Method of Hill Tunneling via Weighted Simplex Centroid for Continuous Piecewise Linear Programming

Zhiming Xu, Yu Bai, Kuangyu Liu, and Shuning Wang*

Abstract: This paper works on a heuristic algorithm with determinacy for the global optimization of Continuous Piecewise Linear (CPWL) programming. The widely applied CPWL programming can be equivalently transformed into D.C. programming and concave optimization over a polyhedron. Considering that the super-level sets of concave piecewise linear functions are polyhedra, we propose the Hill Tunneling via Weighted Simplex Centroid (HTWSC) algorithm, which can escape a local optimum to reach the other side of its contour surface by cutting across the super-level set. The searching path for hill tunneling is established via the weighted centroid of a constructed simplex. In the numerical experiments, different weighting methods are studied first, and the best is chosen for the proposed HTWSC algorithm. Then, the HTWSC algorithm is compared with the hill detouring method and the software CPLEX for the equivalent mixed integer programming, with results indicating its superior performance in terms of numerical efficiency and the global search capability.

Key words: global optimization; piecewise linear; concave minimization; cutting plane method; hill tunneling

1 Introduction

The Continuous Piecewise Linear (CPWL) minimization problem can be stated in the following form:

$$\min \{ f_0(z) | s.t. f_i(z) \leq 0, i = 1, 2, \ldots, N \}$$  (1)

where $z \in \mathbb{R}^n$ and each $f_i(z), i = 1, 2, \ldots, N$ is a Piecewise Linear Function (PLF).

Due to its specific features, CPWL programming has broad application prospects. On the one hand, the arbitrary approximation capability on any nonlinear function[1] makes it more flexible and applicable than Linear Programming (LP) on such problems as black box modeling and system identification. On the other hand, compared with other forms of nonlinear programming, the local linearity of PLFs enables us to propose more efficient algorithms[2–9].

Generally speaking, when dealing with the minimization of separable and convex PLFs with linear inequalities, solutions are easier when performed using the existing algorithms[10–12]. However, achieving the global optimality of a PLF, which is neither separable nor convex, is NP-hard[13]. The most commonly used method is to reformulate the CPWL minimization problems into equivalent Mixed Integer Programming (MIP) problems[14–16]. However, we cannot expect to obtain the global optimum for large-scale problems within an acceptable time period by following this method. Other alternatives are the heuristic algorithms, including the random[17] and deterministic approaches[18–20]. Despite the benefits of high efficiency in actual practice, failing to guarantee the global or even the local optimality results in the unstable performance of the random approaches. Therefore, the current study focuses on the heuristic algorithms with determinacy.

*Zhiming Xu, Yu Bai, Kuangyu Liu, and Shuning Wang are with the National Laboratory for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: xzmudt@163.com; baiyu13@mails.tsinghua.edu.cn; 409890573@qq.com; swang@mail.tsinghua.edu.cn.
*Zhiming Xu is also with Faculty of Air Force Engineering University, Xi’an 710051, China.
*To whom correspondence should be addressed.
Manuscript received: 2017-11-22; accepted: 2018-02-22
In particular, a recent heuristic algorithm with determinacy, named the Hill Detouring (HD) method\cite{18}, is noteworthy. HD provides the “leave and reentry” searching strategy: after a local optimum is obtained, the search leaves the feasible domain first and is then conducted on the contour surface, until it intersects with the feasible domain again. This strategy escapes the local optimum by bypassing the super-level set. Motivated by this, we propose the use of the hill tunneling method to escape the local optima more efficiently by cutting across the super-level set. As each searching path for the hill tunneling is established by using the weighted centroid of a constructed simplex, the proposed algorithm is called Hill Tunneling via Weighted Simplex Centroid (HTWSC).

The rest of this paper is organized as follows. Section 2 contains the theoretical backgrounds and some preliminary works. Section 3 demonstrates the motivation behind the use of the hill tunneling method. Section 4 frames the HTWSC algorithm and discusses the realization issues, and several weighting methods are proposed in Section 5. The numerical experiments are conducted in Section 6, and the results are presented. Finally, the conclusions are given in Section 7.

2 Preliminary

In this section, we first derive the equivalence of the CPWL programming and the concave piecewise linear programming. Then some relevant definitions and properties are introduced. Finally, the method used in searching local optimum is discussed concisely.

2.1 Representation and transformation

2.1.1 Problem representation

Due to the existence of the exact penalty functions in any CPWL problem\cite{19}, there exists $\hat{r} > 0$, which results in problem (1) having the identical set of local optima with the following unconstrained CPWL problem for any $r > \hat{r}$:

$$\min F_r(z) = f_0(z) + r \sum_{i=1}^{N} \max \{0, f_i(z)\}$$

(2)

Owing to the completeness of the Generalized Hinging Hyperplane (GHH) model\cite{20}, any continuous PLF, including $F_r(z)$ in problem (2), can be described in the following GHH form:

$$\max_{1 \leq j \leq n_k} \{a^T_{kj} z + b_{kj}\} - \max_{1 \leq j \leq l_k} \{a_{kj}^T z + b_{kj}\}$$

(3)

where all maximal functions are the basis functions of the GHH model, and $\hat{N}$ and $\bar{N}$ denote the numbers of the positive and negative ones, respectively.

Consequently, problem (2) can be reformulated as a D.C. programming given by

$$\min p(z)$$

(4)

where

$$p(z) = \sum_{k=1}^{\hat{N}} \max_{1 \leq j \leq n_k} \{a^T_{kj} z + \hat{b}_{kj}\} - \sum_{k=1}^{\bar{N}} \max_{1 \leq j \leq l_k} \{a_{kj}^T z + \bar{b}_{kj}\}$$

(5)

According to the above analysis, any CPWL minimization can be represented as problem (4).

In the remainder of this paper, without special remarks, a CPWL programming refers to problem (4).

2.1.2 Equivalent concave transformation

To make the proposal of the HTWSC algorithm sharper in the following sections, a further transformation must be implemented in advance. By introducing new variables $\lambda_k$, min $p(z)$ can be transformed equivalently to the problem stated below.

$$\min \sum_{k=1}^{\hat{N}} \lambda_k - \sum_{k=1}^{\bar{N}} \max_{1 \leq j \leq k} \{a^T_{kj} z + \bar{b}_{kj}\},$$

s.t. $a^T_{kj} z + \bar{b}_{kj} \leq \lambda_k, 1 \leq j \leq n_k, 1 \leq k \leq \hat{N}$

(6)

Clearly, problem (6) is a concave minimization over a convex polyhedron. For simplicity, by denoting $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{\hat{N}})^T$ and $x = (\lambda^T, z^T)^T$, we can express problem (5) in its equivalent form,

$$\min \left\{ f(x) = \sum_{k=1}^{M} \min_{1 \leq j \leq n_k} \{a_{kj}^T z + b_{kj}\} \mid \text{s.t. } Cx \leq g \right\}$$

(7)

where $M$ denotes the number of the minimum functions, $\min \{z, C \in \mathbb{R}^{m \times n}, g \in \mathbb{R}^n, a_{ij} \in \mathbb{R}^n, b_{ij} \in \mathbb{R}, \forall i, j$.

Problem (7) is a concave minimization over a convex polyhedron. Without loss of generality, we assume that problem (7) is nondegenerate, and the feasible domain, denoted by $\Omega = \{x \mid Cx \leq g\}$, is bounded and closed. Actually, when dealing with specific problems where $\Omega$ is unbounded, we can add the bound constraints as $b_l \leq x \leq b_u$, which have practical meanings.

2.2 Definitions and properties

For convenience, some basic definitions and properties are introduced in advance.

2.2.1 Definitions

Given the GHH objective function $f(x)$ in Formula (7) and any $x$, we define the index set at $x$ for $\forall 1 \leq i \leq M$
as
\[
\Phi^i_x = \left\{ j \mid a^T_{ij} x + b_{ij} = \min_{1 \leq k \leq M} \{ a^T_{ik} x + b_{ik} \} \right\}
\]
and the index space at \( x \) as
\[
\Phi_x = \Phi^1_x \times \Phi^2_x \times \cdots \times \Phi^M_x
\]

**Definition 1** For a given point \( x \in \mathbb{R}^n \), given any \( \varphi = (k_1, k_2, \ldots, k_M) \in \Phi_x \), the GHH function \( f(x) \) can be simplified to an associated affine expression, denoted by
\[
a^\varphi x + b^\varphi
\]

Then, the affine function (10) is called a Facet of \( f(x) \) at \( x \), denoted by \( \Psi_x^\varphi \).

Figure 1 illustrates the concept of facet in the situation wherein \( x \in \mathbb{R}^2 \).

**Definition 2** Given \( \gamma \in \mathbb{R} \), the set \( \{ x \mid f(x) = \gamma \} \) is called a Contour Surface of \( f(x) \), denoted by \( \Gamma_\gamma \).

**Definition 3** Given a contour surface \( \Gamma_\gamma \) and a facet \( \Psi : a^\varphi x + b^\varphi \), if
\[
\Gamma_\gamma \cap \{ x \mid f(x) = a^T x + b \} \neq \emptyset
\]
then the associated Contour Facet is defined as
\[
\Gamma_{\gamma, \varphi} = \{ x \mid f(x) = \gamma, f(x) = a^T x + b \}
\]

Figure 2 illustrates the concepts of contour surface and contour facet. The contour surface degenerates to contour line in the situation wherein \( x \in \mathbb{R}^2 \), as is shown by the blue dashed polygonal line. The contour facet degenerates to the contour line segment, as shown by the red dotted line segment. Figure 3 shows their projections to the horizon.

Aside from the concepts defined above, an existing concept of \( \gamma \)-extension will also be involved later in the text. The definition is cited as follows[21].

**Definition 4** Given a concave function \( f \) and a point \( x \in \mathbb{R}^n \), suppose that \( \gamma \) satisfies \( \gamma \leq f(x) \), and
\[
\theta_0 \text{ is a positive number great enough. For the direction } d \in \mathbb{R}^n \setminus \{0\}, \text{ let}
\]
\[
\theta = \min \{ \theta_0, \sup \{ t : f(x + td) \geq \gamma \} \}
\]
then
\[
x' = x + \theta d
\]
is called the \( \gamma \)-Extension of \( f \) in the direction of \( d \) originating from \( x \).

**2.2.2 Properties**

For any concave PLF \( f(x) \), there holds two useful properties as listed below.

**Property 1** If \( f(x) = a^T x + b, \forall x \in D \) holds for an open set \( D \subset \mathbb{R}^n \), then \( f(x) \leq a^T x + b, \forall x \in \mathbb{R}^n \).

This property can be seen obviously in Fig. 4, and it will be utilized later to derive the local optimal
problem. This property has been proved in Ref. [18], and can also be made out in Fig. 2. The super-level set degenerates to polygons in the situation wherein \( x \in \mathbb{R}^2 \). Later, the cutting plane method shall be introduced in the HTWSC algorithm based on this property.

2.3 Local optima obtainment

The global searching procedure for Formula (7) in the following sections will start from a local optimal vertex. Many descend algorithms, such as the subgradient method[22], can be applied in searching the local optima of such problems that are neither separable nor convex. Actually, on account of Property 1, a local optimum of problem (7) can be obtained by solving a series of LP problems.

Given \( \tilde{x} \in \Omega \) and \( \varphi \in \Phi_\tilde{x} \), denote the optimal solution the following LP problem by \( \tilde{x} \).

\[
\min \{ a^T \tilde{x} + b^T \mid C \tilde{x} \leq g \} \tag{16}
\]

Then, according to Property 1, the following must hold:

\[
f (\tilde{x}) \leq a^T \tilde{x} + b^T \leq a^T \tilde{x} + b^T = f (\tilde{x}) \tag{17}
\]

that is to say, \( \tilde{x} \) is a feasible solution of problem (7) that is not worse than \( \tilde{x} \).

Hence, the local optimal routine can be designed as follows. Starting from a feasible point \( \tilde{x} \), pick up a \( \varphi \in \Phi_\tilde{x} \) and solve LP (Formula (16)) to obtain a better solution of problem (7), denoted by \( \tilde{x} \). Update \( \tilde{x} \) with \( \tilde{x} \), and repeat the above steps. Go on until we achieve such an \( x_0 \), where the optimal solution of Formula (16) is \( x_0 \) itself for any \( \varphi \in \Phi_{x_0} \). Then, \( x_0 \) is the obtained local optimum. Further, if the simplex method for LP is exploited, \( x_0 \) must be a vertex of \( \Omega \).

3 Motivation

When a local optimum is obtained, the HD method, which motivates the hill tunneling method proposed in this article, provides a strategy to escape the local trap to go on searching for a better solution. Hence, the HD method must be sketched first.

3.1 HD method and its inadequacy

The procedures of the HD method are sketched in Fig. 5, which is quoted from Ref. [18]. The concave objective function appears to be a “hill” in the situation wherein \( x \in \mathbb{R}^2 \). The red line on the “hill” is the contour with the value of a local optimum \( x_0 \), and the blue line below is the periphery of the feasible domain.

For further illustration, we project Fig. 5 to the horizon, as is shown in Fig. 6. As can be seen, the solid line surrounds the feasible domain, the dashed line is the contour surface \( \Gamma_f(x_0) \) corresponding to \( x_0 \), and \( x^* \) is the global minimum.

The principal steps of the HD method in Fig. 6 can be stated briefly as follows:

- Start from the locally optimum \( x_0 \), and find all the edges of \( \Omega \) which intersect at \( x_0 \). Search along the edges until the contour surface \( \Gamma_f(x_0) \) is reached at \( x_i, i = 1, 2 \);
- Choose \( x_1 \), the intersection point closest to the feasible domain, and project \( x_1 - x_0 \) on the contour facet \( \Gamma^1 \). Then carry out line search to find \( x_3 \);
- Repeat until the contour facet \( \Gamma^* \) is reached, by which \( \Gamma_f(x_0) \) re-intersects with the feasible domain. Then restart the local optimal routine.

As the searching origin of the HD method is closed to the present local optimum while the objective function value is not better than the present within quite an area nearby, the detouring process may take a very long time in performing the iteration. Intuitively, rather than
bypassing the feasible domain gradually from a local optimum to another, it would be much more efficient to cut across the domain directly. Motivated by this idea, we propose a new approach ("hill tunneling") to improve the efficiency of the HD method.

3.2 Hill tunneling strategy

The hill tunneling strategy escapes a local optimum by directly cutting across the super-level set to reach the other side of the objective function “hill”, as is shown in Fig. 7. This approach enhances the escaping efficiency, and thus contributes in improving global searching.

Further, the searching path for the hill tunneling can be determined via the weighted centroid of a constructed simplex, which takes along the information can be determined via the weighted centroid of a constructed simplex, which takes along the information

Thus, the proposed method is called the method of hill tunneling via weighted simplex centroid.

4 Method of Hill Tunneling via Weighted Simplex Centroid

In this section, a detailed description of the HTWSC algorithm is given, including the algorithmic framework, technical analysis, and implementation details.

4.1 Algorithmic framework

The main procedures of the HTWSC in escaping from the present local optimal vertex \( x_0 \) are as follows, and the detailed discussions are provided successively.

1. Initialize the iteration number \( k = 1 \), the present optimal value \( y = f(x_0) \), the current feasible domain \( \Omega_0 \leftarrow \Omega \), and the current tunneling origin \( x_1 \leftarrow x_0 \).

2. Repeat the following steps until the stop criterion is met.

   a. Carry out \( \gamma \)-extensions from \( x_k \) to obtain the extension points \( x_1^k, x_2^k, \ldots, x_n^k \).

   b. Structure an \((n - 1)\)-dimensional simplex \( S_k \) with \( x_1^k, x_2^k, \ldots, x_n^k \) as its vertices.

   c. Take the \((n - 1)\)-dimensional hyperplane \( H_k \), where \( S_k \) is located in as a cutting plane, to cut away a part of the feasible domain that does not contain any point superior to \( x_0 \). This makes the feasible domain shrink to a smaller new domain denoted by \( \Omega_k = \{ x | C^k x \leq g^k \} \).

   d. If \( \Omega_k = \emptyset \), then \( x_0 \) is the global optimum, terminate the algorithm; or else, compute the weighted centroid of \( S_k \) after weighting \( x_1^k, x_2^k, \ldots, x_n^k \), somehow, denoted by \( C_k \).

   e. Tunnel the hill-shaped concave objective function along the direction of \( C_k - x_k \) with the origin \( x_k \), and obtain another \( \gamma \)-contour facet on the other side of the “hill”, denoted by \( F^k \).

   f. Check whether the contour surface \( \Gamma^{k} \) intersects with \( \Omega_k \) on \( F^k \) or other adjacent \( \gamma \)-contour facet near \( F^k \).

   g. If the checking result is yes, step out of the iteration and restart the local optimization process with the intersective point; or else, figure out the nearest point on \( \Gamma^{k} \) to \( \Omega_k \) in the place on or near \( F^k \), denoted by \( x_{k+1} \).

   h. \( k \leftarrow k + 1 \).

4.2 Theoretical analysis and detailed technology

In this section, the detailed algorithm implementation as well as some theoretical analysis will be stated step by step.

4.2.1 \( \gamma \)-extensions in the first iteration

It holds that \( k = 1 \) in the first iteration. By this time, \( x_1 \), the origin of the \( \gamma \)-extensions, is just the current local optimal vertex \( x_0 \), and the directions for extension are the directed edges from \( x_0 \) to the adjacent vertices of the feasible domain.

In other words, we carry out line searches along all edges of \( \Omega_0 = \Omega \) which intersect at \( x_0 \), until they reach the contour surface \( \Gamma^{(x_0)} \) respectively. Then, the intersection points are the \( \gamma \)-extensions in the first iteration.

Given that \( x_0 \) is a vertex nondegenerate, as is assumed previously, there must be \( n \) adjacent vertices to \( x_0 \). So, it will turn out to be \( n \) \( \gamma \)-extensions, denoted by \( x_i^1, i = 1, 2, \ldots, n \), thus making the vector group...
\[ x_i^1 - x_1, \, i = 1, 2, \ldots, n \text{ affine independent.} \]

The approach to calculate the \( n \) directions for the \( y \)-extensions in the first iteration is given in details below as **Proposition 1**.

On account of the nondegeneracy of vertex \( x_0, \) there must be \( n \) inequality constraints critical at \( x_0 \) in problem (7). We denote these \( n \) critical inequality constraints by

\[ Bx \leq \bar{b} \quad (18) \]

then there must hold \( x_0 = B^{-1}\bar{b} \) by this time, where \( B = (b_1, b_2, \ldots, b_n)^T \in \mathbb{R}^{n \times n}, \) and \( \bar{b} = (\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n) \in \mathbb{R}^n. \)

**Proposition 1** As for the coefficient matrix \( B \) in inequality constraint (18), if \( B^{-1} = (d_1, d_2, \ldots, d_n), \) then \(-d_1, -d_2, \ldots, -d_n\) are the directed edges from \( x_0 \) to its adjacent vertices of the feasible domain. Or equivalently, \(-d_1, -d_2, \ldots, -d_n\) are the \( n \) directions for the \( y \)-extensions.

**Proof** Given that \( BB^{-1} = I, \) there should always be \( b_i^T d_i = 1 \) and \( b_i^T d_i = 0 \) for any \( 1 \leq i \leq n \) with \( i \neq j. \) In addition, \( b_j \) is the normal vector of hyperplane \( \{x | b_j^T x = \bar{b}_j\} \), thus \( d_j // \{x | b_j^T x = \bar{b}_j\} \) holds for any \( i \neq j. \) Then, we can assert that

\[ d_j // \bigcap_{j=1}^{n} \{x | b_j^T x = \bar{b}_j\}, \forall 1 \leq i \leq n \quad (19) \]

This shows that \( d_j \) is parallel to the edge line where hyperplanes \( \{x | b_j^T x = \bar{b}_j\}, j \neq i \) intersect in.

Moreover, there holds

\[ b_i^T (x_0 + td_i) = b_i^T x_0 + td_i^T d_i = b_i^T x_0 + t > b_i^T x_0 = \bar{b}_i, \forall i \quad (20) \]

for any \( t > 0. \) This indicates that \( x_0 + td_i \) dissatisfies constraint (18). That is to say, each vector \( d_i \) with the origin \( x_0 \) points to the outside of the feasible domain or the vectors \(-d_i, 1 \leq i \leq n\) are directions pointing to the adjacent vertices from \( x_0. \)

By this way, the directions for the \( y \)-extensions in the first iteration are derived.

### 4.2.2 Simplex construction

In the \( k \)-th iteration, vectors \( x_i^k - x_k, 1 \leq i \leq n, \) generated by the \( n \) \( y \)-extensions \( x_1^k, x_2^k, \ldots, x_n^k \) as well as the origin \( x_k, \) are affine independent. (This is stated in Section 4.2.1 with \( k = 1, \) and will be stated in Section 4.2.7 with \( k \geq 2, \)) Thus an \( (n-1) \)-dimensional simplex \( S_k \) can be constructed with \( x_1^k, x_2^k, \ldots, x_n^k \) as its vertices. Furthermore, combining \( S_k \) with point \( x_k, \) an \( n \)-dimensional simplex \( S_k \) can be formed as well. We define \( S_k \) as the simplex obtained in the \( k \)-th iteration.

#### 4.2.3 Employment of the cutting hyperplane

In the \( k \)-th iteration, considering the convexity of \( \Omega_{k-1} \) and \( S_k, \) overlapping the concavity of the objective function \( f, \) it can be deduced that there is no point superior to the present local optimum \( x_0 \) in \( \Omega_{k-1} \cap \bar{S}_k. \)

Inspired by this, we can cut away \( \Omega_{k-1} \cap \bar{S}_k \) from \( \Omega_{k-1} \) in each iteration, without affecting the global optimum. Concretely, this operation can be carried out by employing the so-called cutting plane.

Actually, \( S_k, \) the simplex obtained in the \( k \)-th iteration, is located in an \((n-1)\)-dimensional hyperplane \( H_k : c_k^T x = g_k. \) Then, we define \( H_k \) to be the cutting plane obtained in the \( k \)-th iteration.

It is obvious that \( x_k, \) as the origin of \( y \)-extensions, locates in the opposite side of \( H_k \) to set \( \Omega_{k-1} \setminus \bar{S}_k. \) Therefore, by adding a new constraint derived from the cutting plane to \( \Omega_{k-1}, \) a shrunk new feasible domain is formed below:

\[ \Omega_k = \begin{cases} \Omega_{k-1} \cap \{x | c_k^T x \geq g_k\}, & \text{if } c_k^T x_k < g_k; \\ \Omega_{k-1} \cap \{x | c_k^T x \leq g_k\}, & \text{else} \\ \{x | (g_k - x_k) c_k^T < 0\} = \{x | c_k^T x \leq g_k\} \end{cases} \]

where \( \text{sign} (\cdot) \) denotes the sign function.

#### 4.2.4 Weighted centroid calculation

If \( \Omega_k \neq \emptyset \) in the \( k \)-th iteration, entrust weights \( w_1^k, w_2^k, \ldots, w_n^k \) singly to \( x_1^k, x_2^k, \ldots, x_n^k \) under a certain rule. Then, we calculate \( C_k, \) the weighted centroid of \( S_k, \) by using

\[ C_k = w_1^k x_1^k + w_2^k x_2^k + \cdots + w_n^k x_n^k \]

The purpose of weighing the vertices of \( S_k \) is to take full advantage of the information of the interrelation between the feasible domain and the current contour surface at points \( x_1^k, x_2^k, \ldots, x_n^k, \) to point out the most efficient direction for tunneling.

Several weighting methods will be discussed later on.

#### 4.2.5 Hill tunneling and facet ascertaining

When the weighted centroid \( C_k \) has been obtained in the \( k \)-th iteration, we carry out \( y \)-extension from \( x_k \) along the direction of \( C_k - x_k. \) This is the hill tunneling operation in the \( k \)-th iteration.

We denote the extension point obtained after hill tunneling as \( x'_k, \) then there is \( x'_k \in \Gamma_y. \) We select a candidate \( \phi \in \Phi_{x'_k} \) and obtain the corresponding facet \( \Psi_{x'_k} \), by using Eq. (10). We take \( \Psi_{x'_k} \) as the facet obtained after the \( k \)-th iteration of hill tunneling.
denoted by
\[ \psi_k : a_k^T x + b_k \]  
(23)

Then, we correspondingly take \( \Gamma_k(x_k^*) \), the contour facet at \( x_k^* \) as the contour facet obtained in the \( k \)-th iteration, denoted by \( \Gamma^k \).

### 4.2.6 Contour facet checking

After the \( k \)-th hill tunneling process, the contour facet \( \Gamma^k \) must be checked in order to assess whether the contour surface \( \Gamma_y \) intersects with the current feasible domain \( \Omega_k \) on \( \Gamma^k \) or other adjacent \( \gamma \)-contour facets. This means that we check if there is \( \Gamma_y \cap \Omega_k \neq \emptyset \) on \( \Gamma^k \) or around it.

The checking can be implemented by solving the following LP programming:

\[
\begin{align*}
\min_{s} & \quad s \\
\text{st.} & \quad C_k x - g_k \leq S, \quad a_k^T x + b_k \leq f(x_k) \\
\end{align*}
\]

(24)

where \( s \in \mathbb{R}, S = (s, s, \ldots, s)^T \). In addition, \( C^k \) and \( g^k \) are the corresponding parameter matrices in Formula (21), which describes \( \Omega_k \). \( a_k^T x + b_k \) is the definition equation of \( \psi_k \) in Formula (23). Suppose that \( (\hat{s}, \hat{x}) \) is the optimal solution of problem (24). Then, under the situation that \( \Omega_k \neq \emptyset \), \( \hat{s} \) can be considered as an index that measures something of the distance between \( \Omega_k \) and \( \Gamma^k \), as illustrated in Fig. 8.

There are three cases when solving LP (24).

1. **Case 1**: \( \hat{s} \leq 0 \)
   - In this case, \( \Gamma^k \cap \Omega_k \neq \emptyset \), and \( \hat{x} \) is a feasible solution not worse than \( x_0 \), as shown in Fig. 9.
   - Then the algorithm should reset the local optimal procedures from \( \hat{x} \).

2. **Case 2**: \( \hat{s} > 0, a_k^T \hat{x} + b_k = f(x_k) \)
   - In this case, \( \hat{s} > 0 \) implies \( \Gamma^k \cap \Omega_k = \emptyset \), and \( a_k^T \hat{x} + b_k = f(x_k) \) indicates \( \hat{x} \in \Gamma_f(x_k) \), as shown in Fig. 8. Then, let \( x_{k+1} = \hat{x} \) and start the next iteration from this point.

3. **Case 3**: \( \hat{s} > 0, a_k^T \hat{x} + b_k < f(x_k) \)

This case means \( \hat{x} \notin \Gamma_y \), as seen in Fig. 10. Additionally, there must be \( f(\hat{x}) < f(x_k) \) according to **Property 1**.

In this case, the algorithm should reset \( \Gamma^k \) to an adjacent \( \gamma \)-contour facet nearer to \( \Omega_k \), and rerun the contour facet checking process. In more detail, we select a candidate facet of \( f(x) \) at \( \hat{x} \), denoted by \( \psi_{\hat{s}} : a_k^T x + b_{\hat{x}} \), to be the new \( \psi_k \), and derive the corresponding new \( \Gamma^k \). Then, we re-solve problem (24). Given the super-level set \( \{ x \mid f(x) \geq f(x_k) \} \) is convex, this reset action must result in a new and smaller \( \hat{s} \).

We repeat these reset and check operations, until it turns into Case 1 or Case 2.

### 4.2.7 \( \gamma \)-extensions with \( k \geq 2 \)

When the contour facet checking leads to Case 2 in the \((k-1)\)-th iteration with \( k \geq 2 \), as shown in Fig. 8, it should proceed into the \( k \)-th iteration. By this time, the origin of the \( \gamma \)-extensions is \( x_{k+1} = \hat{x} \), which is obtained in the contour facet checking process in the \((k-1)\)-th iteration, and the directions can be ascertained as follows.

Under the nondegenerate assumption, there must be \( n + 1 \) constraints of all in problem (24) critical at the optimum \( (\hat{s}, \hat{x}) \). As \( a_k^T \hat{x} + b_k = f(x_k) \) in Case 2, the other \( n \) constraints of the \( n + 1 \) critical ones involve
parameter $s$, denoted by
\[ Bx - \tilde{g} \leq \tilde{S} \] (25)

Similar to the analysis in Proposition 1, denoting $B^{-1} = (d_1, d_2, \ldots, d_n)$, vectors $-d_1, -d_2, \ldots, -d_n$ are parallel to the directions pointing to the adjacent vertices from the nearest feasible vertex to the contour facet $\Gamma^{k-1}$, respectively. That is to say, $-d_1, -d_2, \ldots, -d_n$ are the directions for carrying out the $\gamma$-extensions in the $k$-th iteration. Further, we denote the $\gamma$-extensions in the $k$-th iteration by $x^k_i, i = 1, 2, \ldots, n$, then the vector group $x^k_1 - x_1, i = 1, 2, \ldots, n$, is affine independent.

4.2.8 Loop avoidance

The algorithm should stop the iteration when one of the following criteria is met.

Criterion 1: $\Omega_k = \emptyset$ after employing the cutting plane $H_k$. This means that there is no feasible point that is superior to the current local optimum $x_0$ all over the feasible domain $\Omega$. Thus, we terminate the algorithm by this time and return the global optimum $x_0$.

Criterion 2: The running time meets $t \geq T$ or the iteration number meets $k \geq K$, where $T$ and $K$ are the threshold parameters that are big enough. In this case, we terminate the algorithm and return $x_0$ as the optimal solution. This criterion is set to ensure that the algorithm generates a solution in a sensible time.

4.2.9 Stop criterion

The employment of the cutting plane alters part of the feasible domain closed to $x_k$, where the hill tunneling process starts from in each iteration. This fact ensures that the searching paths for hill tunneling are different from one another, even though any two of the starting contour facets are the same. Thus, the loop of the repetitive hill tunneling can be avoided spontaneously.

4.3 Algorithm scheme

To sum up the above, the scheme of the HTWSC algorithm is given in Algorithm 1.

5 Several Weighting Methods

In order to escape the local optimal trap as efficiently as possible through the established hill tunneling direction, the weighting plan of $\psi_1^k, \psi_2^k, \ldots, \psi_n^k$, the vertices of $S_k$, should be designed rationally. Several weighting methods are proposed in this section from different perspectives, named HTWSC-1 to HTWSC-6.

### Algorithm 1 HTWSC algorithm for concave CPWL programming

**Initialize**
- Set iteration threshold $K$ and time threshold $T$;
- Load parameters of model (7), and find a feasible solution $\hat{x}$.

**while** $k \leq K$ and $t \leq T$ **do**
- Start from $\hat{x}$ to compute a local optimum vertex $x_0$;
- Let $k = 1, \gamma = f(x_0), \Omega_0 = \Omega, x_1 \leftarrow x_0$;
- Find out the matrix $B$ referred in Formula (18) and compute the inverse $B^{-1} = (d_1, d_2, \ldots, d_n)$;
- Let $(d_1, d_2, \ldots, d_n) := -(d_1, d_2, \ldots, d_n)$;
- **repeat**
  - Search along $d_1, d_2, \ldots, d_n$ respectively from $x_k$ to get the $\gamma$-extensions $x^k_1 - x^k_2, \ldots, x^k_n$;
  - Ascertain the cutting hyperplane $H_k : C^T x = g_k$
  - with $x^k_1 - x^k_2, \ldots, x^k_n$;
  - Update $\Omega_k$ via Formula (21), and let $\Omega \leftarrow \Omega_k$;
  - if $\Omega_k = \emptyset$ **then**
    - Terminate the algorithm and return $x_0$;
  - end if
- **end if**
- Weighting $x^k_1, x^k_2, \ldots, x^k_n$ with $w_1^k, w_2^k, \ldots, w_n^k$, and compute $C_k$ via Formula (22);
- Carry out the $\gamma$-extension $x^k_k$ in the direction of $C_k - x_k$ originating from $x_k$;
- **repeat**
  - Pick up a candidate $\varphi \in \Phi_{x^k_k}$, and obtain the corresponding facet $\psi_{x^k_k}^\varphi$ by Formula (10);
  - Let $\varphi_k \leftarrow \psi_{x^k_k}^\varphi$;
  - Solving LP (24) and record the optimum $(\hat{s}, \hat{x})$;
  - Let $x_{k+1}^k \leftarrow \hat{x}$;
- **until** $\hat{s} \leq 0$ or $a_k^T \hat{x} + b_k = f(x_k)$;
- if $\hat{s} > 0$ **then**
  - Find out the matrix $B$ referred to in Formula (25) and compute the inverse $B^{-1} = (d_1, d_2, \ldots, d_n)$;
  - Let $(d_1, d_2, \ldots, d_n) := -(d_1, d_2, \ldots, d_n)$;
  - Let $x_{k+1}^k \leftarrow \hat{x}$ and $k \leftarrow k + 1$;
- **end if**
- **until** $\hat{s} \leq 0$ or $k > K$;
- **end while**
- Algorithm ends and return $x_0$.

5.1 HTWSC-1

The most intuitional weighting method is equal weighting, which we call “HTWSC-1”:
\[ w_1^k = w_2^k = \cdots = w_n^k = \frac{1}{n} \] (26)

The equal weighting method does not consider the structural information and the positional relationship of the objective function and the feasible domain. Consequently, the HTWSC-1 may not perform efficiently despite its simplicity.
5.2 HTWSC-2

In order to take full advantage of the structural information as well as the relative position information of the objective function and the feasible domain, the second weighting method is proposed based on the distances from the contour surface $\Gamma_y$ around each of the points $x^k_1, x^k_2, \ldots, x^k_n$ to the feasible domain $\Omega_k$, called HTWSC-2.

For any $x^k_i$, $i = 1, 2, \ldots, n$, following the process of facet ascertaining in Section 4.2.5, we can obtain a facet of the objective function $f(x)$ at point $x^k_i$, denoted by $\varphi^k_i$, and then the corresponding contour facet $\Gamma^k_i$.

Then, we denote the distance between $\Gamma^k_i$ and $\Omega_k$ by $\text{dist}\{\Gamma^k_i, \Omega_k\}$. The smaller the $\text{dist}\{\Gamma^k_i, \Omega_k\}$ is, the nearer to $\Omega_k$ the current contour surface $\Gamma_y$ is around point $x^k_i$. Thus if the direction for tunneling deflects to $x^k_i$, there would be a greater likelihood that $\Gamma_y$ re-intersects $\Omega_k$ after the hill tunneling process. As can be seen in Fig. 11, $\Gamma^k_1$ is nearer to $\Omega_k$ than $\Gamma^k_2$, and there is a greater likelihood that $\Gamma_y$ re-intersects $\Omega_k$ after the hill tunneling process when the tunneling direction deflects to $x^k_1$.

Hence, it seems to be a reasonable weighting method that each $w^k_i$ becomes inversely proportional to $\text{dist}\{\Gamma^k_i, \Omega_k\}$.

Considering that the quadratic $\text{dist}\{\Gamma^k_i, \Omega_k\}$ cannot be worked out conveniently, further simplification is necessary. Similar to the analysis in Section 4.2.6, if we solve problem (24) while replacing parameters $a_k$ and $b_k$ with the counterparts in the definition equation of $\varphi^k_i$, then the obtained $\hat{s}$, denoted by $s^k_i$, is an index that measures something of $\text{dist}\{\Gamma^k_i, \Omega_k\}$. More specifically, $s^k_i$ is proportional to $\text{dist}\{\Gamma^k_i, \Omega_k\}$ while $s^k_i > 0$, and $s^k_i \leq 0$ indicates $\text{dist}\{\Gamma^k_i, \Omega_k\} = 0$.

Therefore, HTWSC-2 is proposed as follows: if $s^k_i > 0$ for every $i = 1, 2, \ldots, n$, set

$$w^k_i = \frac{1/s^k_i}{1/s^k_1 + 1/s^k_2 + \cdots + 1/s^k_n} \quad (27)$$

or else, pick an index $i$ that meets $s^k_i \leq 0$ and set the corresponding weight as $w^k_i = 1$, while $w^k_j = 0$, $j \neq i$ simultaneously.

HTWSC-2 uses the integrated information of the problem, which is beneficial for optimizing the tunneling direction. However, it needs to solve $n$ LP problems in each iteration of weight computation, which leads to a heavy calculation burden that may affect the algorithmic efficiency. In order to overcome this potential shortcoming, different weighting methods based on HTWSC-2 are proposed in the following to decrease the computation burden.

5.3 HTWSC-3

It can be observed that there are two major factors affecting $\text{dist}\{\Gamma^k_i, \Omega_k\}$: one is the dihedral angle between the contour facet $\Gamma^k_i$ and the contour facet $\Gamma^{k-1}$ where $x_k$ is located, denoted by $\angle_{ik} = \angle_{\Gamma^k_i, \Gamma^{k-1}}$; the other is the distance of the $\gamma$-extension $x^k_i$ from its origin $x_k$, denoted by $\|x^k_i - x_k\|$.

For any given $i = 1, 2, \ldots, n$, a smaller $\angle_{ik}$ would ensure a smaller $\text{dist}\{\Gamma^k_i, \Omega_k\}$ with other conditions remaining stable, as illustrated in Fig. 12. This motivates us to propose HTWSC-3 as

$$w^k_i = \frac{\angle^{-1}_{ik}}{\angle_{1k}^{-1} + \angle_{2k}^{-1} + \cdots + \angle_{nk}^{-1}} \quad (28)$$

5.4 HTWSC-4

HTWSC-3 is proposed based on one of the major factors that affect $\text{dist}\{\Gamma^k_i, \Omega_k\}$, the dihedral angle $\angle_{ik}$. HTWSC-4 is designed based on the other one, the extension distance $\|x^k_i - x_k\|$.

For any given $i = 1, 2, \ldots, n$, a bigger $\|x^k_i - x_k\|$ would ensure a bigger $\text{dist}\{\Gamma^k_i, \Omega_k\}$ with other conditions remaining stable, as illustrated in Fig. 13.
Then from this perspective, HTWSC-4 is proposed as

$$w_i^k = \frac{1}{\|x_i^k - x_k\|} + \frac{1}{\|x_2^k - x_k\|} + \cdots + \frac{1}{\|x_n^k - x_k\|} \tag{29}$$

HTWSC-4 could also be understood from a different point of view. Actually, \(\|x_i^k - x_k\|\) shows the trend of the hill-shaped objective function along the direction \((x_i^k - x_k)\). The bigger \(\|x_i^k - x_k\|\) is, the gentler the objective “hill” slopes, and the more slowly the objective value decreases. HTWSC-4 sets each \(w_i^k\) inversely proportional to \(\|x_i^k - x_k\|\), which in practice makes the obtained tunneling direction deflect to where \(\|x_i^k - x_k\|\) is relatively small. Therefore, this method may accelerate the objective value decreasing process and ultimately improve the algorithmic efficiency.

5.5 HTWSC-5

HTWSC-3 and HTWSC-4 are both simplified approximations of HTWSC-2, and are proposed to decrease the computation burden. However, the factor \(\zeta_{ik}\) in HTWSC-3 and the factor \(\|x_i^k - x_k\|\) are coupled factors that affect \(\text{dist}\{P_i^k, \Omega_k\}\) together, rather than play a part separately.

Considering that the computation burden of either \(\zeta_{ik}\) or \(\|x_i^k - x_k\|\) is rather light, we attempt to integrate HTWSC-3 and HTWSC-4. One of the ways to integrate is to mix the weights in both methods with equal proportion, which can be taken as HTWSC-5. This method is expressed as

$$w_i^k = \frac{1}{2} \left( \frac{1}{\|x_i^k - x_k\|} + \frac{1}{\|x_f^k - x_k\|} \right) \tag{30}$$

5.6 HTWSC-6

Despite the perspectives that the above weighting methods are proposed from, there is still a different one. Given that the purpose of hill tunneling is to go on searching in a place far enough from the current local optimal trap, the suggested direction could be supposed to achieve a region as far as possible from the local optimum after tunneling. From this perspective, the weighting method should be proposed so that each \(w_i^k\) is proportional to \(\|x_i^k - x_k\|\), as

$$w_i^k = \frac{\|x_i^k - x_k\|}{\|x_1^k - x_k\| + \|x_2^k - x_k\| + \cdots + \|x_n^k - x_k\|} \tag{31}$$

This weighting method is the proposed HTWSC-6.

Obviously, HTWSC-6 is just the opposite of HTWSC-4, and which is the superior one that needs to be tested in practice. Actually, all the six weighting methods proposed from various perspectives are different from one another so their performances need to be compared. The comparison will be carried out through numerical experiments in the next section, and the best one will be chosen as the final weighting method for the HTWSC algorithm.

6 Numerical Results

In this section, we focus on the numerical experiments and the results analysis. First, we test HTWSC-1 to HTWSC-6 with randomly generated numerical examples, and choose the best as the weighting methods for the HTWSC algorithm. Then, the performance of HTWSC is compared to the HD algorithm and CPLEX for the equivalent MIP.

Computations are conducted on a Windows machine with Core i3 3.30 GHz processor and 8 GB of RAM. HTWSC and HD are implemented with Matlab 2010a, while MIP with CPLEX 12.4.

6.1 Test objects

The test set \(P\) consists of randomly generated CPWL problems in the following form:

$$\min \sum_{i=1}^{M} \beta_i \max_{1 \leq j \leq N_i} \{a_{ij}^T x + b_{ij}\} \tag{32}$$

where \(\beta_i \in \{1, -1\}\), and \(a_{ij} \in \mathbb{R}^N, b_{ij} \in \mathbb{R}, \forall i, j\), while \(N_i \leq L\) with parameter \(L\) denoting the upper bound of each \(N_i\). Then the problem size obviously depends on parameters \(L, M, \) and \(N\).

Regarding the parameter \(L\), without loss of generality, we let \(L = 3\) uniformly. As to \(M\) and \(N\), various practical problems require various dimension parameters, \(N\), and different basis function numbers, \(M\), satisfy different approximation accuracy requirements. Taking these into account, we set the \(M\)
range as \{10, 30, 50, 80, 100, 120, 150, 180, 200\}, and
the \(N\) range as \{3, 5, 8, 10, 30, 50, 80, 100\}.

Moreover, we denote a group of same-sized test
problems as \(P_{n,m}\), with \(N = n\) and \(M = m\). Then
the test set \(P\) can be expressed as the disjoint union of
subsets \(P_{n,m}\), that means \(P = \bigcup P_{n,m}\).

For convenience, all the test problems are restricted to
the same feasible domain \([0, 1]^N\), and 20 test problems
for each \(P_{n,m}\) are generated. The parameters in each
test problem are generated with the following steps:
(1) \(\beta_i\) takes value from \{1, -1\} with equal possibility;
(2) \(N_i\) takes value from \{1, 2, 3\} with equal possibility;
(3) Generate each component of vector \(a_{ij}\) with
uniform distribution on \([-1, 1]\);
(4) Figure out the upper bound \(b_{ij}^U\) and the
lower bound \(b_{ij}^L\) of scalar \(b_{ij}\), which ensure
that hyperplane \(a_{ij}^T x + b_{ij} = 0\) intersects with
\([0, 1]^N\), then select \(b_{ij}\) with uniform distribution
on \([b_{ij}^L, b_{ij}^U]\).

6.2 Experimental scheme

Briefly speaking, the performances of HTWSC-1
to HTWSC-6 on test set \(P\) are studied in the
first stage. Then, the best weighting methods are
selected for application in the HTWSC algorithm
used in subsequent experiments. These are tested
with randomly generated numerical examples, through
which the efficiencies of solvers HTWSC, HD, and MIP
are analyzed comprehensively.

To be specific, given a test problem \(p \in P\), on the one
hand, we can take it as problem (4), and then transform
it to problem (6) following the process described in
Section 2.1.2. By adding in the constraints \(x \in [0, 1]^N\)
further, the test problem \(p\) is converted to a concave
minimization over a convex polyhedron, which is in the
form of model (7) and can be solved with solver HD
or the HTWSC algorithm with the different proposed
weighting methods proposed.

On the other hand, after the given \(p \in P\) is
transformed to model (7), it could be converted to the
equivalent MIP model (33) further.

\[
\begin{align*}
& \min \sum_{i=1}^{M} \eta_i, \\
& \text{s.t. } C x \leq g, \\
& \quad \eta_i \leq a_{ij}^T x + b_{ij}, \\
& \quad \forall 1 \leq j \leq N_i, 1 \leq i \leq M, \\
& \quad a_{ij}^T x + b_{ij} \leq \eta_i + U y(i, j), \\
& \quad \forall 1 \leq j \leq N_i, 1 \leq i \leq M, \\
& \quad \sum_{j=1}^{N_i} y(i, j) = N_i - 1, \\
& \quad y(i, j) \in \{0, 1\}, \\
& \quad \forall 1 \leq j \leq N_i, 1 \leq i \leq M
\end{align*}
\]

Here \(x, \eta\), and all the \(y(i, j)\) are decision variables,
and \(U\) is a positive scalar that is large enough. At this
time, CPLEX can be applied to show the performance
of solver MIP.

Throughout our numerical experiments, the upper
bound limit on CPU time in each test is \(T = 500\) s.

6.3 Performance profiles

Given the problem set \(P\) and a solver set \(S\), one
can evaluate the performance of a solver \(s \in S\) on \(P\)
from two aspects: the operation times and the optimal
values. The shorter runtimes and the more optimal
values, the more excellent the algorithmic performance.
To ensure the consistency of the evaluations over
problems in different scales, we introduce the following
performance profiles referencing on Ref. [23]. When
analyzing the results of \(s \in S\) on \(p \in P\), the
performance profiles, Performance Ratio (PR) and
Superior Ratio (SR), are defined as

\[
\text{PR}_{p,s} = \frac{E_{p,s}}{\min\{E_{p,s*} : s* \in S\}}
\]

and

\[
\text{SR}_s = \frac{\text{card}\{p \in P : E_{p,s} \leq E_{p,s*}, \forall s* \in S\}}{\text{card} P}
\]

Here “card” is cardinality function, and \(E_{p,s}\) is set as

\[
E_{p,s} = v_{p,s} - \min\{v_{p,s*} : s* \in S\} + \\min\{v_{p,s*} : s* \in S\}
\]

where \(v_{p,s}\) is the optimal value of solver \(s \in S\) in
solving \(p \in P\).

Moreover, based on PR, another performance, profile
General Superior Ratio (GSR), is defined as

\[
\text{GSR}_s(\tau) = \frac{\text{card}\{p \in P : \text{PR}_{p,s} \leq \tau\}}{\text{card} P}
\]

Clearly, \(\text{PR}_{p,s} \geq 1\), thus \(\tau \geq 1\). In fact, \(\text{SR}_s\) is a special case of \(\text{GSR}_s(\tau)\), that is, \(\text{SR}_s = \text{GSR}_s(1)\).

6.4 Numerical results of different weighting
methods

In the first stage of experiments, the solver set \(S\) is
designated as \(S = \{\text{HTWSC-}i : i = 1, 2, \ldots, 6\}\), and
the results are illustrated from two aspects of solving
efficiency and optimizing capacity.
6.4.1 Solving efficiency
The average running time $t_s$ is calculated on each subset $P_{n,m}$ to show the solving efficiency of solver $s \in S$ in solving problems with the same scale of $N = n$ and $M = m$.

As shown in Figs. 14 and 15, HTWSC-1, HTWSC-3, and HTWSC-4 share a similar solving efficiency, which is superior to those of other weighting methods.

6.4.2 Optimizing capacity
To study the optimizing capacity of solver $s \in S$, SR$_s$ caring about the optimal value is calculated on each $P_{n,m}$. Further, $N = 5$, $N = 50$, $M = 50$, and $M = 150$ are taken to represent four typical situations of lower $N$, higher $N$, lower $M$, and higher $M$, to demonstrate the results.

As shown in Figs. 16–19, HTWSC-1, HTWSC-3, and HTWSC-4 possess obvious superiority than the others. Furthermore, HTWSC-4 achieves the best SR in all the four typical situations except that of $M = 50$.

6.4.3 Summary
Numerical results show that HTWSC-1, HTWSC-3, and HTWSC-4 have the best performance among all the weighting methods. They share a similar solving efficiency, whereas HTWSC-4 shows the best optimizing capacity in three of the four typical situations. Therefore, HTWSC-4 is selected to applied in the HTWSC algorithm in subsequent experiments.

6.5 Performance comparison among HTWSC, HD, and MIP
In this stage, the solver set $S$ is designated as $S = \{\text{HTWSC, HD, MIP}\}$, where the solver HTWSC refers
to HTWSC-4 according to the previous stage.

6.5.1 Solving efficiency

It is observed that all the three algorithms share a similar solving efficiency when the problem scale is relatively small.

In Figs. 20 and 21, the running time with MIP has a substantial increase, as the problem scale increases. This is because the branch and bound method used in CPLEX for MIP is an exhaustive method, and the complexity is of the same order as the combinatorial number. By contrast, running times with HD and HTWSC increase slowly, and that with HTWSC is always the relatively smaller one.

6.5.2 Optimizing capacity

Here we still calculate the optimizing SR in the four situations of lower $N$, higher $N$, lower $M$, and higher $M$, to study the optimizing capacities.

As shown in Figs. 22–25, the HTWSC algorithm achieves 100% SR in all the four situations of lower $N$, higher $N$, lower $M$, and higher $M$, respectively. Especially when the problem scale is not too large for the MIP algorithm to provide the true global optimum, the global optimality of the HTWSC solution can be easily verified. Thus, the HTWSC algorithm possesses the best optimizing capacity.

6.5.3 Further comparison on typical $P_{m,n}$

In order to take a closer look at the differences among the optimal performance of the three algorithms, some further comparisons are carried out on several typical $P_{m,n}$, by calculating the optimizing GSR($\tau$).

As can be observed from Figs. 26–29, when $M$ is relatively small, the differences among the optimal values of the three algorithms are uniformly less than...
Fig. 26 Differences of optimal performance on $P_{5,50}$, $N=5$, $M=50$.

Fig. 27 Differences of optimal performance on $P_{10,50}$, $N=10$, $M=50$.

Fig. 28 Differences of optimal performance on $P_{50,50}$, $N=50$, $M=50$.

Fig. 29 Differences of optimal performance on $P_{100,50}$, $N=100$, $M=50$.

5% on the test problems with various $N$. This means that the difference of the optimal performance is not obvious in this case.

Figures 30–33 show that when $M$ is relatively large, the differences of the optimal performance expand.

Fig. 30 Differences of optimal performance on $P_{5,150}$, $N=5$, $M=150$.

Fig. 31 Differences of optimal performance on $P_{10,150}$, $N=10$, $M=150$.

Fig. 32 Differences of optimal performance on $P_{50,150}$, $N=50$, $M=150$.

Fig. 33 Differences of optimal performance on $P_{100,150}$, $N=100$, $M=150$.

With a lower $N$, HD performs relatively poor, and its optimal values on some test problems are over 10% worse than those of HTWSC (see Fig. 30) and even exceeds 50% (Fig. 31).

Conversely with a higher $N$, MIP performs relatively
poor. Its optimal values are over 5%–10% worse than those of HTWSC on 80% of the test problems, as shown in Fig. 32 and Fig. 33.

6.5.4 Summary
Considering the solving efficiency, the HTWSC shares a similar performance with HD and MIP when the problem scale is small, and shows superiority when the problem scale is large. In terms of optimizing capacity, HTWSC shows the best performance in all kinds of problem scales.

In general, compared to HD and MIP, HTWSC possesses an obvious superiority in terms of both numerical efficiency and global search capability.

7 Conclusion
CPWL programming can be converted to a D.C. programming with equivalency, and then to a concave piecewise linear minimization over a polyhedron. Although the local optima can be easily obtained via descent algorithms, how to escape local traps efficiently for global searching remains a contested issues. In this work, we propose the HTWSC algorithm which can escape a local optimum efficiently by cutting across the super-level set to search on the other side of the hill-shaped concave objective function. Each tunneling path is established via a weighted simplex centroid. Various weighting methods are studied to select the best one for the proposed HTWSC. Furthermore, by exploiting mature techniques such as linear search and LP programming, as well as the cutting plane method, the searching efficiency of HTWSC is improved sharply. Finally, the HTWSC algorithm shows better numerical efficiency and the global search capability when compared with CPLEX and the HD method.

Acknowledgment
This project was jointly supported by the National Key Basic Research and Development (973) Program of China (No. 2012CB720505) and the National Natural Science Foundation of China (Nos. 61473165 and 61134012).

References
[1] L. Breiman, Hinging hyperplanes for regression, classification and function approximation, *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 999–1013, 1993.
[2] G. Dantzig, S. Johnson, W. White, and M. Science, A linear programming approach to the chemical equilibrium problem, *Management Science*, vol. 5, no. 1, pp. 38–43, 1958.
[3] S. Zions, *Linear and Integer Programming*. Prentice Hall, 1974.
[4] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1963.
[5] A. Charnes and W. W. Cooper, Management models and industrial applications of linear programming, *Management Science*, vol. 4, no. 1, pp. 38–91, 1957.
[6] A. Charnes and C. E. Lemke, Minimization of nonlinear separable convex functionals, *Naval Research Logistics Quarterly*, vol. 1, no. 4, pp. 301–312, 1954.
[7] H. M. Wagner, Principles of operations research, *Operational Research Quarterly*, vol. 21, no. 4, 1970.
[8] G. B. Dantzig, Recent advances in linear programming, *Management Science*, vol. 2, no. 2, pp. 131–144, 1956.
[9] J. K. Ho, *Relationships among Linear Formulations of Separable Convex Piecewise Linear Programs*. Springer, pp. 126–140, 1985.
[10] R. Fourer, A simplex algorithm for piecewise-linear programming I: Derivation and proof, *Mathematical Programming*, vol. 33, no. 2, pp. 204–233, 1985.
[11] R. Fourer, A simplex algorithm for piecewise-linear programming II: Finiteness, feasibility and degeneracy, *Mathematical Programming*, vol. 41, nos. 1–3, pp. 281–315, 1988.
[12] R. Fourer, A simplex algorithm for piecewise-linear programming III: Computational analysis and applications, *Mathematical Programming*, vol. 53, nos. 1–3, pp. 213–235, 1992.
[13] A. B. Keha, I. R. de Farias, and G. L. Nemhauser, A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization, *Operations Research*, vol. 54, no. 5, pp. 847–858, 2006.
[14] B. Gendron, K. L. Croxton, and T. L. Magnanti, A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems, *Management Science*, vol. 49, no. 9, pp. 1268–1273, 2003.
[15] J. P. Vielma, S. Ahmed, and G. Nemhauser, Mixed-integer models for nonseparable piecewise linear optimization: Unifying framework and extensions, *Operations Research*, vol. 58, no. 2, pp. 303–315, 2010.
[16] A. B. Keha, J. A. G. L. Ismael, and R. De Farias, Models for representing piecewise linear cost functions, *Operations Research Letters*, vol. 32, no. 1, pp. 44–48, 2004.
[17] X. Xi, J. Xu, X. Mu, and S. Wang, Continuous piecewise linear programming via concave optimization and genetic algorithm, in *Proc. Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Osaka, Japan, 2012, pp. 2509–2514.
[18] X. Huang, J. Xu, X. Mu, and S. Wang, The hill detouring method for minimizing hinging hyperplanes functions, *Computers & Operations Research*, vol. 39, no. 7, pp. 1763–1770, 2012.
[19] X. Huang, J. Xu, and S. Wang, Exact penalty and optimality condition for nonseparable continuous piecewise linear programming, *Journal of Optimization Theory and Applications*, vol. 155, no. 1, pp. 145–164, 2012.
[20] S. Wang and X. Sun, Generalization of hinging hyperplanes, *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4425–4431, 2005.

[21] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*. Springer Science & Business Media, 1996.

[22] A. M. Bagirov, L. Jin, N. Karmitsa, A. Al Nuaimat, and N. Sultanova, Subgradient method for nonconvex nonsmooth optimization, *Journal of Optimization Theory and Applications*, vol. 157, no. 2, pp. 416–435, 2013.

[23] E. D. Dolan and J. J. More, Benchmarking optimization software with performance profiles, *Mathematical Programming, Series B*, vol. 91, no. 2, pp. 201–213, 2002.

Zhiming Xu received the BS degree in 2006 and MS degree in 2009 from National University of Defense Technology in China. He is currently a PhD candidate at Department of Automation, Tsinghua University, Beijing, China. He is concurrently a lecturer in Air Force Engineering University, Xi’an, China. He has participated in several research projects granted from MOST and NSFC. His active research areas include nonlinear optimization, system engineering, and mobile ad hoc networks.

Kuangyu Liu received the BS degree from Huazhong University of Science and Technology, China, in 2008, and the MS degree in system engineering from Huazhong University of Science and Technology, China, in 2012. He is currently a PhD candidate in the Department of Automation, Tsinghua University, China. His research interests include compressed sensing and piecewise linear programming.

Shuning Wang received the BS degree from Hunan University, China in 1982, the MS and the PhD degrees both in system engineering from Huazhong University of Science and Technology, China in 1984 and 1998, respectively. He was an associate professor from 1992 to 1993 and a full professor from 1994 to 1995, at the Institute of Systems Engineering, Huazhong University of Science and Technology. Since 1996, he has been a full professor in Department of Automation, Tsinghua University, China. His current research interests are mainly in developing practical methods for nonlinear system identification, control, and optimization via piecewise-linear approximation.

Yu Bai received the BS degree in 2013 from Beihang University in China. He is currently a PhD candidate of Department of Automation, Tsinghua University, Beijing, China. His major research interests focus on continuous piecewise linear optimization and application.