also like to thank each individual person I have met and every single day of my life to make this happen.

Dedication

This is dedicated to my family and my friends I love.

Table of Contents

List of Figures

Chapter 1

Introduction

The Mean-Variance (M-V) portfolio selection problem can be solved by quadratic programming and parametric quadratic programming (PQP) method as discussed, for example, in [5]. The problem can be formulated as:

$$\min \left\{ -t\mu' x + \frac{1}{2} x' C x : l' x = 1, Ax \leq b \right\} \tag{1.1}$$

where μ is an n-vectors of expected returns, C is an $n \times n$ positive semi-definite covariance matrix, x is an asset holdings to be determined, l is an n-vector of $1's$, A is an $m \times n$ matrix, b is an m-vector, and t is non-negative. The constraint $l'x = 1$ is called the budget constraint, which requires the asset holdings to sum to unity. The constraints $Ax \leq b$ represent general linear constraints such as non-negative constraints, upper bounds or lower bounds on asset holdings, sector constraints, and other linear constraints the investor may wish to impose.

In problem (1.1), the quantities $\mu_p = \mu' x$ and $\sigma_p^2 = x' C x$ are defined as the expected return and variance of the portfolio return respectively. The parameter t represents a particular investor's aversion to risk.

Transaction costs can arise when an asset is sold or bought, and practical portfolio optimization requires this transaction costs be included as part of the problem. There are two common types of transaction cost: piecewise linear concave and piecewise constant, illustrated in figure 1. Transaction costs are often relatively large when the amount of transaction is smaller and increase with a small rate. Therefore the transaction cost function is concave. Piecewise linear concave transaction costs are popular in many mathematical models so we use this particular type in our model.

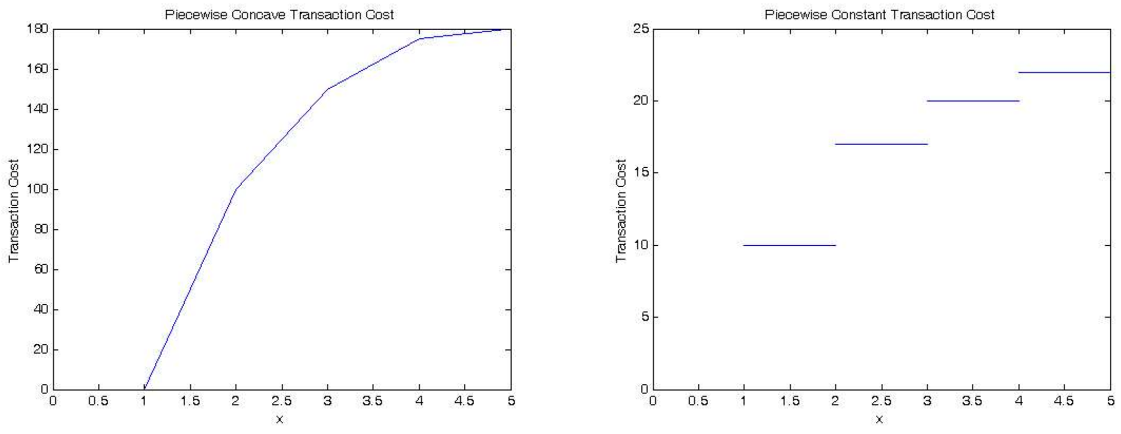

Figure 1.1: 2 Types of Transaction Cost.

Now we can restate the problem (1.1) as

$$\min\left\{-t\mu' x+\frac{1}{2}x'Cx+\sum_{i=1}^{n}Tr_i(x_i) : l'x=1, Ax\le b\right\} \tag{1.2}$$

where $\sum_{i=1}^{n} Tr_i(x_i)$ represents the transaction costs. Since in most cases, we only care about the boundary constraints instead of other equality or inequality constraints. Therefore in this essay, we simplify the problem as:

$$\min\left\{\frac{1}{2}x'Qx+gx+\sum_{i=1}^{n}Tr_i(x_i) : lb\le x\le ub\right\} \tag{1.3}$$

where Q represents the covariance matrix and g represents vector $-t\mu'$ in problem (1.2). We define this equation as the transaction cost function.

We first consider two standard methods that can be applied to this particular optimization problem: Trust-Region Method and Simulated Annealing Method.

A trust-region method defines a region around the current point within which the model is trusted to be an adequate representation of the objective function, and then solve the minimization problem of the model in this region. If the step is not acceptable, which means the objective function value becomes higher than before, the trust-region method reduces the region size and then resolves the minimization problem. If the step reaches the boundary and the result is good, the trust-region method increases the region size for a possible larger step in the next iteration. Therefore, the step size is critical for trust-region method. This method has been proved to be a powerful approach to optimization because it requires fewer iterations than other methods and also has relatively strong convergence properties. Previous trust-region experiments and results can be found in [10] [20] and [21].

The simulated annealing method is designed to deal with the problems that have many local optima. Compared to the traditional iteration algorithms for optimization problem which can only accept a downhill move (i.e., accept the point if the objective function value becoming smaller), simulated annealing algorithm can accept an uphill move (i.e., accept the point which makes objective function value greater) with a probability that depends on a "temperature" parameter. A trust-region method is such a downhill method. Therefore, the simulated annealing method can escape from local optima in its attempt to get closer to the global optima. Theoretically, simulated annealing can find the global optimum if we set up the parameters properly (statistically speaking, in infinite time). Discussion about simulated annealing and its parameters can be found in [4], [17] [24] and [26]. More details are in chapter 3.

In this essay, we illustrate the disadvantages of these two methods when dealing with transaction cost function minimization problem. We add a smoothing technique, which depends on a single parameter, to handle the transaction costs in a simulated annealing framework. The new method is called "Simulated Annealing with Smoothing". It is efficient and effective to solve the transaction cost function optimization problem by our new method.

Chapter 2

Transaction Cost Function Minimization

In this chapter, we first give a description of the transaction cost function and some basic definitions. Then we show how we handle the transaction cost function's kink points, at which the function is not differentiable. Last, we state why it is hard to minimize the transaction cost function using existing continuous optimization software, such as fmincon and fminunc in MATLAB.

2.1 Problem Description

In this essay, the problem we solve is:

$$\min \left\{ \frac{1}{2}x'Qx + gx + \sum_{i=1}^{n} Tr_i(x_i) : lb \leq x \leq ub \right\} \tag{2.1}$$

where Q is positive definite matrix represents the covariance matrix. lb is the lower bound and ub is the upper bound for x. $\sum_{i=1}^{n} Tr_i(x_i)$ is the piecewise transaction costs defined as following:

There are two sets of interest: the set of the kink points and the set of slopes between the kink points.

- $knega_m < knega_{m-1} < ... < knega_1 < 0 < kposi_1 < ... < kposi_{l-1} < kposi_l$ are kink points.

- $mnega_1 < mnega_2 < ... < mnega_m < 0 < mposi_l < ... < mposi_2 < mposi_1$ are the slope, notice that $mnega_i$ is the slope of the line between $knega_i$ and $knega_{i+1}$ and $mposi_k$ is the slope of the line between $kposi_k$ and $kposi_{k+1}$.
- The transaction costs is a straight line with the slope of $mnega_m$ after $knega_m$ as x goes to $-\infty$ and a straight line with the slope of $mposi_l$ after $kposi_l$ as x goes to ∞

Here we give the graph of transaction costs in one dimension to illustrate the definition.

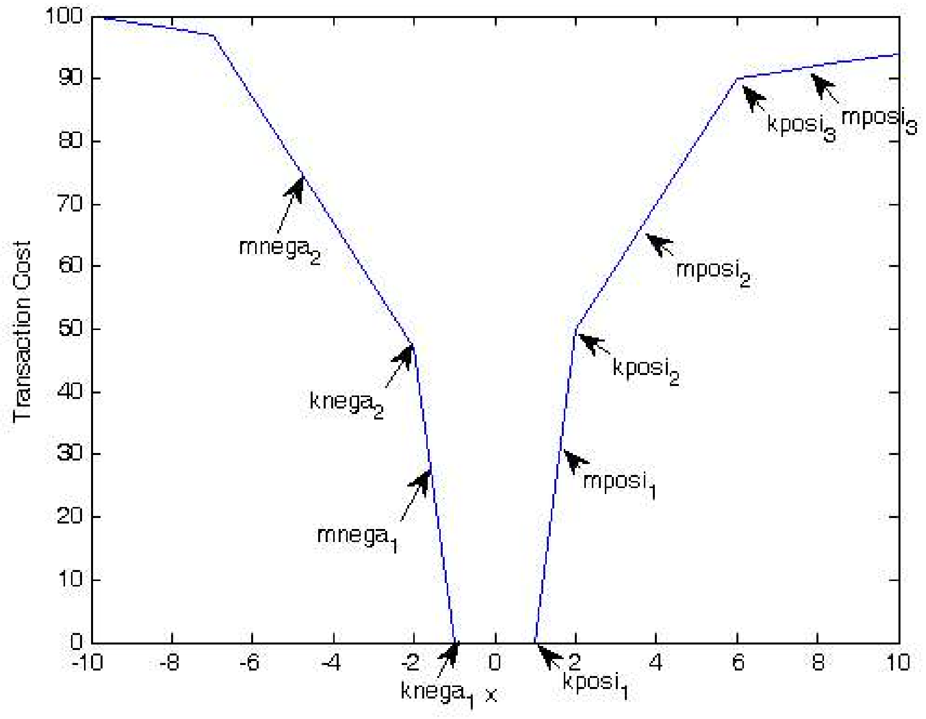

Figure 2.1: Transaction Costs for One Dimension.

It is reasonable for transaction costs to have this shape because often in the real market, the more you buy a product, the lower it's unit price will be. It is also true for the sale situation. That is the reason why the slope of transaction costs for each dimension has the property: $mnega_1 < mnega_2 < ... < mnega_m < 0 < mposi_l < ... < mposi_2 < mposi_1$.

Note: the introduction of the transaction costs have changed an (easy) convex minimization problem into a (hard) nonconvex problem with many local minima.

The transaction cost function is not differentiable at the kink points and is piecewise continuous. We need a way to approximate the first and second derivative to use trust-region method to minimize transaction cost function. We will discuss this in section (2.2).

Here we give the graph of function (2.1) for n=2.

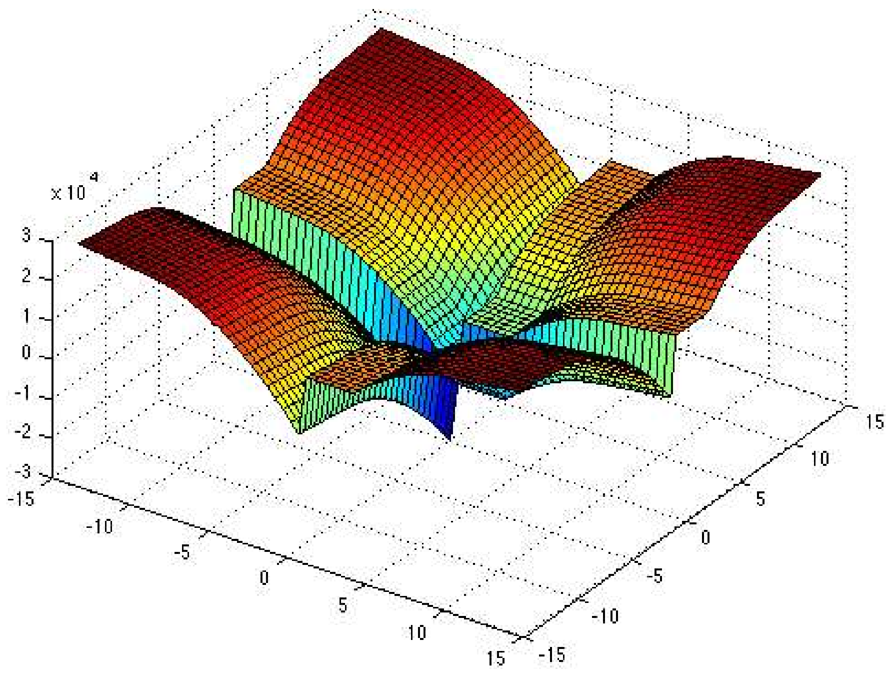

Figure 2.2: 2 Dimensions Transaction Cost Function.

As shown in the graph, we see there is a plat in this small region which includes several local minima and the graph looks very nasty. After we smooth the kink points at 2.2, we may have many local minima in the feasible region. Therefore, it is difficult to minimize such a problem using the trust-region approach. Trust-region finds a local minimum which may be far away from the global minima. However, if we want to try a trust-region approach, the first thing is to find a way to get the gradient and Hessian matrix of the transaction cost function.

2.2 Smoothing Around the Kink Point

A popular way to solve problem (2.1) is using a trust-region method. This method needs the first and second derivative of the objective function, i.e., the gradient and Hessian matrix. Since the kink points of the transaction cost function are not differentiable, the trust-region approach is not directly applicable. Therefore we need a way to smooth the transaction function around the kink points.

We can smooth the kink points as follows. We define $TrS_i(x)$ to be the smooth function for transaction costs $Tr_i(x)$. For each dimension, given the parameter ϵ.

- If $x \notin [k_i - \epsilon, k_i + \epsilon]$, where k_i is the kink point, then $TrS_i(x) = Tr_i(x)$.
- At $k_i - \epsilon$ and $k_i + \epsilon$ two points, the $TrS_i(x)$ has the same function value and first derivative value as $Tr_i(x)$.
- If $x \in [k_i - \epsilon, k_i + \epsilon]$ where k_i is the kink point, $TrS_i(x)$ is a smooth curvature which can approximate the original transaction cost function's value. The curvature can be generated from previous work.

In this way, we can get the approximation of the first and second derivative of the transaction cost function everywhere. After that, we can use the MATLAB's tool-box fmincon to minimize the transaction cost function with trust-region algorithm.

Figure 2.2 show how we smooth around the kink point.

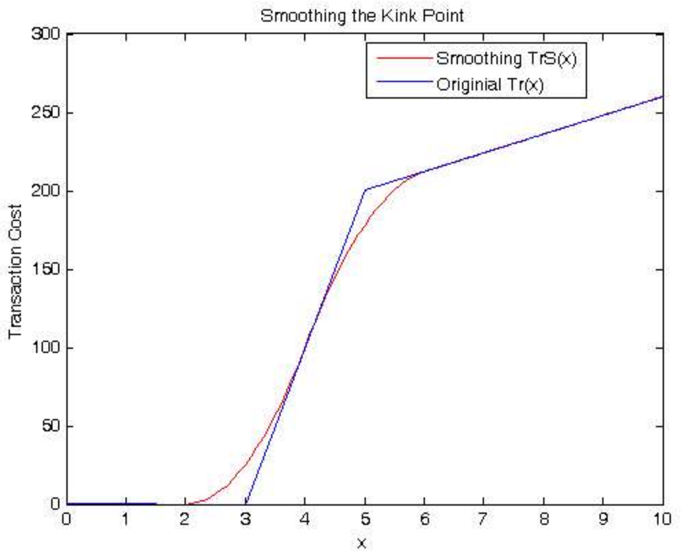

Figure 2.3: Smoothing Around the Kink Points.

2.3 The Problem of Minimizing the Transaction Cost Function

Since the modified transaction cost function is nasty and has many local minima, it is hard to find a sufficiently good solution using the standard local minimization software (such as MATLAB's fmincon or fminunc by trust-region method) because such methods will probably stop at some local minimum which is far away from the global minimum.

Now, we consider tools and methods to improve on an arbitrary local solution.

Chapter 3

Simulated Annealing and Smoothing

In this chapter, we will first give a description of the simulated annealing algorithm and then describe how we smooth the transaction cost function. After adding the smoothing part to the traditional simulated annealing algorithm, it becomes efficient to solve the optimization problem of (2.1). Then, we give numerical results to illustrate that simulated annealing with smoothing can do a better job than fmincon in MATLAB (i.e., the trust-region method).

3.1 Simulated Annealing Algorithm

Simulated annealing is a global search algorithm which can escape from local optima. This algorithm is easy to implement and has a probability to accept the uphill move (i.e., accept the point even it makes objective function value higher). These features make simulated annealing algorithm possible to find the global optima, ideally, in infinite time! It becomes a popular technique over past two decades. It can be used for discrete and continuous optimization problems.

Simulated annealing algorithm is typically designed to solve non-convex problems or problems that have many local optima. It can give us a better solution than purely local techniques. Unlike the traditional iteration algorithm which only accept the downhill move, simulated annealing allows perturbation to move uphill in a certain way. The advantage to accept uphill move is we may escape from the local optima and find a better answer. Traditional algorithms for solving optimization problems may be trapped in the region near the start point and can not escape from it.

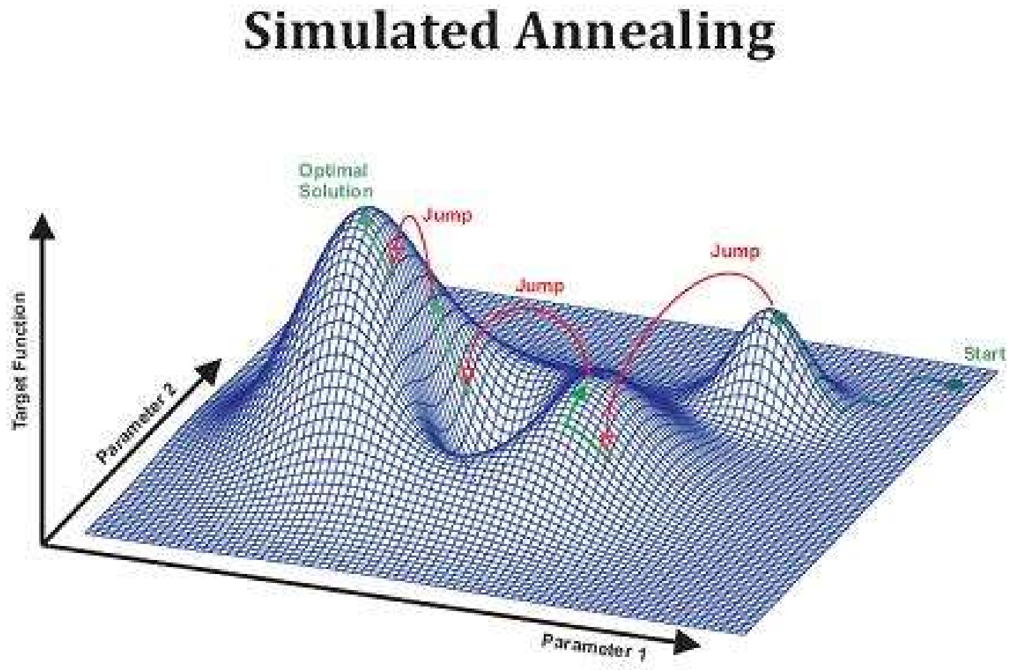

Figure 3.1: Simulated Annealing Jump Example.

There are several key parameters in simulated annealing: temperature sequence, trials for each temperature and so on. Temperature sequence contains a number of temperature from high to low, and it is used to control the probability of moving uphill. For higher temperature, the probability of large uphill move is large. For lower temperature, the probability of uphill move is small. For each temperature, we always accept the downhill move. Trials for each temperature is the number of neighbourhood points we try at each temperature, the way to choose the neighbourhood point can be modified depends on the problem we solve.

Simulated annealing algorithm can be described as this: give the temperature sequence and trials and the way to find the neighbourhood point. Then for each temperature, try different neighbourhood point for number of trials times. If the objective function value is lowered, accept it. If the objective function value is raised, accept it with a probability depends on the temperature. Details of the simulated annealing algorithm will be given later.

There are several ways to decide the rule of accepting uphill move and the neighbourhood point. Most of the uphill accept rules are based on the difference between the new value and old value of the objective function Δf. We use the rule called Metropolis distribution: the probability of an uphill move of size Δf at temperature T is: $Pr[accept] = e^{\frac{\Delta f}{CT}}$ and the way to find the neighbourhood point is: $x_{new} = x_{old} + (2 \cdot rand) \cdot sr - sr$ where sr is defined by users based on the problem.

It is clear that for the high temperature, the probability of accepting uphill move is large. As temperature cools down, the probability of allowing uphill move is becomes smaller and smaller. We stop the search when the objective function value is flat enough for several temperatures.

Notice that the choice of temperature sequence and trials for each temperature and the way to choose neighbourhood point can be modified based on the problem. Different problems may have different parameters. We should also consider the time cost for each parameter since we want the simulated annealing algorithm provides the final result in a reasonable time. It is worth noting that the probability of accepting the uphill move is based on the temperature for each trial, the final answer may not be the same even if we solve for the same problem.

At each temperature, the structure is shown in algorithm 1.

Algorithm 1 Simulated Annealing at Temperature T

```
$M$ = number of moves to attempt, $T$= current temperature, $C$ is a constant.
for $m = 1$ to $M$ do
    Generate a new neighbouring solution, evaluate $f_{new}$.
    if $f_{new} < f_{old}$ then
        (downhill move: accept it)
        Accept this new solution, and update the solution.
    else
        (uphill move: accept maybe)
        Accept with probability $P(T) \leftarrow e^{\frac{-(fnew - fold)}{C \cdot T}}$.
        Update the solution if accepted.
    end if
end for
```

The whole simulated annealing algorithm can be described as follow:

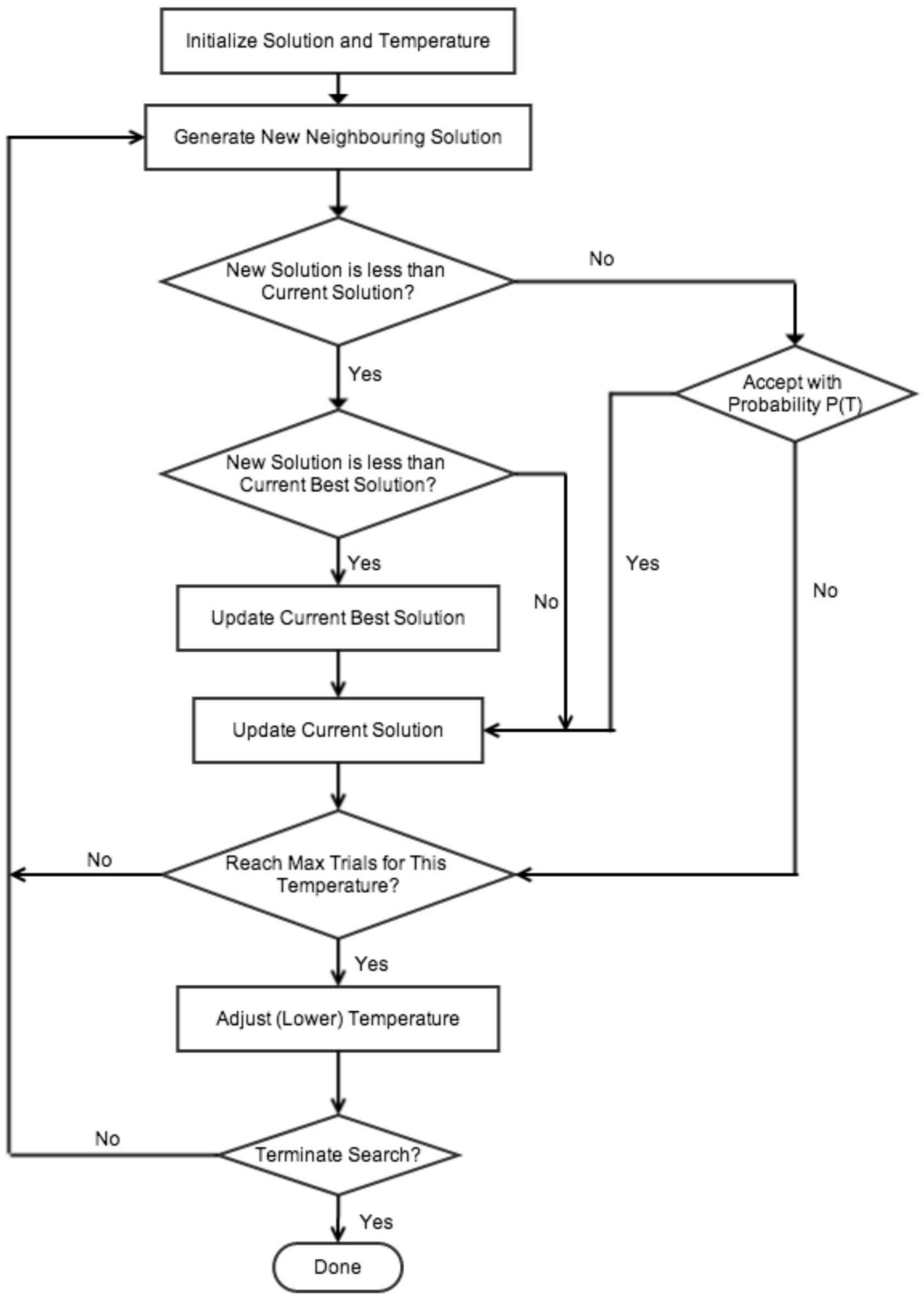
Initialize Solution and Temperature
Generate New Neighbouring Solution
New Solution is less than Current Solution?
No
Yes
Accept with Probability P(T)
New Solution is less than Current Best Solution?
Yes
Yes
No
No
Update Current Best Solution
Update Current Solution
No
Reach Max Trials for This Temperature?
Yes
Adjust (Lower) Temperature
No
Terminate Search?
Yes
Done

Figure 3.2: Simulated Annealing Algorithm.

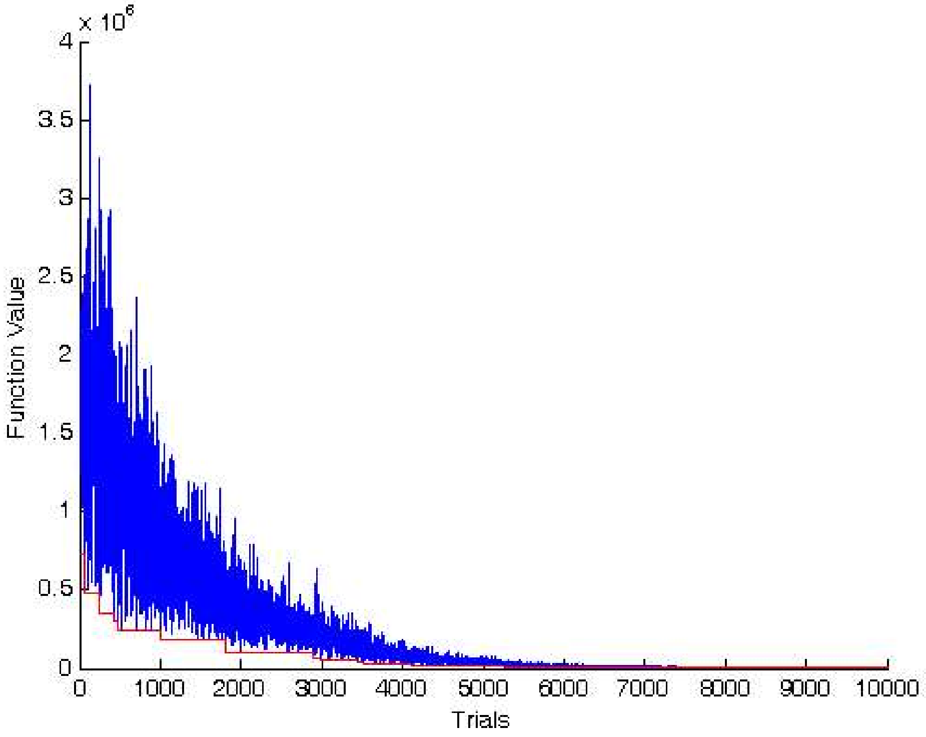

Figure 3.3: Simulated Annealing Tracking Plot.

We give an example on how simulated annealing works on our problem. The vertical bars show the objective function value we have reached at each temperature and the horizontal lines are the global minimum we have stored.

After introducing the simulated annealing algorithm, we find it is efficient to use this method to minimize a problem which has many local minima. It has a probability of "jumping" out of the local minimum zone near the initial point to get a better solution. Previous work and examples described in [1] [3] [8] [19] and [23] show that simulated annealing is a good method to optimize the problems that have many local minima. Also, simulated annealing can be combined into many other algorithm to make the algorithm more efficient for particular problems.

3.2 Smoothing Sequence

Simulated annealing alone is not enough to minimize the transaction cost function efficiently. After doing several experiments at 2 dimensions and 10 dimensions, we find that it takes an unacceptably long time to get the final answer because we have to set the parameters (such as temperature and trials for each temperature) large enough to get a satisfactory result. For 2 dimensions and 10 dimensions problems, simulated annealing can cost 10 minutes to get the final result while trust-region method takes a few seconds. So it would take several hours to get the results if we want to solve 30 or 40 dimensions problems. Therefore, traditional simulated annealing method may not be suitable for transaction problems. So we seek to find another way to make simulated annealing more efficient.

Our purpose is to add some technique to let the simulated annealing get a satisfactory result in a reasonable time under the same parameters. It may take a bit more time after we add the technique but we think it is worth to do it.

Notice that in problem (2.1), matrix Q is a positive definite quadratic. Considering it is easy and efficient to solve an optimization problem for a convex function, we add a smoothing part in simulated annealing method. First, we approximate our objective function by a convex function and solve the optimization problem. Then we made the function closer to the original transaction cost function and solve the optimization problem using the information we get from the last step. Finally, we solve the original function using the information obtained from the previous step.

A key observation is that the degree of smoothing can be controlled by adjusting a single parameter.

Recall that for transaction cost part, we have two sets of interests:

- $knega_m < knega_{m-1} < ... < knega_1 < 0 < kposi_1 < ... < kposi_{l-1} < kposi_l$ are kink points.
- $mnega_1 < mnega_2 < ... < mnega_m < 0 < mposi_l < ... < mposi_2 < mposi_1$ are the slope, notice that $mnega_i$ is the slope of the line between $knega_i$ and $knega_{i+1}$ and $mposi_k$ is the slope of the line between $kposi_k$ and $kposi_{k+1}$.
- The transaction costs is a straight line with the slope of $mnega_m$ after $knega_m$ as x goes to $-\infty$ and a straight line with the slope of $mposi_l$ after $kposi_l$ as x goes to ∞

Given the value of elements of these two sets. We can define an unique transaction costs.

The smoothing part includes a parameter θ, which controls the degree of smoothing part and we called the smoothing function for transaction costs as "θ function". Given the value of θ, θ function is defined as following:

- If $\theta = 0$, the θ function of each dimension should be two straight lines with the slope of $mposi_1$ with the beginning point at $kposi_1$ for positive part and $mnega_1$ with the beginning point at $knega_1$ for negative part.
- If $\theta = 1$, the θ function of each dimension should be the original transaction costs.
- If $0 < \theta < 1$, for each dimension, θ function is defined similar to the transaction costs with: $\overline{mposi_i} = mposi_1 + \theta \cdot (mposi_i - mposi_1)$, and $\overline{mnega_i} = mnega_1 + \theta \cdot (mnega_i - mnega_1)$

Notice that the smoothing part only works for the slope of each segment, the kink points remain the same. To state the idea clearly, we draw the plot of θ function as below.

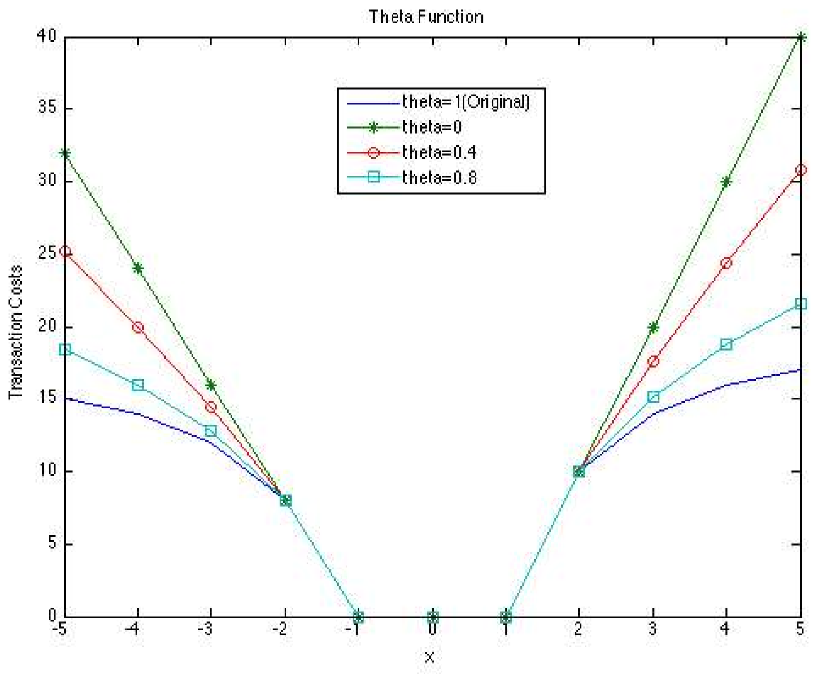

Figure 3.4: θ function.

The main idea of simulated annealing with smoothing is that we have a sequence of θ which is $0 = \theta_1 < \theta_2 < ... < \theta_m = 1$ defined by user.

- First, we use θ function for $\theta = \theta_1 = 0$ and x_0 to minimize the modified transaction cost function, then the modified transaction cost function becomes a convex function, which means we can solve the minimization problem using simulated annealing more quickly and get a better answer: x_1
- Then, we use θ function for $\theta = \theta_2$ and x_1 to minimize the modified transaction cost function using simulated annealing and get x_2. Repeat that until $\theta = \theta_m = 1$ and the final answer x_m

The advantage of using smoothing part in simulated annealing is that we changed the function into a convex function when θ is 0 so that simulated annealing has a much higher probability to get a better answer in fewer trials and temperature sequence. Then we make the smoothing part, which means the θ function become closer to the original transaction cost function. We use the solution comes from the last step which means we use the information from last step. The numerical results also show that the simulated annealing with smoothing does a much better job than trust-region method while it takes less time than simulated annealing alone.

3.3 Numerical Results

Here we present the numerical results for $n = 2, 10, 20, 30$ and 40 where n is the number of dimensions. For each dimension we choose, we do 20 different experiments and draw the final results. Results come from using simulated annealing with smoothing and using the fmincon in MATLAB's tool-box with trust-region method. We use the way discussed in section (2.2) to get the gradient and Hessian matrix of transaction cost function.

The problem we solve is (2.1) and with condition number is equal 10000 of Q. The g and Q is generated randomly. For transaction costs, we also generated it randomly but control the slope value in a certain interval.

For simulated annealing parameter T and $Trials$. We use $T = 10 * 0.9^{(0:199)}$ and $Trials = 50$ for each temperature. The upperbound is [1000,...,1000] and lowerbound is [-1000,...,-1000]. For fmincon, we choose trust-region method and user supplies the gradient and Hessian matrix.

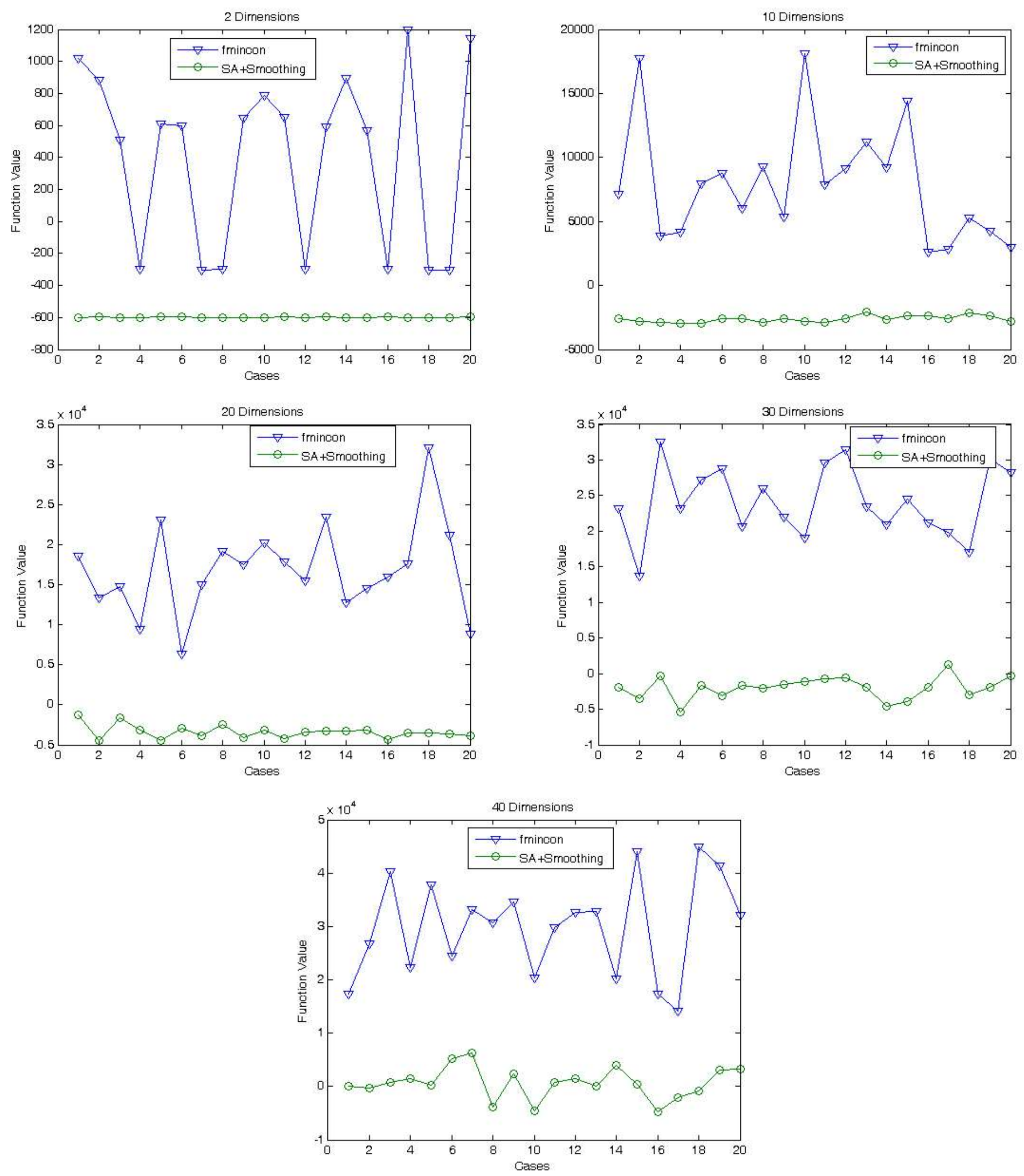

Figure 3.5: Final Results of Simulated Annealing with Smoothing Compared with Trust-Region Method.

3.4 Conclusion

Based on the numerical results, we can conclude that simulated annealing with smoothing is much better in minimizing the transaction cost function than just using the trust-region method. As shown in the graphs, simulated annealing with smoothing will appear to give the global optimum while trust-region can only find a local optimum near the start point. This result emphasizes one of our main points: the traditional algorithm for optimization will probably be trapped in the region near the start point and could not escape from it. But simulated annealing with smoothing can easily escape from local optima and get a much better solution.

If the number of dimensions is higher than 40, we can still get a better answer after adjusting the parameters of temperature sequence and trials properly in simulated annealing with smoothing part.

Chapter 4

Further Comparison

In this chapter, we do some further comparisons between simulated annealing with smoothing and traditional simulated annealing and trust-region. We also investigate the effect of different parameters in simulated annealing on the final results.

4.1 Simulated Annealing with and without Smoothing

For some dimensions we draw the plots of the final results of traditional simulated annealing and simulated annealing with smoothing under the same parameters (same temperature and same trials for each temperature). We want to see if the smoothing part has effect on the traditional simulated annealing method. We use the same parameters for simulated annealing with smoothing and traditional simulated annealing, and solve the same optimization problems.

In this section, the problem we solve is the same as before. The parameters for both methods is $T = 10 * 0.9^{(0:199)}$ and $Trials = 50$ for each temperature. The upperbound is $[1000, ..., 1000]$ and lowerbound is $[-1000, ..., -1000]$.

For each dimension we choose, we do 20 cases and draw the final objective function value we get from simulated annealing with smoothing and traditional simulated annealing. We also draw the time for both method to see the difference.

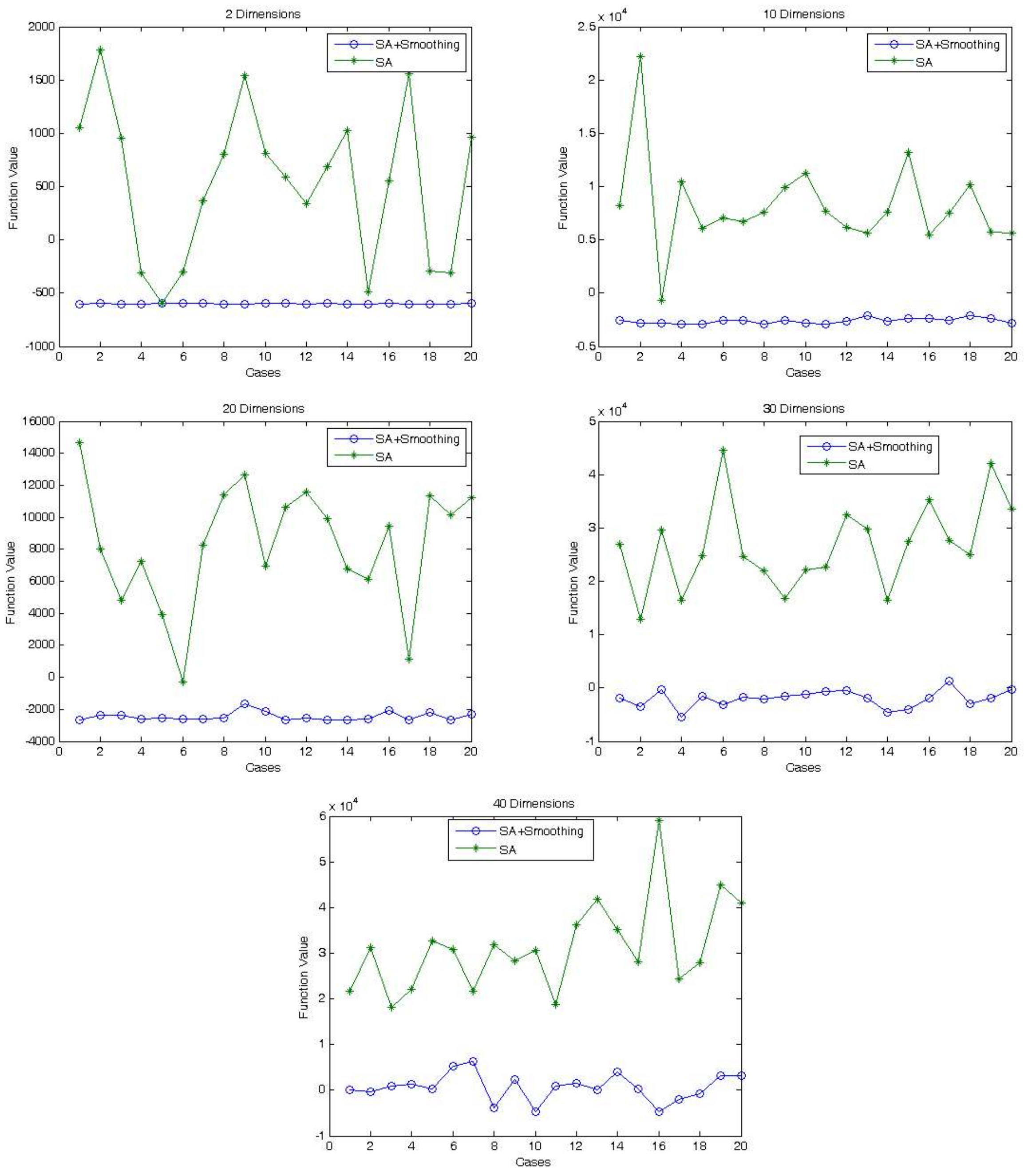

Figure 4.1: Final Results of Simulated Annealing with Smoothing Compared with Traditional Simulated Annealing.

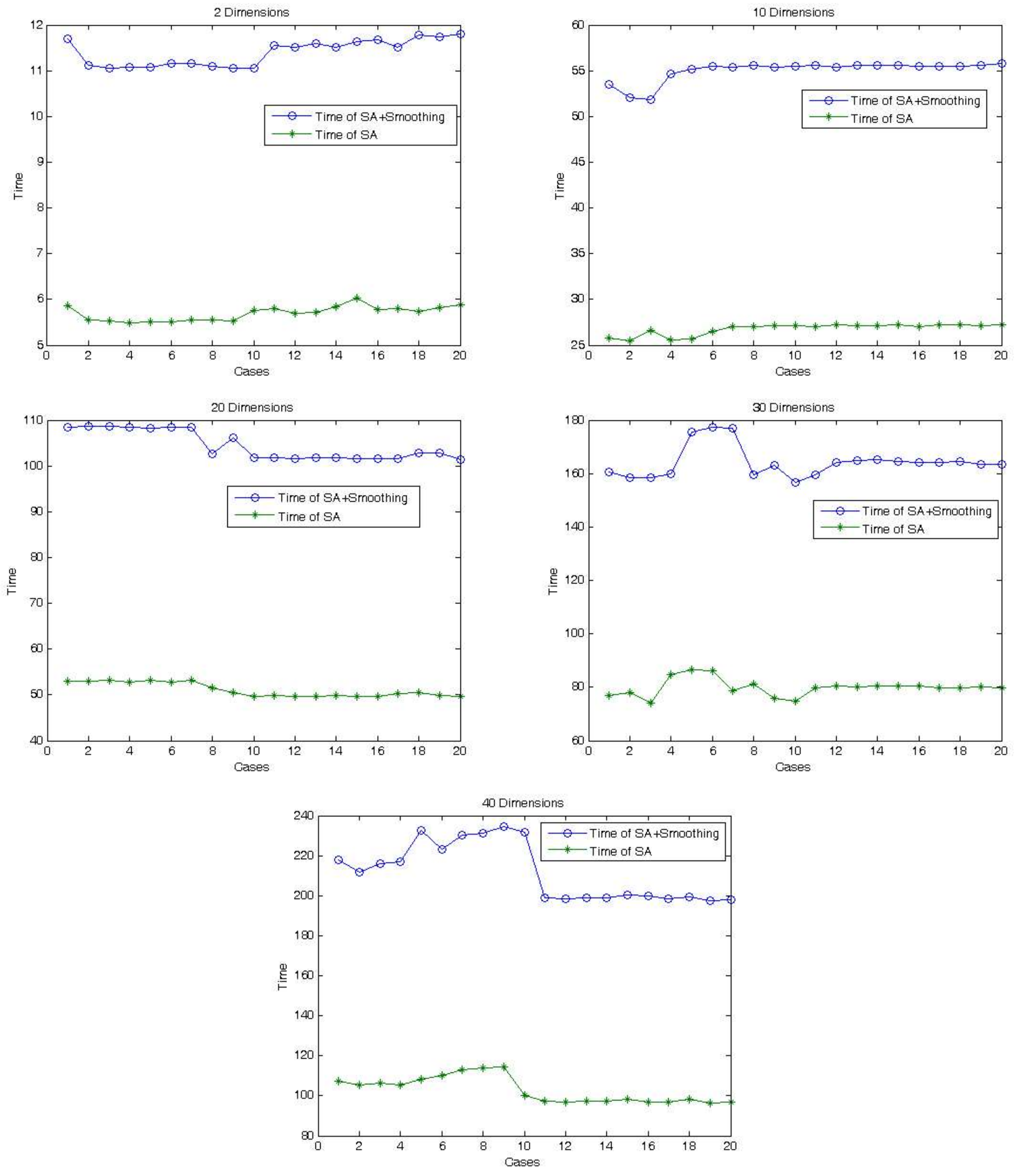

Figure 4.2: Time of Simulated Annealing with Smoothing Compared with Traditional Simulated Annealing.

From the results we can see that under the same parameters, simulated annealing with smoothing does a better job than traditional simulated annealing algorithm. However, the simulated annealing with smoothing takes a longer time to get the final results, but it is worth to take this time cause the results are much better. We believe if we compare these two methods in minimizing the transaction cost function in the same time, simulated annealing and smoothing can do a better job. As we can see, the final objective function's value we get from simulated annealing with smoothing is much lower than the final result of using traditional simulated annealing.

4.2 Simulated Annealing Compared with Trust-Region

In this section we want to see whether the traditional simulated annealing method can do a better job than fmincon (i.e., trust-region method). We choose $n = 2$ and 10 to do the experiments and compare the final results. We use the methods discussed in section (2.2) to get gradient and Hessian matrix of (2.1). As for the simulated annealing method, we adjust the parameter for temperature sequence and trials for each temperature.

The parameter we use for simulated annealing is: $T = 10 * 0.9^{(0:399)}$ and $Trials = 400$ for each temperature. The upperbound is [1000,...,1000] and lowerbound is [-1000,...,-1000]. For fmincon, we choose trust-region method and user supplies the gradient and Hessian matrix.

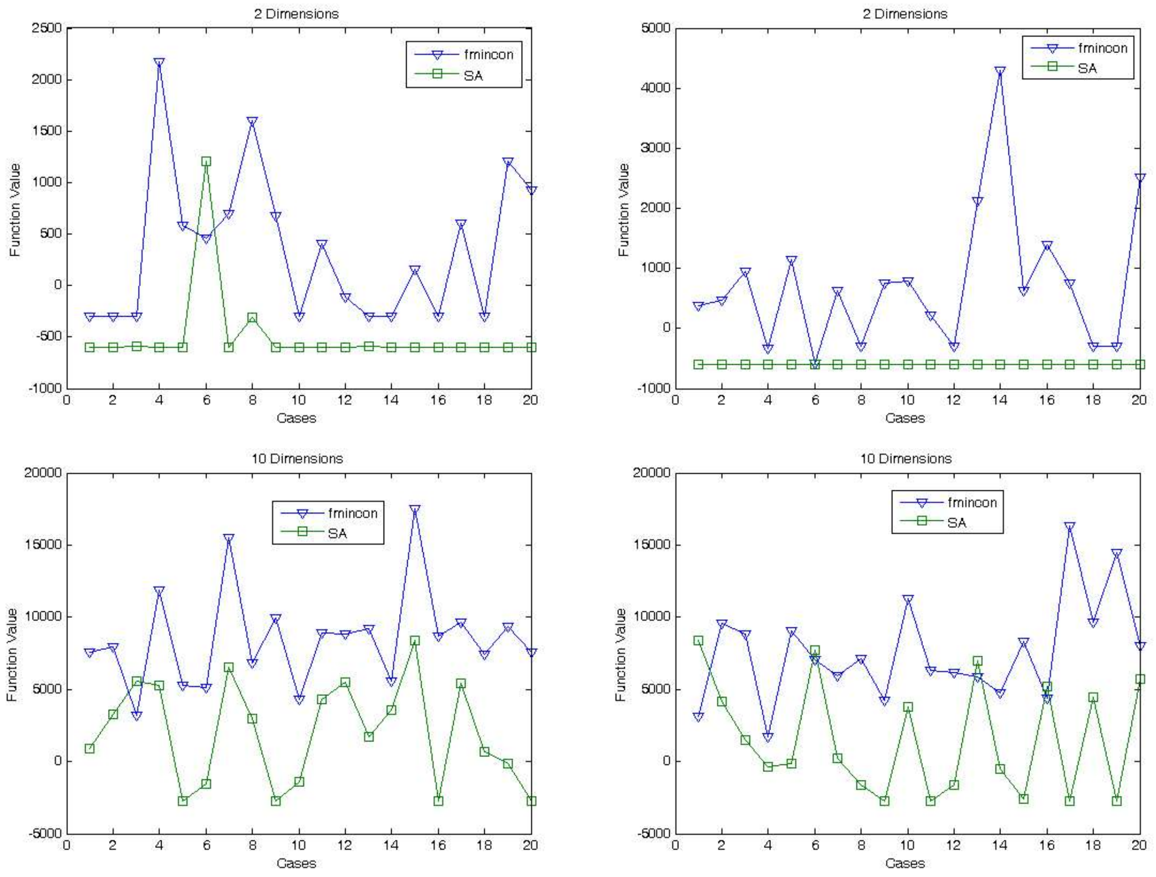

Figure 4.3: Final Results of Traditional Simulated Annealing Compared with Trust-Region Method.

After adjusting the parameters for simulated annealing we can find that it can do a better job than trust-region method, but needs more time. Also, we should notice that for each dimension, there is a probability that simulated annealing does worse than trust-region method but the probability is very small. It is reasonable because simulated annealing is a "random" algorithm since the accept rule depends on a random function. So there is a chance that simulated annealing can not find a better answer than trust-region method.

After all, we can see that as long as the parameters adjusted properly, simulated annealing without smoothing also can do a better job than trust-region method with a high probability.

4.3 Different Parameters Comparison

In this section we do the experiments to find whether the parameters can affect the final results. Here we do 3 different comparison:

1. We compare the final results and time of two different temperature sequence: $T_1 = 10 * 0.9^{(0:199)}$ and $T_2 = 10 * 0.9^{(0:399)}$. Notice that we change the number of sequence.
2. We compare the final results and time of two different number of trials: $Trials = 50$ and $Trials = 200$.
3. We compare the final results of two different T_0: $T_1 = 10 * 0.9^{(0:199)}$ and $T_2 = finitial * 0.9^{(0:199)}$. We know the time for this two different parameters should be the same so we don't compare the time this time.

The problem we solve is (2.1) and for each certain dimension we choose, we do 20 different cases and draw the final objective function value and time.

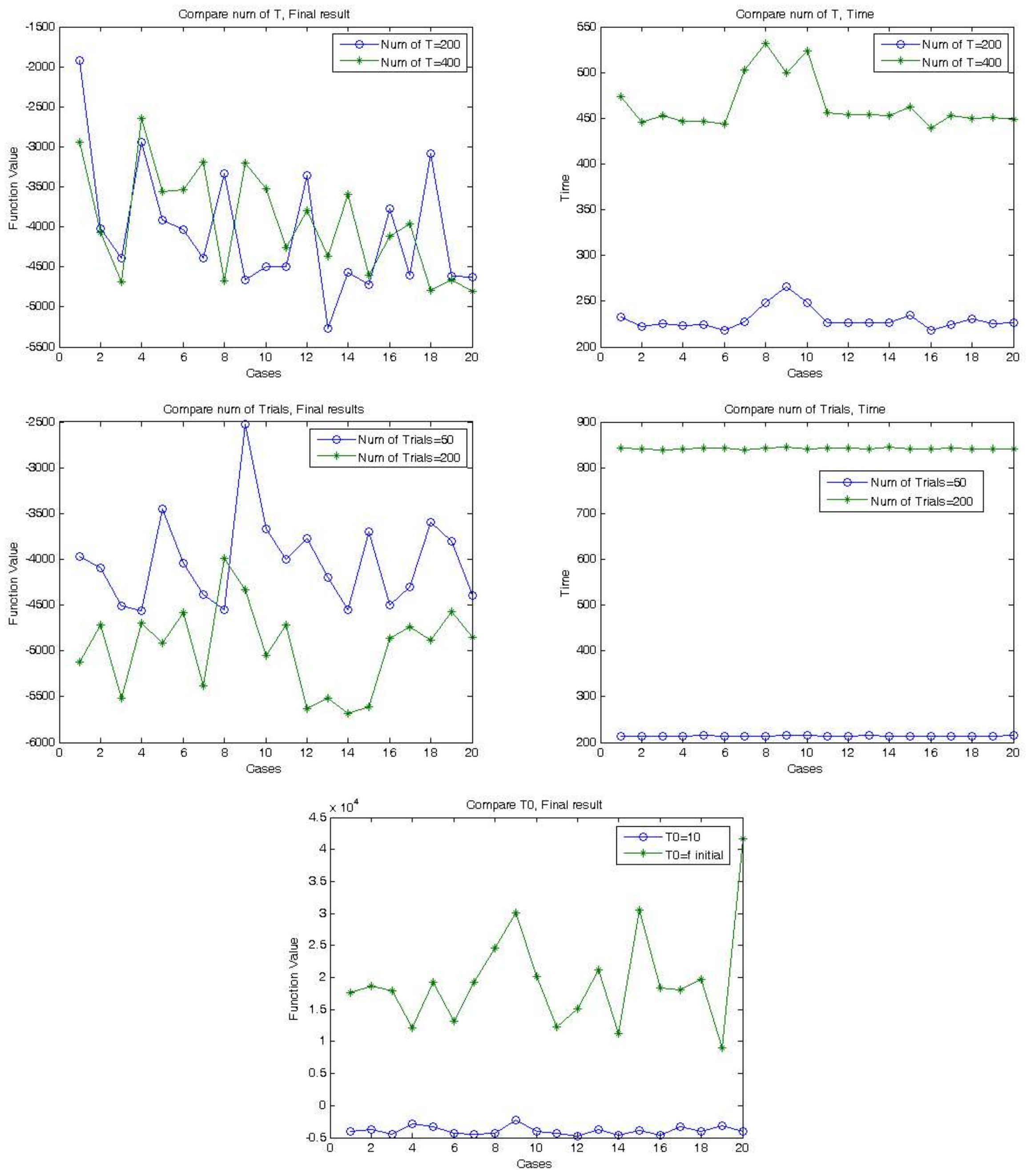

Figure 4.4: Different Parameters Comparison.

The results clearly show that the number of trials for each temperature and the initial temperature T_0 in temperature sequence affect the final results. The number of elements in temperature sequence doesn't have a significant effect for the results. However, the conclusion should be applied for this particular problem. For different problems the conclusion may not be the same.

4.4 Conclusion

We conclude that our smoothing idea can play an important and efficient role in a simulated annealing framework for transaction cost function minimization. Simulated annealing combines with progressive smoothing can be considerable more effective than the straightforward application of simulated annealing. Both simulated annealing and simulated annealing with smoothing outperform a local trust-region method with respect to the quality of the solution found.

We see that some parameters for certain problem also can have effect for the final results. Users should consider the cost for each parameter and determine them properly for each problem. Further work can be applied to determine the parameters properly depends on the problem.

Chapter 5

Summary

Transaction cost function minimization is an important problem in portfolio selection in finance. In this essay, we develop and apply a new method for this problem.

Since the transaction cost function is not differentiable at kink points and has many local optima, it is difficult to get a satisfactory result when minimized by trust-region algorithm. Another way to deal with this situation is to use a global minimization approach such as the simulated annealing algorithm, which is a global search meta-heuristic used to continuous optimization problems. However, when we use simulated annealing to minimize the transaction problem, it takes too much time to get a satisfactory result. Even though the previous work and examples show that simulated annealing can be efficient dealing with the problems that have many local optima, it is not suitable for transaction cost function.

Our new approach is to add a smoothing technique in the simulated annealing methodology where we can approximately convert the objective function to a convex function and solve the optimization problem. We change the function to make it closer to the original transaction cost function while solving the optimization problem using the result we get from the previous step. Numerical results show that simulated annealing with smoothing is an ideal way to optimize transaction cost function since it is more efficient than trust-region algorithm and traditional simulated annealing algorithm. The final objective function value is much lower when we use simulated annealing with smoothing.

We note that the degree of smoothing can be controlled in this problem by adjusting a single parameter.

As shown in section (3.3), simulated annealing with smoothing can get a much lower value than trust-region algorithm in minimizing the transaction cost function and it takes a reasonable time to reach the final result. From the graphs in section (4.1), we conclude that simulated annealing with smoothing can get a lower value than traditional simulated annealing under the same parameters, which means smoothing part makes the simulated annealing more effective. In section (4.2) and (4.3), we know traditional simulated annealing can do better than trust-region if we set up the parameters properly and different parameters may affect the final results.

References

[1] E.H.L. Aarts and P.J.M van Laarhoven. Statistical cooling: A general approach to combinatorial optimization problems. *Phillips Journal of Research*, 40:193–226, 1985.

[2] H. Akaike. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Annals of the Institute of Statistical Mathematics*, 11:1–17, 1959.

[3] M.H. Alrefaei and S. Andradottir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45:748–764, 1999.

[4] S. Anily and A. Federgruen. Simulated annealing methods with general acceptance probabilities. *Journal of Applied Probability*, 24:657–667, 1987.

[5] M.J. Best and R.R. Grauer. The efficient set mathematics when mean-variance problems are subject to general linear constraints. *Journal of Economics and Business*, 42:105–120, 1990.

[6] M.J. Best and J. Hlouskova. Portfolio selection and transaction cost. *Computational Optimization and Applications*, 24:95–116, 2003.

[7] G. Corliss C. Bischof and A. Griewank. Structured second-and higher-order derivatives through univariate taylor series. *Optimization Methods and Software*, 2:211–232, 1993.

[8] I. Charon and O. Hudry. The noising methods-a generalization of some metaheuristics. *European journal of Operational Research*, 135:86–101, 2001.

[9] M.A. Fleischer and S.H. Jacobson. Information theory and the finite-time behavior of the simulated annealing algorithm: experimental results. *INFORMS Journal on Computing*, 11:35–43, 1999.

[10] R.B. Schnabel G.A. Schultz and R.H. Byrd. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical Analysis*, 22:47–67, 1985.

[11] FCE Frankfurt Consulting Engineers GmbH. Optimization. www.frankfurt-consulting.de/English/optimierung_us.htm, 2014.

[12] S.J. Wright J. Nocedal. *Numerical Optimization.* Second edition, 2004.

[13] A.L. Tits J.F. Bonnans, E.R. Panier and J.L. Zhou. Avoiding the maratos effect by means of a nonmonotone line search. ii. inequality constrained problems feasible iterates. *SIAM Journal on Numerical Analysis*, 29:1187–1202, 1992.

[14] A. D. Yucekaya K. Haridass, J. Valenzuela and T. Mcdonald. Scheduling a log transport system using simulated annealing. *Information Sciences*, 264:302–316, 2014.

[15] O.L. Mangasarian. Nonlinear programming. McGraw-Hill: New York, 1969.

[16] H. Markowitz. The optimization of a quadratic function subject to linear constraints. *Naval Research Logistics Quarterly*, March-June:111–133, 1956.

[17] R.L. Salcedo M.F. Cardoso and S.F. de Azevedo. Nonequilibrium simulated annealing: a faster approach to combinatorial minimization. *Industrial Engineering and Chemical Research*, 33:1908–1918, 1994.

[18] M.K. Nadeem, L.K. Sandip, and W.K. Shiv. Simulated annealing technique to design minimum cost exchanger. *Chemical Industry and Chemical Engineering Quarterly*, 17(4):409, 2011.

[19] J. Ma P. Tian and D.M. Zhang. Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation theory: an investigation of generation mechanism. *European Journal of Operational Research*, 1999.

[20] J.C. Gilbert R.H. Byrd and J.Nocedal. A trust-region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 2000.

[21] R.B. Schnabel R.H. Byrd and G.A. Shultz. A trust-region algorithm for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis*, 24:1152–1170, 1987.

[22] K. Ritter. A method for solving nonlinear maximum-problems depending on parameters. *Naval Research Logistics Quarterly*, 14:147–162, 1967.

[23] T.C. Martins R.S. Tavares and M.S.G Tsuzuki. Simulated annealing with adaptive neighbourhood: A case study in off-line robot path planning. *Expert Systems with Applications*, 38(4):2951–2965, 2011.

[24] R. A Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices Magazine*, 26, 1989.

[25] Y.Q. Sheng and A. Takahashi. A simulated annealing based approach to integrated circuit layout design, simulated annealing - single and multiple objective problems. *Simulated Annealing - Single and Multiple Objective Problems*, 12, 2012.

[26] A. Solonen. Proposal adaptation in simulated annealing for continuous optimization problems. *Computational Statistics*, pages 1–17, 2013.