東　　海　　大　　學

工業工程與經營資訊研究所

碩士論文

以改良式利普希茨最佳化演算法求解運輸車隊維修排程問題

研 究 生：葉政祐

指導教授：曾宗瑤 副教授

黃嘉彥 教授

中 華 民 國 九 十 九 年 七 月

An Improved Lipschitz optimization algorithm for solving transportation fleet maintenance scheduling problem

By
Cheng-Yu Yeh

Advisor: Prof. Tsueng-Yao Tseng
Prof. Jia-Yen Huang

A Thesis
Submitted to the Institute of Industrial Engineering and Enterprise Information at Tunghai University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Industrial Engineering and Enterprise Information

July 2010
Taichung , Taiwan , Republic of China

An Improved Lipschitz optimization algorithm for solving transportation fleet maintenance scheduling problem

Student: Cheng-Yu Yeh

Advisor: Prof. Tsueng-Yao Tseng
Prof. Jia-Yen Huang

Department of Industrial Engineering and Enterprise Information
Tunghai University

ABSTRACT

In this study, we propose an improved Lipschitz optimization algorithm secure a near-optimal solution for the Transportation Fleet Maintenance Scheduling Problem (TFMSP). By employing the proposed algorithm and a fine-tune procedure based on slope-checking, step-size comparison mechanisms, the search steps can be significantly reduced and the solutions can be secured within a very short run time. We provide a numerical example to demonstrate the efficiency of the proposed algorithm. To compare the computational performance, we test these search algorithms by random experiments with different values of the major setup cost and different size of vehicle groups. Based on our numerical experiments, we conclude that the proposed search algorithm can significantly outperform the dynamic Lipschitz optimization algorithm presented by Yao and Huang (2007) in run time as well as the quality of solutions.

Keywords: Lipschitz, optimization algorithm, maintenance scheduling

以改良式利普希茨最佳化演算法求解運輸車隊維修排程問題

學生：葉政祐　　指導教授：曾宗瑤 副教授
黃嘉彥 教授

東海大學工業工程與經營資訊研究所

摘要

在本研究中，提出以 Evtushenko 演算法為架構的一改良式利普希茨最佳化演算法，並用以求解運輸車對維修排程問題。透過本研究所提出之演算法及能夠確保搜尋速度的步距比較和斜率判斷機制，可使搜尋時間及次數大幅降低，提高搜尋的效率。而在不同車隊大小以及維修的整備成本考量之下，本研究提出隨機數值實驗，並從中得到此改良式利普希茨最佳化演算法不論是在搜尋速度或是求解品質皆比過去研究還要好之結論。

關鍵字詞：利普希茨、最佳化演算法、維修排程

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

In the past decade, logistics service providers have generally experienced low profit margins due to the intensive competition that exists in the industry and prices skyrocketing of the crude oil. It is necessary to have a fleet maintenance scheduled economically. A well planned maintenance schedule can not only bring down the cost but also raise the utilization rate of a transportation fleet.

In this study, we devote our efforts to investigate a mathematical model for determining the economic maintenance frequency of a transportation fleet. We name this problem as "the Transportation Fleet Maintenance Scheduling Problem", which is abbreviated as the TFMSP. In the TFMSP, the decision maker needs to determine T (*i.e.*, the basic period) and $\{k_1,k_2,\ldots,k_m\}$ (*i.e.*, the frequency of maintenance for vehicles in each group) so as to minimize the total costs incurred per unit time.

The mathematical model for the TFMSP was first proposed by Goyal and Gunasekaran (1992) and refuted by Dekker and Wildeman (1995). Huang and Yao (2007) proposed a dynamic Lipschitz algorithm to secure an ε-optimal solution rapidly, but the search process can be meliorated for a better searching procedure. Therefore, in this study, we propose an improved Lipschitz optimization algorithm which can enhance the searching efficiency.

This paper is structured as follows. First, we review the studies in the literature for the TFMSP in chapter 2. The mathematical model of TFMSP is presented in chapter 3. Then, in chapter 4, we present on the optimal cost curve of the problem. Based on Huang and Yao's (2007) study, the lower and upper bounds can be defined. An improved Lipschitz optimization algorithm with slope-checking, step-size comparison mechanisms is proposed in chapter 5. In the first part of Section 6, a numerical example is given to demonstrate the implementation of the proposed algorithm. Then, in the second part of Section 6, randomly generated examples are presented to show that the proposed algorithm significantly outperforms the search procedure of the traditional Lipschitz algorithm. Finally, we address our concluding remarks in chapter 7.

2. Literature review

In this chapter, we review the literature of the studies in the Transportation Fleet Maintenance Scheduling Problem.

2.1 Transportation fleet maintenance scheduling problem

In the past, Goyal and Gunasekaran (1992) mentioned some researches have been dealt with the determining of economic maintenance scheduling in management science/operations research/industrial engineering (see Luss and Kander 1974, Luss 1976, Christer and Doherty 1977, Sule and Harmon 1979, Goyal and Kusy 1985). Recently, many researchers have been addressing their efforts to the studies on the scheduling of production facilities or machines (see Wildeman and Dekker 1997, Dekker, *et al*. 1997, Anily, Glass and Hassin 1998, 1999, Amotz, *et al*. 2002).

However, these studies did not take the characteristics of maintenance for the vehicles in a transportation fleet into consideration. We note that the objective functions in these studies are significantly different in their theoretical properties from that for the TFMSP. On the other hand, researchers pay limited attention to the problem of determining the operating and maintenance schedules for a transportation fleet.

Although some of the researches showed that the aircraft fleet maintenance scheduling problem has been raised to reduce the cost for many years, the constraints are completely different due to the special characteristics such as heterogeneous fleet of aircraft, the regulations of routine inspection given by Federal Aviation Administration, the consideration of flight hours and number of take-off and landing cycles (Sriram and Haghani, 2003). Besides, the aircraft maintenance scheduling is related to assignment (Moudani and Mora-Camino, 2000), maintenance routing and crew scheduling as well (Papadakos, 2009), which is different from the TFMSP.

2.2 Goyal and Gunasekaran's method

Notation

Z total cost per unit of time

m number of groups of vehicles

T basic maintenance cycle time

S fixed cost incurred in each maintenance cycle for the i^{th} group of vehicles

$f_i(t)$ operating cost per unit of time at t time units after the last maintenance

a_i fixed operating cost per unit of time

b_i increased in the operating cost per unit of time

s_i fixed cost of maintenance for a vehicle

n_i number of vehicles in the group

X_i time required for maintenance work on the vehicle

k_i an integer which when multiplied by the basic maintenance cycle time T gives the maintenance cycle time for the vehicles in the group

Y_i utilization factor of a vehicle on the road

z_i total cost per unit time for a vehicle

Goyal and Gunasekaran (1992) proposed an approach for TFMSP based on two equations that are derived by setting the first derivative of $Z(T,k_i)$ with respect to the decision variables to zero:

$$\min Z(T,k_i)=\frac{S}{T}+\sum_{i=1}^{m} n_i \frac{s_i+\int_0^{Y_i(Tk_i-X_i)}(a_i+b_i t)dt}{Tk_i},\ T>0 \text{ and } k_i\in\{1,2,3,\cdots\} \quad (1)$$

$$T(k_1,k_2,\cdots,k_m)=\sqrt{2\left(\frac{S+\sum_{i=1}^{m}(n_i(s_i-X_iY_i(a_i-b_iX_iY_i))/k_i)}{\sum_{i=1}^{m}n_ib_ik_iY_i^2}\right)} \quad (2)$$

$$k_i=\frac{1}{TY_i}\sqrt{\frac{2(s_i-X_iY_i(a_i-0.5b_iX_iY_i))}{b_i}} \quad (3)$$

Step 1.For the first iteration, assume $k_i=k_i^{(0)}=1$ for all i, and obtain the first estimate of $T=T^{(1)}$ from (2). At $T=T^{(1)}$, determine $k_i=k_i^{(1)}$ from (3) for all i. If $k_i^{(1)}$ values are not integers, then select the nearest

non-zero integer.

Step 2. Using $k_i = k_i^{(0)}$ from (3) for i=1,…,m, we obtain $T = T^{(2)}$ from (2) and then $k_i = k_i^{(2)}$ from (3) using $T = T^{(2)}$. Repeat the process until the r^{th} iteration and stop when $k_i^{(r)} = k_i^{(r-1)}$ for i=1,…,m. The economic policy is obtained at $T^* = T^{(r)}$ and $k_i^* = k_i^{(r)}$

Later, in van Egmond, Dekker and Wildeman's (1995) paper, they have a full discussion on Goyal and Gunasekaran's search procedure. They indicate that the objective function is not convex as Goyal and Gunasekaran (1992) assumed. And, since the values of k_i need to be integers, the determination of the global optimization is not as easy as Goyal and Gunasekaran suggested. They also show that it is not necessarily the k_i minimizing Z when one rounds (3) to the nearest non-zero integer. Finally, they indicate that Goyal and Gunasekaran's search procedure often stops after its first iteration without obtaining an optimal solution since they assumed $k_i = k_i^{(0)} = 1$.

These three problems explain why Goyal and Gunasekaran's solution does not always obtain an optimal solution. In fact, it is often stuck in a local optimal solution. However, van Egmond, Dekker and Wildeman's (1995) only mentioned that one needs to try different starting values to find the global optimum, but without proposing a new solution approach to solve the TFMSP.

2.3 Yao and Huang's junction-point search algorithm

Yao and Huang (2006) conducted a full analysis for the TFMSP. By their theoretical results, they proposed an efficient search algorithm that finds the optimal solution within a very short run time and out-performs Goyal and Gunasekaran's search procedure.

In their study, they indicated the objective function is piece-wise convex with respect to T, and they defined "junction point" as a particular value of T where two consecutive convex curves concatenate. The search algorithm is based on locating all the junction points and calculating the local-minimum $\breve{T}(k)$ between each pair of junction points. If $\breve{T}(k)$ exists and within two

consecutive junction points, then record the objective function value $\Psi(\mathbf{k},T)$ by substituting $\breve{T}(k)$ into the objective function and compared with the optimal value on hand.

Besides, they defined the searching range by a lower and upper bound on the *T*-axis that can make the searching algorithm more efficient. They used Common Cycle approach proposed by Hanssmann (1962) to be the upper bound, *i.e*., all the vehicle groups share a common maintenance cycle, and they derived the lower bound from the optimal objective function value Ψ^* and optimal basic cycle time T^*. The iteratively searching procedure stops until *T* is out of the lower bound.

2.4 Huang and Yao's dynamic Lipschitz optimization algorithm

Huang and Yao (2007) devoted their efforts to TFMSP by adopting a dynamic Lipschitz optimization algorithm that can secure an ε-optimal solution in a very short run time and outperforms Goyal and Gunasekaran's search procedure.

In their study, they first conducted theoretical analysis on the mathematical model of the TFMSP. Based on their theoretical results, the important foundation is then established to show that the objective function of the TFMSP is Lipschitz. Next, they employed a relaxed problem to solve the TFMSP. If the solution is not within an allowance of ε, the Evtushenko algorithm (see Horst and Pardalos, 1995) is then applied as the Lipschitz optimization tool after locating the lower and upper bounds by some line search methods (e.g., bisection; Bazaraa, *et al*., 1993).

According to their numerical experiments results, one may discover that the Lipschitz optimization algorithm is significantly better than Goyal and Gunasekaran's method. However, due to the nature of Evtushenko algorithm, the searching step-size will become very small while the objective function value on hand is better than the existing one, which lead to great number of searching iterations, the searching procedures will be endured as well.

2.5 A brief summary

To the best of the authors' knowledge, there exists few researches deal with transportation fleet maintenance scheduling problem. Although Huang and Yao proposed a dynamic Lipschitz optimization algorithm which can shorten the run time compared to a traditional Lipschitz algorithm, it can be further improved by considering some characteristics of the theoretical properties of the objective function. In the rest of this study, we will dedicate our efforts to propose an improved Lipschitz optimization algorithm based on Huang and Yao's (2007) study.

3. The mathematical model for the TFMSP

In this chapter, before presenting the mathematical model, we first introduce the assumptions made and the notation used later.

3.1 The assumptions and notation

To discuss the transportation fleet maintenance scheduling problem, we redefine the following notation for more clarity based on Goyal and Gunasekaran's (1992) research.

Z total cost per unit of time

m number of groups of vehicles

T basic maintenance cycle time

S fixed cost incurred in each maintenance cycle

$f_i(t)$ operating cost per unit of time at t time units after the last maintenance for a vehicle of the i^{th} group

a_i fixed operating cost per unit of time for a vehicle of the i^{th} group

b_i increased in the operating cost per unit of time for a vehicle of the i^{th} group

s_i fixed cost of maintenance for a vehicle of the i^{th} group

n_i number of vehicles of the i^{th} group

X_i time required for maintenance work on each vehicle of the i^{th} group

k_i an integer variable; k_iT gives the maintenance cycle time for the vehicles of the i^{th} group

Y_i utilization factor of a vehicle of the i^{th} group on the road

There are m groups of vehicles, and the number of vehicles is denoted as n_i for the i^{th} group. In the TFMSP, the decision maker plans the schedules of maintenance for vehicle groups in some basic period, denoted by T (e.g., in days, weeks, or bi-weeks, etc.). The maintenance work of vehicles in a group is executed at a fixed, equal-time interval that is called the maintenance cycle for that group of vehicles. The vehicles in the i^{th} group are sent for maintenance once in k_i basic periods, where k_i is positive integer. Therefore, k_iT is maintenance cycle for vehicles in the i^{th} group. We note that the model for the TFMSP is for preventive maintenance, and the model does not take unplanned

fleet vehicle failures into consideration. Therefore, the maintenance capacity is not limited resource since it requires less man-power and maintenance time for routine maintenance schedule.

3.2 The mathematical model

With regard to costs of the TFMSP, we consider two categories: operating cost and the maintenance cost. The operating cost of a vehicle depends on the length of the maintenance cycle and it is assumed to increase linearly with respect to time since the maintenance work on the vehicle. Specifically, the operating cost at time t after the last maintenance for a vehicle in group i is given by $f_i(t) = a_i + b_i t$, where a_i is the fixed cost and b_i indicates the increase in the operating cost per unit of time. In addition, for each vehicle in group i, we assume that it takes X_i units of time for its maintenance work and Y_i as the utilization factor of a vehicle in the i^{th} group on the road, where X_i and Y_i are known constants (One may refer to Yanagi, 1992 for further discussions on the utilization factor of a vehicle.). Accordingly, the actual time during which a vehicle can operate is equal to $Y_i(k_i T - X_i)$, and the total operating cost for a vehicle in group i is given by $\int_0^{Y_i(k_i T - X_i)} f_i(t)dt = \int_0^{Y_i(k_i T - X_i)} (a_i + b_i t)dt$

$= Y_i(a_i - b_i X_i Y_i)\, k_i T + 0.5\, Y_i^2\, k_i^2 T^2 - X_i Y_i(a_i - 0.5 b_i X_i Y_i)$

Notice that, the expression above is the integral outcome for total operating cost, and we do not go deep into the meaning for each term.

On the other hand, the average fixed cost of maintenance for a vehicle in group i is given by $s_i/(k_i T)$. Besides, as maintenance work is carried out at intervals of T, a fixed cost, denoted by S, will be incurred for all vehicle groups scheduled for maintenance in each basic period. We define the average total cost

$$Z(\{k_1, k_2, \cdots, k_m\}, T) := \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T) + u\,, \text{ where } \Phi_i(k_i, T) = \frac{n_i C_{1i}}{k_i T} + n_i C_{2i} k_i T\,,$$

$C_{1i} = s_i - X_i Y_i (a_i - 0.5 b_i X_i Y_i)$, $C_{2i} = 0.5 b_i Y_i^2$, and $u = \sum_{i=1}^{m} n_i Y_i (a_i - b_i X_i Y_i)$ is a constant since all the parameters are given in its expression. Therefore, the mathematical model for the TFMSP can be expressed as problem (P_0).

$$(P_0) \quad \inf_{T>0, k_i \in Z^+, i=1,\cdots,m} Z(\{k_1, k_2, \cdots, k_m\}, T) = \inf_{T>0, k_i \in Z^+, i=1,\cdots,m} \left\{ \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T) + u \right\} \quad (4)$$

We note that since the fixed cost for the maintenance of a group of vehicle s_i is generally greater than the fixed operating cost a_i in practice, it is reasonable to assume that $C_{1i} > 0$ with $s_i > a_i$ and $Y_i < 1$ (which is the utilization factor of a vehicle). Also, it is obvious that $C_{2i} > 0$.

We define $\Psi(\{k_1, k_2, \cdots, k_m\}, T) := \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T)$ since u is a constant. Then, solving the problem (P_0) is equivalent to obtain the optimal solution for the problem (P) as follows.

$$(P) \quad \inf_{T>0, k_i \in Z^+, i=1,\cdots,m} \Psi(\{k_1, k_2, \cdots, k_m\}, T) = \inf_{T>0, k_i \in Z^+, i=1,\cdots,m} \left\{ \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T) \right\} \quad (5)$$

In the TFMSP, we dedicated our efforts to minimize the total costs incurred per unit of time, *i.e.*, to determine the basic period T and the frequency of maintenance for vehicles in each group for decision maker.

4. Theoretical analysis

In this chapter, we conduct theoretical analysis on the mathematical model of the TFMSP based on Huang and Yao's (2007) research. In section 4.1, we discuss the maintenance frequency for each group of vehicles, which can also provide us more insight about the optimal objective function curve. A relaxation method of the TFMSP is introduced to set the search range in section 4.2. A brief summary is presented in section 4.3.

4.1 The closed form for finding maintenance frequency

As presented by Huang and Yao (2007), the right-side of (5), *i.e.*, $\Phi_i(k_i,T)$, which has the following properties for $T>0$, $i\in\{1,\ldots,m\}$ with given $k_i\in Z^+$.

1. $\Phi_i(k_i,T)$ *is strictly convex*;
2. $\Phi_i(k_i,T)$ *has a minimum for* $T=x_i^*/k_i$ *with* x_i^* *given by*:

$$x_i^*=\sqrt{C_{1i}/C_{2i}} \tag{6}$$

3. *The function* $\Phi_i(k_i,T)$ *obtains its minimal objective function value by*

$$2n_i\sqrt{C_{1i}C_{2i}} \tag{7}$$

Moreover, Huang and Yao defined a new function $g_i(T)$ by taking the optimal value of k_i at any value $T'>0$ for the function $\Phi_i(k_i,T)$ as follows.

$$g_i(T):=\inf_{k_i\in Z^+}\{\Phi_i(k_i,T)\} \tag{8}$$

Therefore, the function $g_i(T)$ is a curve that concatenates $\Phi_i(k_i^*(T'),T')$ where $k_i^*(T')$ is the optimal value of k_i at a given value of $T=T'$. By graphically displaying the $g_i(T)$ curve, one may observe that $g_i(T)$ is actually the lower envelop of all the functions of $\Phi_i(k_i,T)$ with respect to T.

Besides, one may notice that there is a particular value of T where two consecutive convex curves $\Phi_i(k_i,T)$ and $\Phi_i(k_i+1,T)$ concatenate. Huang and Yao defined such points as "junction point" for $g_i(T)$, and derived a closed-form for the location of the junction points. They defined the difference

function $\Delta_i(k,T)$ by

$$\Delta_i(k,T) = \Phi_i(k+1,T) - \Phi_i(k,T)$$
$$= \frac{n_i C_{1i}}{(k+1)T} + n_i C_{2i}(k+1)T - \frac{n_i C_{1i}}{kT} - n_i C_{2i} kT = -\frac{n_i C_{1i}}{k(k+1)T} + n_i C_{2i} T \quad (9)$$

and located the junction points by letting $\Delta_i(k,T) = 0$ as follows.

$$\delta_i(k) = \sqrt{\frac{C_{1i}}{C_{2i}(k+1)k}} = \sqrt{\frac{2(s_i - X_i Y_i(a_i - 0.5 b_i X_i Y_i))}{b_i Y_i^2 (k+1)k}} \quad (10)$$

Importantly, such a junction point provides us the information on at "what value of *T*" where one should change the value of *k* so as to obtain the optimal value for the $g_i(T)$ function.

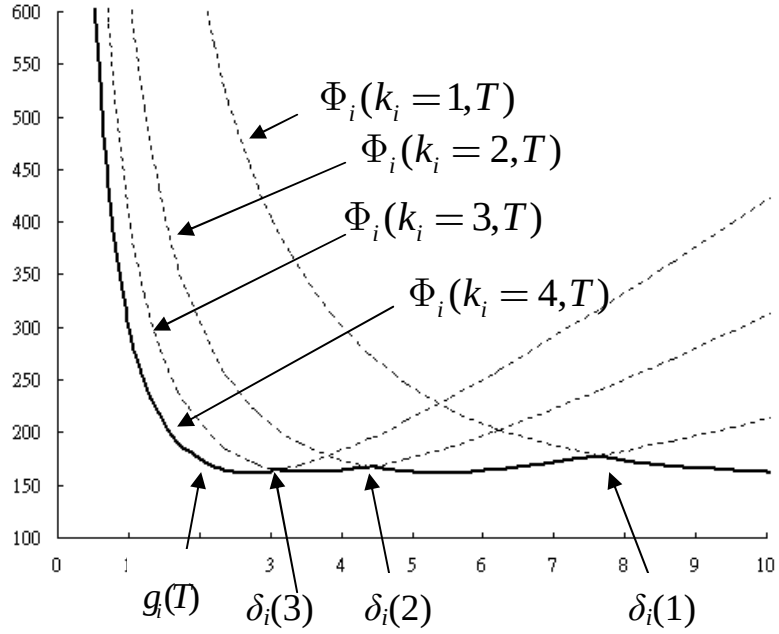

Figure 1. The function curve of the $g_i(T)$ function

Also, Huang and Yao proved the following lemma as an easier way to obtain the optimal multiplier $k_i^*(T) \in Z^+$ for the $g_i(T)$ function for any given $T > 0$ ($\delta_i(k) < \delta_i(k-1)$ by the closed-form of $\delta_i(k)$ in equation (10)).

Lemma 1. *For any given T > 0, an optimal value of* $k_i^*(T) \in Z^+$ *for the* $g_i(T)$ *function is given by*

$$k_i^*(T) = \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{4C_{1i}}{C_{2i}T^2}} \right\rceil \quad (11)$$

with $\lceil . \rceil$ denoting the upper-entier function.

By the rationale discussed above, we know that the $g_i(T)$ function is a piece-wise convex function since it is made up of several different $\Phi_i(k_i,T)$ curves. In addition, the average cost of S/T is obviously a convex curve. Therefore, we further define $\Gamma(T)=\frac{S}{T}+\sum_{i=1}^{m} g_i(T)$ and rewrite the problem (P) by

$$(P_1)\quad \inf_{T>0}\Gamma(T)=\inf_{T>0}\left\{\frac{S}{T}+\sum_{i=1}^{m} g_i(T)\right\} \tag{12}$$

where the function $\Gamma(T)$ is the optimal objective function value curve with piece-wise convex property of a univariate function with respect to T.

4.2 The upper and lower bounds

Based on Huang and Yao's (2007) study, we employ a relaxation method to set the search range. By relaxing the constraints $k_i \in Z^+$ by $k_i \geq 1$, we obtain a relaxation of (P) in (5), namely (R) as follows.

$$(R)\quad \inf_{k_i\geq 1, i=1,\ldots,m; T>0}\Psi(\{k_1,k_2,\ldots,k_m\},T)=\inf_{k_i\geq 1, i=1,\ldots,m; T>0}\left\{\frac{S}{T}+\sum\nolimits_{i=1}^{m}\Phi_i(k_i,T)\right\} \tag{13}$$

Similarly, we also define $g_i^{(R)}(T)$ as a relaxation of $g_i(T)$ in (8) by replacing $k_i \in Z^+$ with $k_i \geq 1$, *i.e.*, $g_i^{(R)}(T) \coloneqq \inf_{k_i\geq 1}\{\Phi_i(k_i,T)\}$, where $T>0$ and $i=1,...,m$. If we define $h(T)$ by $h(T) \coloneqq S/T+\sum_{i=1}^{m} g_i^{(R)}(T)$, we may consider $h(T)$ is a relaxation of the function $\Gamma(T)$ defined right before (12). Then, it is obvious that solving (R) is equivalent to obtain an optimal solution for the problem (R_1) as follows.

$$(R_1)\quad \inf_{T>0} h(T)=\inf_{T>0}\left\{S/T+\sum_{i=1}^{m} g_i^{(R)}(T)\right\} \tag{14}$$

If we ignore the constraint $k_i \geq 1$, then k_i becomes a continuous variable. In such a case, for any given value of $T>0$, we may easily obtain the optimal value for $\Phi_i(k_i,T)=\frac{n_i C_{1i}}{k_i T}+n_i C_{2i} k_i T$ by

$$k_i^*(T) = \frac{1}{T}\sqrt{C_{1i}/C_{2i}} \tag{15}$$

Recall that $x_i^* = \sqrt{C_{1i}/C_{2i}}$, which is expressed in (6). When $T \le x_i^*$, we have $k_i^*(T) \ge 1$, which satisfies the constraint $k_i \ge 1$. Therefore, $g_i^{(R)}(T)$ obtains its optimal value as a constant by $2n_i\sqrt{C_{1i}C_{2i}}$ for $T \le x_i^*$.

On the other hand, when $T > x_i^*$, we have $k_i^*(T) < 1$, and we are forced to take $k_i^*(T) = 1$ if we take the constraint $k_i \ge 1$ into accounts. By summarizing both cases, it follows that

$$g_i^{(R)}(T) = \begin{cases} 2n_i\sqrt{C_{1i}C_{2i}}, & \text{if } T \le x_i^*. \\ \Phi_i(1,T) = n_iC_{1i}/T + n_iC_{2i}T, & \text{if } T > x_i^*. \end{cases} \tag{16}$$

Also, we could easily obtain the first derivative of $g_i^{(R)}(T)$: when $T \le x_i^*$, $\frac{dg_i^{(R)}(T)}{dT} = 0$, and when $T > x_i^*$,

$$\frac{dg_i^{(R)}(T)}{dT} = \frac{d\Phi_i(1,T)}{dT} = -n_iC_{1i}/T^2 + n_iC_{2i}. \tag{17}$$

Therefore, we conclude that the function $g_i^{(R)}(T)$ is convex, increasing, and continuously differentiable on $(0,\infty)$.

Without loss generality, we assume that $x_1^* \le x_2^* \le ... \le x_m^*$, the strictly increasing derivative $h'(\cdot)$ is given by

$$h'(T) = \begin{cases} \dfrac{-S}{T^2} & \text{if} \quad T \le x_1^* \\ \displaystyle\sum_{i=1}^{l} n_iC_{2i} - \left(S + \sum_{i=1}^{l} n_iC_{1i}\right)\Big/T^2 & \text{if} \quad x_l^* \le T \le x_{l+1}^*, 1 \le l \le m-1 \\ \displaystyle\sum_{i=1}^{m} n_iC_{2i} - \left(S + \sum_{i=1}^{m} n_iC_{1i}\right)\Big/T^2 & \text{if} \quad T \ge x_m^* \end{cases} \tag{18}$$

By setting the derivative of $h'(\cdot)$ in (18) to zero, we have the following lemma to locate the optimal solution $T^{(R)}$ for (R_1).

Lemma 2. *Assume without loss generality that* $x_1^* \le x_2^* \le ... \le x_m^*$. *If it holds that* $i^* := \max\{1 \le i \le m : h'(x_i^*) < 0\}$, *then the optimal solution* $T^{(R)}$ *of* (R_1) *is given by*

$$T^{(R)} = \sqrt{\left(S + \sum_{i=1}^{i^*} n_i C_{1i}\right) \Big/ \sum_{i=1}^{i^*} n_i C_{2i}} \,. \tag{19}$$

Let $v(R)$ be the optimal objective function value of (R_1). Then, $v(R)$ can be obtained by plugging $T^{(R)}$ into the objective function of the problem (R_1). We note that $v(R)$ serves as a lower bound on the optimal objective function value of the problem (P).

Sometimes $T^{(R)}$ is the optimal solution for the problem (P), as we show it by the following lemma.

Lemma 3. *Assume without loss generality that* $x_1^* \le x_2^* \le ... \le x_m^*$. *If it holds that* $T^{(R)} \ge x_m^*$, *then* ({1,…,1}, $T^{(R)}$) *is an optimal solution for the problem* (P).

Proof. Since $T^{(R)} \ge x_m^*$ is an optimal solution of (R_1), we have the corresponding optimal $k_i^* = 1$, for $i = 1,...,m$, by (15). And, obviously, it is also a feasible solution for the problem (P). Hence, it implies that ({1,…,1}, $T^{(R)}$) is an optimal solution for the problem (P). ■

When $T^{(R)} < x_m^*$, $T^{(R)}$ may not be an optimal solution of the problem (P). (This is due to the fact that the values of k_i corresponding to $T^{(R)}$ are not necessary integers. This implies that the optimal solution of (R_1) is generally not feasible for the problem (P).) In such a case, we propose to use Lipschitz optimization algorithm to search for an optimal solution for (P). Define $v(FP)$ as the objective function value at a feasible solution of the problem (P). In this section, we discuss how to use $T^{(R)}$ and $v(FP)$ to determine the search range in our improved Lipschitz optimization algorithm.

Though $T^{(R)}$ may not be an optimal solution of (P) for $T^{(R)} < x_m^*$, we could easily obtain $\{k_i^*(T^{(R)})\}$ at $T^{(R)}$ by (11) in Lemma 1, and $(\{k_i^*(T^{(R)})\}, T^{(R)})$ is a feasible solution of (P).

Denote $\mathbf{k}(T')$ as the set of optimal maintenance frequencies at a given value of T'. Let $v(FP)$ be the objective function value of the problem (P) at $T^{(R)}$, *i.e.*, $v(FP) = \Psi(\mathbf{k}(T^{(R)}), T^{(R)})$. Obviously, $v(FP)$ serves as an upper bound on the optimal objective function value of the problem (P). If $v(FP)$ is very close to $v(R)$ (say within an allowance of ε), we have found an acceptable and feasible solution for (P). If it is not good enough, we shall apply a global-optimization procedure to solve (P). To this end, we need to determine an interval that

contains an optimal solution of (P), denoted by $T^{(P)}$.

Denote T_{low} and T_{up} as the lower and the upper bounds, respectively, of the search range for Lipschitz optimization algorithm. In the following lemma, we will show that a lower and an upper bound on $T^{(P)}$ are given by the two values of T where the objective function of (R_1) equals to $v(FP)$.

Lemma 4. *Let T_{low} and T_{up} be the smallest and the largest T, respectively, for which the objective function of (R_1) is equal to $v(FP)$. Then, the optimal value of T for the problem (P) must lie between T_{low} and T_{up}, i.e., $T^* \in [T_{\text{low}}, T_{\text{up}}]$.*

Proof. Since the objective function of (R_1) is strictly convex, we clearly have the results that $T_{\text{low}} \le T^{(R)} \le T_{\text{up}}$. Consequently, the objective function value is larger than $v(FP)$ for $T < T_{\text{low}}$. Since (R_1) is a relaxation of (P), so that T_{low} is a lower bound on $T^{(P)}$.

Similarly, we may proof that $T^{(P)} \le T_{\text{up}}$. ■

We note that the bounds T_{low} and T_{up} may easily located by some line search methods (*e.g.*, bisection; Bazaraa, *et al.*, 1993). Therefore, if $v(R)$ is not good enough, we would apply a global-optimization technique to solve the problem (P) on the interval $[T_{\text{low}}, T_{\text{up}}]$.

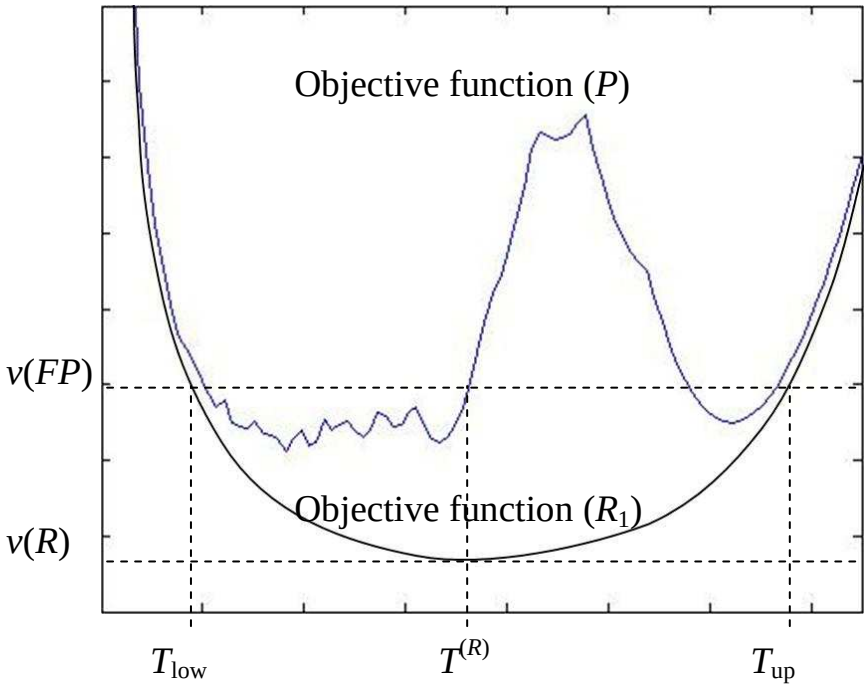

Figure 2. The lower and upper bounds are given by the two values of T where the objective function of (R_1) equals to $v(FP)$.

4.3 A brief summary

In this chapter, Huang and Yao's (2007) research provide us a clear knowledge about the TFMSP. After having the search range, we then ready to apply the Evtushenko algorithm as the Lipschitz optimization algorithm tool. The improved Lipschitz optimization algorithm will be proposed in next chapter.

5. The proposed improved Lipschitz optimization algorithm

In this chapter, we first review the Evtushenko algorithm with the rationale of Lipstchitz optimization. The procedure and detail situations for expediting the search in Evtushenko algorithm is presented in section 5.2. In section 5.3, a fine-tune searching procedure based on slope-checking and step-size comparison is proposed to complete the improved Lipschitz optimization algorithm. A brief summary of the improved searching algorithm is given in section 5.4, and the steps of proposed algorithm are presented.

5.1 A review on the Evtushenko algorithm

Before presenting the Evtushenko algorithm, it is important to learn more about Lipschitz optimization algorithm. It is a global-optimization approach when the objective functions are univariate (Huang and Yao, 2007). More formally, a real-valued function *h* defined on a compact set $X \subseteq \mathrm{R}^n$ that is said to be *Lipschitz* must satisfy the condition

$$\forall x \in X,\ \forall y \in X \qquad \left|h(x)-h(y)\right| \le L\left\|x-y\right\|$$

where *L* is a constant (called *Lipschitz constant*) and $\|\cdot\|$ denotes the Euclidean norm (Horst and Pardalos, 1995). ($n=1$ for univariate)

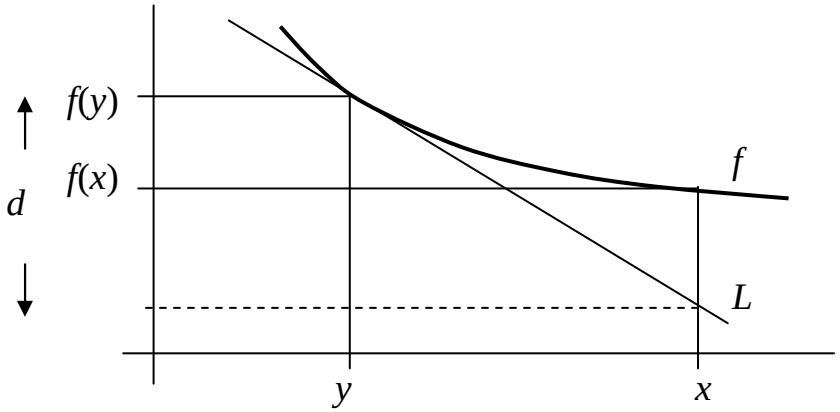

Figure 3. Lipschitz function

One can notice by Figure 3 that an univariate function *f* is Lipschitz when the absolute difference of the objective function value for each pair of *x* and *y* is

smaller than or equal to the *Lipschitz constant*. In other words, the optimal solution is secured by $d=L\|x-y\|$ within range [*y*, *x*]. However, it is worth to mention that often *L* is unknown beforehand, *i.e.*, only an overestimate of *L* will be available for bounding the Lipschitz function.

The Evtushenko algorithm is designed for maximization of multivariate Lipschitz functions, but it solves repeatedly univariate maximization problems obtained by fixing all variables but one. The idea of the algorithm is to make use of the information on the current best known function value to determine the largest valid step size $\delta = \left(2\varepsilon + f_{opt} - f(x_k)\right)/L$ for next iteration, while securing an ε-optimal solution simultaneously. Besides, it is an ordered sequential method, *i.e.*, the evaluation points at successive iterations are increasing values of *x* belonging to [*a*, *b*] (Horst and Pardalos, 1995). Its steps are as follows:

Initialization

$k \leftarrow 1$;

$x_1 \leftarrow a+\varepsilon/L$;

$x_{opt} \leftarrow x_1$

$f_{opt} \leftarrow f(x_{opt})$

Evtushenko's saw-tooth cover

While $x_k < b$ **do**

$$x_{k+1} \leftarrow x_k + \frac{f_{opt} + 2\varepsilon - f(x_k)}{L};$$

If $f(x_{k+1}) > f_{opt}$ **then** $f_{opt} \leftarrow f(x_{k+1})$; $x_{opt} \leftarrow x_{k+1}$ **endif**;

$k \leftarrow k+1$

EndWhile.

In the worst case, the step size is equal to $\frac{2\varepsilon}{L}$ (the incumbent evaluation value is also the optimal value on hand) when the function is constant functions or all monotonously increasing functions (Horst and Pardalos, 1995). Even if part of the function value is increasing in some ranges, it takes considerable length of searching runtime and great number of iterations which is very

time-consuming.

Later, Huang and Yao (2007) transferred the objective function of the TFMSP into an univariate function with respect to T (eq. (5)) for a minimization problem, and proved that it is Lipschitz on the interval $[T_{\text{low}}, T_{\text{up}}]$. Recall that the *Lipschitz constant* is often unknown, however, Huang and Yao showed that it can be obtained on some interval by the maximum of its derivative in absolute value, and it is given by

$$L = L_0 + \sum_{i=1}^{m} L_i \tag{20}$$

where

$$L_0 = \frac{S}{T_{\text{low}}^2} \tag{21}$$

Noted that the derivative for $\frac{S}{T}$ is equal to $-\frac{S}{T^2}$, and $\left|-\frac{S}{T^2}\right|$ is maximal on $[T_{\text{low}}, T_{\text{up}}]$ for T_{low}. Similarly, the function $g_i(T)$ has maximum slope in absolute value at the junction point $\delta_i(k)$ (see eq. (10)). Although $g_i(T)$ is not differentiable at these junction points, the left and right-hand derivatives exist. By (8) and (10), it follows that the left and right-hand derivatives of $g_i(T)$ are given by $(k+1)\Phi_i'(k+1, \delta_i(k))$ and $-k\Phi_i'(k, \delta_i(k))$, respectively. It is easy to verify that

$$(k+1)\Phi_i'(k+1, \delta_i(k)) = -k\Phi_i'(k, \delta_i(k)) = n_i C_{2i}, \tag{22}$$

and hence

$$L_i = n_i C_{2i},\ i = 1, \ldots, m. \tag{23}$$

By combining (20), (21) and (23), the *Lipschitz constant* for the objective function of problem (P) on $[T_{\text{low}}, T_{\text{up}}]$ is given by

$$L = \frac{S}{T_{\text{low}}^2} + \sum_{i=1}^{m} n_i C_{2i} \tag{24}$$

In addition, Huang and Yao apply the algorithm of Evtushenko as the Lipschitz optimization tool with a *dynamic Lipschitz constant* which was first proposed by Wildeman and Dekker (1997); after each function evaluation (going from left to right), the *Lipschitz constant* is recomputed as we take the current searching step to be the new T_{low}, denoted by L_c. Therefore, a smaller *Lipschitz constant* is generated due to a larger lower bound, and a larger searching step can be taken, which will speed up the searching process.

Next, we are going to propose the procedure and detail situations that can expedite the Evtushenko algorithm with *dynamic Lipschitz constant* for the TFMSP.

5.2 A procedure for expediting the Evtushenko algorithm

In this section, we propose a procedure which can speed up the search in the Evtushenko algorithm. The procedure can be separated into three parts: the searching at starting point, the searching during finding a local minimum, and the searching after a local minimum is located. By graphically displaying the possible situations in these three parts, we have full discussion in subsection later. The rationale to perform the procedure is as follows.

It is important to realize that with a given T, one could obtain a specific set of $\mathbf{k}=\{k_1,k_2,\ldots,k_m\}$ (see eq.(11) for each k_i), and each with a corresponding objective function curve

$$\gamma(\mathbf{k},T) := \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i,T). \qquad (25)$$

Namely, for a given T, we pick up a particular $\Phi_i(k_i,T)$ from $g_i(T)$ of each group, and these $\Phi_i(k_i,T)$ can form a unique $\gamma(\mathbf{k},T)$. One should notice that the optimal objective function $\Gamma(T)$ is exactly the lower bound of these $\gamma(\mathbf{k},T)$ on T-axis. Therefore, for each $\mathbf{k}$ set, its local minimum $\breve{T}(\mathbf{k})$ could be calculated by equating the derivative of the $\Gamma(T)$ function to zero. The $\breve{T}(\mathbf{k})$ is given by

$$\breve{T}(\mathbf{k}) = \sqrt{\frac{S + \sum_{i=1}^{m} \frac{n_i C_{1i}}{k_i}}{\sum_{i=1}^{m} n_i C_{2i} k_i}}. \qquad (26)$$

Since such a local minimum obtained from a specific **k** set could be a candidate for the optimal solution, we are motivated to search $\breve{T}(\mathbf{k})$ consecutively, which can reduce the computational efforts of Evtushenko algorithm between a given T and $\breve{T}(\mathbf{k})$ of a particular **k** set.

Besides, it is easy to learn that for a given T and its corresponding set of multipliers, say, $\mathbf{k}^0$, a particular $\breve{T}(\mathbf{k}^0)$ can be obtained. Therefore, if **k** set corresponding to $\breve{T}(\mathbf{k}^0)$ has changed, say, $\mathbf{k}^1$, we know that $\breve{T}(\mathbf{k}^0)$ is not a candidate of optimal solution because the **k** set is unique for each local minimum of $\Gamma(T)$, and we set $T=\breve{T}(\mathbf{k}^0)$, $\mathbf{k}^1=k_i(\breve{T}(\mathbf{k}^0))$ for next iteration. On the other hand, if $\mathbf{k}^0 = \mathbf{k}^1$, the $\breve{T}(\mathbf{k}^0)$ must be a local minimum of $\Gamma(T)$, and its corresponding optimal objective function value, denoted by TC_{opt}, will be recorded and compared to the best-on-hand solution, denoted by TC_{best}.

Since such a procedure can locate the local minimum $\breve{T}(\mathbf{k})$ of $\gamma(\mathbf{k},T)$ directly with a given T value and its specific **k** set, and the procedure can be executed consecutively if we obtain a new set of **k** at $\breve{T}(\mathbf{k})$, it is obvious that the procedure can expedite the searching process compares to the Evtushenko algorithm. Its process is as follows.

1. Start from a given T, calculate $\mathbf{k}^0 = \{k_i(T)\}$ for each group i.
2. Calculate $\breve{T}(\mathbf{k}^0)$ and $\mathbf{k}^1$ at $\breve{T}(\mathbf{k}^0)$
3. If $\mathbf{k}^1=\mathbf{k}^0$, calculate the TC_{opt} at $\breve{T}(\mathbf{k}^0)$ and record it if it is less than the TC_{best}. If $\mathbf{k}^1 \neq \mathbf{k}^0$, set $T=\breve{T}(\mathbf{k}^0)$, repeat step 1.

Recall that $\Gamma(T)$ is the optimal objective function curve with piece-wise convex property, it is impossible for us to employ the proposed algorithm through the whole search range without interruption except for a monotonously decreasing curve (*i.e.*, we will stuck in a local minimum of $\Gamma(T)$). Therefore, during the search section with an ascending curve, we still have to use Evtushenko algorithm since the proposed algorithm is designed for locating the

local minimum of $\Gamma(T)$ and Evtushenko algorithm can search fast when the curve is ascending. We will discuss the proposed algorithm in three parts: the searching at starting point, the searching during finding a local minimum, and the searching after a local minimum is located, and have full discussion in the following subsection.

5.2.1 The proposed algorithm at the starting point

Apparently, we have to investigate the shape of the optimal objective function $\Gamma(T)$ at the lower bound since $\breve{T}(\mathbf{k})$ may be out of the lower bound if $\Gamma(T)$ is in ascending trend at T_{low}, as shown in Figure 4. An easy method to see the curve trend could be done by checking function values at adjacent point T_{low} and T_{low}^{+}. We use Evtushenko algorithm to obtain the searching iteration which is denoted by X**(***m***)** (*m*=1 to *n*) and we take T_{low}^{+} as X**(**1**)**, then calculate its TC_{opt}. If $TC_{\text{opt}}(T_{\text{low}}^{+})$ is larger than $TC_{\text{opt}}(T_{\text{low}})$, the curve $\Gamma(T)$ is ascending. By the usage of Evtushenko algorithm, we denote the following steps to be X(*m*+1)=X(*m*)+δ. The increment step is defined as $\delta = \dfrac{f - f_{opt} + 2\varepsilon}{L_c}$. The searching speed would become faster when the step-size is getting larger while *Lipschitz constant* become smaller, as well as the optimal value on hand f_{opt} is fixed and objective function value *f* is increasing.

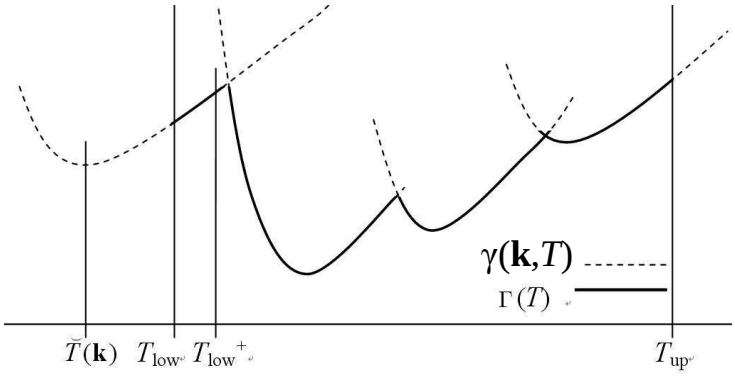

Figure 4. An increasing curve trend leads $\breve{T}(\mathbf{k})$ to be out of lower bound

On the other hand, if $TC_{\text{opt}}(T_{\text{low}}^{+})$ is smaller than $TC_{\text{opt}}(T_{\text{low}})$ or $TC_{\text{opt}}(\text{X}(m))$ is smaller than $TC_{\text{opt}}(\text{X}(m\text{-}1))$ after m^{th} iterations, then it is suitable to adopt the proposed algorithm. One may notice that the searching step will move to a local

minimum of $\Gamma(T)$, and the determination of the steps during searching for a local minimum are judged from the condition of one of the three possible situations as we discuss in the following subsection.

5.2.2 Three possible situations when searching for a local minimum

When TC_{opt} is decreasing, we can directly jump the searching step to $\breve{T}(\mathbf{k})$ of a particular **k** set, and replace the TC_{best} if the TC_{opt} at $\breve{T}(\mathbf{k})$ is better than the existing one. Whether the jump could happen depends on three possible situations when searching for a local minimum as we stated in the following.

1. Normal situation (Forward search)

Generally, we can apply the proposed algorithm consecutively without missing local minimum if only one component of the **k** set has changed at each jump step. This situation is mostly happened during the searching process with a descending curve, and we therefore named it as the normal situation. One demonstrative example is given in Figure 5, where the step X(6) is the local minimum $\breve{T}(\mathbf{k})$ corresponding to X(5) with $\mathbf{k}=\{k_i(\text{X}(5))\}$. As depicted in Figure 5, by checking the $\gamma(\mathbf{k},T)$ (dashed curve) with the lower envelop $\Gamma(T)$ (the bold curve), we found that the optimal **k** set at X(6) should be different from $\mathbf{k}=\{k_i(\text{X}(5))\}$, so we forward the searching step to the local minimum at X(7) corresponding to the new $\mathbf{k}=\{k_i(\text{X}(6))\}$ at X(6).

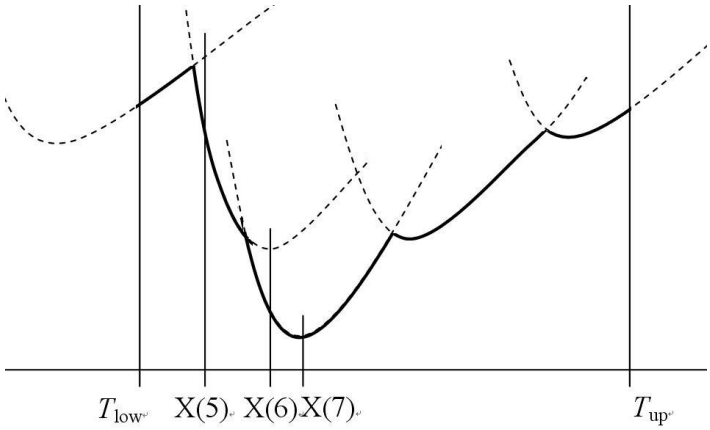

Figure 5. Normal situation of the proposed algorithm

2. Backward search without missing local minimum

Sometimes more than one component in the **k** set has changed during

searching from current step (a given T) to its $\breve{T}(\mathbf{k})$. In this case, a backward search mechanism will be triggered to prevent the search from missing the local minimum. As demonstrated in Figure 6, when the current step X(5) jump to its $\breve{T}(\mathbf{k})$, the corresponding new $\mathbf{k}=\{k_i(\mathrm{X}(6))\}$ will in term lead to a jump backward to X(7). Likewise, the new $\mathbf{k}=\{k_i(\mathrm{X}(7))\}$ will again jump backward to X(8). Therefore, no local minimum will be missed. Note that, in this case, we will start next step directly from $\mathrm{X}(9)=\mathrm{X}(6)+(f-f_{\mathrm{opt}}+2\varepsilon)/L_c$, which will save the effort for not to duplicate the searching within X(8) to X(6).

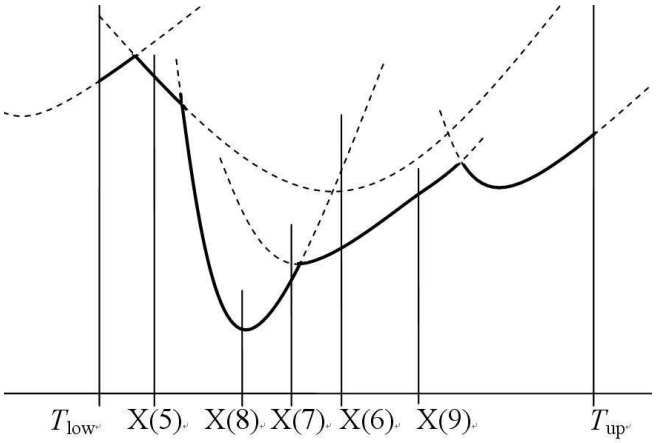

Figure 6. Backward search without missing local minimum

3. Missing local minimum

However, there were times when more than one component in the **k** set has changed during searching and the local minimum will be missed. As shown in Figure 7, when the searching move from the current step X(5) to its $\breve{T}(\mathbf{k})$ (*i.e.*, X(6)), more than one component in the **k** set has changed. However, unlike the situation shown in Figure 6, the searching will forward to X(7) since the $\Gamma(T)$ is descending, the local minimum located between X(5) and X(6) will not be detected.

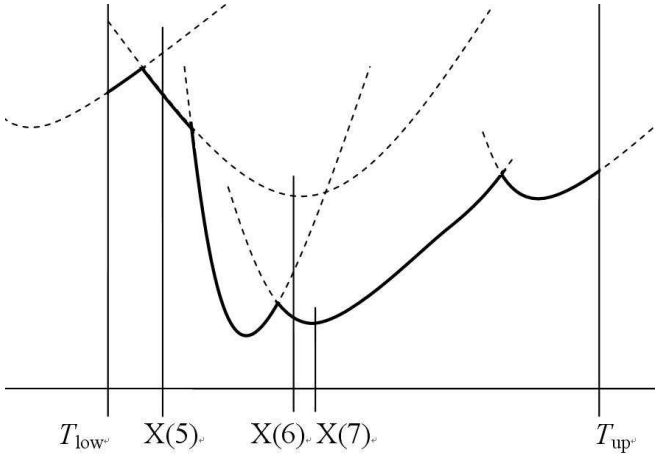

Figure 7. Forward search missing local minimum

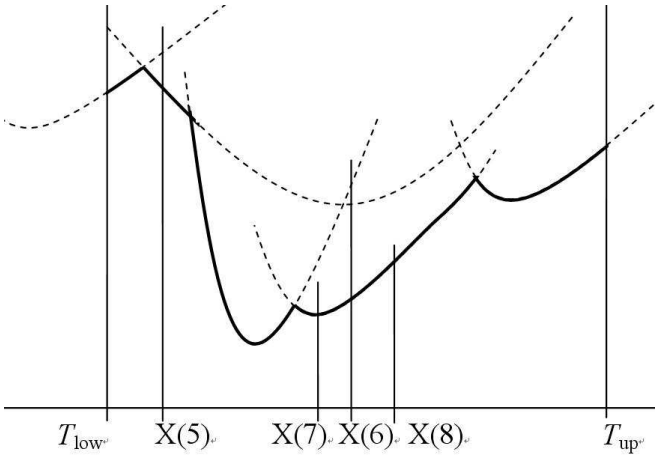

Figure 8. Backward search missing local minimum

Another possible situation of missing local minimum is shown in Figure 8, which depicts that when the searching move from X(5) to X(6), it will only move backward to X(7), and then move forward to X(8)=X(6)+(f-fopt+2ε)/Lc. Hence, one local minimum will be missed.

5.2.3 Two possible situations when a local minimum is located

Obviously, the current step must be moved to a local minimum of $\Gamma(T)$ if it has the same **k** set compare to the previous step, and we will record the current step and the **k** set if the TC_{opt} is lower than the TC_{best}. Besides, due to the characteristic of piece-wise convex property, the TC_{opt} is fluctuating, it may increase or decrease when the next step moves forward on T-axis after locating a local minimum. If it is increasing, we will keep searching by Evtushenko algorithm until it is smaller than previous step so as to make sure the $\Gamma(T)$ is

descending and then adopt the proposed algorithm. If it is decreasing, the proposed algorithm will be triggered for finding another local minimum again. Note that the next step after locating a local minimum is moved forward for $\frac{2\varepsilon}{L_c}$ by Evtushenko algorithm since $f=f_{opt}$.

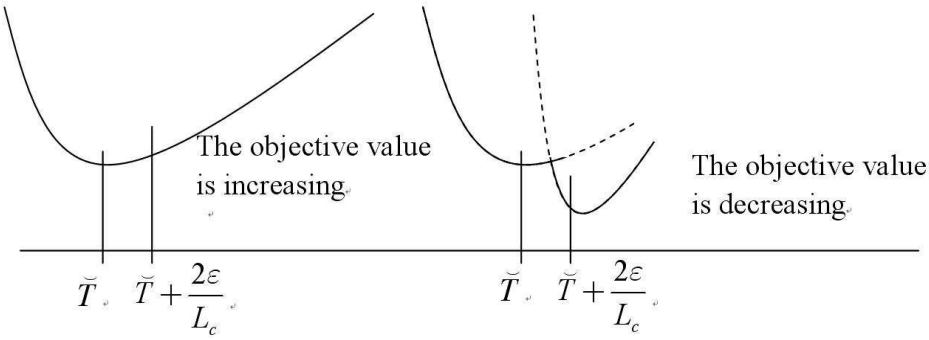

Figure 9. Two possible situations when a local minimum is located

Basically, subsection 5.2.1, 5.2.2 and 5.2.3 contain the main concepts of the proposed algorithm. It shows that the Evtushenko algorithm can be improved within the decreasing sections of TC_{opt}. We summarize the procedures as follows.

1. Check the curve trend by comparing $TC_{\text{opt}}(T_{\text{low}})$ and $TC_{\text{opt}}(T_{\text{low}}{}^{+})$, if it is increasing, use Evtushenko algorithm to search along the T-axis until the function value is decreasing.
2. While TC_{opt} is decreasing, the proposed algorithm is then adopted to find $\breve{T}$ (**k**) of different **k** sets consecutively, and it will move to a local minimum of $\Gamma(T)$.
3. While a local minimum is located, we use Evtushenko algorithm to restart the search and proceed with step size $\frac{2\varepsilon}{L_c}$ to the next step. Comparing $TC_{\text{opt}}(\breve{T})$ with $TC_{\text{opt}}(\breve{T}+\frac{2\varepsilon}{L_c})$ to check if the curve is ascending or descending. If TC_{opt} is increasing, adopt Evtushenko algorithm until TC_{opt} is decreasing. Otherwise, go to step 2.

   However, the right side of Figure 9 shows that the **k** set may have changed

within a very tiny range, which implies the possibilities of infinitesimal jumping distance of current step and its $\breve{T}$, and this lead to the discussion in section 5.3.

5.3 A fine-tune search procedure

As we mentioned above, the Evtushenko algorithm can be enhanced within TC_{opt} decreasing area since we can forward the searching step directly to its local minimum $\breve{T}(\mathbf{k})$ with specific **k** set. However, the function curves $\gamma(\mathbf{k},T)$ with different **k** sets sometimes make a over-frequent fluctuating behavior of the $\Gamma(T)$ within some tight sections. Such behavior will lead to numerous searching iterations since the backward search mechanism would be performed to prevent from missing the local minimum when the step propelled by Evtushenko algorithm is across from descending curve to ascending part.

However, the backward search is unnecessary because the quality of the solution could be secured by Evtushenko algorithm even if a local minimum is missed. Moreover, for a given *T*, if the step propelled by Evtushenko algorithm is larger than its $\breve{T}(\mathbf{k})$, it is reasonable to choose Evtushenko algorithm in the view of searching efficiency. Therefore, in order to meliorate the proposed algorithm, we propose a fine-tune search procedure based on slope-checking for current step and step-size comparison mechanisms that can eliminate unnecessary steps and adopt an efficient next step while remain the quality of solution simultaneously.

Here, we take the marching step-size into account by two methods: One is the distance between current step and local minimum $\breve{T}(\mathbf{k})$ corresponding to its **k** set, we named it the “jumping step size” since the movement on *T*-axis is like a long jump, and the other is the step size predicted by Evtushenko algorithm, we named it the “Evtushenko step size”. The following are four situations we may encounter:

1. **Evtushenko step size larger than jumping step size with negative slope.**

   If the step size determined by Evtushenko algorithm is larger than the jumping step size and the current step has negative slope, as shown in

Figure 10, we adopt Evtushenko step size as the next step. In this case, the quality of the solution is still guaranteed, since the Evtushenko algorithm can secure an ε-optimal solution.

2. **Evtushenko step size smaller than jumping step size with negative slope.**

   If the movement from current location to $\breve{T}(\mathbf{k})$ is larger than the step size determined by Evtushenko algorithm, as shown in Figure 11, then we take jumping step size as the next marching step. Since the jumping step size is generally larger than the Evtushenko step size, the searching during the section with negative slope can normally be speed up. This is actually the key mechanism for shortening the searching procedure by using our proposed algorithm instead of the Evtushenko algorithm.

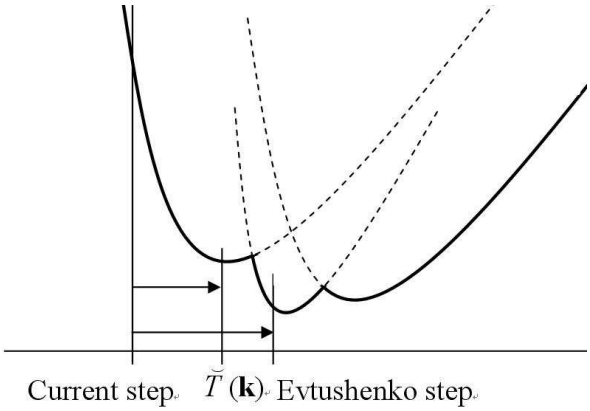

Figure 10. The step size determined by Evtushenko algorithm is larger than the jumping step size

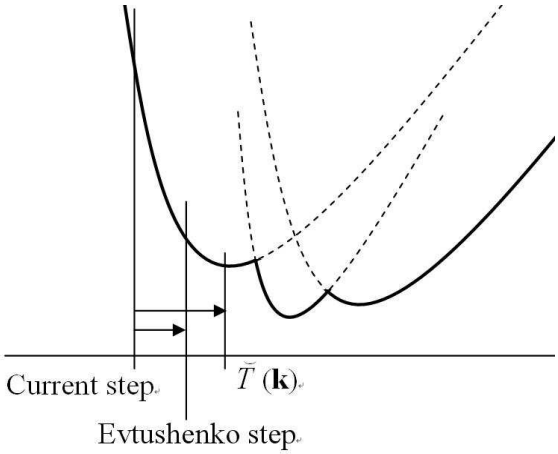

Figure 11. Evtushenko step size smaller than the jumping step size

3. Current step is propelled by the proposed algorithm with positive slope

If the slope of the objective function at the current searching step is positive, and it is also the local minimum $\breve{T}(\mathbf{k})$ corresponding to the previous step, as shown in Figure 12, then it is unnecessary to compare the step size since the jumping step size is always negative (moving backward). Although the Evtushenko algorithm should be adopted to proceed in ascending trend, we had the $\breve{T}(\mathbf{k})$ of current step first priority to be the next step since the proposed algorithm is still in progress, i.e., there is no local minimum of $\Gamma(T)$ has been confirmed. Note that, this is exactly the backward search situation, and it will keep searching until detecting a local minimum.

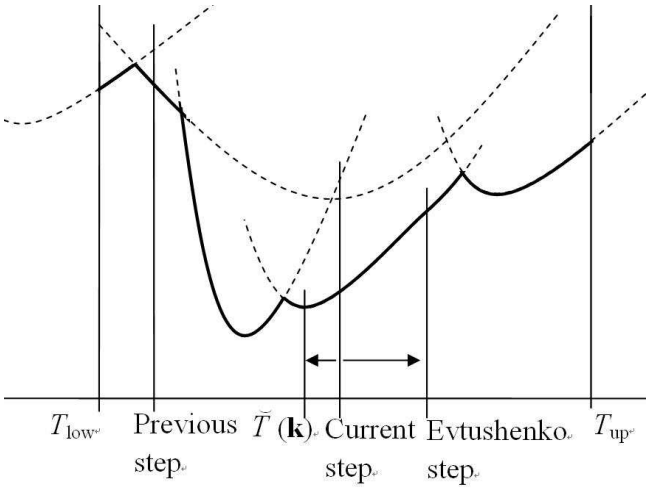

Figure 12. Use $\breve{T}(\mathbf{k})$ to be the next step while the proposed algorithm is still in progress (Backward search situation)

4. Current step is propelled by Evtushenko algorithm with positive slope

If the Evtushenko step size propels the step from negative to positive slope, then there must exists at least one local minimum between previous step and current step, as shown in Figure 13. In such a case, a backward search is not needed since Evtushenko algorithm can obtain an ε-optimal solution, and we proceed with Evtushenko steps.

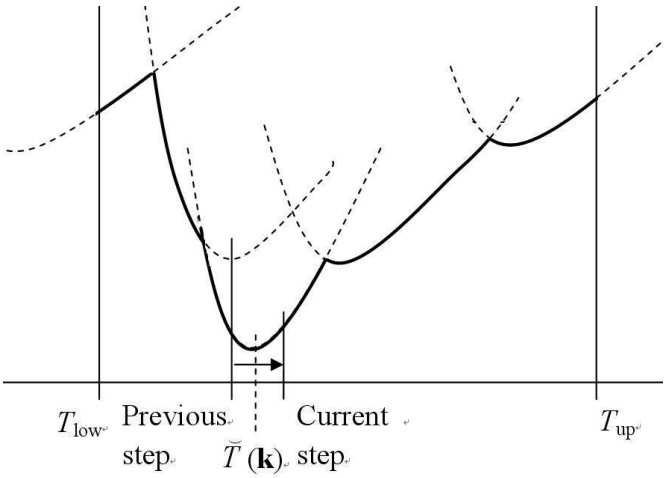

Figure 13. Backward search for $\breve{T}$ (k) is not needed while current step is propelled by Evtushenko algorithm and across local minimum

In a word, if the current step has negative slope, a larger step size is preferred to expedite the searching process. Otherwise, a checking procedure for whether the current step is propelled by Evtushenko algorithm is required in order to eliminate the excess backward search. Therefore, by checking the slope at current step and the step size, the searching efficiency can be assured as well as the quality of solution. We propose a flow chart to help us judging the next step in Figure 14.

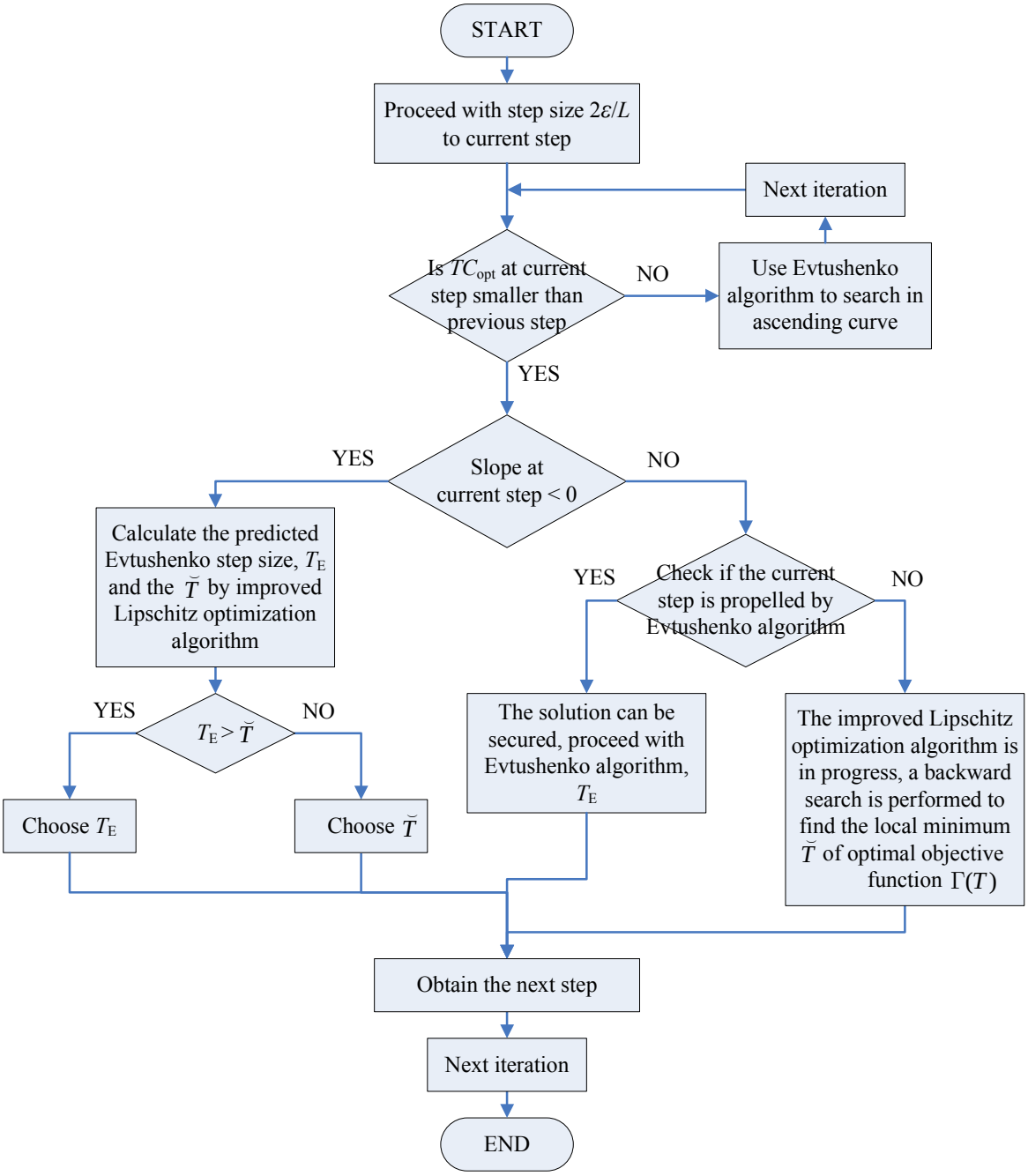

Figure 14. A flow chart for determination of the next step

5.4 A summary of the improved Lipschitz optimization algorithm

According to the discussion above, we can assure that the improved Lipschitz optimization algorithm is reasonable to have a better searching efficiency in a function value decreasing condition. Besides, duplicating search work can be avoided by identifying a backward search, and the searching efficiency can be further proved since we eliminate excess backward searching iterations caused by the steps propelled by Evtushenko algorithm from descending curve to ascending section. Also, we take into account the step size

and slope of current iteration to decide the next best T place so as to expedite the searching speed. Therefore, we are motivated to examine how significant affect can the improved Lipschitz optimization algorithm reach. We hereby propose an improved Lipschitz optimization algorithm as follows.

Relaxation Begin

Step 1. Obtain $T^{(R)}$ by eq.(19), set $v(R)$ as the solution of (R_1) at $T^{(R)}$.

Step 2. **if** $T^{(R)} \geq x_m^*$, set $\mathbf{k} = [1,1,\cdots,1]$ and $T^{(R)}$ the optimal solution for the problem (P).

Step 3. **if** $T^{(R)} < x_m^*$, obtain $\mathbf{k}=\{k_i(T^{(R)})\}$ by eq.(11) and obtain a feasible solution $v(FP)$ for the problem (P) at $T^{(R)}$.

Step 4. **if** $v(FP)$-$v(R) \leq \varepsilon v(R)$, **then** set $v(FP)$, $T^{(R)}$, and $\mathbf{k}=\{k_i(T^{(R)})\}$ as an ε-optimal solution. **else** $v(FP)$-$v(R) > \varepsilon v(R)$, use bisection search to locate T_{low} and T_{up} by finding two values of T where the objective function of (R_1) equals $v(FP)$.

End

Improved Lipschitz optimization algorithm Begin
(*Initiation setting and check the curve trend*)

TC_{best} the best known solution on hand

TC current optimal objective function value

TC_{pre} optimal objective function value of previous step

TC^* the optimal objective function value

T^* the optimal T

$\mathbf{k}^*$ the optimal k set

X(m) searching step at m iteration

T_{cur} the current searching step

L_c dynamic Lipschitz constant

T_{E} Evtushenko step

$\breve{T}$ the step of proposed algorithm

slope slope at current searching step

propelbyEv index of whether current step is propelled by Evtushenko

algorithm

Step 1. Initiation of the improved Lipschitz optimization algorithm:

Set $TC_{\text{best}}= v(FP)$, $TC^{*}= v(FP)$, $T^{*}= T^{(R)}$, $\mathbf{k}^{*}=\{k_i(T^{(R)})\}$, start the search from the lower bound T_{low}. Set $T_{\text{cur}}=T_{\text{low}}$. Obtain $\mathbf{k}=\{k_i(T_{\text{cur}})\}$, get $TC=TC(\mathbf{k}, T_{\text{cur}})$, set $TC_{\text{pre}}=TC$.

Step 2. Set m=1, compute L_c at T_{cur}, set $\text{X}(1)= T_{\text{cur}} +\varepsilon/L_c$, obtain $\mathbf{k}=\{k_i(\text{X}(1))\}$, get $TC=TC(\mathbf{k}, \text{X}(1))$.

(*Use Evtushenko algorithm for function value increasing area*)

Step 3. **While** $TC > TC_{\text{pre}}$, set $T_{\text{cur}}=\text{X}(m)$, Compute L_c at current search step T_{cur}, set $m=m+1$, $\text{X}(m) = T_{\text{cur}} + \dfrac{TC - TC_{\text{best}} + 2\varepsilon}{L_c}$

Step 4. **if** $\text{X}(m) > T_{\text{up}}$, **break, else** set $T_{\text{cur}}=\text{X}(m)$, $TC_{\text{pre}}=TC$, obtain $\mathbf{k}=\{k_i(T_{\text{cur}})\}$, get $TC=TC(\mathbf{k}, T_{\text{cur}})$, go to **Step 3**.

(*Initiation setting finished and having a decreased function value. Start to adopt the proposed algorithm*)

Step 5. **While** $\text{X}(m) \le T_{\text{up}}$, set $T_{\text{cur}}= \text{X}(m)$, obtain $\mathbf{k}=\{k_i(T_{\text{cur}})\}$, get $TC= TC(\mathbf{k}, T_{\text{cur}})$

Step 6. if $TC < TC^{*}$, set $TC^{*}=TC$, $T^{*}= T_{\text{cur}}$, $\mathbf{k}^{*}=\{k_i(T_{\text{cur}})\}$

Step 7. compute *slope* at T_{cur}, obtain $\breve{T}$ by eq.(25), obtain Evtushenko step $T_{\text{E}} = T_{\text{cur}} + \dfrac{TC - TC_{\text{best}} + 2\varepsilon}{L_c}$

(*Confirm a local minimum*)

Step 8. **if** $T_{\text{cur}}=\breve{T}$, get $TC=TC(\mathbf{k}, T_{\text{cur}})$, **do step 6**. Initialize *propelbyEv*= -1.

(*Confirm a forward search and use Evtushenko algorithm after locating a local minimum*)

Step 8-1. **if** $T_{\text{cur}}=\max(\text{X}(m))$, set $TC_{\text{pre}}=TC$, compute L_c, set $m=m+1$, $\text{X}(m) = T_{\text{cur}} + \dfrac{TC - TC_{\text{best}} + 2\varepsilon}{L_c}$. **if** $\text{X}(m) > T_{\text{up}}$, **break**, **else** set $T_{\text{cur}}=\text{X}(m)$, obtain $\mathbf{k}=\{k_i(T_{\text{cur}})\}$, get $TC=TC(\mathbf{k},T_{\text{cur}})$, set

propelbyEv=1, **do step 3 and 4.**

(*Confirm a Backward search and use Evtushenko algorithm after locating a local minimum*)

Step 8-2. **else** set $X_{max}=\max(X(m))$, $T_{cur}=X_{max}$, compute L_c, get $TC=TC(\mathbf{k},T_{cur})$, set $TC_{pre}=TC$, set $m=m+1$, $X(m)=T_{cur}+\frac{TC-TC_{best}+2\varepsilon}{L_c}$, **if** $X(m)>T_{up}$, **break**, **else** set $T_{cur}=X(m)$, obtain $\mathbf{k}=\{k_i(T_{cur})\}$, get $TC=TC(\mathbf{k},T_{cur})$, set *propelbyEv*=1, **do step 3 and 4**.

(***Check Evtushenko step size and slope of current step*, *and use* propelbyEv *as index to help judging the next step***)

Step 9. **elseif** $T_E<\breve{T}$ and *slope* < 0, set $m=m+1$, $X(m)=\breve{T}$, set *propelbyEv*=0

elseif $T_E\geq\breve{T}$ and *slope* < 0, set $m=m+1$, $X(m)=T_E$, set *propelbyEv*=1

elseif $T_E\geq\breve{T}$ and *slope* > 0 and *propelbyEv*=1, set $m=m+1$, $X(m)=T_E$

else set $m=m+1$, $X(m)=\breve{T}$.

Step10. go to **Step 5**.

End of the Improved Lipschitz optimization algorithm

6. Numerical experiments

In the first part of this chapter, we employ a numerical example to demonstrate the implementation of the improved Lipschitz optimization algorithm. Then, we use randomly generated instances to show that the improved Lipschitz optimization algorithm outperforms Huang and Yao's (2007) search procedure.

6.1 A demonstrative example

In this subsection, we hereby use a five-group example presented in Goyal and Gunasekaran's (1992) paper to demonstrate the implementation of the improved Lipschitz optimization algorithm. The data set of this five-group example is shown in Table 1.

In this example, we set the error allowance as $\varepsilon = 0.01\%$ for the improved Lipschitz optimization algorithm.

Table 1. The data set of the five-group example

m=5 S=800					
n_i	X_i	Y_i	a_i	b_i	s_i
10	0.8	0.90	80	3	198
24	0.6	0.95	50	2	192
30	0.4	0.85	90	1	193
16	0.6	0.95	85	1.5	205
12	0.5	0.94	95	2.5	204

First, we locate the optimal $T^{(R)}$ of the problem (R_1) by $T^{(R)}$=12.5349, and the $v(R)$ is given by \$2,020.66. Since $T^{(R)} < x_m^*$=21.20637, we use eq.(11) to get the set of optimal maintenance frequencies $\mathbf{k}(T^{(R)})$={1,1,2,1,1} to obtain a feasible solution for the problem (P) at $T^{(R)}$. Therefore, we have $v(FP)=\Psi(\mathbf{k}(T^{(R)}),\ T^{(R)})$=\$2,034.87. Since the error of the feasible solution is $(v(FP)-v(R))/\ v(R)$=0.7028%>ε=0.01%, it is not ε-optimal.

Next, we locate the bounds T_{low} and T_{up} by bisection search method to find two values of T where the objective function of (R_1) equals $v(FP)$. The search

range is obtained by T_{low} =9.8418 and T_{up} =14.9888. We note that the T_{up} is exactly the same value of T_{CC} by common cycle approach in this example. Besides, we set $\Psi(\mathbf{k}(T^{(R)}), T^{(R)})$=\$2,034.87 as the initial TC_{best} in order to enhance the searching speed, which is different to Huang and Yao's (2007) initial optimal solution $f_{opt}=\Psi(\mathbf{k}(T_{low}), T_{low})$.

Then, we apply the algorithm of Evtushenko as the improved Lipschitz optimization tool. The search start from X(1)=$T_{low}+\varepsilon/L_c$=9.8444, and we have the cost value $\Psi(\mathbf{k}(X(1)),X(1))$=\$2,083.52 (without considering the constant u). Therefore, by examining the total cost at T_{low} and X(1), it has decreased from $\Psi(\mathbf{k}(T_{low}), T_{low})$=\$2,083.61 to $\Psi(\mathbf{k}(X(1)),X(1))$=\$2,083.52. Obviously, it is suitable to apply the proposed algorithm.

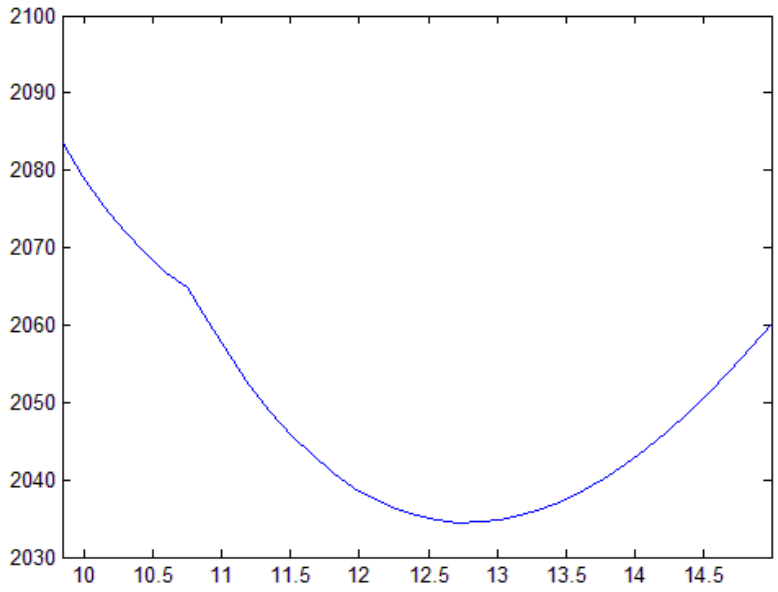

Figure 15. Optimal function value curve of demonstrative example

It is worth to note that the improved Lipschitz optimization algorithm take total 74 steps to complete the search, and only 3 steps (9.8444, 11.4009, and 12.7843) to locate the optimal solution at T^*=12.7843 with $\mathbf{k}^*$={1,1,2,1,1}. Therefore, we obtain the optimal average total cost by $Z=TC_{best}+u$=\$2,034.47+\$6,438.25=\$8,472.72. On the other hand, Huang and Yao's (2007) procedure locates T^*=12.7818 after 85 iterations and terminates after 157 steps, it is obvious to see that the improved Lipschitz optimization algorithm has significant reduction of searching iterations in cost decreasing

condition.

In fact, the proposed algorithm can obtain the *real* optimal solution if we did not miss the global minimum, whereas the traditional dynamic Lipschitz optimization algorithm can only secure an ε-optimal solution. Hence, we are motivated to generate random instances for showing that the improved Lipschitz optimization outperforms Yao and Haung's (2007) research.

6.2 Numerical results from random instances

In this subsection, we present a summary of our random experiments. We design our experimental settings by referring to the settings in Table 1 brought by Goyal and Gunasekaran's (1992) paper. We select six different values for the number of groups of vehicles (m = 3, 5, 7, 10, 25, 50), and seven different values for the fixed cost in each basic period T (S = 10, 50, 100, 200, 500, 750, 1000). This yields 42 combinations from these parameter settings. Then, for each combination, we randomly generate 1,000 instances by randomly choosing the values for X_i, Y_i, a_i, b_i and s_i by using uniform distribution functions. Table 2 indicates the ranges of these uniform-distributed random variables.

After randomly generating totally 42,000 instances, we solve each one of them by the proposed improved Lipschitz optimization algorithm as well as Huang and Yao's (2007) search procedure on a Core 2 Duo processor P8600 with 4GB RAM. We set the error allowance in each algorithm by ε=0.01%. We summarize our experimental results for the smaller-size (with m = 3, 5, 7) and larger-size (with m = 10, 25, 50) in Tables 3 and 4, respectively.

Table 2. The settings of the parameters in our random experiments

m	3, 5, 7, 10, 25, 50
S	10, 50, 100, 200, 500, 750, 1000
n_i	U[10-30]
X_i	U [0.4-0.8]
Y_i	U [0.9-0.95]
a_i	U [5-10]
b_i	U [1-3]
s_i	U [25-40]

In the view of searching efficiency, we calculate the average run time for each instance by repeating the search of 25 times. One may observe that the run time of Huang and Yao's (2007) search procedure is very fast. On the other hand, the proposed improved Lipschitz optimization algorithm solves the TFMSP with extremely short run time, and an average 71.79% run time improving rate to each problem. However, for the same value of *S*, with the increasing size of *m*, the run time improving rate is decreasing and become negative while *m*=50 and *S*=10, whereas the solution on average is better than traditional dynamic Lipschitz optimization algorithm.

On the aspect of solution quality, one should realize that we set the error allowance equal to 0.01% for both searching algorithms. However, the improved Lipschitz optimization algorithm sometimes miss the local minimum, so we compare the outcome and indicate the number of instances out of the 1,000 instances for each *m* and *S* combination that the difference greater than 0.01% in the last 3^{rd} column. The record shows that the differences of solutions obtained by two algorithms are all within 0.01%.

Besides, in the last two columns of Tables 3 and 4, we present the maximum error and average error of the improved Lipschitz optimization algorithm in percentages. Surprisingly, we observe that 37 out of 42 combinations have negative maximum error value, which means the proposed algorithm in these problems can always obtain better solutions than the traditional dynamic Lipschitz optimization algorithm. Although there are 5 combinations (when *m*=25 and 50) have positive maximum error, the largest one is only 0.000129% and all of these 5 problems have negative average error value, it is obvious that the proposed algorithm is better than the traditional one.

In addition, one may notice that, the fine-tune procedure can help us choosing the largest step in each iteration, which means the number of iterations of the proposed algorithm is equal to the Evtushenko algorithm's in the worst case. However, although the number of iterations is smaller than Evtushenko algorithm's when *m*=50 and *S*=10, the runtime is larger since we

apply step-size comparison and slope-checking mechanisms, and we consider it is the reason that the searching efficiency become negative. On the other hand, we observe that when *m* is increasing, the time improving rate is decreasing with different *S* values in different group, this is due to the fact that the larger group it is, the more step-size comparison and slope-checking are needed.

Table 3. Experimental results for the smaller-size (m = 3, 5 and 7) problems

		dynamic Lipschitz optimization algorithm		Improved Lipschitz optimization algorithm					
m	*S*	Avg. iterations	Avg. Run time (1)	Avg. iterations	Avg. Run time (2)	Rum time Improving Rate (%) ((1)-(2))/(1)	Objective function value		
							Error more than 0.01%	Max Error (%)	Avg. Error (%)
3	10	302	0.0269	89	0.0090	66.40%	0/1000	-2.30E-05	-3.10E-06
	50	243	0.0206	33	0.0043	79.13%	0/1000	-7.50E-06	-2.20E-06
	100	219	0.0230	21	0.0044	81.04%	0/1000	-7.10E-06	-2.20E-06
	200	221	0.0225	14	0.0038	83.04%	0/1000	-6.80E-06	-2.10E-06
	500	267	0.0248	7	0.0028	88.57%	0/1000	-5.90E-06	-1.80E-06
	750	326	0.0286	5	0.0025	91.36%	0/1000	-5.20E-06	-1.70E-06
	1000	385	0.0344	4	0.0025	92.72%	0/1000	-4.70E-06	-1.50E-06
5	10	446	0.0374	172	0.0157	58.15%	0/1000	-2.70E-05	-4.50E-06
	50	330	0.0317	68	0.0080	74.74%	0/1000	-9.90E-06	-2.40E-06
	100	293	0.0256	41	0.0051	79.93%	0/1000	-7.50E-06	-2.10E-06
	200	261	0.0230	26	0.0041	82.21%	0/1000	-7.10E-06	-2.10E-06
	500	263	0.0232	13	0.0031	86.65%	0/1000	-6.50E-06	-1.80E-06
	750	283	0.0235	8	0.0025	89.36%	0/1000	-5.70E-06	-1.70E-06
	1000	305	0.0288	5	0.0027	90.59%	0/1000	-5.30E-06	-1.70E-06
7	10	514	0.0492	225	0.0237	51.90%	0/1000	-3.80E-05	-6.30E-06
	50	413	0.0363	105	0.0102	71.96%	0/1000	-1.30E-05	-2.60E-06
	100	359	0.0320	63	0.0070	78.13%	0/1000	-8.10E-06	-2.10E-06
	200	313	0.0281	36	0.0050	82.37%	0/1000	-7.20E-06	-2.00E-06
	500	286	0.0256	19	0.0037	85.48%	0/1000	-6.70E-06	-2.00E-06
	750	285	0.0260	11	0.0031	88.08%	0/1000	-6.10E-06	-1.90E-06
	1000	293	0.0270	7	0.0029	89.26%	0/1000	-5.90E-06	-1.80E-06

Table 4. Experimental results for the larger-size (m = 10, 25 and 50) problems

		dynamic Lipschitz optimization algorithm		Improved Lipschitz optimization algorithm					
m	*S*	Avg. iterations	Avg. Run time (1)	Avg. iterations	Avg. Run time (2)	Rum time Improving Rate (%) ((1)-(2))/(1)	Objective function value		
							Error more than 0.01%	Max Error (%)	Avg. Error (%)
10	10	549	0.0503	275	0.0288	42.77%	0/1000	-4.40E-05	-8.60E-06
	50	511	0.0469	156	0.0151	67.78%	0/1000	-1.80E-05	-2.90E-06
	100	442	0.0414	100	0.0105	74.67%	0/1000	-9.30E-05	-2.10E-06
	200	373	0.0341	53	0.0064	81.37%	0/1000	-7.50E-06	-2.00E-06
	500	320	0.0294	27	0.0042	85.53%	0/1000	-6.80E-06	-2.00E-06
	750	308	0.0280	17	0.0035	87.44%	0/1000	-6.60E-06	-2.10E-06
	1000	301	0.0283	12	0.0032	88.71%	0/1000	-6.20E-06	-1.90E-06
25	10	566	0.0548	416	0.0515	5.97%	0/1000	6.00E-05	-1.80E-05
	50	642	0.0640	266	0.0311	51.43%	0/1000	2.70E-07	-5.90E-06
	100	646	0.0684	217	0.0253	62.99%	0/1000	-1.70E-05	-3.00E-06
	200	580	0.0586	154	0.0168	71.33%	0/1000	-1.00E-05	-1.90E-06
	500	413	0.0428	58	0.0078	81.86%	0/1000	-7.40E-06	-1.40E-06
	750	380	0.0371	41	0.0058	84.27%	0/1000	-7.20E-06	-1.60E-06
	1000	361	0.0352	33	0.0052	85.19%	0/1000	-7.00E-06	-1.60E-06
50	10	555	0.0615	494	0.0759	-23.40%	0/1000	0.000129	-1.10E-05
	50	653	0.0676	358	0.0481	28.85%	0/1000	4.45E-05	-1.00E-05
	100	688	0.0721	291	0.0374	48.06%	0/1000	1.49E-05	-6.40E-06
	200	711	0.0763	237	0.0289	62.06%	0/1000	-1.30E-05	-3.10E-06
	500	615	0.0657	143	0.0166	74.77%	0/1000	-8.40E-06	-1.20E-06
	750	504	0.0552	89	0.0113	79.46%	0/1000	-7.40E-06	-1.20E-06
	1000	435	0.0478	58	0.0081	82.98%	0/1000	-7.30E-06	-1.10E-06

According to numerical experiments results shown in Table 3 and 4 above, we conclude that the improved Lipschitz optimization algorithm outperforms Huang and Yao's (2007) research is evident.

7. Conclusion and future research

In this study, based on Huang and Yao's (2007) theoretical analysis on the mathematical model for the Transportation Fleet Maintenance Scheduling Problem (TFMSP), we propose an improved Lipschitz optimization algorithm with slope-checking and step-size comparison mechanisms to enhance the searching efficiency. As we start the searching from the starting point, the searching step can be speed up during the section with a descending objective function. The searching iterations as well as the runtime can be therefore reduced significantly.

Although the improved Lipschitz optimization algorithm sometimes may miss the local minimum, we enunciate that the quality of the solution is even better referring to our numerical experiments results. In addition, we considered it is pretty fair that the searching efficiency become negative while m=50 and S=10 since the group is large and the applied checking mechanisms, whereas the S is possibly much significant in real world. Therefore, we give a conclusion that the improved Lipschitz optimization algorithm is better than Huang and Yao's (2007) searching algorithm.

Since we are motivated to modify the traditional dynamic Lipschitz optimization algorithm, we did not take Yao and Huang's (2006) junction-point search algorithm into account. Besides, similar to most of other research works, the capacity of the maintenance team (or, the maintenance facility) is not a limited resource in this study. In order to make the research perfect resolved, one should compare with the junction-point search algorithm and take the capacity constraint of maintenance in the future research.

Reference

1. Amotz, B.N., Bhatia, R., Naor, J. and Schieber, B., “Minimizing Service and Operation Costs of Periodic Scheduling”, *Mathematics of Operations Research*, **27**, 518-544, 2002.
2. Anily, S., Glass, C.A. and Hassin, R., “The Scheduling of Maintenance Service”, *Discrete Applied Mathematics*, **82**, 27-42, 1998.
3. Anily, S., Glass, C.A. and Hassin, R., “Scheduling of Maintenance Services to Three Machines”, *Annals of Operations Research*, **86**, 375-391, 1999.
4. Bazaraa, M.S., Sherali, H.D., Shetty, C.M., *Nonlinear Programming: Theory and Algorithms*. 2nd Ed. John Wiley & Sons, New York, 1993.
5. Christer, A.H., Doherty, T., “Scheduling Overhauls of Soaking Pits”, *Operational Research Quarterly*, **28**, 4, 915-926, 1977.
6. Dekker, R., Wildeman, R., van der Duyn Schouten, F., “A review of multi-component maintenance models with economic dependence”, *Mathematical methods of Operations Research*, **45**, 411-435,1997.
7. Goyal, S.K., Gunasekaran, A. “Determining Economic Maintenance Frequency of a Transportation Fleet”, *International Journal of Systems Science*, **23**, 4, 655-659, 1992.
8. Goyal, S.K., Kusy, M.I., “Determining Economic Maintenance Frequency for a Family of Machines”, *Journal of the Operational Research Society*, **36**, 12, 1125-1128, 1985.
9. Hanssmann, F., “Operations research in production and inventory control”, John Wiley & Sons, New York, 1962.
10. Horst, R., Pardalos, P.M., *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1995.
11. Huang, J.Y., Yao, M.J., “A Dynamic Lipschitz Algorithm for Determining Economic Maintenance Frequency of a Transport Fleet”, *Journal of Information and Optimization Sciences*, **28**, 3, 357-376, 2007.
12. Luss, H., “Maintenance Policies When Deterioration can Be Observed by Inspections”, *Operations Research*, **24**, 2, 359-366, 1976.
13. Luss, H., Kander, Z., “Preparedness Model Dealing With N Systems Operating Simultaneously”, *Operations Research*, **22**, 1, 117-128, 1974.
14. Moudani W.E., Mora-Camino F., “A Dynamic Approach for Aircraft Assignment and Maintenance Scheduling by Airlines”, *Journal of Air Transport Management*, **6**, 4, 233-237, 2000.
15. Papadakos, N., “Integrated airline scheduling”, *Computers & Operations Research*, **36**, 1, 176-195, 2009.

16. Sriram, C., Haghani, A., “An Optimization Model for Aircraft Maintenance Scheduling and Re-assignment”, *Transportation Research Part A: Policy and Practice*, 37, 1, 29-48, 2003.

17. Sule, D.R., Harmon, B., “Determination of Coordinated Maintenance Scheduling Frequencies for a Group of Machines”, *AIIE Transactions*, **11**, 1, 48-53, 1979.

18. van Egmond, R., Dekker, R. Wildeman, R.E., “Correspondence: Determining Economic Maintenance of a Transportation Fleet”, *International Journal of Systems Science*, **26**, 9, 1755-1757, 1995.

19. Wildeman, R. E., Dekker, R., “Dynamic Influence in Multi-Component Maintenance”, *Quality and Reliability Engineering International*, **13**, 199-207, 1997.

20. Wildeman, R.E., Frenk, J.B.G., Dekker, R. “An Efficient Optimal Solution Method for the Joint Replenishment Problem”, *European Journal of Operational Research*, **99**, 2, 433-444, 1997.

21. Yanagi, S., “Iteration method for Reliability Evaluation for a Fleet System”, *Journal of the Operational Research Society*, **43**, 9, 885-896, 1992.

22. Yao, M.J., Huang, J.Y., “A New Optimal Search Algorithm for the Transportation Fleet Maintenance Scheduling Problem”, *Journal of the Operations Research of Japan*, **49**, 1, 33-48, 2006.