J Optim Theory Appl
DOI 10.1007/s10957-012-0032-7

Exact Penalty and Optimality Condition for Nonseparable Continuous Piecewise Linear Programming

Xiaolin Huang · Jun Xu · Shuning Wang

Received: 29 October 2011 / Accepted: 12 March 2012

Abstract Utilizing compact representations for continuous piecewise linear functions, this paper discusses some theoretical properties for nonseparable continuous piecewise linear programming. The existence of exact penalty for continuous piecewise linear programming is proved, which allows us to concentrate on unconstrained problems. For unconstrained problems, we give a sufficient and necessary local optimality condition, which is based on a model with universal representation capability and hence applicable to arbitrary continuous piecewise linear programming. From the gained optimality condition, an algorithm is proposed and evaluated by numerical experiments, where the theoretical properties are illustrated as well.

Keywords Piecewise linear · Nonlinear programming · Exact penalty · Local optimality condition

1 Introduction

Continuous piecewise linear (CPWL) programming stands for optimization problems with CPWL objective functions and CPWL constraints. Intuitively, CPWL programming technique is useful for continuous optimization. On the one hand, a CPWL programming can be transformed into a series of linear problems, for which there

Communicated by Gianni Di Pillo.

X. Huang · J. Xu · S. Wang (✉)
Department of Automation, Tsinghua University, Beijing, P.R. China
e-mail: swang@mail.tsinghua.edu.cn

X. Huang
e-mail: huangxl06@mails.tsinghua.edu.cn

J. Xu
e-mail: yun-xu05@mails.tsinghua.edu.cn

Published online: 30 March 2012

are some very efficient algorithms. On the other hand, any continuous function can be approached by a CPWL function to arbitrary precision, so that any continuous nonlinear programming can be solved approximately by CPWL programming.

In the 1960s, Beale has applied CPWL approximation and CPWL programming to minimize a separable nonlinear function; see [1, 2] for details. Since then, CPWL programming has attracted the attention of researchers. For example, problems without explicit formulation have been discussed in [3] and a local optimality condition has been given, which is applicable to a special class of CPWL functions. In order to analyze complicated CPWL problems, it is necessary to represent a CPWL function explicitly. For example, via expressing one-dimensional CPWL functions by the breakpoints, a generalized simplex algorithm for minimizing separable CPWL functions has been proposed in [4–6]. Similarly, a series of formulations for CPWL functions have been given in [7–10], and so on. These models have been summarized and insightfully compared in [11]. Another kind of methods for modeling CPWL function, called compact representation, is motivated by circuit analysis. This kind of models can be found in [12–17]. Compact representation methods have been used in identification, control, and other applications, but researches on CPWL programming in compact representation are rare. In this paper, we first give the comparison between the above two classes of models and then study CPWL programming in compact representation.

In the sequel, we first review some properties of CPWL functions and introduce representation methods in Sect. 2, where the comparison among these representations is discussed. Section 3 proves the existence of exact penalty. This property allows us to concentrate on unconstrained CPWL programming. For the unconstrained case, a local optimality condition is proved and an algorithm is established in Sect. 4. Then, we use numerical experiments to illustrate the gained theoretical results and evaluate the proposed algorithm in Sect. 5. Finally a brief conclusion is given in Sect. 6.

2 Continuous Piecewise Linear Functions and Representations

Generally, a CPWL programming problem takes the following form:

$$\min f_0(x) \quad \text{s.t.} \quad f_j(x) \le 0, \quad 1 \le j \le J, \tag{1}$$

where $f_j(x), 0 \le j \le J$, are continuous piecewise linear functions, and the domain of the problem is denoted by Ω.

Basically, a CPWL function is a continuous function, which equals to one of the finite distinct affine functions at any point in its domain. Let $f(x)$ be a CPWL function on a convex set Ω. We have

$$f(x) \in \{l_1(x), l_2(x), \ldots, l_N(x)\}, \quad \forall x \in \Omega, \tag{2}$$

where $l_i(x), 1 \le i \le N$, are finite distinct affine functions. From (2), the subregions can be defined as $\Omega_i = \{x : f(x) = l_i(x), x \in \Omega\}$ and the domain is partitioned into

N subregions, i.e.,

$$\Omega = \bigcup_{i=1}^{N} \Omega_i \quad \text{and} \quad \overset{\circ}{\Omega}_i \cap \overset{\circ}{\Omega}_j = \emptyset, \quad \forall i \neq j,$$

where $\overset{\circ}{\Omega}_i$ denotes the interior of Ω_i. And the continuity of $f(x)$ requires $l_i(x) = l_j(x), \forall x \in \Omega_i \cap \Omega_j, \forall i, j$. Furthermore, it has been proved in [18] that if Ω is a polyhedron, then each subregion Ω_i is either a polyhedron or a union of polyhedra. Thus, without any loss of generality, we assume that Ω and $\Omega_i, \forall i$, are polyhedra and we refer to Ω_i as *linear subregion*. Then, one can represent a CPWL function as

$$f(x) = l_i(x), \quad \forall x \in \Omega_i.$$

In the above expression, $f(x)$ is defined by affine functions in subregions. Hence we name it as *piecewise representation*. If (1) is given in piecewise representation, then the optimum can be obtained via solving problems in the linear subregions one by one. However, it is not easy to construct CPWL functions in piecewise representation to describe complicated systems, since it is very hard to satisfy the continuity condition when adjusting the parameters. Therefore, it is more interesting to consider CPWL programming expressed by some explicit formulations given in Sects. 2.1 and 2.2.

2.1 Vertex Representation

It is popular to use the vertices of Ω_i and some binary variables or special ordered sets to represent a CPWL function. A typical method is called a convex combination model, which has been discussed insightfully in [7, 9, 19, 20]. This formulation can represent any separable CPWL function $f(x) = \sum_{j=1}^{n} f^j(x(j))$, where $f^j(x(j))$ is one-dimensional piecewise linear function, which is affine on the segment $[d_j^k, d_j^{k+1}]$ for any $k = 0, 1, \ldots, K-1$. If the breakpoints (the vertices in one dimension) d_j^k and their function values are all known, then we can write

$$x(j) = \sum_{k=0}^{K} d_j^k \lambda_j^k, \quad \forall j \quad \text{and} \quad f(x) = \sum_{j=1}^{n} \sum_{k=0}^{K} f^j\big(d_j^k\big) \lambda_j^k, \tag{3}$$

where $\sum_{k=0}^{K} \lambda_j^k = 1$, $\lambda_j^k \geq 0$, and each $\{\lambda_j^k\}$ must be a *special ordered set of type 2 (SOS2)*, which has been discussed in [2] and defined in [21]. For a given j, set $\{\lambda_j^k\}$ being a SOS2 means that: (i) all the variables are nonnegative; (ii) at most two variables can be positive; (iii) if there are two positives, then their subscripts k must be adjacent. SOS2 is actually a kind of logistic constraint. Hence, when the functions in CPWL programming (1) are represented by a convex combination model, (1) can be formulated as a mixed-integer programming (MILP).

For a comparison, another formulation, named incremental model, has been proposed in [9]. This model uses another special constraint called *SOSX*. The original forms of convex combination model and incremental model are restricted to separable

CPWL functions. When extending them to high dimensional space, the breakpoints become the vertices of the linear subregions. If all the vertices of Ω_i are known, then nonseparable CPWL functions can be represented by some models and most of these models are locally ideal. For details, one can see [11], where the computational complexity and the sizes of the variables are compared. Since the formulations in [11] are built from vertices, we call them *vertex representations* of CPWL functions.

In some particular problems, such as merge-in-transit, transportation problem and network flow problem, the objective functions can be represented by one of the vertex representations; see [8, 20] for details. Then, the corresponding problem can be posed as a MILP and solved. Also, we want approximately to solve nonlinear problems via CPWL programming. Thus, there have been some works on nonlinear interpolation or fitting using vertex representations. Recently, a method has been proposed by [22] to interpolate nonlinear function in 2 or 3 dimensions. This method partitions the domain into simplices and uses the convex combination of the vertices to get a CPWL function. Similarly, a method has been discussed in [23] for approximating two-dimensional nonlinear functions by CPWL functions constructed from simplicial partition, whose structure can be optimized.

2.2 Compact Representation

In order to describe piecewise linear resistors, the first *compact representation* for CPWL functions has been proposed in [12]. The main property of compact representation models is that CPWL functions are represented in some closed-form formulations and the continuity on the boundaries of the linear subregions holds naturally. Basically, these representations can be written as linear combinations of basis functions. Take the model proposed by [13] and called hinging hyperplanes (HH) as an example. The HH has the following form:

$$f(x) = l_0(x) + \sum_{m=1}^{M} \beta_m H_m(x) = l_0(x) + \sum_{m=1}^{M} \beta_m \max\{0, l_m(x)\}, \tag{4}$$

where $H_m(x)$ are the basis functions, the number of which is M, and $l_m(x)$ are affine functions. Since the composition of continuous functions is still continuous, the continuity of (4) holds naturally and no additional constraints, such as SOS2, are needed. The HH can be used to approximate nonlinear systems and its parameters can be identified from observed data by algorithms in [24, 25]. These algorithms are effective and the HH has been applied widely; see e.g. [26–29].

Another popular compact representation is named high level canonical piecewise linear representation (HL-CPWL), which has been given by [14] and discussed in [30]. The HL-CPWL can approximate high dimensional functions and has been applied in [31–33], and so on. The HL-CPWL is based on simplicial partition: the basis functions are constructed by the boundaries of the simplices and the linear coefficients of the basis functions are calculated through the least-squares method. Adjusting the coefficients is actually configuring the values for the vertices of the simplices. This is quite different from the methods used in [22, 23] where the function values at the vertices are unchanged and should be given.

Although the HH and HL-CPWL perform well in approximating some nonlinear functions, they cannot represent all the CPWL functions in 2 or higher dimensions. The lack of universal representation capability essentially affects the identification performance, which has been analyzed and explained in [18]. Without this capability, we may need a lot of basis functions to achieve satisfactory approximation precision. Moreover, only when a model has such a capability, the corresponding theoretical results are applicable to all CPWL functions. Therefore, researchers pursue CPWL models with universal representation capability. From (4), it is not hard to see that the boundaries of the subregions of the HH must be lines throughout the whole space. Hence, the structure of the HH is not flexible enough. This is the reason why the HH cannot represent all the CPWL functions. Naturally, the capability of representation can be extended via adding affine functions to generalize the basis function, i.e., letting the basis function be $B_m(x) = \max\{l_1^m(x), l_2^m(x), l_3^m(x), \ldots\}$. The model using such basis function is called generalized hinging hyperplanes (GHH). It has been proved in [15] that in n-dimensional space, using at most $n+1$ affine functions in $B_m(x)$, arbitrary CPWL function can be represented. That is to say, for arbitrary CPWL function $f(x) : \mathbb{R}^n \to \mathbb{R}$, there are M basis functions $B_m(x)$ and coefficients β_m such that

$$f(x) = \sum_{m=1}^{M} \beta_m B_m(x) = \sum_{m=1}^{M} \beta_m \max\left\{l_1^m(x), l_2^m(x), \ldots, l_{n+1}^m(x)\right\}, \tag{5}$$

where $\beta_m = \pm 1$, $B_m(x)$ are basis functions and $l_i^m(x)$, $1 \le i \le n+1$, are affine functions. The identification algorithm for the GHH has been given by [34] and since the GHH has universal representation capability, we can analyze CPWL programming by using GHH to represent CPWL functions.

2.3 Comparison of Vertex and Compact Representation

Vertex representation and compact representation provide two ways to express CPWL functions. Although in theory both the methods can represent all CPWL functions, the transformation between them is not a tractable problem except for special cases, e.g., the separable functions. Vertex representation (3) can be expressed by the HH as follows:

$$f(x) = \sum_{j=1}^{n} f^j\big(x(j)\big) = \sum_{j=1}^{n} \sum_{k=1}^{K} a_j^k \max\left\{0, x(j) - d_j^{k-1}\right\},$$

where

$$a_j^k = \begin{cases} \frac{f^j(d_j^1) - f^j(d_j^0)}{d_j^1 - d_j^0}, & k = 1, \\ \frac{f^j(d_j^k) - f^j(d_j^{k-1})}{d_j^k - d_j^{k-1}} - \sum_{l=1}^{k-1} a_j^l, & k = 2, 3, \ldots, K. \end{cases}$$

Meanwhile, when the HH (4) is separable, i.e., the basis functions take the form of $H_m(x) = \beta_m \max\{0, a_m x(j) - b_m\}$, we can calculate all the breakpoints, such as

$d_j^k = \frac{b_m}{a_m}$, and sort them to obtain the corresponding convex combination model. However, for nonseparable functions, the transformation is not easy. On the one hand, when a CPWL function is given in compact representation, to obtain the equivalent vertex representation we should know all the vertices and the topological relations, but the number of the vertices may be huge. Consider (4) as an example. Each basis function $\max\{0, l_m(x)\}$ defines a boundary of linear subregions, i.e. $l_m(x) = 0$, and in n-dimensional space, a point is determined by n lines. Thus, the function represented by (4) has $\binom{M}{n}$ vertices and the translation to any vertex representation is very time-consuming when M and n are large. On the other hand, transforming a CPWL function in vertex representation to any compact representation is very hard as well and no practical method has been established. Therefore, CPWL programming in both vertex and compact representations is worthy of study. In practice, one should choose suitable technique according to the specifical formulation of (1).

Using a CPWL function to approach the nonlinear objective function and then minimizing the obtained surrogate model is the major motivation of the research on CPWL programming. Especially, when the concerned relationship is unclear or too complicated to be optimized directly, we need to do approximation before applying optimization technique. For this purpose, some methods for approximation using vertex representations have been proposed in [22, 23]. However, for high dimensional problems, it may be better to consider compact representation models.

The significant advantage of vertex representation is that the corresponding CPWL programming is a MILP and can be solved by mature techniques. From the equivalence of vertex representation and compact representation, one can expect that a CPWL programming represented by compact representation can be written as a MILP as well. Consider the following linearly constrained problem, whose objective function takes an HH formulation:

$$\min\ f(x) = l_0(x) + \sum_{m=1}^{M} \beta_m \max\{0, l_m(x)\} \quad \text{s.t.} \quad Ax \le b. \tag{6}$$

By introducing M binary variables $y_1, y_2, \ldots, y_M$, the above problem can be equivalently transformed into the following MILP:

$$\begin{aligned}
&\min\ f(x) = l_0(x) + \sum_{m=1}^{M} \beta_m z_m \\
&\text{s.t.} \quad Ax \le b \\
&\qquad -(1 - y_m)U \le l_m(x) \le y_m U, \quad m = 1, 2, \ldots, M \\
&\qquad 0 \le z_m \le y_m U, \\
&\qquad\quad l_m(x) - U(1 - y_m) \le z_m \le l_m(x) + U(1 - y_m), \quad m = 1, 2, \ldots, M \\
&\qquad y_m \in \{0, 1\}, \quad m = 1, 2, \ldots, M,
\end{aligned} \tag{7}$$

where U is positive and large enough. Notice that although (7) provides an easy and convenient way to obtain the equivalent MILP for (6), it is not a good formulation,

since a MILP with large positive U may have bad LP relaxation. Meanwhile, transforming (6) into one-vertex representation and then getting the equivalent MILP is not good either, because the translation is difficult. A better formulation can be gained via further study. In this paper, we simply use (7) to calculate the optima in numerical examples.

A CPWL programming can be transformed into a MILP, but the equivalent MILP has been proved to be NP-hard in [20] and we cannot expect to get the global optimum for large scale problems in acceptable time. In some applications, there is a strict requirement on computing time. For example, in [31, 35, 36], compact representations have been used to identify some chemical processes and model predictive control method has been established, which involves finite horizon open-loop optimization problems with CPWL (or continuous piecewise quadratic, due to different norm) objective functions. It is required that the corresponding programming be solved within a short time (about 30 seconds) in order to obtain the control signal in time. In another application, the power consumption of a chiller plant is described as a CPWL function and the operation point is optimized online via CPWL programming; see [37] for details. For such problems, it is not practical to pursue the global optimum and we should consider local optimality condition. For this, a closely related work has been given in [3], which is applicable only to a special class of CPWL functions. In this paper, a local optimality condition, which is applicable to any CPWL programming, will be given by utilizing the GHH, which has the capability of representing all the CPWL functions.

3 Exact Penalty

First, we discuss the optimization problem with CPWL objective function and CPWL constraints. According to the properties of CPWL functions, the functions in (1) equal to one of the following vector functions at any point in Ω, that is:

$$\begin{bmatrix} f_0(x) \\ f_1(x) \\ \vdots \\ f_J(x) \end{bmatrix} \in \left\{ \begin{bmatrix} p_{10}(x) \\ p_{11}(x) \\ \vdots \\ p_{1J}x \end{bmatrix}, \begin{bmatrix} p_{20}(x) \\ p_{21}(x) \\ \vdots \\ p_{2J}(x) \end{bmatrix}, \ldots, \begin{bmatrix} p_{T0}(x) \\ p_{T1}(x) \\ \vdots \\ p_{TJ}(x) \end{bmatrix} \right\}, \quad \forall x \in \Omega, \tag{8}$$

where $p_{ij}(x), \forall i, j$, are affine functions. And the domain Ω is partitioned into the following T subregions:

$$\Omega_i = \left\{ x : \begin{bmatrix} f_0(x) \\ f_1(x)) \\ \vdots \\ f_J(x) \end{bmatrix} = \begin{bmatrix} p_{i0}(x) \\ p_{i1}(x) \\ \vdots \\ p_{iJ}(x) \end{bmatrix} = \begin{bmatrix} a_{i0}^T x + b_{i0} \\ a_{i1}^T x + b_{i1} \\ \vdots \\ a_{iJ}^T x + b_{iJ} \end{bmatrix} \right\}, \quad 1 \le i \le T. \tag{9}$$

Applying a l_1-norm penalty function, we can approach problem (1) by minimizing the following unconstrained function:

$$F_\lambda(x) = f_0(x) + \lambda \sum_{j=1}^{J} \max\{f_j(x), 0\}, \tag{10}$$

for a succession of increasing positive values of the penalty parameter λ. $F_\lambda(x)$ is obviously a CPWL function, and it can be expressed by the following piecewise representation:

$$F_\lambda(x) = p_{i0}(x) + \lambda \sum_{j=1}^{J} \max\{p_{ij}(x), 0\} = a_{i0}^T x + b_{i0} + \lambda \sum_{j=1}^{J} \max\{a_{ij}^T x + b_{ij}, 0\},$$
$$\forall x \in \Omega_i.$$

Then, each Ω_i is divided into at most 2^J linear subregions Ω_{il} by hyperplanes $a_{ij}^T x + b_{ij} = 0$, $j = 1, 2, \ldots, J$. In general, the number of the linear subregions is much less than 2^J, since lots of hyperplanes $a_{ij}^T x + b_{ij} = 0$ may not intersect with Ω_i. In every subregion Ω_{il}, $F_\lambda(x)$ is an affine function, i.e.,

$$F_\lambda(x) = a_{i0}^T x + b_{i0} + \lambda \sum_{j=1}^{J} (c_{ilj}^T x + d_{ilj}), \quad \forall x \in \Omega_{il},$$

where

$$\text{if} \quad \forall x \in \overset{\circ}{\Omega}_{il}, \quad a_{ij}^T x + b_{ij} > 0, \quad c_{ilj} = a_{ij} \quad \text{and} \quad d_{ilj} = b_{ij},$$
$$\text{if} \quad \forall x \in \overset{\circ}{\Omega}_{il}, \quad a_{ij}^T x + b_{ij} < 0, \quad c_{ilj} = 0 \quad \text{and} \quad d_{ilj} = 0.$$

As assumed, Ω_i are polyhedra, thus Ω_{il} are polyhedra as well and can be determined by some linear inequalities. Hence, Ω_{il} can be written as $\Omega_{il} = \{x : P_{il} x \le q_{il}\}$. Let $x \in \Omega$ be a given point and $\Psi(x) = \{\Omega_{il} : x \in \Omega_{il}\}$ stand for the subregions which contain x. Then, $\Psi(x)$ is not empty and the number of its elements is finite.

A CPWL function is continuous in its domain but not differentiable at some points. Therefore, the concept of the directional derivative is needed in CPWL analysis. The directional derivative of an arbitrary continuous function f at $x \in \mathbb{R}^n$ in the direction d can be defined as

$$\nabla^d f(x) = \lim_{t \mapsto 0^+} \frac{f(x + td) - f(x)}{t}.$$

Using directional derivative, we can prove the following theorem which can derive the existence of exact penalty for (1).

Theorem 3.1 *For CPWL function* (10), *there exists a* $\bar{\lambda}$ *which ensures that for all* $\lambda > \bar{\lambda}$, $F_\lambda(x)$ *have the same local minima.*

Proof $\hat{x}$ is a local minimum of $F_\lambda(x)$ if and only if $\nabla^d F_\lambda(\hat{x}) \geq 0, \forall d \in \mathbb{R}^n$. We use $D_{il}(\hat{x})$ to represent the set of directions along which the ray from $\hat{x}$ still stays in Ω_{il}, i.e., $D_{il}(\hat{x}) = \{d : \exists \bar{t} > 0, \forall 0 \leq t \leq \bar{t}, \hat{x} + td \in \Omega_{il}\}$. Then, there is

$$\nabla^d F_\lambda(\hat{x}) = \left(a_{i0}^T + \lambda \sum_{j=1}^{J} c_{ilj}^T\right) d, \quad \forall d \in D_{il}(\hat{x}).$$

For a given direction d, there is at least one $\Omega_{il} \in \Psi(\hat{x})$ such that $d \in D_{il}(\hat{x})$. Contrarily, in a given subregion $\Omega_{il} \in \Psi(\hat{x})$, there is an $\tilde{x}$ which is different from $\hat{x}$ and belongs to Ω_{il}; hence we can get a direction $d = \tilde{x} - \hat{x} \in D_{il}(\hat{x})$. Therefore, the condition $\nabla^d F_\lambda(\hat{x}) \geq 0, \forall d \in \mathbb{R}^n$ equals

$$\left(a_{i0}^T + \lambda \sum_{j=1}^{J} c_{ilj}^T\right) d \geq 0, \quad \forall d \in D_{il}(\hat{x}),\ \Omega_{il} \in \Psi(\hat{x}). \tag{11}$$

Ω_{il} is determined by $\Omega_{il} = \{x : P_{il}x \leq q_{il}\}$. We denote the kth row of P_{il} by $P_{il}(k)$ and the kth component of q_{il} by $q_{il}(k)$. Then, $\Gamma_{il}(\hat{x}) = \{k : P_{il}(k)\hat{x} = q_{il}(k)\}$ stands for the set of the indices of the active constraints at $\hat{x}$. It is easy to see that $D_{il}(\hat{x}) = \{d : P_{il}(k)d \leq 0, \forall k \in \Gamma_{il}(\hat{x})\}$ and it is convex. According to the Minkowski–Weyl theorem, $D_{il}(\hat{x})$ can be expressed as $D_{il}(\hat{x}) = \{d : d = \sum_m \theta_m \nu_m, \forall \theta_m \geq 0\}$, where the number of ν_m is finite and (11) is equal to

$$\left(a_{i0}^T + \lambda \sum_{j=1}^{J} c_{ilj}^T\right) \sum_m \theta_m \nu_m \geq 0, \quad \forall \theta_m \geq 0,\ \forall \Omega_{il} \in \Psi(\hat{x}).$$

Therefore, $\hat{x}$ is locally minimal if and only if for any $\Omega_{il} \in \Psi(\hat{x})$,

$$\left(a_{i0}^T + \lambda \sum_{j=1}^{J} c_{ilj}^T\right) \nu_m \geq 0, \quad \forall \nu_m. \tag{12}$$

If $\sum_{j=1}^{J} c_{ilj}^T \nu_m = 0, \forall m$, then whether (12) holds or not is independent of λ. Otherwise, we set

$$\lambda_{il}^*(\hat{x}) = \max_{m:\sum_{j=1}^{J} c_{ilj}^T \nu_m \neq 0} \left\{ -\frac{a_{i0}^T \nu_m}{\sum_{j=1}^{J} c_{ilj}^T \nu_m} \right\}.$$

Then, λ has no effect on the validity of inequality (12) as long as $\lambda > \lambda_{il}^*(\hat{x})$, because all $\lambda > \lambda_{il}^*(\hat{x})$ can ensure that

$$\left(a_{i0}^T + \lambda \sum_{j=1}^{J} c_{ilj}^T\right) \nu_m > 0 \quad \Leftrightarrow \quad \sum_{j=1}^{J} c_{ilj}^T \nu_m > 0$$

for any m satisfying $\sum_{j=1}^{J} c_{ilj}^T \nu_m \neq 0$. This property means that $\hat{x}$ is a local optimum of $F_\lambda(x)$ for all $\lambda > \max_{\Omega_{il} \in \Psi(\hat{x})}\{\lambda_{il}^*(\hat{x})\}$ or it is not a local optimum for any $\lambda > \max_{\Omega_{il} \in \Psi(\hat{x})}\{\lambda_{il}^*(\hat{x})\}$.

From the above we can see that if the value of $\sup_{\hat{x}} \max_{\Omega_{il} \in \Psi(\hat{x})} \{\lambda_{il}^*(\hat{x})\}$ is finite, then there exists $\bar{\lambda}$ ensuring that for all $\lambda > \bar{\lambda}$, $F_\lambda(x)$ have the same local minima. One can verify that

$$\sup_{\hat{x}} \max_{\Omega_{il} \in \Psi(\hat{x})} \left\{\lambda_{il}^*(\hat{x})\right\} = \max_{\Omega_{il}} \sup_{\hat{x} \in \Omega_{il}} \left\{\lambda_{il}^*(\hat{x})\right\}.$$

When Ω_{il} is given, ν_m is directly related with $D_{il}(\hat{x})$, which is determined by $\Gamma_{il}(\hat{x})$. The number of different values of $\Gamma_{il}(\hat{x})$ is finite, so is the number of different groups of ν_m. Moreover, there are finite Ω_{il}, thus the number of different values of $\lambda_{il}^*(\hat{x})$ is finite and the maximal value can be achieved, i.e.,

$$\bar{\lambda} = \max_{\Omega_{il}} \max_{\hat{x} \in \Omega_{il}} \left\{\lambda_{il}^*(\hat{x})\right\}.$$

Therefore, all the $F_\lambda(x)$ have the same local minima as long as $\lambda > \bar{\lambda}$. □

If the original problem is unbounded, then (10) is also unbounded. Otherwise, we denote a global optimum for (1) by x^*. If the minimal value of $F_\lambda(x)$ is unbounded for all $\lambda > 0$, then we can sum the following additional constraints for (1),

$$\mathrm{LB} \le x \le \mathrm{UB}, \tag{13}$$

where the vector LB (UB) is the lower (upper) bound for x. With these constraints, x^* is still the optimum as long as $\mathrm{LB} \le x^* \le \mathrm{UB}$. Obviously, for the problem with constraints (13), the penalty function $F_\lambda(x)$ has a lower bound with sufficiently large but finite λ. Moreover, we can design LB and UB satisfying $\mathrm{LB} \le x^* \le \mathrm{UB}$ easily, even when x^* is not exactly known. Thus, without any loss of generality, we can assume that if an optimum of (1) exists, then there is a λ_0 such that a global optimum of $F_\lambda(x), \forall \lambda > \lambda_0$, is achievable. Under this assumption, the existence of exact penalty is guaranteed by the following theorem.

Theorem 3.2 *If x^* is globally optimal for (1), then there exists a $\tilde{\lambda}$ such that for any $\lambda > \tilde{\lambda}$, x^* is a global minimum of $F_\lambda(x)$.*

Proof Define L_λ to be the set of the local minima of $F_\lambda(x)$, i.e., $L_\lambda = \{x : x \text{ is a local minimum of } F_\lambda(x)\}$. From Theorem 3.1 it can be concluded that L_λ keeps unchanged as long as $\lambda > \bar{\lambda}$, and we denote $L = L_\lambda, \forall \lambda > \bar{\lambda}$. As assumed, for any $\lambda > \lambda_0$, the minimal value of $F_\lambda(x)$ is bounded and achievable. Therefore, its global minimum must be locally minimal. Then, for any $\lambda > \max\{\bar{\lambda}, \lambda_0\}$, x^* is a global minimum of $F_\lambda(x)$ if and only if $F_\lambda(x^*) \le F_\lambda(x), \forall x \in L$.

For any $x \in L$, if $f_j(x) \le 0, \forall j = 1, 2, \ldots, J$, then there is $F_\lambda(x^*) \le F_\lambda(x)$ obviously. Otherwise, we can set

$$\lambda_x = \frac{f_0(x^*) - f_0(x)}{\sum_{j=1}^{J} \max\{0, f_j(x)\}}.$$

And one can verify that $F_\lambda(x^*) \le F_\lambda(x), \forall \lambda \ge \lambda_x$. Hence, for any $x \in L$, there is $F_\lambda(x^*) \le F_\lambda(x), \forall \lambda \ge \sup_{x \in L} \lambda_x$.

For any $x \in L$, there exists an Ω_{il} such that $x \in \Omega_{il}$. Since $F_\lambda(x)$ is affine in Ω_{il} and x is a local optimum of $F_\lambda(x)$, x must be an optimum of the corresponding linear programming in Ω_{il}. The number of Ω_{il} is finite, so is the number of different values of $F_\lambda(x), \forall x \in L$. That means when $x \in L$, the number of different values of $f_0(x)$ and that of $\sum_{j=1}^{J} \max\{0, f_j(x)\}$ are both finite. Therefore, $\sup_{x\in L} \lambda_x$ can be achieved and one can expect to get an optimum of (1) using a finite exact penalty factor by choosing $\lambda > \tilde{\lambda}$, where $\tilde{\lambda} = \max\{\bar{\lambda}, \lambda_0, \max_{x\in L} \lambda_x\}$. □

4 Local Optimality Condition

In Sect. 3, the existence of exact penalty is proved. This property allows us to concentrate on the unconstrained continuous piecewise linear programming as the following:

$$\min\ f(x) = f_0(x) + \lambda \sum_{j=1}^{J} \max\big\{0, f_j(x)\big\},$$

where $f_j(x), \forall j$, are CPWL functions. Since the composition of CPWL functions is still continuous and piecewise linear, $f(x)$ is a CPWL function and there exists a GHH formulation to represent it. Hence, the above problem can be represented in the following closed form:

$$\min f(x) = \sum_{m=1}^{M} \beta_m B_m(x) = \sum_{m=1}^{M} \beta_m \max\big\{l_1^m(x), l_2^m(x), \ldots, l_{n+1}^m(x)\big\}, \tag{14}$$

where $\beta_m = \pm 1$, $B_m(x)$ are the basis functions and $l_i^m(x) = a_{mi}^T x + b_{mi}$ are affine functions.

When $\hat{x}$ is given, $B_m(\hat{x})$ is determined and we can define $\mathcal{A}_m(\hat{x}) = \{i : l_i^m(\hat{x}) = B_m(\hat{x})\}$. Since only the functions $l_i^m(x), i \in \mathcal{A}_m(\hat{x})$ can affect the value of $B_m(x)$ in the neighborhood of $\hat{x}$, we call $\mathcal{A}_m(\hat{x})$ the *active index set* of $B_m(x)$ at $\hat{x}$. Then, the sufficient and necessary condition for $\hat{x}$ being a local minimum of (14) is given by the following theorem.

Theorem 4.1 *$\hat{x}$ is a local minimum for (14) if and only if for any $I = [I(1), I(2), \ldots, I(M)]^T$, $I(m) \in \mathcal{A}_m(\hat{x}), \forall m$, there exist scalars $\theta_{mi} \geq 0$ which make the following equality hold*:

$$\sum_{m=1}^{M} \beta_m a_{mI(m)} = \sum_{m=1}^{M} \sum_{i\in\mathcal{A}_m(\hat{x})} \theta_{mi}(a_{mI(m)} - a_{mi}). \tag{15}$$

Proof According to the definition of $\mathcal{A}_m(\hat{x})$, one can find a neighborhood of $\hat{x}$, denoted by $\mathcal{B}_m(\hat{x})$, satisfying

$$B_m(x) = \max_{i\in\mathcal{A}_m(\hat{x})} \big\{l_i^m(x)\big\}, \quad \forall x \in \mathcal{B}_m(\hat{x}).$$

Define $\mathcal{B}(\hat{x}) = \bigcap_{m=1}^{M} \mathcal{B}_m(\hat{x})$. The radius of $\mathcal{B}(\hat{x})$, denoted by r, is positive and in $\mathcal{B}(\hat{x})$, (14) is equal to the following function:

$$f(x) = \sum_{m=1}^{M} \beta_m \max_{i \in \mathcal{A}_m(\hat{x})} \{l_i^m(x)\} = \sum_{m=1}^{M} \beta_m \max_{i \in \mathcal{A}_m(\hat{x})} \{a_{mi}^T x + b_{mi}\}, \quad \forall x \in \mathcal{B}(\hat{x}). \tag{16}$$

Using vector I, we define function $f_I(x)$ as

$$f_I(x) = \sum_{m=1}^{M} \beta_m \big(a_{mI(m)}^T x + b_{mI(m)}\big),$$

and the corresponding polyhedron as

$$\Phi_I = \big\{x : a_{mI(m)}^T x + b_{mI(m)} \geq a_{mi}^T x + b_{mi}, \forall i \in \mathcal{A}_m(\hat{x}), \forall m\big\}.$$

It is easy to see that $f(x) = f_I(x), \forall x \in \mathcal{B}(\hat{x}) \cap \Phi_I$. Next we show that $\hat{x}$ is a local minimum of $f(x)$ if and only if

$$f_I(\hat{x}) \leq f_I(x), \quad \forall x \in \Phi_I, \ \forall I, \tag{17}$$

where $I = [I(1), I(2), \ldots, I(M)]^T$, $I(m) \in \mathcal{A}_m(\hat{x}), \forall m$.

We first prove the necessity by supposing $\exists I_0, \exists \tilde{x} \in \Phi_{I_0}$ and $f_{I_0}(\hat{x}) > f_{I_0}(\tilde{x})$. Since Φ_{I_0} is a polyhedron and $f_{I_0}(x)$ is affine, we know that for any $t \in (0, 1]$, $x_t = \hat{x} + t(\tilde{x} - \hat{x})$ satisfies $x_t \in \Phi_{I_0}$ and $f_{I_0}(\hat{x}) > f_{I_0}(x_t)$. For any $\delta \in (0, r]$, we can find $0 < t_0 \leq \min\{1, \delta/\|\tilde{x} - \hat{x}\|\}$, then, $\|x_{t_0} - \hat{x}\| \leq \delta \leq r$ and $f_{I_0}(\hat{x}) > f_{I_0}(x_{t_0})$. That means both $\hat{x}$ and x_{t_0} belong to $\mathcal{B}(\hat{x}) \cap \Phi_{I_0}$. Therefore, $f(\hat{x}) = f_{I_0}(\hat{x}) > f_{I_0}(x_{t_0}) = f(x_{t_0})$, i.e., $\hat{x}$ is not locally minimal.

On the other hand, for any $x \in \mathcal{B}(\hat{x})$, one can find an I_x satisfying $x \in \mathcal{B}(\hat{x}) \cap \Phi_{I_x}$ and $f(x) = f_{I_x}(x)$. It is noted that $\hat{x} \in \bigcap \Phi_I$ and $f(\hat{x}) = f_{I_x}(\hat{x})$. According to (17), $f(\hat{x}) \leq f(x)$, i.e., $\hat{x}$ is locally minimal. Hence, (17) is sufficient for $\hat{x}$ being locally minimal.

From the above discussion, we know that (17) is the sufficient and necessary condition for $\hat{x}$ being locally minimal. Consider a given I satisfying $I(m) \in \mathcal{A}_m(\hat{x}), \forall m$. (17) means

$$\sum_{m=1}^{M} \beta_m \big(a_{mI(m)}^T \hat{x} + b_{mI(m)}\big) \leq \sum_{m=1}^{M} \beta_m \big(a_{mI(m)}^T x + b_{mI(m)}\big) \tag{18}$$

is valid for all $x \in \Phi_I$, where Φ_I is determined by

$$a_{mI(m)}^T x + b_{mI(m)} \geq a_{mi}^T x + b_{mi}, \quad \forall i \in \mathcal{A}_m(\hat{x}), \forall m. \tag{19}$$

Since the following relationship holds:

$$a_{mI(m)}^T \hat{x} + b_{mI(m)} = a_{mi}^T \hat{x} + b_{mi}, \quad \forall i \in \mathcal{A}_m(\hat{x}), \forall m,$$

(19) is equal to

$$\left(a_{mI(m)}^T - a_{mi}^T\right)(x - \hat{x}) \geq 0, \quad \forall i \in \mathcal{A}_m(\hat{x}), \forall m.$$

Meanwhile, (18) can be written as

$$\sum_{m=1}^{M} \beta_m a_{mI(m)}^T (x - \hat{x}) \geq 0.$$

Therefore, (17) holds if and only if for any I there is no feasible point for the following inequalities:

$$\begin{aligned} &\sum_{m=1}^{M} \beta_m a_{mI(m)}^T (x - \hat{x}) < 0, \\ &(a_{mI(m)}^T - a_{mi}^T)(x - \hat{x}) \geq 0, \quad \forall i \in \mathcal{A}_m(\hat{x}), \forall m. \end{aligned} \tag{20}$$

According to Farkas' lemma, (20) has no feasible point if and only if there are nonnegative scalars $\theta_{mi} \geq 0$ ensuring

$$\sum_{m=1}^{M} \beta_m a_{mI(m)} = \sum_{m=1}^{M} \sum_{i \in \mathcal{A}_m(\hat{x})} \theta_{mi}(a_{mI(m)} - a_{mi}). \tag{21}$$

From the above discussion, we know that $\hat{x}$ is a local minimum if and only if (21) is valid for arbitrary I. The sufficient and necessary condition for $\hat{x}$ being locally optimal is proved. □

An important advantage of CPWL programming is that it can be transformed into a series of linear problems in subregions, called *sub-LP* in this paper. If $\hat{x}$ is not locally minimal, then the objective value can be strictly decreased through sub-LP. According to Theorem 4.1, when $\hat{x}$ is not a local minimum of (14), there is at least an I_0 for which (15) does not hold for any $\theta_{mi} \geq 0$. Then, we can construct the following sub-LP:

$$\begin{aligned} &\min \sum_{m=1}^{M} \beta_m l_{I_0(m)}^m(x) \\ &\text{s.t.} \quad l_{I_0(m)}^m(x) \geq l_i^m(x), \quad \forall i = 1, 2, \ldots, n+1, \forall m = 1, 2, \ldots, M. \end{aligned} \tag{22}$$

From the proof of Theorem 4.1, one can easily verify that the objective value can be strictly decreased from $f_0(\hat{x})$ by solving the above sub-LP. Following this idea, we give an algorithm for (14). This algorithm uses sub-LP to decrease the objective function from an initial point and hence is named Descent Algorithm using LP Technique (DALPT), as shown in Algorithm 1.

The number of the distinct linear subregions is finite, DALPT hence can converge to a local optimum naturally. Obviously, in DALPT, to verify the local optimality

Algorithm 1: Descent Algorithm using LP Technique (DALPT)

Initialize
- Randomly select an initial solution $\hat{x}$

while do
- Identify $\mathcal{A}_m(\hat{x})$ and let $\mathcal{I}$ be the set of all possible I, i.e., $\mathcal{I} = \{I : I \in Z^M, I(m) \in \mathcal{A}_m(\hat{x})\}$
- **while** $\mathcal{I} \neq \emptyset$ **do**
  - Randomly select I_0 from $\mathcal{I}$
  - **if** *(15) can be satisfied by some nonnegative scales* **then**
    - let $\mathcal{I} = \mathcal{I} \setminus \{I_0\}$
  - **else**
    - break
  - **end**
- **end**
- **if** $\mathcal{I} \neq \emptyset$ **then**
  - Solve sub-LP (22) for I_0 and use its optimum to update $\hat{x}$
- **else**
  - break
- **end**

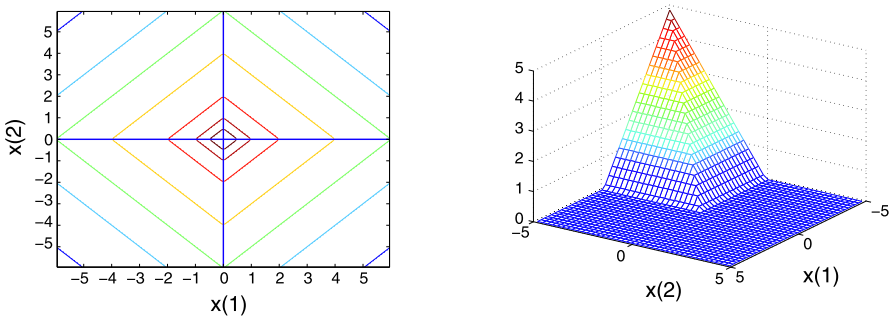
end

Result: $\hat{x}$ is the obtained locally optimal solution

of $\hat{x}$, at most $\prod_{m=1}^{M} \|\mathcal{A}_m(\hat{x})\|$ linear equations should be solved, where $\|\mathcal{A}_m(\hat{x})\|$ denotes the number of elements in $\mathcal{A}_m(\hat{x})$. According to the definition of the active index set $\mathcal{A}_m(\hat{x})$, one can see that $\hat{x}$ must satisfy $\sum_{m=1}^{M} \|\mathcal{A}_m(\hat{x})\| - 1$ linear equations. Therefore, when these linear equations are linearly independent, we have $\sum_{m=1}^{M} \|\mathcal{A}_m(\hat{x})\| \leq n + M$ and $\|\mathcal{A}_m(\hat{x})\| \geq 1, \forall m$. Although $\prod_{m=1}^{M} \|\mathcal{A}_m(\hat{x})\|$ may be very large, it is acceptable in regular cases. To show the effectiveness of DALPT, we report the computing time in numerical experiments. The basic thought of DALPT, i.e., solving LP in one subregion and choose another to test, is the same as the idea of the algorithm in [37] and the generalized simplex algorithm in [4–6]. The algorithm in [37] deals with a minimization problem of adaptive hinging hyperplanes (AHH), which is a nesting CPWL representation proposed in [17], by solving sub-LPs repeatedly in allowed time. In that algorithm, the local optimality of the result cannot be guaranteed, since the local optimality condition for the AHH has not been derived. To make algorithm more effective, local optimality condition is considered in both DALPT and the generalized simplex algorithm, the latter of which is established for minimizing a separable CPWL function in vertex representation. As mentioned in Sect. 2.3, transforming a nonseparable compact expression to vertex representation is not easy, we cannot simply extend the generalized simplex algorithm to solving (14).

According to Theorem 4.1, if the objective function can be represented by the HH, i.e.,

$$f(x) = a_0^T x + b_0 + \sum_{m=1}^{M} \beta_m \max\{0, a_m^T x + b_m\}, \tag{23}$$



(a) Contour map of (24) and its ridges

(b) Plots of function (25)

Fig. 1 Decomposable and non-decomposable functions

then the optimality condition is given by the following corollary. Using this corollary, the local optimality of (23) can be verified by solving one linear equation.

Corollary 4.1 *For function* (23), *if* $\{a_m\}_{m:a_m^T\hat{x}+b_m=0}$ *are linearly independent, then the sufficient and necessary condition for* $\hat{x}$ *being locally minimal is*: *for any* m *satisfying* $a_m^T\hat{x}+b_m=0$, (i) $\beta_m=1$; (ii) *there are scalars* η_m *such that*

$$-1\le\eta_m\le 0,$$
$$a_0+\sum_{m:a_m^T\hat{x}+b_m>0}\beta_m a_m=\sum_{m:a_m^T\hat{x}+b_m=0}\eta_m a_m.$$

Conn and Mongeau have given a local optimality condition for CPWL programming (see Theorem 2 in [3]) and that condition is equivalent to Corollary 4.1. Inspired by this equivalence, one can see that Conn's condition is discussing the CPWL objective function which can be represented by the HH model in the neighborhood of the concerned point. As HH cannot represent all CPWL functions, there are lots of functions that Conn's condition cannot handle. To show this fact, we compare two simple functions both taken from [3]. The two functions are shown below and illustrated by Fig. 1:

$$f_1\big(x(1),x(2)\big)=\begin{cases}-x(1)-x(2), & \text{if } x(1)\ge 0 \text{ and } x(2)\ge 0,\\ x(1)-x(2), & \text{if } x(1)<0 \text{ and } x(2)\ge 0,\\ -x(1)+x(2), & \text{if } x(1)\ge 0 \text{ and } x(2)<0,\\ x(1)+x(2), & \text{otherwise.}\end{cases}\tag{24}$$

$$f_2\big(x(1),x(2)\big)=\begin{cases}-x(2), & \text{if } x(2)<0 \text{ and } x(2)>x(1),\\ -x(1), & \text{if } x(1)<0 \text{ and } x(2)\le x(1),\\ 0, & \text{otherwise.}\end{cases}\tag{25}$$

Conn's condition is proved via decomposing a CPWL function by *ridge*. In [3], a ridge of piecewise linear function f is a specified hyperplane "containing a relative neighborhood where the derivative of f is not defined." One can verify that

the ridges of f_1 are the two lines $x(1) = 0$ and $x(2) = 0$, and the ridges of f_2 are $x(1) = 0, x(2) = 0$ and $x(1) = x(2)$. Using the optimality condition in [3], one can analyze the local optimality for a CPWL function with known ridges by decomposing the function into a smooth part and a non-smooth part. However, f_2 is non-decomposable at $[0, 0]^T$ and the reason is rigorously explained in [3]. Compactly represent the two functions, i.e.,

$$f_1\big(x(1), x(2)\big) = x(1) + x(2) - 2\max\big\{0, x(1)\big\} - 2\max\big\{0, x(2)\big\},$$
$$f_2\big(x(1), x(2)\big) = \max\big\{0, x(1), x(2)\big\} - \max\big\{x(1), x(2)\big\}.$$

One can see the main difference between f_1 and f_2 is that f_1 can be represented by the HH model but f_2 cannot be. We call the part of ridges that actually has relative neighborhood where the derivative is not defined as the *active part of ridge*. The active parts of ridges form the boundaries of the linear subregions. Consider the function f_2: the active parts of its ridges are

$$\big\{x : x(1) = 0, x(2) \le 0\big\}, \qquad \big\{x : x(2) = 0, x(1) \le 0\big\},$$
$$\big\{x : x(1) - x(2) = 0, x(1) \ge 0, x(2) \ge 0\big\},$$

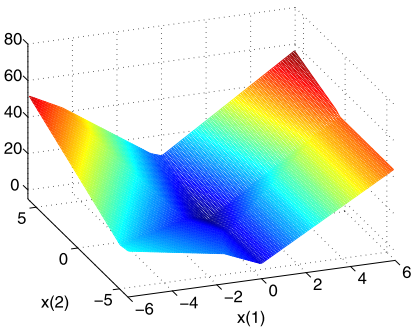
which are rays. This is the reason why f_2 is non-decomposable. In fact, when a CPWL function can be represented by the HH, it is decomposable at any point in its domain. From another point of view, if there are some boundaries of the linear subregions which are not lines but segments or rays, then the CPWL function is non-decomposable at the terminals of such segments or rays. In this case, Conn's condition does not work but Theorem 4.1 is applicable to any CPWL programming.

5 Numerical Examples

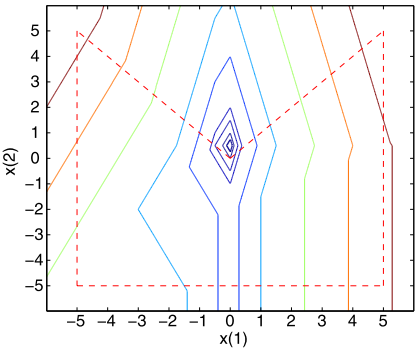
Some properties of CPWL programming are discussed above and an algorithm is proposed. Two examples are considered below. The first example is a problem with nonseparable CPWL objective function and CPWL constraints. This example is used to illustrate some theoretical results. In the second example, we use DALPT to minimize a CPWL function under linear constraints. Then, the results are compared with the global optimum calculated by CPLEX.

Example 5.1 Consider the following CPWL programming:

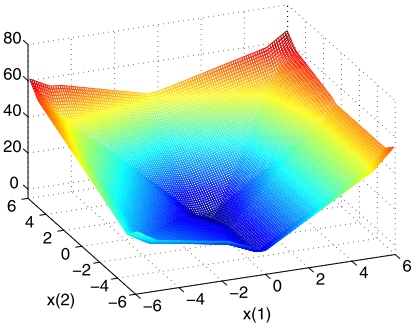
$$\begin{aligned}\min\ & 2\max\big\{x(1) + x(2), x(1) - x(2) + 1, -x(1) + x(2) - 1\big\} \\ & - \max\big\{x(1) - 1, 2x(2) + 3, -2x(1) + x(2)\big\} \\ & + 2\max\big\{-3x(1) + x(2), 3x(1) + x(2)\big\} \\ \text{s.t.}\quad & -5 \le x(1) \le 5, \quad -5 \le x(2) \le 5, \quad x(2) \le \max\big\{x(1), -x(1)\big\}.\end{aligned}$$



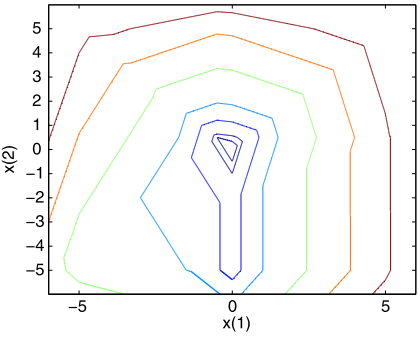
(a) Plots of the objective function



(b) Contour map and the feasible domain



(c) Plots of $F_5(x)$



(d) Contour map of $F_5(x)$

Fig. 2 Features of Example 5.1

The plots of the objective function and the feasible domain are shown in Fig. 2(a) and (b). It is noted that $\max\{0, x(2) - \max\{x(1), -x(1)\}\} = \max\{\max\{x(1), -x(1)\}, x(2)\} - \max\{x(1), -x(1)\}$. We can transform Example 5.1 into a unconstrained problem using penalty parameter λ as follows:

$$\min\ F_\lambda(x) = \sum_{m=1}^{9} \beta_m B_m(x),$$

where $\beta_2 = -1$, $\beta_9 = -1$, $\beta_m = 1, m = 1, 3, 4, \ldots, 8$, and

$$\begin{aligned}
B_1(x) &= 2\max\{x(1) + x(2), x(1) - x(2) + 1, -x(1) + x(2) - 1\},\\
B_2(x) &= \max\{x(1) - 1, 2x(2) + 3, -2x(1) + x(2)\},\\
B_3(x) &= 2\max\{-3x(1) + x(2), 3x(1) + x(2)\},\\
B_4(x) &= \lambda\max\{0, -x(1) - 5\},\\
B_5(x) &= \lambda\max\{0, x(1) - 5\},
\end{aligned}$$

$$B_6(x) = \lambda \max\{0, -x(2) - 5\},$$
$$B_7(x) = \lambda \max\{0, x(2) - 5\},$$
$$B_8(x) = \lambda \max\{x(1), -x(1), x(2)\},$$
$$B_9(x) = \lambda \max\{x(1), -x(1)\}.$$

To show the local optimality condition, we let $\hat{x} = [0, 0]^T$, for which $\mathcal{A}_1(\hat{x}) = \{2\}$, $\mathcal{A}_2(\hat{x}) = \{2\}$, $\mathcal{A}_3(\hat{x}) = \{1, 2\}$, $\mathcal{A}_4(\hat{x}) = \mathcal{A}_5(\hat{x}) = \mathcal{A}_6(\hat{x}) = \mathcal{A}_7(\hat{x}) = \{1\}$, $\mathcal{A}_8(\hat{x}) = \{1, 2, 3\}$, $\mathcal{A}_9(\hat{x}) = \{1, 2\}$. Considering $I = [2, 2, 1, 1, 1, 1, 1, 3, 1]^T$, Eq. (15) becomes

$$2\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 2 \end{bmatrix} + 2\begin{bmatrix} -3 \\ 1 \end{bmatrix} + \lambda\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \lambda\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$= 2\theta_1\begin{bmatrix} -6 \\ 0 \end{bmatrix} + \theta_2\begin{bmatrix} -\lambda \\ \lambda \end{bmatrix} + \theta_3\begin{bmatrix} \lambda \\ \lambda \end{bmatrix} + \theta_4\begin{bmatrix} 2\lambda \\ 0 \end{bmatrix}.$$

Obviously, the above equation can be satisfied by nonnegative θ_m when $\lambda \geq 2$. Following similar analysis, we know that Theorem 4.1 can be met, i.e., $[0, 0]^T$ is locally optimal for $F_\lambda(x)$, when $\lambda \geq 2$. For the special case of $\lambda = 5$, the penalized function is illustrated in Fig. 2(c) and 2(d). According to Theorem 3.1, one can see that $\hat{x} = [0, 0]^T$ is also a local optimum of Example 5.1.

Example 5.2 Consider the HH minimization problem below:

$$\min\ f(x) = l_0(x) + \sum_{m=1}^{M} \beta_m \max\{0, l_m(x)\} \quad \text{s.t.} \quad 0 \leq x(i) \leq 1, \quad \forall i = 1, 2, \ldots, n.$$

The objective function takes HH form and $l_m(x) = a_m^T x + b_m$. We consider the cases $[n, M] = [3, 30], [3, 100], [5, 30], [5, 100]$, separately. Parameters a_m are selected from $[-1, 1]^n$ following the uniform distribution. Then, we randomly generate b_m and guarantee each hyperplane $l_m(x) = 0, m = 1, 2, \ldots, M$, to intersect with the hypercube $[0, 1]^n$ (otherwise, the basis function $\max\{0, l_m(x)\}$ reduces to an affine function in the feasible domain). β_m takes value from $\{-1, 1\}$ with equal possibility.

DALPT is established for unconstrained problem. Meanwhile, it can be directly used in linearly constrained problem. We apply DALPT to handle Example 5.2. Since the result of DALPT may differ from different initial points, we run DALPT 10 times from random initial points and choose the best one as the optimized result. Meanwhile, we can transform this problem into a MILP like (7) and solve it by CPLEX, whose result is globally optimal and can be used to evaluate the performance of DALPT. For a pair of M and n, 10 instances are generated and Table 1 shows the experimental results, including the average and standard derivation of computing time of DALPT, the average computing time of CPLEX and the percentage of global optimum are achieved within repeating DALPT 10 times.

Table 1 Performance of DALPT in Example 5.2

n	M	Avg. time (s)	Std. of time (s)	Percentage of gaining optimum	Avg. time of CPLEX (s)
3	30	0.0797	0.0514	90%	0.005
3	100	1.7925	0.9855	100%	8.494
5	30	0.1306	0.0690	100%	0.107
5	100	1.8650	0.7904	100%	406.5

DALPT is run by Matlab 2007a in Pentium 4.3 GHz, 1 GB RAM and CPLEX is 11.1 solver run in Xeon 2.33 GHz, 11.9 GB RAM

6 Conclusions

Continuous piecewise linear functions represented by compact models have been widely applied in circuit analysis and nonlinear system identification. Therefore, continuous piecewise linear programming in a compact model is an important problem worthy of study. In this paper, we prove the existence of exact penalty and give a local optimality condition for nonseparable CPWL programming. Based on the gained optimality condition, a decreasing algorithm is established. Then, we illustrate the theoretical results and evaluate the proposed algorithm by a numerical study. This paper gives some preliminary discussions on CPWL programming in compact model. Further study on efficient algorithm will make CPWL programming in compact representation a promising tool in nonlinear analysis and optimization.

Acknowledgements This research was supported jointly by the National Natural Science Foundation of China (61074118, 60974008, 61104218, 041306020) and the Research Fund of Doctoral Program of Higher Education (200800030029).

The authors appreciate the reviewers for their insightful comments and helpful suggestions.

References

1. Beale, E.M.L., Coen, P.J., Flowerdew, A.D.J.: Separable programming applied to an ore purchasing problem. J. R. Stat. Soc., Ser. C, Appl. Stat. **14**, 89–101 (1965)
2. Beale, E.M.L.: Numerical methods: the theory of separable programming. In: Abadie, J., Vajda, S. (eds.) Nonlinear Programming, pp. 174–177. North-Holland, Amsterdam (1967)
3. Conn, A.R., Mongeau, M.: Discontinuous piecewise linear optimization. Math. Program. **80**, 315–380 (1998)
4. Fourer, R.: A simplex algorithm for piecewise-linear programming I: Derivation and proof. Math. Program. **33**, 204–233 (1985)
5. Fourer, R.: A simplex algorithm for piecewise-linear programming II: Finiteness, feasibility and degeneracy. Math. Program. **41**, 281–315 (1988)
6. Fourer, R.: A simplex algorithm for piecewise-linear programming III: Computational analysis and applications. Math. Program. **53**, 213–235 (1992)
7. Padberg, M.W.: Approximation separable nonlinear functions via mixed zero-one programs. Oper. Res. Lett. **27**, 1–5 (2000)
8. Croxton, K.L., Gendron, B., Magnanti, T.L.: Models and methods for merge-in-transit operations. Transp. Sci. **37**, 1–22 (2003)
9. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: Models for representing piecewise linear cost functions. Oper. Res. Lett. **32**, 44–48 (2004)

10. Vielma, J.P., Keha, A.B., Nemhauser, G.L.: Nonconvex, lower semicontinuous piecewise linear optimization. Discrete Optim. **5**, 467–488 (2008)
11. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Oper. Res. **58**, 303–315 (2010)
12. Chua, L.O., Kang, S.M.: Section-wise piecewise-linear functions: canonical representation, properties, and applications. Proc. IEEE **65**, 915–929 (1977)
13. Breiman, L.: Hinging hyperplanes for regression, classification and function approximation. IEEE Trans. Inf. Theory **39**, 999–1013 (1993)
14. Julián, P., Desages, A., Agamennoni, O.: High-level canonical piecewise linear representation using a simplicial partition. IEEE Trans. Circuits, I **46**, 463–480 (1999)
15. Wang, S., Sun, X.: Generalization of hinging hyperplanes. IEEE Trans. Inf. Theory **12**, 4425–4431 (2005)
16. Wang, S., Huang, X., Junaid, K.M.: Configuration of continuous piecewise-linear neural networks. IEEE Trans. Neural Netw. **19**, 1431–1445 (2008)
17. Xu, J., Huang, X., Wang, S.: Adaptive hinging hyperplanes and its applications in dynamic system identification. Automatica **45**, 2325–2332 (2009)
18. Tarela, J.M., Martinez, M.V.: Region configurations for realizability of lattice piecewise-linear models. Math. Comput. Model. **30**, 17–27 (1999)
19. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Manag. Sci. **49**, 1268–1273 (2003)
20. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Oper. Res. **54**, 847–858 (2006)
21. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In: Proceedings of the Fifth International Conference on Operational Research, pp. 447–454 (1970)
22. Misener, R., Floudas, C.A.: Piecewise-linear approximations of multidimensional functions. J. Optim. Theory Appl. **145**, 120–147 (2010)
23. Toriello, A., Vielma, J.P.: Fitting piecewise linear continuous functions. Report of Georigia Institue of Technology (2010)
24. Ernst, S.: Hinging hyperplane trees for approximation and identification. In: Proceedings of the 37th IEEE Conference on Decision and Control, pp. 1266–1271 (1998)
25. Pucar, P., Sjöberg, J.: On the hinge-finding algorithm for hingeing hyperplanes. IEEE Trans. Inf. Theory **44**, 1310–1319 (1998)
26. Karniel, A., Meir, R., Inbar, G.F.: Polyhedral mixture of linear experts for many-to-one mapping inversion and multiple controllers. Neurocomputing **37**, 31–49 (2001)
27. Özkan, L., Kothare, M.V., Georgakis, C.: Control of a solution copolymerization reactor using multimodel predictive control. Chem. Eng. Sci. **58**, 1207–1221 (2003)
28. Ramírez, D.R., Camacho, E.F., Arahal, M.R.: Implementation of min-max MPC using hinging hyperplanes: application to a heat exchanger. Control Eng. Pract. **12**, 1197–1205 (2004)
29. Zanma, T., Fuke, K., Ma, S.C., Ishida, M.: Simultaneous identification of piecewise affine systems and number of subsystems using mixed logical dynamical systems theory. Electron. Commun. Jpn. **91**, 1–10 (2008)
30. Julián, P., Desages, A., D'Amico, B.: Orthonormal high-level canonical PWL functions with applications to model reduction. IEEE Trans. Circuits, I **47**, 702–712 (2000)
31. Cervantes, A.L., Agamennoni, O.E., Figueroa, J.L.: A nonlinear model predictive control system based on wiener piecewise linear models. J. Process Control **13**, 655–666 (2003)
32. Castro, L.R., Figueroa, J.L., Agamennoni, O.E.: An NIIR structure using HL-CPWL functions. Lat. Am. Appl. Res. **35**, 161–166 (2005)
33. Shafiee, G., Arefi, M.M., Jahed-Motlagh, M.R., Jalali, A.A.: Nonlinear predictive control of a polymerization reactor based on piecewise linear wiener model. Chem. Eng. J. **143**, 282–292 (2008)
34. Wen, C., Wang, S., Jin, X., Ma, X.: Identification of dynamic systems using piecewise-affine basis function models. Automatica **43**, 1824–1831 (2007)
35. Chikkula, Y., Lee, J.H., Ogunnaike, B.A.: Dynamically scheduled MPC of nonlinear processes using hinging hyperplane models. AIChE J. **44**, 2658–2674 (1998)
36. Xu, J., Huang, X., Wang, S.: Nonlinear model predictive control using adaptive hinging hyperplanes model. In: Proceedings of the 48th IEEE Conference on Decision and Control, pp. 2598–2603 (2009)
37. Huang, X., Xu, J., Wang, S.: Operation optimization for centrifugal chiller plants using continuous piecewise linear programming. In: Proceedings of 2010 IEEE International Conference on Systems Man and Cybernetics, pp. 1121–1126 (2010)