Optimization Methods and Software
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/goms20

Scheduling of transportation fleet maintenance service by an improved Lipschitz optimization algorithm

Ming-Jong Yao[a] & Jia-Yen Huang[b]

[a] Department of Transportation and Logistics Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu City 300, Taiwan, Republic of China

[b] Department of Information Management, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung City 41170, Taichung, Taiwan, Republic of China

Accepted author version posted online: 12 Aug 2013.Published online: 01 Oct 2013.

To cite this article: Ming-Jong Yao & Jia-Yen Huang (2014) Scheduling of transportation fleet maintenance service by an improved Lipschitz optimization algorithm, Optimization Methods and Software, 29:3, 592-609, DOI: 10.1080/10556788.2013.833615

To link to this article: http://dx.doi.org/10.1080/10556788.2013.833615

Optimization Methods & Software, 2014
Vol. 29, No. 3, 592–609, http://dx.doi.org/10.1080/10556788.2013.833615

Taylor & Francis
Taylor & Francis Group

Scheduling of transportation fleet maintenance service by an improved Lipschitz optimization algorithm

Ming-Jong Yao[a] and Jia-Yen Huang[b*]

[a]*Department of Transportation and Logistics Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu City 300, Taiwan, Republic of China;* [b]*Department of Information Management, National Chin-Yi University of Technology, No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung City 41170, Taichung, Taiwan, Republic of China*

(*Received 17 July 2012; final version received 4 August 2013*)

This study proposes a new and improved Lipschitz optimization algorithm to obtain an ε-optimal solution for solving the transportation fleet maintenance-scheduling problem. It employs a procedure based on slope-checking and step-size comparison mechanisms to improve the computational efficiency of the Evtushenko algorithm. Our numerical experiments used 36,000 randomly generated instances to compare the run time and solution quality of our proposed algorithm with the alternative approach in the literature. Our results indicate that the run time of the proposed algorithm could be significantly improved by more than 80% in over 50% of instances in our numerical experiments. We conclude that our proposed algorithm significantly improves the computational efficiency of the conventional Evtushenko algorithm.

Keywords: Lipschitz programming; optimization; Evtushenko algorithm; maintenance scheduling

1. Introduction

For the past decade, logistics service providers have faced the challenge of low profit margins due to intensive competition among peers and the skyrocketing price of crude oil. It is well known that the maintenance cost of their transportation fleet is a major operating cost for most logistics service providers. An appropriate maintenance schedule may not only lower the total operating costs, but also ensure a satisfactory utilization rate for the transportation fleet. Therefore, it is crucial to determine an optimal maintenance schedule for the fleet. The literature refers to this as 'the transportation fleet maintenance-scheduling problem' (the TFMSP). In the TFMSP, the decision-maker must determine a basic planning period and the frequency of maintenance for vehicles in each group to minimize the total costs incurred per unit time.

Some studies have dealt with the determination of economic maintenance scheduling in the fields of management science, operations research, and industrial engineering [5,8,11,12,17]. Many researchers have recently attempted to solve the problems regarding the scheduling of production facilities or machines [1–3,6,18]. However, few studies have examined the problem of determining the operating and maintenance schedules for a transportation fleet. The

*Corresponding author. Email: jygiant@ncut.edu.tw

objective functions in these studies, compared with the TFMSP, differ significantly in terms of their theoretical properties.

Some researchers have studied the aircraft fleet maintenance-scheduling problem (AFMSP). The AFMSP takes into account unique features such as heterogeneous fleets of aircraft and the regulations for routine inspections mandated by the Federal Aviation Administration. Flight hours and the number of take-off and landing cycles [16] are considered constraints. Additionally, the AFMSP is closely associated with the assignment of aircraft [13], maintenance routing, and crew scheduling [14]. Thus, the mathematical models for the AFMSP differ significantly from those of the TFMSP.

Goyal and Gunasekaran [7] designed a mathematical model for the TFMSP. They proposed a simple heuristic based on two equations that are derived by setting the first derivative of the objective function (namely, the average total cost) with respect to the decision variables to zero. Dekker *et al.* [6] argued that Goyal and Gunasekaran's heuristic does not guarantee an optimal solution. Huang and Yao [10] showed the objective function of the TFMSP is Lipschitz and proposed a dynamic Lipschitz optimization algorithm that finds an ε-optimal solution for this problem. They first solved the optimal solution for a relaxed problem of the TFMSP. If the solution was not within an allowance of ε, they located the lower and upper bounds of the search range using the relaxed solution by some line search methods (e.g. bisection [4]), and then applied the Evtushenko algorithm [9] as the Lipschitz optimization tool for finding an ε-optimal solution. Based on numerical results, they claimed their Lipschitz optimization algorithm offers a significantly better solution than Goyal and Gunasekaran's method.

However, we observed from the application of the Evtushenko algorithm in Huang and Yao [10] that the search step-size may become very small as the objective function value available is lower than the existing one, which leads to many search iterations and a significant run time. By taking into account some special characteristics of the objective function, we should be able to improve the run time of Huang and Yao's algorithm. We therefore propose a more efficient Lipschitz optimization algorithm for solving the TFMSP.

The rest of the paper is organized as follows. Section 2 presents the mathematical model and theoretical analysis of the TFMSP. Section 3 briefly reviews the Evtushenko algorithm proposed in Huang and Yao [10]. Section 4 presents the mechanism for speeding up the search process and discusses in detail three possible cases that may be encountered during the process. Section 5 presents a numerical example to demonstrate the implementation of the proposed search algorithm. The effectiveness of the proposed algorithm is verified by comparing the run time and the solution quality with the other existing solutions for a variety of parameter settings through randomly generated instances. Finally, Section 6 makes a number of concluding remarks.

2. The mathematical model

The following assumptions and notations have been made to formulate the problem.

2.1 *The assumptions and notation*

In order to compare our study with the approaches in the literature, we use the same notation and assumptions as those in Huang and Yao's paper [10], which are as follows:

Parameters

m number of groups of vehicles

S fixed cost incurred in each maintenance cycle

a_i fixed operating cost per unit of time for a vehicle of the ith group
b_i increased operating cost per unit of time for a vehicle of the ith group
s_i fixed cost of maintenance for a vehicle of the ith group
n_i number of vehicles in the ith group
X_i time required for maintenance work on each vehicle in the ith group
Y_i utilization factor of a vehicle of the ith group on the road

Decision variables
T basic maintenance cycle time
k_i an integer; k_iT gives the maintenance cycle time for the vehicles of the ith group

There are m groups of vehicles, and the number of vehicles is denoted as n_i for the ith group. In the TFMSP, the decision-maker plans the maintenance schedules for vehicle groups in a basic period, denoted by T (e.g. in days, weeks, or biweekly). The maintenance work of vehicles in a group is performed at a fixed, equal-time interval referred to as the 'maintenance cycle' for that group of vehicles. The vehicles in the ith group are sent for maintenance once in k_i basic periods, where k_i is a positive integer. Thus, k_iT is the maintenance cycle for vehicles in the ith group. We note that the model for the TFMSP is for preventive maintenance, and the model does not consider unplanned fleet vehicle failures.

2.2 *The mathematical model*

The TFMSP considers two categories of cost: the operating cost and the maintenance cost. A vehicle's operating cost depends on the length of the maintenance cycle and it is assumed to increase linearly with respect to time following 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 $f_i(t) = a_i + b_it$, 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 it takes X_i units of time for its maintenance work and Y_i is its utilization factor on the road, where X_i and Y_i are known constants. Accordingly, the actual time during which a vehicle can operate is equal to $Y_i(k_iT - X_i)$, and the total operating cost for a vehicle in group i is given by

$$\begin{aligned}\int_0^{Y_i(k_iT-X_i)} f_i(t)\,\mathrm{d}t &= \int_0^{Y_i(k_iT-X_i)} (a_i + b_it)\,\mathrm{d}t \\ &= Y_i(a_i - b_iX_iY_i)k_iT + 0.5Y_i^2k_i^2T^2 - X_iY_i(a_i - 0.5b_iX_iY_i).\end{aligned}$$

The average fixed cost of maintenance for a vehicle in group i is given by $s_i/(k_iT)$. 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. The average total cost is defined as

$$Z(\{k_1, k_2, \ldots, k_m\}, T) := \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T) + u,$$

where $\Phi_i(k_i, T) = n_iC_{1i}/k_iT + n_iC_{2i}k_iT$, $C_{1i} = s_i - X_iY_i(a_i - 0.5b_iX_iY_i)$, $C_{2i} = 0.5b_iY_i^2$ and $u = \sum_{i=1}^{m} n_iY_i(a_i - b_iX_iY_i)$. (Note: u is a constant because all the parameters are given in its expression.) Therefore, the mathematical model for the TFMSP can be expressed as problem (P_0):

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

Because 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). In addition, it is obvious that $C_{2i} > 0$. We define $\Psi(\{k_1, k_2, \ldots, k_m\}, T) := S/T + \sum_{i=1}^{m} \Phi_i(k_i, T)$ because u is a constant. Then, solving the problem (P_0) is equivalent to finding the optimal solution for the problem (P) as follows.

$$(P) \quad \inf_{T>0,\ k_i \in Z^+,\ i=1,\ldots,m} \Psi(k_1, k_2, \ldots, mk_m, T) = \inf_{T>0,\ k_i \in Z^+,\ i=1,\ldots,m} \left\{ \frac{S}{T} + \sum_{i=1}^{m} \Phi_i(k_i, T) \right\}. \tag{2}$$

In the TFMSP, we determine the basic period T and the frequency of maintenance for vehicles in each group $\{k_i\}$ to minimize the total costs incurred per unit of time for the decision-maker.

3. Review of Huang and Yao's [10] algorithm

Recall that this study focuses on improving the effectiveness of the Evtushenko algorithm proposed in Huang and Yao [10]. Therefore, we briefly review Huang and Yao's (2007) algorithm before proposing our alternative.

As presented by Huang and Yao [10], $\Phi_i(k_i, T)$ is strictly convex, and it has a minimum, for a given $k_i \in Z^+$, at $T = x_i^*/k_i$, where $x_i^* = \sqrt{C_{1i}/C_{2i}}$. By taking the optimal value of k_i at any value $T > 0$, i.e. $g_i(T) := \inf_{k_i \in Z^+}\{\Phi_i(k_i, T)\}$ for each ith group, Huang and Yao [10] transferred the objective function of the TFMSP in Equation (2) into an univariate function with respect to T, and rewrote the problem (P) as

$$(P_1) \quad \inf_{T>0} \Gamma(T) = \inf_{T>0} \left\{ \frac{S}{T} + \sum_{i=1}^{m} g_i(T) \right\}, \tag{3}$$

where the function $\Gamma(T)$ is the optimal objective function value curve with a piecewise convex property of an univariate function with respect to T. They proved that $\Gamma(T)$ is Lipschitz on the search range $[T_{\text{low}}, T_{\text{up}}]$. (We will have a further discussion on the search range later.) Therefore, they can apply the algorithm of Evtushenko [9] as the Lipschitz optimization tool with a dynamic Lipschitz constant to find an ε-optimal solution for the TFMSP.

Note that a relaxation method of the TFMSP plays an important role in Huang and Yao's [10] algorithm. By relaxing the constraints $k_i \in Z^+$ by $k_i \geq 1$, we obtain a relaxation of (P_1) in (3); namely (R_1), as follows:

$$(R_1) \inf_{T>0} h(T) = \inf_{T>0} \left\{ \frac{S}{T} + \sum_{i=1}^{m} g_i^{(R)}(T) \right\}. \tag{4}$$

The following lemma (which is Lemma 2 in [10]) indicates the location of the optimal solution $T^{(R)}$ for (R_1).

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

$$T^{(R)} = \sqrt{\frac{(S + \sum_{i=1}^{i^*} n_i C_{1i})}{\sum_{i=1}^{i^*} n_i C_{2i}}}. \tag{5}$$

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

optimal objective function value of the problem (P). Next, we denote $\mathbf{k}(T)$ as the set of optimal maintenance frequencies at a given value of T. Huang and Yao [10] presented (6) for obtaining the optimal multiplier $k_i^*(T) \in Z^+$ in $\mathbf{k}(T)$ as follows:

$$k_i^*(T) = \left\lceil -\frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{4C_{1i}}{C_{2i}T^2}} \right\rceil \tag{6}$$

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

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 $v(R)$ (e.g. within an allowance of ε), we have found an acceptable and feasible solution for (P). If it does not satisfy the required allowance is not good enough, Huang and Yao [10] suggested applying 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. They indicated that the lower and the upper bound on $T^{(P)}$ are given by the two values of T, where the objective function of (R_1) is equal to $v(FP)$. The bounds T_{low} and T_{up} may be easily located by some line search methods (e.g. bisection [4]). Therefore, if $v(R)$ is not close enough to the optimal solution, the Lipschitz optimization algorithm will be applied to solve the problem (P) on the interval $[T_{\text{low}}, T_{\text{up}}]$. The algorithm is a global-optimization approach when the objective functions are univariate [10]. More formally, a real-valued function f defined on a compact set $X \subseteq R^n$ that is said to be *Lipschitz* must satisfy the condition [9]

$$|f(x) - f(y)| \leq L||x - y||, \quad \forall x \in X, \ \forall y \in X, \tag{7}$$

where L is a constant (called *Lipschitz constant*) and $||\cdot||$ denotes the Euclidean norm, and $n = 1$ for the univariate. One may notice that a univariate function f is Lipschitz when the absolute difference of the objective function value for each pair of x and y is less than or equal to the product of the *Lipschitz constant* and the absolute difference of x and y. Huang and Yao applied the algorithm of Evtushenko as the Lipschitz optimization tool with a *dynamic Lipschitz constant*, which was first proposed by Wildeman and Dekker [18]. Note that the purpose of the Evtushenko algorithm is to make use of the information on the current best-known function value f_{opt} to determine the largest valid step-size for the next iteration, while simultaneously securing an ε-optimal solution. The step-size of the Evtushenko algorithm is as follows:

$$\delta = \frac{(f - f_{\text{opt}} + 2\varepsilon)}{L}. \tag{8}$$

In this study, the univariate function f is no other than $\Gamma(T)$. The Lipschitz constant L can be obtained on some interval by taking the maximum of its derivative in absolute value; Huang and Yao [10] showed that the Lipschitz constant L for the problem (P) on $[T_{\text{low}}, T_{\text{up}}]$ is given by

$$L = \frac{S}{T_{\text{low}}^2} + \sum_{i=1}^{m} n_i C_{2i}. \tag{9}$$

Before presenting our proposed algorithm, we next summarize Huang and Yao's [10] Lipschitz optimization algorithm.

Huang and Yao's [10] algorithm

(1) Use Equation (5) to find the optimal $T^{(R)}$ of the problem (R_1). If $T^{(R)} \geq x_m^*$, then $T^{(R)}$ is optimal for the problem (P) and $k_i = 1$, for $i = 1, \ldots, m$; stop.

(2) If $T^{(R)} < x_m^*$, we use (6) in Lemma 1 to derive the set of optimal maintenance frequencies $\mathbf{k}(T^{(R)}) = \{k_i(T^{(R)})\}$, and $(\mathbf{k}(T^{(R)}), T^{(R)})$ is a feasible solution of (P). If $\upsilon(FP) - \upsilon(R) \leq \varepsilon\upsilon(R)$, then we have secured an ε-optimal solution $\upsilon(FP) = \Psi(\mathbf{k}(T^{(R)}), T^{(R)})$; stop.
(3) If the solution available is not ε-optimal, locate the bounds T_{low} and T_{up} of the search range through a line search method by finding the two values of T where the objective function of (R_1) equals $\upsilon(FP)$.
(4) Apply the Evtushenko algorithm as the Lipschitz optimization tool with a dynamic Lipschitz constant as shown in (9) on the interval $[T_{\text{low}}, T_{\text{up}}]$ to solve an ε-optimal solution for the TFMSP.

Note that in Step (4) we need to evaluate the objection function after each iteration. A larger lower bound may lead to a smaller *Lipschitz constant* and consequently, a larger step-size δ. However, one may observe that in the worst case, the step-size is equal to $2\varepsilon/L$ (the incumbent evaluation of the optimal objective function value is also the best available) when the function is either a constant function or a monotonically decreasing function. When passing through some descending sub-ranges, the application of the Evtushenko algorithm may require one to conduct many iterations, which could be very time consuming.

In this study, we propose an improved version of the Evtushenko algorithm with better computational efficiency as the search passes through those decreasing sub-ranges. In the next section, we propose the procedure and discuss the situations in which we may expedite the Evtushenko algorithm for solving the TFMSP.

4. An improved Lipschitz optimization algorithm

The proposed algorithm makes use of the information of local minimum to expedite the search process, especially when the Evtushenko algorithm passes through some descending subranges during the search. The first part of this section focuses on elaborating our idea of utilizing the local minima. We also highlight the search at the starting point. The second part discusses three possible cases that one may encounter in the search. Finally, we summarize our proposed algorithm.

4.1 *Expedite the search by making use of the local minima*

In this section, we first present our idea about the local minima on the function $\Gamma(T)$ to expedite the search. We then discuss three possible conditions as the search approaches a local minimum.

4.1.1 *The mechanism for shortening the search procedure*

Recall that one may use (6) to obtain a vector of optimal multipliers $\mathbf{k}(\overline{T}) = (k_1^*(\overline{T}), k_2^*(\overline{T}), \ldots, k_m^*(\overline{T}))$ for a given $\bar{T}$. In addition, for a given vector of $\overline{\mathbf{k}} = (\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_m)$, one may easily locate its local minimum $\breve{T}(\overline{\mathbf{k}})$ by equating the derivative of the $\Gamma(T)$ function to zero where $\breve{T}(\overline{\mathbf{k}})$ is given by

$$\breve{T}(\overline{\mathbf{k}}) = \sqrt{\frac{S + \sum_{i=1}^{m} (n_i C_{1i}/\bar{k}_i)}{\sum_{i=1}^{m} (n_i C_{2i}\bar{k}_i)}}. \tag{10}$$

Such a local minimum $\breve{T}(\overline{\mathbf{k}})$ may serve as a candidate for the optimal solution. Thus, we are motivated to search $\breve{T}(\overline{\mathbf{k}})$ along the T-axis. In addition, as the search passes through some monotonically decreasing subranges, we may keep 'sliding down' a series of local minima to reduce

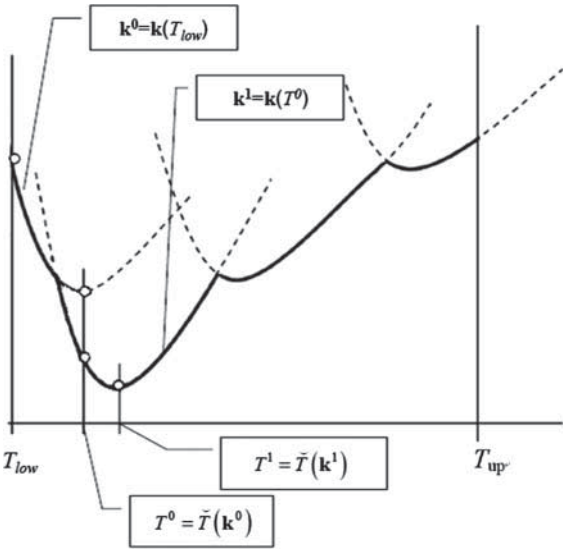

Figure 1. The idea of 'sliding down' a series of local minima.

the computational efforts and to improve the Evtushenko algorithm. Figure 1 depicts our suggested improvement. Assume that we start our search at T_{low}, and its corresponding set of optimal multipliers is $\mathbf{k}^0$, i.e. $\mathbf{k}^0 = \mathbf{k}(T_{\text{low}})$. We may obtain the local minimum at $\breve{T}(\mathbf{k}^0)$ by (10), and let $T^0 = \breve{T}(\mathbf{k}^0)$. Then we apply the same approach to derive the set of optimal multipliers $\mathbf{k}^1$ at T^0, and locate its corresponding local minimum $T^1 = \breve{T}(\mathbf{k}^1)$. The $\mathbf{k}$ set is unique for each local minimum of $\Gamma(T)$. When two consecutive $\mathbf{k}$ sets of multipliers are identical, it indicates that the search meets a local minimum of $\Gamma(T)$; that is, one may repeat the procedure until the optimal set of multipliers becomes invariant. One may observe that in following the above steps, the search will continue to slide down until it reaches a local minimum of the function $\Gamma(T)$. Following this rationale, one may use such a procedure to improve the computational efficiency of the Evtushenko algorithm during the search within some descending sub-ranges. Whenever the algorithm reaches a local minimum of the function $\Gamma(T)$, we shall record and compare the objective function value and try to update the best available solution.

4.1.2 *Three conditions approaching a local minimum*

When the search reaches the subranges where $\Gamma(T)$ is decreasing, one should employ the proposed 'sliding down' procedure to locate a local minimum. Three possible conditions arise as the search approaches a local minimum.

Condition 1: Forward search

Generally, we can apply the proposed algorithm consecutively without missing a local minimum if only one component of the $\mathbf{k}$ set has changed at each jump step. This situation mostly arose during the search process with a descending curve, and we therefore regarded it as the normal situation. A demonstration example is given in Figure 2, where step $X(6)$ is the local minimum $\breve{T}(\mathbf{k})$ corresponding to $X(5)$ with $\mathbf{k} = \{k_i(X(5))\}$. As depicted in Figure 2, by checking the $\psi(\mathbf{k}, T)$ (dashed curve) with the lower envelope $\Gamma(T)$ (the bold curve), we found that the optimal $\mathbf{k}$ set

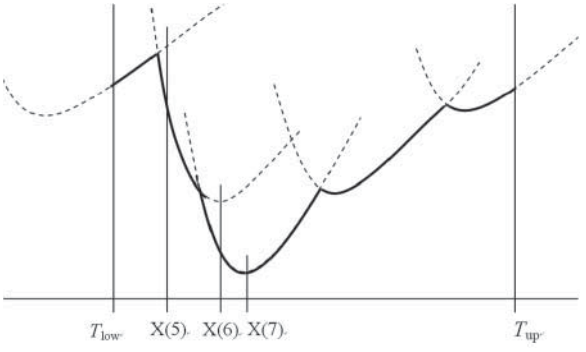

Figure 2. Normal situation of the proposed algorithm.

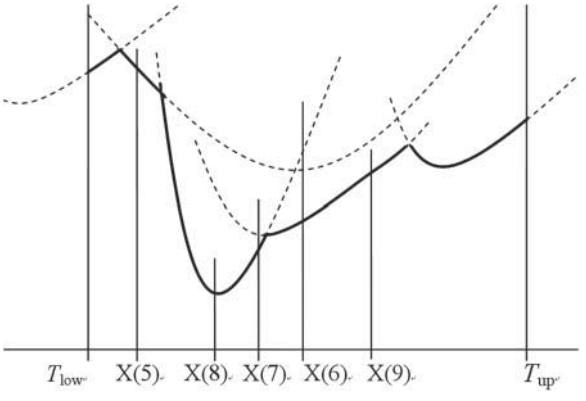

Figure 3. Backward search without missing a local minimum.

at $X(6)$ should be different from $\mathbf{k} = \{k_i(X(5))\}$, so we moved forward to the local minimum at $X(7)$ corresponding to the new $\mathbf{k} = \{k_i(X(6))\}$ at $X(6)$.

Condition 2: Backward search without missing a local minimum

Sometimes more than one component in the $\mathbf{k}$ set changes during a search from the 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 3, when the current step $X(5)$ jumps to its $\breve{T}(\mathbf{k})$, the corresponding new $\mathbf{k} = \{k_i(X(6))\}$ will in turn lead backward to $X(7)$. Likewise, the new $\mathbf{k} = \{k_i(X(7))\}$ will again move backward to $X(8)$. Thus, no local minimum will be missed. Note that, in this case, we will start the next step of the search directly from $X(9) = X(6) + (f - f_{\text{opt}} + 2\varepsilon)/L_{\text{c}}$, which will save us the effort of searching from $X(8)$ to $X(6)$.

Condition 3: Missing a local minimum during the search

There were situations in which two (or more) components in the $\mathbf{k}$ set were changed during the search and the local minimum will be missed, as shown in Figure 4, when the search moves from the current step $X(5)$ to its $\breve{T}(\mathbf{k})$ (i.e. $X(6)$). However, unlike the situation shown in Figure 3, the search moves forward to $X(7)$ because the $\Gamma(T)$ is descending; the local minimum located between $X(5)$ and $X(6)$ was not detected.

Another possible situation of a missing local minimum is shown in Figure 5, which depicts how when the search moves from $X(5)$ to $X(6)$, it turns only backward to $X(7)$, and then goes forward to $X(8) = X(6) + (f - f_{\text{opt}} + 2\varepsilon)/L_{\text{c}}$. In such a case, the search also misses a local minimum.

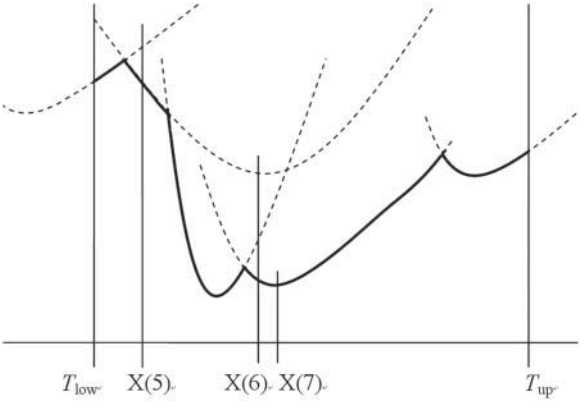

Figure 4. Forward search missing a local minimum.

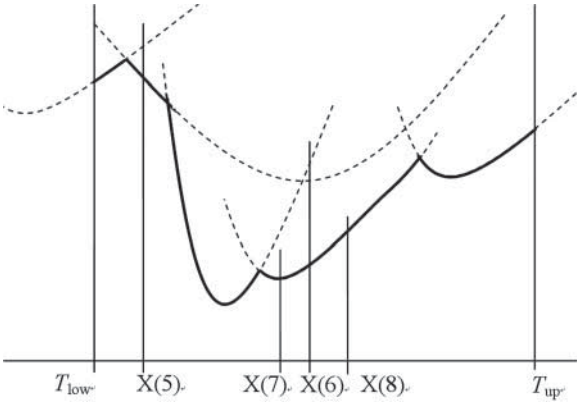

Figure 5. Backward search missing a local minimum.

Apparently, by starting the search at T_{low}, we have to examine if the function $\Gamma(T)$ is increasing or decreasing as T increases. We do this as follows. Denote $X(m)$ as the point located by the mth step of the Evtushenko algorithm, $m = 1, 2, \ldots$. Recall that the (conventional) Evtushenko algorithm goes from $X(m)$ to $X(m+1)$ by a step-length of $\delta = (f - f_{\text{opt}} + 2\varepsilon)/L_c$; namely, $X(m+1) = X(m) + \delta$. For simplicity in notation, we set $T_{\text{low}} \equiv X(0)$. Obviously, if $\Gamma(X(0)) < \Gamma(X(1))$, $\Gamma(T)$ increases, we should proceed with the Evtushenko algorithm until it detects that $\Gamma(T)$ no longer increases as T increases.

In contrast, if $\Gamma(X(0)) > \Gamma(X(1))$, then $\Gamma(T)$ decreases, and we should therefore seek a local minimum. We discuss this case further in the next subsection.

4.2 *Three possible cases when proceeding with the search*

Due to the piecewise convex property feature, the $\Gamma(T)$ may increase or decrease as the search moves from T_{low} toward T_{up}. In general, we may have three possible cases: (1) $\Gamma(T)$ increases, (2) $\Gamma(T)$ increases, and its slope at the current step is negative or zero, and (3) $\Gamma(T)$ increases, and its slope at the current step is positive. (Note: The three conditions discussed in Section 4.1 are manifested only in the latter two cases.) We present the details for these three possible cases next.

Case 1: $\Gamma(T)$ *increases*

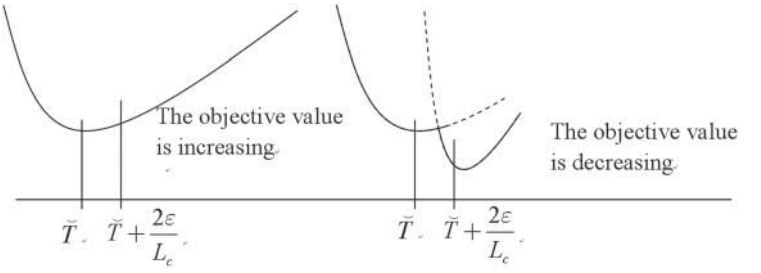

Figure 6. Two possible situations when a local minimum is located.

As $\Gamma(T)$ increases, the search moves with an upward momentum. In such a case, we apply the conventional Evtushenko algorithm for the search because its step-size $\delta = (f - f_{\text{opt}} + 2\varepsilon)/L_c$ is large (see Figure 6).

Case 2: $\Gamma(T)$ *decreases, and its slope at the current step is negative or zero*

As discussed in Section 4.1, we may improve the efficiency of the search by 'sliding down' to its local minimum $\breve{T}(\mathbf{k})$ with a specific $\mathbf{k}$ set as $\Gamma(T)$ decreases. Recall that we shall perform a backward search to prevent a missing local minimum between the current and the previous steps, as discussed in Conditions 2 and 3 in Section 4.1.

Note that $\Gamma(T)$ may 'fluctuate' frequently if many optimal $\mathbf{k}$ sets exist within a short range of T. Such behavior may trigger many iterations of a backward search to prevent missing the local minimum. However, a backward search would be unnecessary if the step-size recommended by the Evtushenko algorithm is larger than the step-size proposed by our improved algorithm. It is reasonable to adopt the step-size recommended by the Evtushenko algorithm in this case because the quality of the solution could be guaranteed (within an ε-% error allowance), even if a local minimum is missed.

We hope to eliminate unnecessary search steps to improve computational efficiency while retaining the quality of the solution. Therefore, we propose a procedure based on slope and step-size checking mechanisms. To facilitate our presentation, we name the step-sizes recommended by the Evtushenko algorithm and our improved algorithm 'the Evtushenko step-size' and 'the jumping step-size', respectively. We present our procedure for the two possible situations in Case 2 as follows:

(a) The Evtushenko step-size is larger than the jumping step-size. We adopt the Evtushenko step-size as the next step in this case because the Evtushenko algorithm guarantees an ε-optimal solution.
(b) The Evtushenko step-size is smaller than the jumping step-size. If the jumping step-size (i.e. the movement from the current location to $\breve{T}(\mathbf{k})$) is larger than the Evtushenko step-size, we use the jumping step-size in our proposed algorithm. Because the jumping step-size is usually larger than the Evtushenko step-size, our proposed algorithm is able to speed up the search when it encounters Case 2. This is actually the key to improving the computational efficiency of our proposed algorithm in comparison to the Evtushenko algorithm.

Case 3: $\Gamma(T)$ *decreases, and its slope at the current step is positive*

We discuss the two possible situations in Case 3 as follows.

(a) The current step is driven by the jumping step-size. Because $\Gamma(T)$ decreases, our proposed algorithm may employ a jumping step-size that is so large that it surpasses local minima in between. One may refer to Figure 7 for an example where $\Gamma(T)$ decreases, but its slope is

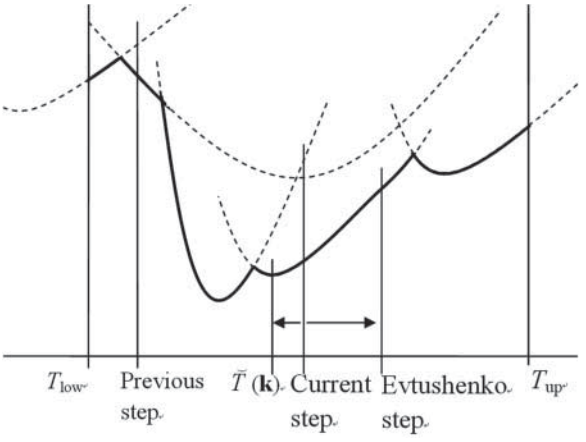

Figure 7. An example of a backward search situation with a decreasing objective function and a positive current slope.

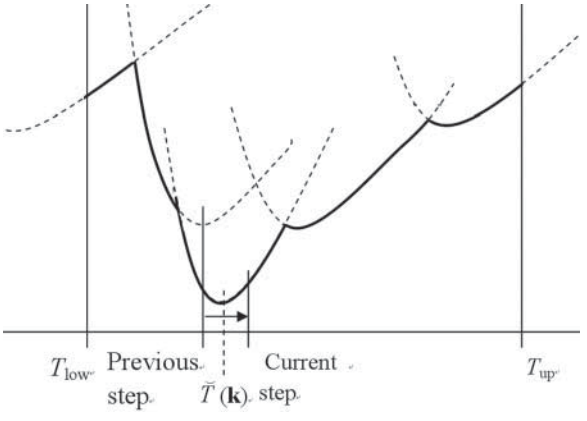

Figure 8. An example showing a current step propelled by the Evtushenko algorithm and across a local minimum.

positive. Here is Condition 2 of Section 4.1.2; we apply a backward search from the current step until a local minimum is detected. We note that such a backward search mechanism enables our proposed algorithm to find a better solution than the conventional Evtushenko algorithm in some instances of our numerical experiments in Section 5.

(b) The current step is driven by the Evtushenko step-size. If $\Gamma(T)$ decreases and the slope at the current step is positive, there is then at least one local minimum in between, as shown in Figure 8. In such a case, a backward search is unnecessary because the Evtushenko algorithm guarantees an ε-optimal solution. We will keep applying the Evtushenko algorithm to proceed with the search.

4.3 *A summary of our proposed algorithm*

Following the above discussion, we propose a new algorithm for improving the computational efficiency of the conventional Evtushenko algorithm. We present a flow chart of our proposed algorithm and summarize the pseudo-code as follows (see Figure 9).

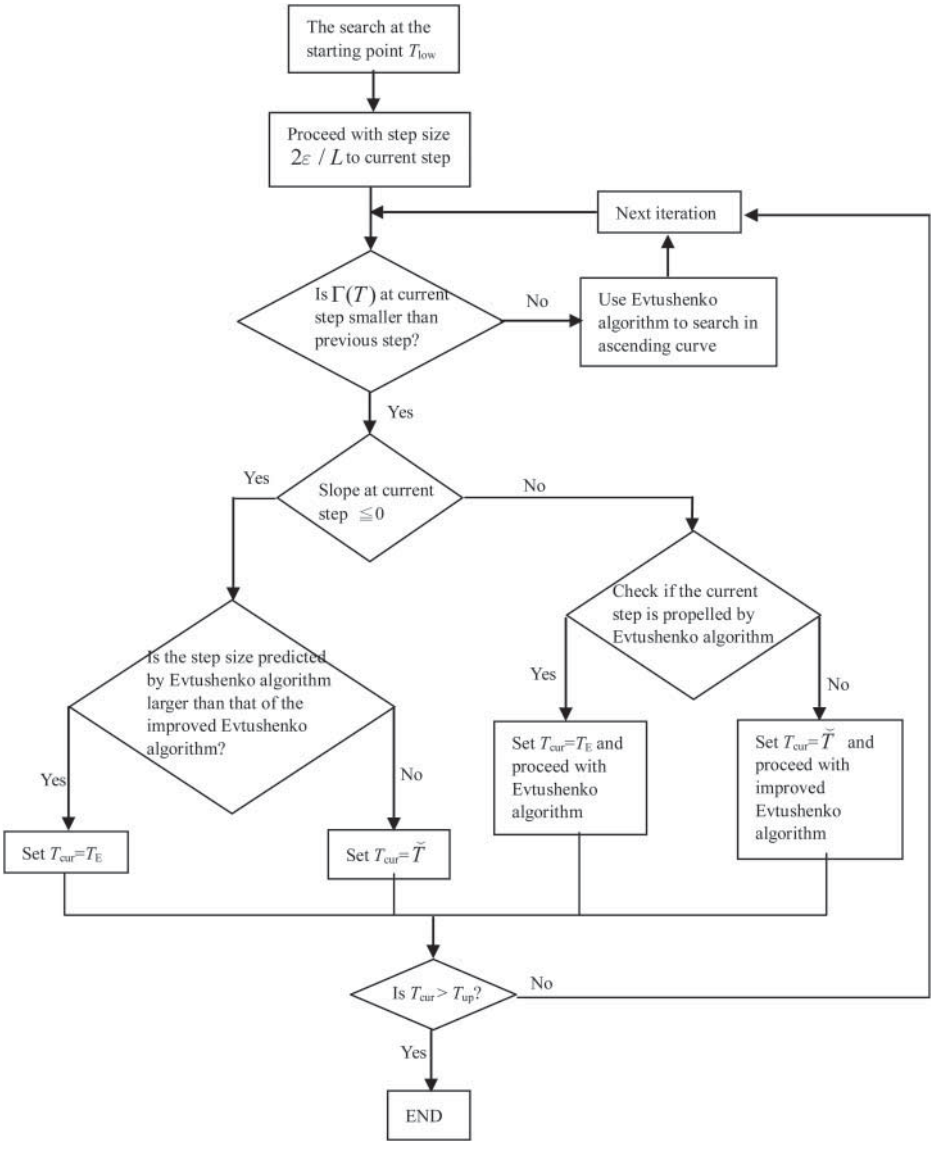

Figure 9. A flow chart of our proposed algorithm.

Notation definition:

TC_{best}	the best-known solution available
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
$X(m)$	search step at m iteration
T_{cur}	the current search step
L_{c}	dynamic Lipschitz constant
T_E	Evtushenko step

$\breve{T}$ the step of our proposed algorithm (it will end with a local minimum)
S_{cur} slope at current search step

set Υ_{E} to true if the current step is driven by the Evtushenko algorithm, otherwise set Υ_{E} to false
Our proposed 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 $TC_{\text{pre}} = TC$.

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

While $TC > TC_{\text{pre}}$
  set $T_{\text{cur}} = X(m)$, Compute L_{c} at current search step T_{cur}, set $m = m + 1$,
  $X(m) = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
  If $X(m) > T_{\text{up}}$
    stop
  else
    set $T_{\text{cur}} = X(m)$, $TC_{\text{pre}} = TC$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$
  Endif
EndWhile

While $X(m) \le T_{\text{up}}$
  set $T_{\text{cur}} = X(m)$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$
  If $TC < TC^*$, set $TC^* = TC$, $T^* = T_{\text{cur}}$, $\mathbf{k}^* = \{k_i(T_{\text{cur}})\}$, Endif
  Compute S_{cur}, obtain $\breve{T}$ by Equation (10), obtain Evtushenko step
    $T_{\text{E}} = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
  If $T_{\text{cur}} = \breve{T}$
    get $TC = TC(\mathbf{k}, T_{\text{cur}})$
    If $TC < TC^*$, set $TC^* = TC$, $T^* = T_{\text{cur}}$, $\mathbf{k}^* = \{k_i(T_{\text{cur}})\}$, Endif
    Initialize Υ_{E} to false
  Endif
  If $T_{\text{cur}} = \max(X(m))$,
    Set $TC_{\text{pre}} = TC$, compute L_{c}, set $m = m + 1$,
    $X(m) = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
  if $X(m) > T_{\text{up}}$
    stop
  else
    set $T_{\text{cur}} = X(m)$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$, set $\Upsilon_{\text{E}} = 1$
  Endif
  While $TC > TC_{\text{pre}}$
  set $T_{\text{cur}} = X(m)$, Compute L_{c} at current search step T_{cur}, set $m = m + 1$,
  $X(m) = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
  If $X(m) > T_{\text{up}}$
    stop
  else
    set $T_{\text{cur}} = X(m)$, $TC_{\text{pre}} = TC$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$
  Endif
  EndWhile
else
  set $X_{\max} = \max(X(m))$, $T_{\text{cur}} = X_{\max}$, compute L_{c}, get $TC = TC(\mathbf{k}, T_{\text{cur}})$,

set $TC_{\text{pre}} = TC$, set $m = m + 1$, $X(m) = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
if $X(m) > T_{\text{up}}$
stop
else
set $T_{\text{cur}} = X(m)$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$, set Υ_{E} to true
Endif
While $TC > TC_{\text{pre}}$
set $T_{\text{cur}} = X(m)$, Compute L_{c} at current search step T_{cur},
set $m = m + 1$, $X(m) = T_{\text{cur}} + (TC - TC_{\text{best}} + 2\varepsilon)/L_{\text{c}}$
If $X(m) > T_{\text{up}}$
stop
else
set $T_{\text{cur}} = X(m)$, $TC_{\text{pre}} = TC$, obtain $\mathbf{k} = \{k_i(T_{\text{cur}})\}$, get $TC = TC(\mathbf{k}, T_{\text{cur}})$.
Endif
EndWhile
Endif.
If $T_{\text{E}} < \breve{T}$ and $S_{\text{cur}} < 0$
set $m = m + 1$, $X(m) = \breve{T}$, set Υ_{E} to false
elseif $T_{\text{E}} \geq \breve{T}$ and $S_{\text{cur}} < 0$
set $m = m$+1, $X(m) = T_{\text{E}}$, set Υ_{E} to true
elseif $T_{\text{E}} \geq \breve{T}$ and $S_{\text{cur}} > 0$ and Υ_{E} to true
set $m = m$+1, $X(m) = T_{\text{E}}$
else
set $m = m + 1$, $X(m) = \breve{T}$
Endif
EndWhile

5. Numerical experiments

In the first part of this chapter, we employ a numerical example to demonstrate the implementation of our proposed algorithm. We then use randomly generated instances to show that our proposed algorithm significantly improves the computational efficiency of the conventional Evtushenko algorithm.

5.1 *A demonstration example*

In this subsection, we use a five-group example in Goyal and Gunasekaran [7] to demonstrate the implementation of our proposed algorithm. The error allowance is set as $\varepsilon = 0.01\%$. The data set of this five-group example is shown in Table 1.

In this example, we start by locating the optimal $T^{(R)}$ of the problem (R_1) at $T^{(R)} = 12.5349$ and set the $v(R)$ at \$2020.66. Because $T^{(R)} < x_m^* = 21.20637$, we use Equation (6) to obtain the set of optimal maintenance frequencies $\mathbf{k}(T^{(R)}) = \{1, 1, 2, 1, 1\}$ and to obtain a feasible solution for the problem (P) at $T^{(R)}$. We have $v(FP) = \Psi(\mathbf{k}(T^{(R)}), T^{(R)}) = \$2034.87$. Because the error of the feasible solution $(v(FP) - v(R))/v(R) = 0.7028\%$ is larger than $\varepsilon = 0.01\%$, it is not ε-optimal.

Next, using a bisection search method, we may set the search range at $T_{\text{low}} = 9.8418$ and $T_{\text{up}} = 14.9888$. Note that the T_{up} is exactly the same value of T_{CC} by the common cycle approach in this example. We set $\Psi(\mathbf{k}(T^{(R)}), T^{(R)}) = \$2034.87$ at the initial TC_{best} to expedite the search.

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

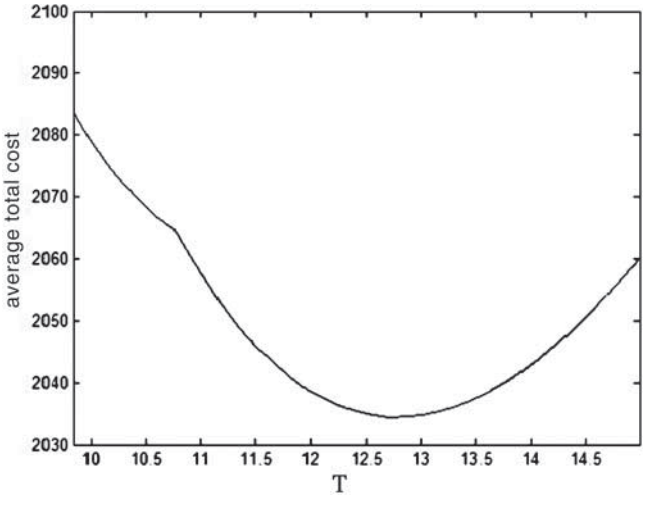

Figure 10. The optimal function value curve of the demonstration example.

The search starts from $X(1) = T_{\text{low}} + \varepsilon/L_c = 9.8444$ (by the traditional Evtushenko algorithm), and we have $\Psi(\mathbf{k}(X(1)), X(1)) = \$2083.52$ (without considering the constant u). Because the objective function value at $X(1)$ is less than that at T_{low}, i.e. $\Psi(\mathbf{k}(T_{\text{low}}), T_{\text{low}}) = \$2083.61$, it is suitable to apply our proposed algorithm to predict the next step.

It is worth noting that it takes our proposed algorithm 74 steps to complete the search, and the optimal solution located at $T^* = 12.7843$ with $\mathbf{k}^* = \{1, 1, 2, 1, 1\}$. Taking the constant u into account, we derive the optimal average total cost by $Z = TC_{\text{best}} + u = \$2034.47 + \$6438.25 = \$8472.72$. The optimal function value curve of the example demonstration is shown in Figure 10.

By using Huang and Yao's [10] search algorithm, we located the optimal solution at $T^* = 12.7818$ after 85 iterations and finished the complete search in 157 steps. It is clearly evident that our proposed algorithm is more efficient than Huang and Yao's [10] algorithm.

Our proposed algorithm can obtain the *real* optimal solution if we did not miss the global minimum, because the local minima will be located during the descending search range. In contrast, Huang and Yao's [10] search algorithm can only find an ε-optimal solution because the search is driven by a small step-size and it may sometimes surpass a local minimum.

5.2 *Numerical results from random instances*

Here, we use randomly generated scenarios to show that our proposed algorithm significantly improves the computational efficiency of the conventional Evtushenko algorithm. Again, one may compare our experimental settings to that of Goyal and Gunasekaran [7]. Six different numbers

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

m	3, 5, 7, 10, 25, 50
S	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]

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

		Huang and Yao's [10] search algorithm		Our proposed algorithm					
							Objective function value		
m	S	Avg. iterations	Avg. run time	Avg. iterations	Avg. run time	Run time improving rate (%)	Error more than 0.01%	Max. error (%)	Avg. error (%)
3	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	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	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

of groups of vehicles ($m = 3, 5, 7, 10, 25, 50$) and six different values for the fixed cost in each basic period T ($S = 50, 100, 200, 500, 750, 1000$) are selected. This yields 36 combinations from these parameter settings. Then, for each combination, we randomly generate 1000 instances by randomly choosing the values for X_i, Y_i, a_i, b_i, and s_i using uniform distribution functions. Table 2 shows the ranges of these uniform-distributed random variables.

After randomly generating 36,000 instances, we solve each of them using our proposed algorithm, along with Huang and Yao's [10] search algorithm, on a PC with a Core 2 Duo processor P8600 with 4 GB RAM. The error allowance in each algorithm is set by $\varepsilon = 0.01\%$. The experimental results for the smaller size (with $m = 3, 5, 7$) and larger size (with $m = 10, 25, 50$) are summarized in Tables 3 and 4, respectively.

One may observe that our proposed algorithm solves the TFMSP within an extremely short run time. It has an average reduction of 71.79% in its run time compared with Huang and Yao's [10] search algorithm. Regarding solution quality, we set the error allowance equal to 0.01% for both approaches. As mentioned in Condition 3 of Section 4.1.2, our proposed algorithm may sometimes miss the local minimum, so we compare the outcome and indicate the number of instances out of the 1000 instances for each m and S combination where the difference is greater than 0.01% in

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

		Huang and Yao's [10] search algorithm		Our proposed algorithm					
							Objective function value		
m	S	Avg. iterations	Avg. run time	Avg. iterations	Avg. run time	Run time improving rate (%)	Error more than 0.01%	Max. error (%)	Avg. error (%)
10	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	0.028	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	50	642	0.064	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	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

the last third column. Our results show that the differences of the solutions obtained by the two approaches are all within 0.01%.

In the last two columns of Tables 3 and 4, we present the maximum (relative) error and average (relative) error of our proposed algorithm in percentages. An interesting finding is that 37 out of 42 combinations have a negative maximum error value, which means our proposed algorithm in these instances can mostly obtain better solutions than the traditional dynamic Lipschitz optimization algorithm (i.e. the Evtushenko algorithm). Although five combinations (when $m = 25$ and 50) have a positive maximum error, the largest is only 0.000129%, and all these five problems have a negative average error value. It is thus obvious that our proposed algorithm is better than the traditional one in terms of the quality of the solution.

6. Conclusion and future research

In this study, we propose a new Lipschitz optimization algorithm for solving the TFMSP. Our proposed algorithm employs slope-checking and step-size comparison mechanisms to significantly reduce the number of iterations in Huang and Yao's Evtushenko algorithm [10] as the search passes through decreasing subranges of the objective function. Our numerical experiments show that our proposed algorithm solves the TFMSP with an average reduction of 71.79% in its run time compared with Huang and Yao's [10] Evtushenko algorithm. Therefore, we conclude that our proposed algorithm significantly improves computational efficiency.

Note that our proposed algorithm may sometimes miss the local minimum, although it still guarantees an ε-% error allowance. The quality of the solution of our proposed algorithm is even better than the conventional Evtushenko algorithm because we incorporate the mechanism of a backward search for a local minimum in our proposed algorithm.

We note that both Huang and Yao [10] and this study apply the Evtushenko algorithm as Lipschitz optimization algorithms for solving the TFMSP. One may apply other global-optimization

approaches, such as the Piyavskii [15] algorithm, to solve the TFMSP and to propose fine-tuned mechanisms (similar to those in Section 4 in this study) following the characteristics of the objective function. These applications may serve as a future extension of this study.

Acknowledgements

The authors would like to acknowledge the grant received from the National Science Council: NSC 100-2410-H-167-004-MY2.

References

[1] B.N. Amotz, R. Bhatia, J. Naor, and B. Schieber, *Minimizing service and operation costs of periodic scheduling*, Math Oper. Res. 27 (2002), pp. 518–544.
[2] S. Anily, C.A. Glass, and R. Hassin, *The scheduling of maintenance service*, Discrete Appl. Math. 82 (1998), pp. 27–42.
[3] S. Anily, C.A. Glass, and R. Hassin, *Scheduling of maintenance services to three machines*, Ann. Oper. Res. 86 (1999), pp. 375–391.
[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, *Nonlinear Programming Theory and Algorithms*, 2nd ed., John Wiley & Sons, New York, 1993.
[5] A.H. Christer and T. Doherty, *Scheduling overhauls of soaking pits*, Oper. Res. Quart. 28 (1977), pp. 915–926.
[6] R. Dekker, R. Wildeman, and F. van der Duyn Schouten, *A review of multi-component maintenance models with economic dependence*, Math. Method Oper. Res. 45 (1997), pp. 411–435.
[7] S.K. Goyal and A. Gunasekaran, *Determining economic maintenance frequency of a transportation fleet*, Int. J. Syst. Sci. 23 (1992), pp. 655–659.
[8] S.K. Goyal and M.I. Kusy, *Determining economic maintenance frequency for a family of machines*, J. Oper. Res. Soc. 36 (1985), pp. 1125–1128.
[9] R. Horst and P.M. Pardalos, *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht, 1995.
[10] J.Y. Huang and M.J. Yao, *A dynamic Lipschitz algorithm for determining economic maintenance frequency of a transport fleet*, J. Infor. Optim. Sci. 28 (2007), pp. 357–376.
[11] H. Luss, *Maintenance policies when deterioration can be observed by inspections*, Oper. Res. 24 (1976), pp. 359–366.
[12] H. Luss and Z. Kander, *Preparedness model dealing with N systems operating simultaneously*, Oper. Res. 22 (1974), pp. 117–128.
[13] W.E. Moudani and F. Mora-Camino, *A dynamic approach for aircraft assignment and maintenance scheduling by airlines*, J. Air Transp. Manag. 6 (2000), pp. 233–237.
[14] N. Papadakos, *Integrated airline scheduling*, Comput. Oper. Res. 36 (2009), pp. 176–195.
[15] S.A. Piyavskii, *An algorithm for finding the absolute extremum of a function*, USSR Comp. Math. Math. 12 (1972), pp. 57–67.
[16] C. Sriram and A. Haghani, *An optimization model for aircraft maintenance scheduling and re-assignment*, Transport Res. A-Pol. 37 (2003), pp. 29–48.
[17] D.R. Sule and B. Harmon, *Determination of coordinated maintenance scheduling frequencies for a group of machines*, AIIE T. 11 (1979), pp. 48–53.
[18] R.E. Wildeman and R. Dekker, *Dynamic influence in multi-component maintenance*, Qual. Reliab. Eng. Int. 13 (1997), pp. 199–207.