2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Optimization based on adaptive hinging hyperplanes and genetic algorithm

Jun Xu, Xiangming Xi and Shuning Wang

***Abstract*— This paper describes an optimization strategy based on the model of adaptive hinging hyperplanes (AHH) and genetic algorithm (GA). The sample points of physical model are approximated by the AHH model, and the resulting model is minimized using a modified GA. In the modified GA, each chromosome corresponds to a local optimum. A criterion based on γ-valid cut is used to judge whether the global optimum is reached. Simulation results show that if the parameters are carefully chosen, the global optimum of AHH minimization is close to the optimum of the original function.**

I. Introduction

In industry optimization, usually high-fidelity models are hard to derive, or the evaluation of the model requires long running time and intensive computation. Hence, direct optimization of these models is not easy. Thus, the optimization strategy based on the surrogate model describing the high-fidelity model is attractive, which is applied in the area of aerospace design [1], [2], [3], distillation process [4], [5], air conditioning system [6], [7], and so on.

There are many kinds of surrogate models describing the physical system, such as neural network [6], [8], polynomial regression model [9], Kriging model [10], [11] and continuous piecewise affine (CPWA) model [7], [5]. To optimize these surrogate models, two kinds of optimization methods are employed. The first one is the conventional gradient-based optimization, including steepest descent and conjugate-gradient algorithms, the second one is derivative-free method, such as simulated annealing (SA) and evolutionary algorithm (GA). If local optimum is satisfactory, then fast gradient-based method can be used, however, if one wants to derive the global optimum, in general, the derivative-free method should be applied.

In this paper, we consider a CPWA model called adaptive hinging hyperplanes (AHH), which is proposed as a method for function approximation and dynamic system identification [12]. It is based on the well-known multivariable regression splines (MARS) [13], and inherits the way of constructing the model, i.e., recursive partitioning of domain. MARS is piecewise polynomial, i.e., the whole domain is partitioned into convex polyhedron, and in each one, the relationship is polynomial. In contrast, the AHH model admits a linear relationship in each subregion. The AHH model can be seen as a special form of previously proposed generalized hinging hyperplanes (GHH) model [14], which has the ability of representing any CPWA function in any dimension. AHH performs well in approximating continuous functions, especially for CPWA functions.

Now that the high-fidelity model is modeled as an AHH function, if the constraints are linear, the resulting optimization problem is a continuous piecewise linear programming (CPWLP). For this kind of problem, analog to the simplex algorithm in linear programming, a modified simplex algorithm is developed for CPWLP with a separable objective function [15], [16], [17]. An alternative solution is to derive an equivalent mixed-integer linear programming formulation of the CPWLP, such as [18], [19] and [20], of which the former two are for separable CPWA objective functions, and the last one is for a more general class, non-separable CPWA objective functions.

In this paper, we adapt the GA in this special situation to give a global optimal solution. Using GA, abundant local optima are provided and thanks to the CPWA property of the optimization problem, the criterion based on γ-valid cut can be used to judge whether the global optimum is reached. As is known to all, GA is a stochastic algorithm mimicking the natural evolution among generations, after the evaluation of GA, superior offsprings (solutions) are left, while inferior ones are deleted. [21] proves that if the parameters are properly selected, the GA can converges to a quite good solution.

The paper is organized as follows. Section 2 introduces the model of AHH. The minimization of AHH is formulated in Section 3 as a concave programming. Section 4 proposes a descent algorithm to search vertices which is locally optimal for the concave programming. To fasten the process of finding global optimum, Section 5 proposes a modified GA. Simulation study is done in Section 6, and the paper ends in Section 7 with conclusion.

II. AHH Modeling

The AHH model $f : \mathbb{R}^n \to \mathbb{R}$ can be seen as a linear combination of different basis functions,

$$f = a_0 + a_m \sum_{m=1}^{M} B_m, \tag{1}$$

where $B_m(\cdot)$ and a_m, $0 \le m \le M$ are the basis function and correponding linear coefficient. Apparently, $B_0 = 1$. The

Jun Xu is with the Research Institute of Automation, China University of Petroleum, Beijing, China, and also with Delft Center for System and Control, Delft University and Technology, The Netherlands, Xiangming Xi and Shuning Wang are with the Department of Automation, Tsinghua University, Beijing, China (email: {yun-xu05, xxm10}@mails.tsinghua.edu.cn, swang@mail.tsinghua.edu.cn).

This work was supported by National Natural Science Foundation of China(21006127; 61104218), the National Basic Research Program of China (973 program: 2012CB720500), and the Science Foundation of China University of Petroleum(YJRC-2013-12)

m-th basis function is defined as,

$$B_m(x) = \max_{k\in\{1,\dots,K_m\}}\{\min\{0, s_{km}(x_{v_{km}} - \beta_{km})\}\}, \quad (2)$$

in which $s_{km} = \pm 1$, K_m is the number of "min" terms in the "max" bracket, $x_{v_{km}}$ is the v_{km}-th variable and β_{km} is a splitting point in that dimension. The corresponding subregion where B_m is nonzero is,

$$s_{km}(x_{v_{km}} - \beta_{km}) < 0, \ \forall k \in \{1, \dots, K_m\}. \quad (3)$$

Assume B_m is nonzero in some cube subregion (3), then in the cube, the basis B_m further introduces at most K_m subregions where B_m is affine. Therefore, there are at most $\prod_{m=1}^{M} K_m$ subregions in which f is affine, we call these subregions linear subregions. In each basis, the variables $x_{v_{km}}$ must be different, thus there are no more than n "min" items, therefore, the largest number of linear subregions is n^M, which may be large when n and M is large.

In order to approximate the sample data $(x(t), y(t))_{t=1}^{N}$, the parameters in f should be chosen carefully to ensure a small approximation error. In fact, the parameters of the basis functions as well as the linear coefficients are obtained via the recursive partitioning of the domain, i.e., f is obtained similar to the growth of a tree. In the forward procedure, first, the domain relates to the constant basis function $B_0 = 1$ is fixed, besides, the number of basis functions needed, denoted by M, as well as the candidate splitting points N_{split} in each dimension, is preset. Then in each step, a parent basis (subregion) is chosen from all existed basis functions, say B_i, and a split is performed at some candidate point $x_j = \beta_j$, which together results in the new born basis functions $\max\{B_i(x), \min\{0, x_j - \beta_j\}\}$ and $\max\{B_i(x), \min\{0, -(x_j - \beta_j)\}\}$, i.e., new subregions (branches) are generated. By searching through all available parent basis functions, splitting variables and candidate points, a greedy strategy, i.e., doing so will cause the largest decrease in the approximation error, is employed to choose B_i and $x_j, \ \beta_j$. It is noted that here the approximation error is measured by the sum of square, hence the least square technique is used to fix the linear coefficients. The new born basis are added to the pool of the available parent bases, and the next step repeats until the number of basis functions is M. It is noted that in this forward stepwise growth, the parent basis (subregion) is not removed after splitting, which makes AHH adjustable to model both additive functions (always having $B_0 = 1$ as the parent basis) and relationships involving interactions among variables.

Once the AHH model with the needed number of basis functions is generated, a backward procedure is performed to remove redundant splits in order to avoid over-fitting, i.e., delete subregions (branches). In the backward stepwise deletion, one basis function is deleted in each iteration, again adopting the greedy strategy, then a sequence of AHH models are obtained, each one having one less basis functions than the previous one. The model in the sequence which fits the best is chosen as the final model.

The detailed approximation algorithm of AHH can be seen in [12]. It is proved in [12] that for a continuous function in a compact set, AHH can approximate it with arbitrary precision.

III. Formulation of the AHH minimization

Once the unknown relationship is modeled as an AHH function, we optimize the AHH function with polyhedral constraints. The problem can be expressed as follows,

$$\begin{aligned} \min \quad & f(x) = a_0 \\ & + \sum_{m=1}^{M} a_m \max_{1\le k\le K_m}\{\min\{0, s_{km}(x_{v_{km}} - \beta_{km})\}\} \\ s.t. \quad & Ax \le b. \end{aligned} \quad (4)$$

Obviously, the basis $B_m(x)$ in the cost function can be written as,

$$\begin{aligned} B_m(x) &= \max_{m\in\{1,\dots,K_m\}}\{\min\{0, s_{km}(x_{v_{km}} - \beta_{km})\}\} \\ &= \max_{m\in\{1,\dots,K_m\}}\{s_{km}(x_{v_{km}} - \beta_{km})\} \\ &\quad - \max_{m\in\{1,\dots,K_m\}}\{0, s_{km}(x_{v_{km}} - \beta_{km})\}. \end{aligned}$$

Hence, the problem (4) can be rewritten as,

$$\begin{aligned} \min \quad & \sum_{m=1}^{M} a_m \left(\max_{k\in\{1,\dots,K_m\}}\{s_{km}(x_{v_{km}} - \beta_{km})\} \right. \\ & \qquad \left. - \max_{k\in\{1,\dots,K_m\}}\{0, s_{km}(x_{v_{km}} - \beta_{km})\} \right) \\ s.t. \quad & Ax \le b. \end{aligned} \quad (5)$$

Introduce the auxiliary variable $y = [y_1, \dots, y_m, \dots, y_M]^T$ in which

$$y_m = \begin{cases} \max_{k\in\{1,\dots,K_m\}}\{s_{km}(x_{v_{km}} - \beta_{km})\} & \text{if } a_m > 0 \\ \max_{k\in\{1,\dots,K_m\}}\{0, s_{km}(x_{v_{km}} - \beta_{km})\} & \text{if } a_m < 0, \end{cases}$$

then we can reconstruct the problem (5) as

$$\begin{aligned} \min \quad & \sum_{m:a_m>0} |a_m| \left(y_m - \max\{0, y_m\}\right) \\ & + \sum_{m:a_m<0} |a_m| \left(y_m - \max_{k\in\{1,\dots,K_m\}}\{s_{km}(x_{v_{km}} - \beta_{km})\} \right) \\ s.t. \quad & s_{km}(x_{v_{km}} - \beta_{km}) \le y_m, \quad \forall m \\ & 0 \le y_m, \quad \forall m : a_m < 0 \\ & Ax \le b. \end{aligned} \quad (6)$$

Suppose $z = [x, y_1, \dots, y_m]^T$, denote $n_z = n + M$, then z is an n_z-dimensional optimized variable, define $\mathcal{M}^+ = \{m|a_m > 0\}$ and $\mathcal{M}^- = \{m|a_m < 0\}$, the above problem can be cast as a concave programming of which the objective is CPWA and the constraints are linear inequalities (equalities),

$$\begin{aligned} \min \quad & J = \theta_0^T z - \sum_{m\in\mathcal{M}^+} \max\{0, \theta_m^T z\} \\ & - \sum_{m\in\mathcal{M}^-} \max_{k\in\{1,\dots,K_m\}}\{\theta_{km}^T z + \varphi_{km}\} \\ s.t. \quad & \bar{A}z \le \bar{b}, \end{aligned} \quad (7)$$

where the parameters θ_m, θ_{km}, φ_{km} and $\bar{A}$, $\bar{b}$ can be determined from the problem (6).

For the above concave programming, denote the feasible region as $\mathcal{D}$, it is demonstrated in [22] that the global minimum is always obtained in some vertex of $\mathcal{D}$, if $\mathcal{D}$ has vertices. Therefore, traversing of all vertices will give the minimum, which becomes intractable when the number of vertices is large. Actually, we can apply a cutting plane method for possible early termination of the traversal, of which the cut adopted is the γ-valid cut.

Definition 1: [23] Given a vertex z_0 which is a strict local minimum of J over the feasible polyhedron, assume $\gamma = J(z_0)$, the hyperplane

$$\pi^T(z - z_0) \geq 1 \tag{8}$$

is called a γ-valid cut for the problem (7) if the following holds,

$$\{z \in \mathcal{D} | J(z) < \gamma\} \subset \{z \in \mathcal{D} | \pi^T(z - z_0) \geq 1\}. \tag{9}$$

Suppose that we have obtained N vertices $z_1, \ldots, z_N$, then the following lemma gives a sufficient condition for global optimality [22].

Lemma 1: The global minimum of problem (7) can be chosen as the one giving the least cost value among $z_1, \ldots, z_N$, if there exists,

$$\bigcap_{i=1}^{N} \{z \in \mathcal{D} | \pi_i^T(z - z_i) \geq 1\} = \emptyset, \tag{10}$$

in which the hyperplane $\pi_i^T(z - z_i) \geq 1$ is the γ-valid cut stemmed from z_i.

IV. Vertex searching via linear programming

Then what we interested in is finding the vertex globally optimal. Traditional ways of finding vertex is listing all combinations of constraints. In the following, we find a vertex which is a local minimum of the optimization problem (7), and in the next section, the global optimum is chosen among those local minuma. Given a vertex $\hat{z}$, we can fix the corresponding linear subregions $\hat{z}$ lying in, deonted by $\mathcal{D}_{\hat{z}}$. Denote the affine functions in $\mathcal{D}_{\hat{z}}$ as $l_{\hat{z}}$, there may be multiple $l_{\hat{z}}(z)$ for a point $\hat{z}$, and share a common function value at $\hat{z}$. Denote the affine function pool of $l_{\hat{z}}(z)$ as $\mathcal{L}(\hat{z})$, then the following lemma gives the conditions for the local optimality of $\hat{z}$.

Lemma 2: A vertex $\hat{z}$ is locally optimal if and only if $\forall\, l_{\hat{z}} \in \mathcal{L}(\hat{z})$, $\hat{z}$ minimizes the following linear programming (LP),

$$\begin{array}{ll} \min & l_{\hat{z}}(z) \\ s.t. & z \in \mathcal{D}. \end{array} \tag{11}$$

Proof: It is clear that the optimal solution of (11) is a vertex of $\mathcal{D}$.

The following proof employs the necessary and sufficient conditions proposed in [24] for local optimality of CPWLP, i.e., $\hat{z}$ is the local optimum if and only if it minimizes the following LP,

$$\begin{array}{ll} \min & l_{\hat{z}}(z) \\ s.t. & z \in \mathcal{D} \\ & l_{\hat{z}}(z) = J(z). \end{array} \tag{12}$$

Compared with LP (11), the above problem has an additional constraint that the optimization is confined within the linear subregion $\{z | l_{\hat{z}}(z) = J(z)\}$. In this proof, we state that for locally optimal vertex, the linear subregion constraint in (12) can be removed.

(i) Necessity. We complete this part of proof by contradiction. As $\hat{z}$ is locally optimal, and with conclusions in [24], it is optimal for LP (12). Suppose there exists another vertex $\tilde{z} \in \mathcal{D}$ such that $l_{\hat{z}}(\tilde{z}) < l_{\hat{z}}(\hat{z})$. Apparently, $\tilde{z}$ is not feasible for the problem (12), i.e., $\tilde{z} \notin (\mathcal{D} \bigcap \{z | l_{\hat{z}}(z) = J(z)\})$. From the convexity of $\mathcal{D}$, we can construct one

$$z = \eta\hat{z} + (1 - \eta)\tilde{z}$$

such that z is a feasible point for (12). Then we have,

$$l_{\hat{z}}(z) = \eta l_{\hat{z}}(\hat{z}) + (1 - \eta) l_{\hat{z}}(\tilde{z}) < l_{\hat{z}}(\hat{z}),$$

which contradicts that $\hat{z}$ is optimal for the problem (12).

(ii) Sufficiency. As $\hat{z}$ is optimal for LP (11) and feasible for LP (12), it is optimal for LP (12). According to conclusions in [24], $\hat{z}$ is locally optimal. ■

Based on Lemma 2, starting from a feasible point z_0, evaluating the following algorithm will yield vertex which is locally optimal.

Algorithm 1: Local optimum searching algorithm.

Initialize

- Initial feasible point z_0.

repeat

- Construct the linear programming

$$\begin{array}{ll} \min & l_{z_0}(z) \\ s.t. & z \in \mathcal{D}, \end{array} \tag{13}$$

  the optimum is denoted as z^*.
- $z_0 = z^*$

until z_0 *is locally optimal*;

- Algorithm ends and return the local optimum.

The convergency of Algorithm 1 is proved in the following theorem.

Theorem 1: Algorithm 1 is a descent algorithm, i.e.,

$$J(z^*) \leq J(z_0), \tag{14}$$

it can converge to a locally optimal vertex within finite steps.

Proof: First we prove the validity of the equation (14). Denote Φ_0 as

$$\Phi_0 = \{z | l_{z_0}(z) = J(z)\}, \tag{15}$$

of which l_{z_0} is one of the affine functions in $\mathcal{L}(z_0)$.

If $z^* \in \Phi_0$, then

$$J(z^*) = l_{z_0}(z^*) \leq l_{z_0}(z_0) = J(z_0).$$

If $z^* \notin \Phi_0$, randomly choose a $\tilde{z} \in \overset{\circ}{\Phi}_0$ ($\overset{\circ}{\Phi}_0$ indicates the interior of Φ_0).

From the concavity of J, there exists $0 < \delta < 1$ such that $z_\delta = \delta\tilde{z} + (1 - \delta)z^* \in \Phi_0$ and

$$J(z_\delta) > \delta J(\tilde{z}) + (1 - \delta) J(z^*). \tag{16}$$

As $z_\delta \in \Phi_0$, we have,

$$J(z_\delta) = l_{z_0}(z_\delta) = \delta l_{z_0}(\tilde{z}) + (1-\delta) l_{z_0}(z^*). \tag{17}$$

From the equations (16) and (17), we have

$$(1-\delta) J(z^*) < (1-\delta) l_{z_0}(z^*).$$

Since $\delta \neq 1$ (which comes from the fact that $\tilde{z} \in \overset{\circ}{\Phi}_0$), the following holds,

$$J(z^*) < l_{z_0}(z^*) < l_{z_0}(z_0) = J(z_0), \tag{18}$$

confirming that Algorithm 1 is a descent algorithm.

Then, as the optimum of LP (13) is always obtained as the vertex of $\mathcal{D}$, and the number of vertices is finite, the algorithm converges to some vertex of $\mathcal{D}$ which is locally optimal. ■

It is noted that when using Algorithm 1, the linear programming is done in the domain $\mathcal{D}$, which is actually the feasible domain for the concave programming. This greatly facilitates the evaluation of the algorithm. Then, the local optimum is always the vertex of $\mathcal{D}$, which makes the use of γ-valid cut convenient.

Starting from different initial points z_0, a series of vertices which are locally optimal can be obtained and Lemma 1 can be used to test whether we reach the global optimum. In order to generate enough vertices locally optimal, the next section employs a modified GA.

V. Global minimum searching via GA

GA is basically a stochastic optimization algorithm. Compared with deterministic optimization, the stochastic version intends to visit the candidate solution randomly, thus resulting in a widespread searching, and may converge to the global optimum more efficiently. GA is evolved generation by generation, and each generation consists of a number of strings, also known as chromosomes. Through selection, crossover and mutation of the chromosomes of the previous generation, the next generation comes out. The evolution is repeated until some criterion of termination is satisfied.

A. Encoding

We relate each local optimal vertex to a chromosome in GA, and the operations crossover and mutation give rise to other chromosomes, which are also vertices locally optimal. And how is the relationship built?

Thanks to the CPWA property of the optimization problem, for a vertex z which is locally optimal, the chromosome is encoded in according to that whether the affine functions $\theta_m^T z$, $\theta_{km}^T z + \varphi_{km}$ take effects in (7), i.e., $\max\{0, \theta_m^T z\} = \theta_m^T z$ or $\max\limits_{j\in\{1,\ldots,K_m\}} \{\theta_{jm}^T z + \varphi_{jm}\} = \theta_{km}^T z + \varphi_{km}$. Specifically, we stipulate a string C with M characters, say $C = c_1 \cdot c_M$, then

$$\begin{aligned} &c_m \in \{0,1\}, \text{ if } m \in \mathcal{M}^+ \\ &c_m \in \{1,\ldots,K_m\}, \text{ if } m \in \mathcal{M}^- \end{aligned} \tag{19}$$

It is noted that $K_m \leq n$, hence when $m \in \mathcal{M}^-$, the number of possible values of c_m does not exceed n.

For the local optimum z, we can obtain the value of c_m as follows,

$$\begin{aligned} c_m = 0 &\iff \theta_m^+ z_0 < 0 \\ c_m = 1 &\iff \theta_m^+ z_0 \geq 0 \end{aligned}, \ \forall m \in \mathcal{M}^+ \tag{20}$$

and $\forall m \in \mathcal{M}^-$, we have

$$\begin{aligned} c_m = \quad k \iff \theta_{km}^T z_0 + \varphi_{km} \geq \theta_{jm}^T z_0 + \varphi_{jm}, \\ \forall j \in \{1,\ldots,K_m\}, j \neq k \end{aligned} \tag{21}$$

There may be more than one chromosomes corresponding to a vertex locally optimal, we randomly choose one of those.

The number of chromosomes in one generation is decided according to the problem and the performance of a chromosome C (vertex z) is evaluated through a fitness function [25],

$$\mathrm{FIT}(z) = \exp\{-\alpha f(z)\}$$

in which α is a positive scalar, and we choose appropriate α to make $\mathrm{FIT}(z)$ in the range of $(0, 1]$.

B. Selection, Mutation, and Crossover

Next, we demonstrate how we obtain offsprings from a parent population through mutation and crossover.

To do the mutation operation, we randomly select chromosome in the present generation, and preset a mutation probability p_0. Then we generate a probability vector $p = [p_1, \ldots, p_M]^T$, and each $p_m \in [0,1]$, $(1 \leq m \leq M)$. If $p_m \leq p_0$, the m-th bit will mutate. For this problem, the mutation means that if $m \in \mathcal{M}^+$, $c_m = 1 - c_m$, and if $m \in \mathcal{M}^-$, $c_m = \mod(c_m + 1, K_m)$, in which $\mod(c_m + 1, K_m)$ represents the remainder of division of $c_m + 1$ by K_m. The resulting chromosome can fix a linear programming and result in a vertex. If the resulting optimum is not locally optimal, we employ Algorithm 1 to find a vertex locally optimal. The chromosome relating to the locally optimal vertex is added to the offspring population.

For the crossover operation, we utilize the property of the concave programming (7) to select candidate local optimal vertices far away from each other. This is straightforward, as we want to escape from the current local optima and generate abundant local optima. Here we again use the information contained in the γ-valid cut, suppose we have obtained N vertices locally optimal, we attempt to construct a structure matrix $L_{N\times N}$ to approximately reflect the location of these vertices.

Definition 2: Suppose we have obtained N local optimal vertices $z_1, \ldots, z_N$, and their γ-valid cut are $\pi_i^T(z - z_i) \geq 1$, define $H_i(z) = \pi_i^T(z - z_i) - 1$, the elements of matrix L are,

$$L(i,j) = H_i(z_j), \ i,j = 1,\ldots,N.$$

Obviously, the diagonal element satisfies $L(i,i) = H_i(z_i) \leq 0$. And if $J(z_i) > J(z_j)$, we have $L(i,j) = H_i(z_j) > 0$.

In fact, if the point z_i and z_j are far away from each other, they may remain feasible after doing the γ-valid cut of each other, thus both $L(i,j)$ and $L(j,i)$ may be positive. Conversely, if the two points are close to each other, it is

likely that one point is cut off by the γ-valid cut of the other point. Hence, we give the following rules for crossover operation.

Rule 1: Given matrix $L \in \mathbb{R}^{N\times N}$, select the chromosomes corresponding to z_i and z_j to do the crossover operation, if both $L(i,j)$ and $L(j,i)$ are positive.

Rule 2: If there are no such z_i and z_j according to Rule 1, we adopt Roulette Wheel method to pick up chromosome pair to apply crossover operation. For each pair the fitness function is $|\alpha_1 L(i,j)+\alpha_2 L(j,i)|$, α_1 and α_2 are the weighting factors and can be adjusted for specific problem.

C. Termination criterion

The GA terminates once (10) is fulfilled. Otherwise, if (10) does not hold all the time, the GA terminates according to the general terminate criteria of GA, i.e., if the performance of the offspring gets no improved or the number of generations reaches a maximum number. In this paper, we adopt the latter one, the framework of the global optimum searching algorithm is illustrated in the following Fig. 1.

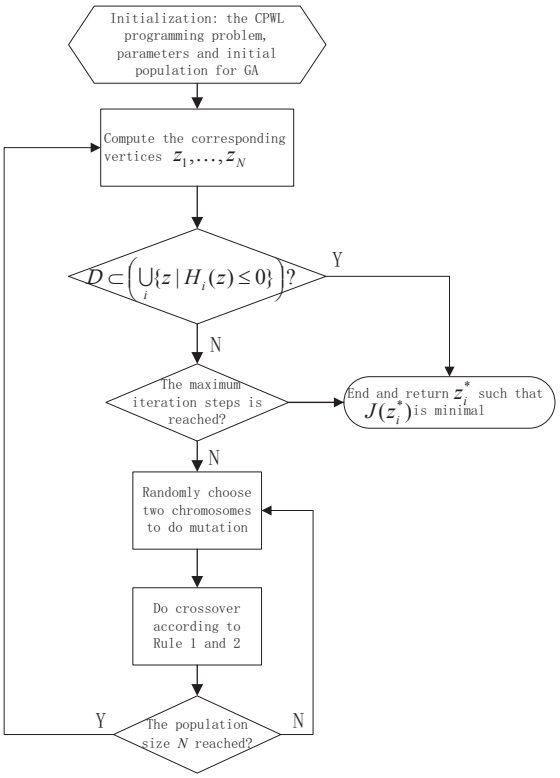

Fig. 1. Minimizing AHH through modified GA.

VI. Simulation Results

Considering two nonlinear functions listed in [26] for comparing the performance of different approximating strategies, assume we don't know the specific relationship but only a list of sample points drawn from them, we approximate those sample points with the model of AHH, and then minimize AHH.

A. Example 1: 2-dimensional optimization

$$\begin{aligned} y = \;& 1.9[1.35(1-x_1) \\ & +e^{x_1}\cdot\sin(13(x_1-0.6)^2)\cdot e^{-x_2}\cdot(\sin(7x_2))] \end{aligned} \tag{22}$$

in which $x_1, x_2 \in [0,1]$. The plot of the function can be seen in Fig. 2. The function (22) has three local optima,

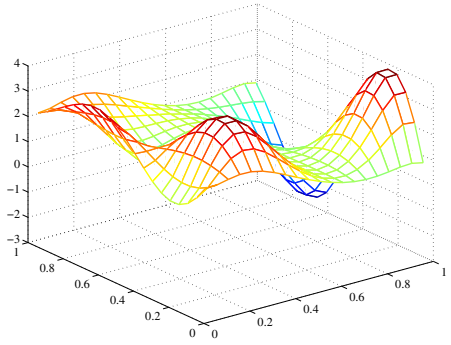

Fig. 2. Plot of the 2-dimensional function in Example 1.

$[1\ 0]^T$, $[0.2965\ 0.6529]^T$, $[0.9698\ 0.6529]^T$, and the corresponding function values are 0, 0.5782, -2.4496, we can find that the point $[0.9698\ 0.6529]^T$ is the global optimum.

The data points $(x(t),y(t))_{t=1}^N$ are uniformly generated in $[0,\ 1]^2$, assume $N = 400$, we approximate the 400 data points $(x(t),y(t))$ with AHH models different in M and N_{split}. For the parameters of the GA, the population size is 8, the mutation probability is 0.001, and the maximum number of iterations is 50. The optimum obtained is listed in Table I, and in this situation, each experiment reaches a global optimum. Also listed in the table is the approximation error e_{app}, which can be expressed as,

$$e_{\text{app}} = \frac{\sum_{t=1}^{N}(y(t)-\hat{y}(t))^2}{\sum_{t=1}^{N}(y(t)-\bar{y})^2}, \tag{23}$$

where $\hat{y}(t)$ is the AHH approximated output, and $\bar{y}$ is the mean of $y(t)$ over the 400 samples. To make a comparison, we also use SA, which is fulfilled through MATLAB. We use the subscript "$_{\text{s}}$" to denote the results for the SA method.

It can be seen from the table that when the number of basis functions increases, the deviation between the approximated AHH model and the original function is smaller, and the resulting optimum may be closer to the true global optimum. However, when we increase the number of candidate splitting points, the global optimum obtained is not closer to the true one, which is not as expected. This is because we aim at minimizing e_{app} when approximating the sample points, though overall precision is higher, the local precision around the global optimum may not be improved. Therefore, in real applications, the approximated model should be carefully chosen, moreover, if necessary, local approximation around

TABLE I

RESULTS OF MINIMIZING AHH MODELS WITH DIFFERENT M AND N_{split}.

M	N_{split}	e_{app}	x^*	x_{s}^*	$f(x^*)$	$f_{\text{s}}(x_{\text{s}}^*)$	$t(s)$	$t_{\text{s}}(s)$
15	20	0.1103	$[0.925\ 0.625]^T$	$[0.926\ 0.626]^T$	-2.3734	-2.3731	0.5303	0.8202
15	50	0.0769	$[1\ 0.62]^T$	$[1\ 0.62]^T$	-3.0142	-3.0142	0.5108	0.8966
40	20	0.0182	$[0.95\ 0.65]^T$	$[0.9505\ 0.6496]^T$	-2.5556	-2.5550	1.9307	0.6116
40	50	0.0088	$[1\ 0.66]^T$	$[1\ 0.66]^T$	-2.7270	-2.7270	1.7753	1.3178

those points with a large approximation error may be employed.

For this problem, both the modified GA and SA method perform well. Although the time needed for evaluating our strategy is more than that of SA in some situations, the possibility of gaining a global optimum is larger.

In order to obtain a precise approximation, the number of M and N_{split} may be large. However, the computational cost of approximation and optimzation also increase. Hence we should carefully choose the value of the two parameters to make a tradeoff between the precision and computational complexity.

B. Example 2: 4-dimensional Optimization

$$y = e^{2x_1 \sin(\pi x_4)} + \sin(x_2 x_3) \tag{24}$$

in which $x \in [-0.25,\ 0.25]^4$.

We generate 1000 data points conforming to the uniform distribution in $[-0.25,\ 0.25]^4$. For this problem, the AHH model with 40 basis functions is employed to approximate the data, and the number of candidate splitting points in each dimension, N_{split}, is set to 10. The parameters in GA is set the same as the previous example.

For this example, the AHH approximation error is

$$e_{\text{app}} = 0.0249.$$

The global solution for the AHH minimization found by the modified GA is

$$[-0.25\ -0.25\ 0.25\ 0.25]^T.$$

It is actually a global optimum of the original function. Hence in this example, the global optimum of AHH coincides with the original global optimum. Actually, there are another three global optima of the original function, $[-0.25\ 0.25\ -0.25\ 0.25], [0.25\ -0.25\ 0.25\ -0.25]^T$, and $[0.25\ -0.25\ 0.25\ -0.25]^T$.

The running time elapsed is 50.5s. Apparently, for problem of higher dimension, the computational cost of searching for global optimum is much more.

As is indicated in [12], the computational cost for AHH approximation increases abruptly with the problem dimension n. Moreover, the optimization complexity is higher when n increases, which can be seen from the running time for the two examples. Hence, for high-dimensional problem, the proposed strategy can only be done offline.

VII. CONCLUSIONS

In this paper, we propose the optimization based on adaptive hinging hyperplanes and genetic algorithm, i.e., the unknown or difficult obtaining relationship is approximated by the AHH model through approximating sample points. Then, a modified GA is evaluated to minimize AHH. Considering the optimization is a CPWLP, we transform it to a concave programming in which the cost function is concave piecewise affine and the feasible region is a polyhedron. Thus in the modified GA, starting from a chromosome, we can get a vertex that is locally optimal, then through selection, mutation and crossover, we gain plenty of locally optimal vertices. The termination of GA is judged by the criterion based on the γ-valid cuts of all the vertices. Simulation results concerning two functions show that if the approximated AHH model as well as the parameters of the GA is carefully chosen, the proposed strategy will yield a solution close to the original global optimum.

REFERENCES

[1] K. S. Won and T. Ray, "A framework for design optimization using surrogates," *Engineering Optimization*, vol. 37, no. 7, pp. 685–703, 2005.

[2] A. Forrester and A. Keane, "Recent advances in surrogate-based optimization," *Progress in Aerospace Sciences*, vol. 45, pp. 50–79, 2009.

[3] N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, "Surrogate-based analysis and optimization," *Progress in Aerospace Sciences*, vol. 41, pp. 1–28, 2005.

[4] A. Safavi, A. Nooraii, and J. Romagnoli, "A hybrid model formulation for a distillation column and the on-line optimisation study," *Journal of Process Control*, vol. 9, no. 2, pp. 125–134, 1999.

[5] W. Lv, Y. Zhu, D. Huang, Y. Jiang, and Y. Jin, "A new strategy of integrated control and on-line optimization on high-purity distillation process," *Chinese Journal of Chemical Engineering*, vol. 18, no. 1, pp. 66–79, 2010.

[6] T. Chow, G. Zhang, Z. Lin, and C. Song, "Global optimization of absorption chiller system by genetic algorithm and neural network," *Energy and buildings*, vol. 34, no. 1, pp. 103–109, 2002.

[7] X. Huang, J. Xu, and S. Wang, "Operation optimization for centrifugal chiller plants using continuous piecewise linear programming," in *2010 IEEE International Conference on Systems, Man, and Cybernetics*, 2010, pp. 1121–1126.

[8] D. Cook, C. Ragsdale, and R. Major, "Combining a neural network with a genetic algorithm for process parameter optimization," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 4, pp. 391–396, 2000.

[9] R. Jin, W. Chen, and T. W. Simpson, "Comparative studies of metamodelling techniques under multiple modelling criteria," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, 2001.

[10] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree, "Kriging models for global approximation in simulation-based multidisciplinary design optimization," *AIAA journal*, vol. 39, no. 12, pp. 2233–2241, 2001.

[11] K.-D. Lee and K.-Y. Kim, "Surrogate based optimization of a laidback fan-shaped hole for film-cooling," *International Journal of Heat and Fluid Flow*, vol. 32, pp. 226–238, 2011.
[12] J. Xu, X. Huang, and S. Wang, "Adaptive hinging hyperplanes and its applications in dynamic system identification," *Automatica*, vol. 45, no. 10, pp. 2325–2332, 2009.
[13] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.
[14] S. Wang and X. Sun, "Generalization of hinging hyperplanes," *IEEE Transactions on Information Theory*, vol. 12, no. 51, pp. 4425–4431, 2005.
[15] R. Fourer, "A simplex algorithm for piecewise-linear programming i: Derivation and proof," *Mathematical Programming*, vol. 33, no. 2, pp. 204–233, 1985.
[16] ——, "A simplex algorithm for piecewise-linear programming ii: Finiteness, feasibility and degeneracy," *Mathematical Programming*, vol. 41, no. 1, pp. 281–315, 1988.
[17] ——, "A simplex algorithm for piecewise-linear programming iii: Computational analysis and applications," *Mathematical Programming*, vol. 53, no. 1, pp. 213–235, 1992.
[18] A. Keha, I. de Farias Jr, and G. Nemhauser, "A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization," *Operations research*, vol. 54, no. 5, pp. 847–858, 2006.
[19] H. Sherali, "On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions," *Operations Research Letters*, vol. 28, no. 4, pp. 155–160, 2001.
[20] J. 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.
[21] R. R. Sharapov and A. V. Lapshin, "Convergence of genetic algorithms," *Pattern Recognition and Image Analysis*, vol. 16, no. 3, pp. 392–397, 2006.
[22] R. Horst and T. Hoang, *Global optimization: Deterministic approaches*. Berlin : Springer Verlag, 1990.
[23] H. Benson, "Generalized γ-valid cut procedure for concave minimization," *Journal of Optimization Theory and Applications*, vol. 102, no. 2, pp. 289–298, 1999.
[24] 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, pp. 145–164, 2012.
[25] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, "Genetic algorithm toolbox for use with matlab," Department of Automation Control and System Engineering, University of Sheffield, Tech. Rep., 1994.
[26] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of adaptive methods for function estimation from samples," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 969–984, 1996.