ELSEVIER

European Journal of Operational Research 106 (1998) 676–688

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Hybrid extreme point tabu search

Jennifer A. Blue [1], Kristin P. Bennett *

Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Received 1 April 1996; received in revised form 1 December 1996

Abstract

We develop a new hybrid tabu search method for optimizing a continuous differentiable function over the extreme points of a polyhedron. The method combines extreme point tabu search (EPTS) with traditional descent algorithms based on linear programming. The tabu search algorithm utilizes both recency-based and frequency-based memory and oscillates between local improvement and diversification phases. The hybrid algorithm iterates between using a descent algorithm to find a local minimum and tabu search to improve locally and then move to a new area of the search space. The descent algorithm acts as a form of intensification within the tabu search. This algorithm can be used on many important classes of problems in global optimization including bilinear programming, multilinear programming, multiplicative programming, concave minimization, and complementarity problems. The algorithm is applied to two practical problems: the quasistatic multi-rigid-body contact problem (QCP) in robotics and the global tree optimization (GTO) problem in machine learning. We perform computational experiments on problems of significant real world dimensions: the smallest machine learning problem tested has on the order of 10^{485} extreme points. The results show our method obtains high quality solutions both by comparison with the embedded descent algorithm and by comparison to a version of tabu search that does not make use of descent.

Keywords: Tabu search; Global optimization; Bilinear programming; Machine learning; Classification

1. Introduction

Minimization of a function subject to linear constraints is one of the most fundamental problems in optimization. In particular we are interested in problems for which an extreme point or vertex solution is desired or known to exist. The most common example of this problem is linear programming. Fast and powerful techniques developed for linear programming exist for traversing the extreme points of a polyhedral constraint region to an optimal solution [18,17]. There are many global optimization problems with nonlinear objectives and linear constraints in which an optimal extreme point solution is desired or known to exist. Examples can be found in bilinear programming, multilinear programming, multiplicative programming, concave minimization, fixed charge [24,14], and complementarity problems [12,7]. Our goal is to develop an algorithm for

* Corresponding author. E-mail: bennek@rpi.edu.
[1] E-mail: bluej@rpi.edu.

0377-2217/98/$19.00
PII S0377-2217(97)00297-X

nonlinear objective problems that searches the extreme points for an optimal or near-optimal solution.

One strategy used for bilinear and other differentiable objective functions is to iteratively linearize the objective and solve the resulting linear program. Examples of these algorithms are the uncoupled bilinear programming algorithm (UBPA) [6,19], Frank–Wolfe type algorithms (FW) [6,9], and successive linear programming [2]. These iterative linear programming algorithms find a local minimum relatively quickly and then stop. Powerful simplex method codes such as MINOS [17] can be used quickly and efficiently to solve the linear subproblems. The algorithms are simple and have few parameters. Searches of adjacent extreme points have been added to make such algorithms more robust [19]. But their functionality is limited on global optimization problems. Gradient descent methods such as the reduced-gradient method can also be applied to such problems, but are slow due to a large number of gradient calculations.

Tabu search is well suited for minimizing nonlinear functions with linear constraints. There is a natural neighborhood structure. Each extreme point corresponds to a basic feasible solution (BFS). We can examine an adjacent extreme point by exchanging a variable outside the basis for a variable inside the basis. This is the pivot used in the simplex method for linear programming [18]. Recency memory is maintained by keeping track of when variables are pivoted in and out of the basis. Frequency memory is incorporated by keeping track of how often variables appear in the basis. Tabu search algorithms using extreme points have been successfully applied to integer programming, fixed charge problems [24,22,23], and bilinear programming applications [5,11,1,13]. One limitation of extreme point tabu search (EPTS) is that if the evaluation of the objective function is expensive and the neighborhoods are large, then tabu search can be slow when compared with the local descent methods described above. In addition, the parameter choices and features added to tabu search can make it less aggressive than the iterative linear programming algorithms.

In hybrid extreme point tabu search (HEPTS), we cycle between the two approaches. We use an appropriate local descent algorithm to move rapidly to a local minimum. Tabu search is used to investigate the area around the local minimum and then to diversify to a new part of the search space. Both the descent method and the tabu search use the same neighborhood structure. The descent method acts as a form of intensification within the tabu search. The descent methods use gradient information to quickly choose decreasing moves without an expensive search of the neighborhood. The tabu search becomes more efficient and aggressive while maintaining global robustness. The algorithm can be viewed as a form of restart with long-term memory. The frequency-based memory allows us to target new areas of the search space.

To illustrate the practicality of this approach we experimented with two applications. The quasistatic multi-rigid-body contact problem (QCP) is used to predict the motion of a passive rigid body in contact with robot manipulators [25]. This NP-complete problem is an uncoupled linear complementarity problem that can be formulated as an uncoupled bilinear program [19]. The second problem is global tree optimization (GTO) in machine learning [4]. In this problem, we try to construct a decision tree with a given structure to correctly classify points from two classes. This NP-complete problem can be posed as an extended linear complementarity problem that can be formulated as a coupled multilinear program [5].

This paper is organized as follows. In Section 2 we define the problem we are interested in and discuss its possible applications. Our version of EPTS is described in Section 3. A brief review of two local descent techniques is provided in Section 4. The new hybrid approach is developed in Section 5. Computational results on the two types of practical problems are given in Section 6. Directions for future work are given in Section 7.

We use the following notation. For a vector x in the n-dimensional real space $\mathbb{R}^n$, $(x)_i$ denotes the ith component of x and x_+ denotes the vector in $\mathbb{R}^n$ with components $(x_i)_+ := \max\{x_i, 0\}$, $i = 1, \ldots, n$. The dot product of two vectors x

and w is indicated by xw. The outer product is never used.

2. Basic problem

We are interested in problems that involve minimization of an objective function subject to linear constraints. To ensure that the hybrid algorithm can be used, we restrict the objective to functions with continuous first partial derivatives. In general, continuity and differentiability are not required for EPTS. We assume without loss of generality that the constraints are in the "standard form" used in linear programming [18]. Thus the problem becomes:

$$\begin{aligned} &\min_{x} \quad f(x) \\ &\text{s.t.} \quad x \in \mathscr{X}, \quad \mathscr{X} := \{x \mid Ax = b,\ x \geqslant 0\}, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $f: \mathbb{R}^n \to \mathbb{R}$. For this paper we will assume that $n > m$. The set $\mathscr{X}$ is a polyhedron. We do not assume that $\mathscr{X}$ is bounded.

The linear constraints in $\mathscr{X}$ form a simplex. We review briefly the properties of the simplex and advise the reader to consult a text on linear programming for more details [18]. Any point in $\mathscr{X}$ that cannot be written as a linear combination of any two other distinct points in $\mathscr{X}$ is called an extreme point. Each extreme point corresponds to a BFS. Each BFS consists of n variables. Thus, the polyhedron has $n!/m!(n-m)!$ extreme points. The relatively small problem described in Section 6.1 with $m = 9$ and $n = 18$ has 48620 extreme points. The smallest machine learning problem attempted in Section 6.2 with $m = 800$ and $n = 1618$ has on the order of 10^{485} extreme points. The largest problem solved by Sun's similar algorithm [22] for the fixed charge problem was only $m = 110$ and $n = 120$ which has on order of 10^{14} extreme points.

We can move from one extreme point to an adjacent extreme point by doing a pivot. Once a variable is chosen to enter the basis, the ratio-test is used to determine the exiting variable. The basis is then updated. The BFSs and the pivot operation provide a natural definition for the neighborhood and memory structures needed for tabu search.

3. Extreme point tabu search

Our version of extreme point tabu search (EPTS) is an extension of the algorithm described in [5]. Other versions of tabu search applied to extreme points can be found in [11,1,13,24]. We begin with a description of the features of the algorithm and recommend parameter choices. The resulting algorithm is summarized in Algorithm 3.1.

The tabu search neighborhood of each extreme point consists of the adjacent extreme points. Since the choice of variables in the basis uniquely determines the extreme point, the memory structures need only track which variables are in the basis. For recency-based memory, the iteration number is recorded whenever a variable enters or exits the basis. For frequency-based memory, we keep count of the number of times each variable is used in the basis. We increment the count of the variables in the basis at every iteration, not just for critical events as suggested in [10]. We explored the critical event strategy but found it did not enhance the quality of our results. The best solution found so far in terms of the given objective function is used as the aspiration criterion.

Moves can become tabu in two ways. If a nonbasic variable is required to reenter the basis before its tabu-tenure has expired, the move is tabu. A move may also be tabu if a basic variable is required to leave the basis before its tabu-tenure has expired. The tabu-tenure of nonbasic variables is longer than that of basic variables since the number of possible entering variables is much larger than that of exiting variables. We set the tabu-tenure for nonbasic variables to $\sqrt{\textit{total number of variables}/8}$ and for basic variables to 1/4 of the tabu-tenure of the nonbasic variables. We found the results were relatively insensitive to changes in the tabu-tenure. These parameters and all other parameters were chosen through experimentation with the generated classification problems described in Section 6.2.

Since the number of adjacent extreme points and cost of function evaluation can be quite large, a candidate list is used to limit the number of function evaluations performed when selecting a move. The candidate list contains the k best pivots at

some iteration and is not regenerated at every iteration. Possible moves from the candidate list are evaluated until a non-tabu improving move is found or the aspiration criterion is satisfied. The selected move is then removed from the candidate list. If no improving non-tabu moves are found then the candidate list is remade. If still no improving moves are found then a tabu move is taken. To ensure a good selection of moves, the candidate list is remade if it becomes less than 1/8 of its original length.

EPTS oscillates between a local improvement mode and a diversification mode. In the local improvement mode, the objective function is used to evaluate the qualities of the move. The diversification strategy is a variation of the approach suggested in [10]. The following frequency penalty function is used to force the algorithm into unexplored areas of the search space.

$$penalty(x) = f(x) + \frac{\rho}{iteration} \sum_{i \in basis} frequency(i), \tag{2}$$

where i belongs to the set of indices of the variables in the current basis, ρ is a large positive penalty constant, *iteration* the number of the current iteration, *frequency*(i) a count of the number of times the variable has appeared in the basis, and $f(x)$ the original objective function. Since we are only ranking adjacent extreme points at any iteration, we need only calculate

$$penalty(x) = f(x) + \frac{\rho}{iteration}(frequency(entering\ i) - frequency(exiting\ i)) + C, \tag{3}$$

where *entering i* is the index of the entering variable, *exiting i* the index of the departing variable and C the sum of frequency of the variables in the old basis. In practice, a large constant can be used for C provided it is sufficiently large to avoid negative values of *penalty*(x).

The algorithm dynamically determines when to change modes. The algorithm starts in the local improvement mode. If the objective function has not improved significantly in the last k iterations, the algorithm switches to the diversification mode. For small problems with 200 points or less we use $k = 0.25n$, and $k = 0.03n$ for larger problems. EPTS continues in diversification mode until the original objective begins to improve or a maximum number of diversification iterations is reached. The maximum number of diversification iterations is set to $0.1n$ for small problems and $0.25n$ for larger problems. We say the objective has not improved if it has not changed by δ % in the last ζ iterations. We choose $\delta = 2.0$ % with $\zeta = 0.25n$ for small problems and $\delta = 0.5$ % with $\zeta = 0.1n$ for larger problems. These parameters were chosen by tuning the algorithm on the generated classification problems described in Section 6.2. In the diversification mode, we also find it is necessary to increase the tabu-tenure of both nonbasic and basic variables to avoid cycling. The stopping criteria are met when the optimal solution is found or an iteration limit is reached.

Algorithm 3.1 (EPTS).

0. Start with an initial basic feasible solution for Problem (1).
1. Start in local improvement mode with original objective.
2. If iteration limit is exceeded or solution is optimal, stop.
3. (a) If in local improvement mode and no progress is being made then switch to diversification mode with penalized frequency objective (3).
   (b) Else if in diversification mode and the actual objective is improving or the maximum number of diversification steps has been reached, switch to local improvement mode with original objective.
4. Until a move is chosen
   (a) If the candidate list is too small or all the moves are tabu, then remake the candidate list.
   (b) Find first improving move on the candidate list.
   (c) If not tabu, take that move.
   (d) Else if tabu and aspiration criterion met, take that move.
   (e) Else if all moves are tabu and the candidate list has just been remade, take the best tabu move.
5. Perform move and update frequency and recency memories.
6. Go to step 2.

The basic EPTS algorithm is very similar to the tabu search algorithms proposed for fixed charge problems by Sun and McKeown [24], Sun [22] and Sun et al. [23]. We both define the tabu move as a pivot to an adjacent extreme point and use the same tabu structure, but a slightly different aspiration criterion is used. The approach in the original paper of Sun and McKeown [24] is adapted to the fixed charge transportation problem in [22,23]. It is interesting to note that they independently enhanced the algorithm with many of the same approaches used in this paper. Specifically, long-term frequency memory was added and a penalized objective function based on the frequency is used for diversification. Sun et al. also use this frequency memory for intensification, a feature that is not used in EPTS but may lead to better results. In the HEPTS algorithm, a gradient descent method is used for intensification within the tabu search. Later work of Sun et al. incorporates a candidate list strategy customized for transportation problems. Sun et al. applied their algorithm to relatively small problems. The largest problem solved by Sun's algorithm [22] for the fixed charge problem has on order of 10^{14} extreme points while the smallest machine learning problem EPTS was applied to had on order of 10^{485}. Note that EPTS is a general purpose algorithm that could be applied directly to fixed charge problems.

4. Review of descent algorithms

Many descent algorithms can be used within HEPTS. We looked at two algorithms: the UBPA and FW.

UBPA [6] can be applied to uncoupled bilinear problems with the following form:

$$\begin{aligned} \min_{x,y} \quad & xy \\ \text{s.t.} \quad & x \in \mathcal{X}, \quad \mathcal{X} := \{x \mid Ax = b,\ x \geqslant 0\} \\ & y \in \mathcal{Y}, \quad \mathcal{Y} := \{y \mid Cy = d,\ y \geqslant 0\}. \end{aligned} \tag{4}$$

The problem is called uncoupled because the constraint sets $\mathcal{X}$ and $\mathcal{Y}$ are independent. UBPA takes advantage of the fact that the constraints are uncoupled.

Algorithm 4.1 (UBPA [6]). Start with any feasible point (x^0, y^0) for Eq. (4). Determine (x^{i+1}, y^{i+1}) from (x^i, y^i) as follows:

$$x^{i+1} \in \arg\operatorname{vertex}\min_x \{xy^i \mid Ax = b,\ x \geqslant 0\},$$

$$y^{i+1} \in \arg\operatorname{vertex}\min_y \{x^{i+1}y \mid Cy = d,\ y \geqslant 0\}$$

and such that $x^{i+1}y^{i+1} < x^iy^i$. Stop when impossible.

In the above algorithm, "arg vertex min" denotes an extreme point in the solution set of the indicated linear program. UBPA terminates at a local minimum satisfying the minimum principle necessary optimality condition [6].

FW can be applied to any instance of Problem (1) provided f has continuous first partial derivatives on $\mathcal{X}$ and f is bounded below on $\mathcal{X}$. In [6], FW was shown to terminate at a local minimum satisfying the minimum principle necessary optimality condition.

Algorithm 4.2 (FW [9,6]). Start with any $x^0 \in \mathcal{X}$.

1. $v^i \in \arg\operatorname{vertex}\min_{x\in\mathcal{X}} \nabla f(x^i)x$.
2. Stop if $\nabla f(x^i)v^i = \nabla f(x^i)x^i$.
3. $x^{i+1} = (1-\lambda^i)x^i + \lambda^i v^i$ where $\lambda^i \in \arg\min_{0\leqslant\lambda\leqslant 1} f((1-\lambda)x^i + \lambda v^i)$.
4. Set $i = i+1$. Go to Step 1.

Both UBPA and FW iteratively linearize the objective and then solve the resulting objective. FW can be applied to bilinear programs with uncoupled or coupled constraints. FW is slightly more expensive because it requires a line search. In practice both algorithms find a local minimum after a small number of linear programs are solved [6,19].

5. Hybrid extreme point tabu search

HEPTS combines a fast local descent method with the robust global optimization properties of EPTS. The basic idea is to use a descent algorithm such as UBPA or FW to get to a local minimum. Then tabu search is used to further explore the lo-

cal area and diversify into a new search area. The descent algorithm performs the role of intensification within the tabu search.

The basic algorithm is as follows.

Algorithm 5.1 (HEPTS). Start with any basic feasible point (x^0).

1. Use appropriate descent algorithm to find x^i.
2. If x^i is optimal or stopping criteria are met then stop.
3. Use EPTS (3.1) to find new point x^{i+1} in new region of search space.
4. Set $i = i + 1$. Go to Step 1.

The stopping criteria for the hybrid algorithm are a set number of runs through the algorithm with termination after the appropriate descent algorithm. We used 10 cycles for our results. In HEPTS, the basic EPTS algorithm is used unchanged except that the frequency information is retained from cycle to cycle and the parameters are adjusted to limit the time spent in the local improvement phase. The descent algorithms are far more efficient at local improvement, so if EPTS does not quickly make progress we switch to the diversification phase. The EPTS algorithm starts in the local improvement mode. If the objective function has not changed by 2% in the last $k = 0.03n$ iterations, the algorithm switches to the diversification mode. EPTS continues in diversification mode until the original objective begins to improve or a maximum number of diversification iterations is reached. The maximum number of diversification iterations is set to $0.15n$. An iteration limit of $0.18n$ is used within the EPTS algorithm. These parameters were tuned using the same generated problems that were used to tune EPTS.

6. Computational results

HEPTS, EPTS, UBPA, and FW were implemented and tested on two practical applications from robotics and machine learning. The MINOS linear programming package [17] was customized to perform all four algorithms. Both applications can be formulated as complementarity problems. The multi-rigid-body contact problem requires the solution of an uncoupled bilinear program. The GTO problem requires the solution of coupled bilinear or multilinear programs.

6.1. Quasistatic multi-rigid-body contact problem

QCP predicts the motion of a quasistatic rigid object when it is in contact with multiple rigid robot manipulators. The goal is to use QCP to aid in automatically planning the motion of a robot manipulating an object. We briefly describe QCP and refer the reader to [25,19] for more details. Since QCP is NP-complete and has extreme point solutions, it is an ideal problem for HEPTS. Our computational results in this paper are compared to solutions of problems given in [19].

In [19], QCP was formulated as an uncoupled complementarity problem (4) with the following general form: Find $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ such that

$$xy = 0, \quad Ax = b, \quad Cy = d, \quad x \geqslant 0, \quad y \geqslant 0, \tag{5}$$

where A is an $l \times n$ matrix, $b \in \mathbb{R}^l$, C a $k \times n$ matrix and $d \in \mathbb{R}^k$. An algorithm, we will call QCPA, minimizes an uncoupled bilinear program to find a solution of QCP. The problem minimizes xy subject to the remaining constraints. QCPA uses the above UBPA (4.1) as a subproblem. A local minimum is found using UPBA and then the adjacent extreme points are searched for improved solutions. If no improved move is found a random move is taken. Then UBPA is restarted from the new point. This restart strategy is very similar to how UBPA is used within HEPTS. The difference is that HEPTS maintains long-term memory to help guide the search of the adjacent extreme points. QCPA performed very well on this problem. It successfully solved 78 problems out of the 82 problems attempted.

We tested HEPTS, EPTS, and UBPA on six problems solved by QCPA. Results for these six problems using QCPA can be found in [19]. Data sets for three of the problems (Problems 1–3) were given in [19, p. 151]. These three problems have 18

Table 1
Objective values of four methods: Pang et al. QCPA, local descent algorithm UBPA, EPTS, and HEPTS

Problem	Variables	Constraints	QCPA	UBPA	EPTS	HEPTS
1	18	9	0	0	0	0
2	18	9	0	1.00129	0	0
3	18	9	0	0	0	0
10	74	120	0	0	11.12245	0
18	55	90	14.61551	14.61566	125.39537	0.53738
19	109	180	24.32773	33.85649	82.59379	10.23229

variables and nine constraints each. Data sets for Problems 10, 18, and 19 were obtained from Pang and Lo. These problems have 74, 55, and 109 variables with 120, 90, and 180 constraints, respectively. Results from the QCPA algorithm as reported in [19] are included in Table 1. UBPA found globally optimal solutions in Problems 1, 3, and 10. Thus, HEPTS also found globally optimal solutions on these problems as UBPA is the first step of the HEPTS algorithm. UBPA failed to find the global optimal solution in Problems 2, 18, and 19. EPTS found the globally optimal solution for Problems 1–3, but failed to do so for Problems 10, 18, and 19. HEPTS found the global optimal solution for Problems 1–3, and 10. While failing to find the globally optimal solution for Problems 18 or 19, it did find better solutions than any of the other three methods. It is not known if a globally optimal solution of zero exists for Problems 18 and 19.

EPTS was run with an iteration limit of 200 (both by itself and within HEPTS). The UBPA, EPTS, and HEPTS algorithms we tested all ran in under 10 s on a SUN workstation, except for Problem 19. Problem 19 took 386 s for HEPTS and 25 s for EPTS to complete. In Problems 18 and 19 we encountered feasibility problems within MINOS. Most likely, this is due to the fact that our data does not have as many significant digits as the data used in [19]. The feasibility tolerance or maximum infeasibility allowed per constraint was increased for these problems. The solutions given in Table 1 for Problems 18 and 19 are for a feasibility tolerance of 10^{-3}, while the solutions to Problems 1–3 have a feasibility tolerance of 10^{-6}, and the solution to Problem 10 has a feasibility tolerance of 10^{-5}.

6.2. Global tree optimization

In GTO, the problem of constructing a decision tree with a given structure to recognize points from two classes is formulated as a multilinear program [4]. If a tree with the given structure exists that completely classifies the points in the two sets then the solution satisfies an extended linear complementarity problem [21,5]. Even a problem using a tree with only two decisions is NP-Complete [15,6]. If a tree cannot be constructed to completely classify the points, a tree that minimizes the classification error is desired. The goal is to construct trees that generalize well, i.e., trees that correctly predict future points.

GTO can be used in nongreedy algorithms to construct multivariate decision trees. Suppose we are given two sets of points. Set $\mathscr{A}$ consists of m_A points in $\mathbb{R}^N$ from Class $\mathscr{A}$. Set $\mathscr{B}$ consists of m_B points in $\mathbb{R}^N$ from Class $\mathscr{B}$. We are also given a multivariate decision tree with fixed structure. For example, Fig. 1(a) is a tree with three decisions and Fig. 1(b) is a tree with seven decisions. Each decision consists of a linear combination of the attributes of the points. For example, for the root of the tree and the point x, if $xw^1 > \gamma^1$ then the point follows the right path and if $xw^1 \leqslant \gamma^1$ then the point follows the left path. We call $w^1 \in \mathbb{R}^N$ the weights and $\gamma^1 \in \mathbb{R}$ the threshold of the decision. The error of the entire tree is formulated as a bilinear or multilinear program and the error is minimized. This contrasts with greedy decision tree methods such as C4.5 [20] that construct a decision tree one decision at a time until a desired accuracy is reached. Both algorithms, FW and EPTS, have been applied with reasonable success to the GTO problem.

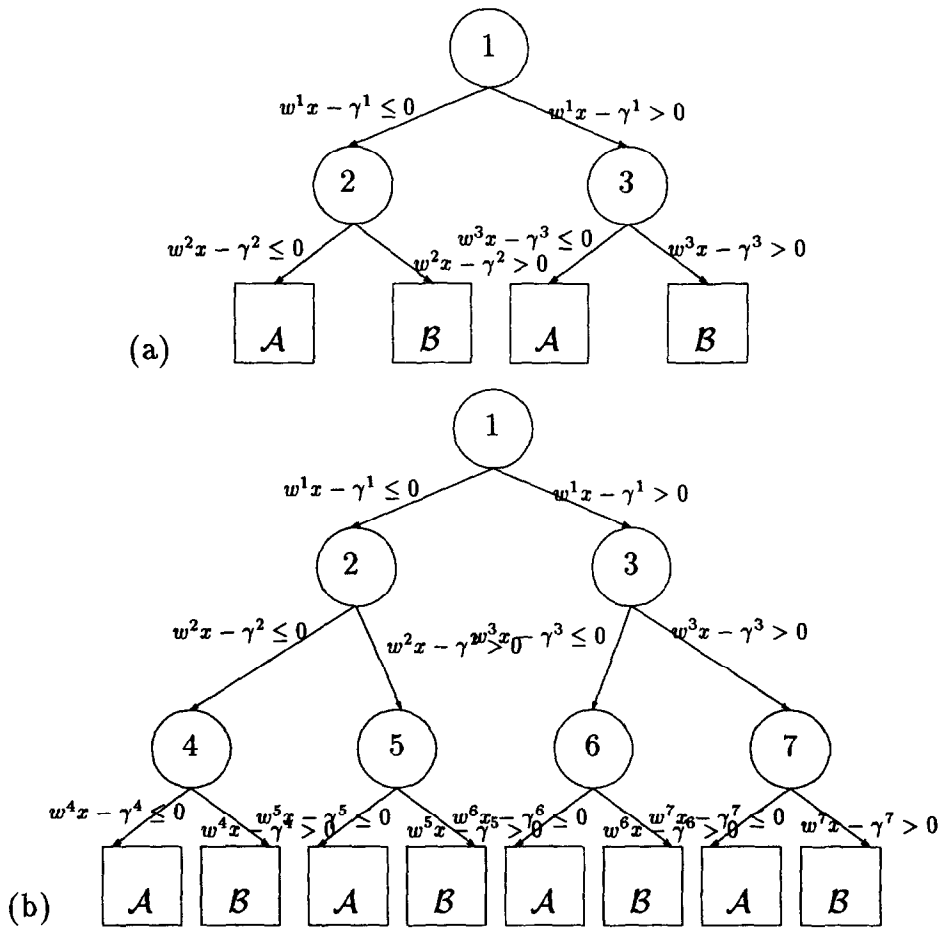

Fig. 1. Multivariate decision trees with (a) three and (b) seven decisions.

Computational results showed that the quality of solutions found were very good in terms of generalization, but the global minima were frequently not found.

We explored constructing trees with three decisions (seven total nodes) and seven decisions (15 total nodes). The reader should consult [5] for details on how the problems are constructed. The three-decision tree requires solution of the following multilinear program:

$$\min_{y,z,w,\gamma} \quad \sum_{i=1}^{m_A}((y_l^1)_i + (y_l^2)_i)((y_g^1)_i + (y_l^3)_i) + \sum_{j=1}^{m_B}((z_l^1)_j + (z_g^2)_j)((z_g^1)_j + (z_g^3)_j)$$

$$\begin{aligned}
\text{s.t.} \quad & (y_l^1)_i \geq A_i w^1 - \gamma^1 + 1, \\
& (y_g^1)_i \geq -A_i w^1 + \gamma^1 + 1, \\
& (y_l^2)_i \geq A_i w^2 - \gamma^2 + 1, \\
& (y_l^3)_i \geq A_i w^3 - \gamma^3 + 1, \\
& (z_l^1)_j \geq B_j w^1 - \gamma^1 + 1, \\
& (z_g^1)_j \geq -B_j w^1 + \gamma^1 + 1, \\
& (z_g^2)_j \geq -B_j w^2 + \gamma^2 + 1, \\
& (z_g^3)_j \geq -B_j w^3 + \gamma^3 + 1, \\
& i = 1, \ldots, m_A, \quad j = 1, \ldots, m_B, \quad y, z \geqslant 0,
\end{aligned} \tag{6}$$

where $A_i \in \mathbb{R}^N$ is the ith point in $\mathcal{A}$, and $B_j \in \mathbb{R}^N$ the jth point in $\mathcal{B}$. This problem can be simplified to form a coupled bilinear program.

The seven-decision tree results in the following multilinear program:

$$
\begin{aligned}
\min_{y,z,w,\gamma} \quad & \sum_{i=1}^{m_A}\Big(((y_l^1)_i + (y_l^2)_i + (y_l^4)_i)((y_l^1)_i + (y_g^2)_i \\
& +(y_l^5)_i)((y_g^1)_i + (y_l^3)_i + (y_l^6)_i)((y_g^1)_i \\
& + (y_g^3)_i + (y_l^7)_i)\Big) \\
& +\sum_{j=1}^{m_B}\Big(((z_l^1)_j + (z_l^2)_j + (z_g^4)_j)(z_l^1)_j \\
& +(z_g^2)_j + (z_g^5)_j)((z_g^1)_j + (z_l^3)_j + (z_g^6)_j)((z_g^1)_j \\
& + (z_g^3)_j + (z_g^7)_j)\Big) \\
\text{s.t.} \quad & (y_l^k)_i \geqslant A_i w^k - \gamma^k + 1, \\
& (y_g^k)_i \geqslant -A_i w^k + \gamma^k + 1 \\
& (z_l^k)_j \geqslant B_j w^k - \gamma^k + 1, \\
& (z_g^k)_j \geqslant -B_j w^k + \gamma^k + 1 \\
& i = 1,\ldots,m_A,\ j = 1,\ldots,m_B, \\
& k = 1,\ldots,7, \quad y,z \geqslant 0.
\end{aligned} \tag{7}
$$

To evaluate how well HEPTS performed, we randomly generated problems that could be classified by the three-decision and seven-decision trees in Fig. 1. For the three-decision tree, we generated sets of 200 and 500 points with 5, 10, and 20 dimensions. For the seven-decision tree, we generated 200 points in five dimensions. These "training" sets were used to construct the trees. To measure how well the trees generalized, we used the trees to classify 2000 additional randomly generated points. The additional points are called the "testing" set. This process was repeated five times to obtain average values for the optimal objective found, the training set error, and the testing set error. The random data was created by first generating the decisions in the tree. Each weight and threshold were uniformly generated between −1 and 1. Then points were randomly generated in the unit cube and classified using the generated tree. Since a starting point is required for FW, an initial tree was generated using the greedy MSMT decision tree algorithm [3]. MSMT used a linear program to recursively construct the decisions. The tree was then pruned to the appropriate number of decisions. This starting point was used for each of the algorithms we tested.

We compared the performance of EPTS, FW, HEPTS, and the nonlinear solver (MINOS) implemented within MINOS 5.4 [17] on these generated, separable data sets. The MINOS nonlinear solver is a reduced-gradient algorithm used in conjunction with a quasi-Newton method and an active set strategy for constraint handling. To evaluate their effectiveness as global optimization methods, we compared the optimal objective values obtained. For all of these problems the optimal objective value is known to be zero. Table 2 gives the average of the objective values of the three-decision trees over five trials. To see how well these methods performed as classification algorithms, we compared the training and testing set errors for each problem. The average of the training and testing errors for the three-decision trees is given in Table 3.

The computational results on the random data clearly show that EPTS performed better than FW, EPTS alone, or MINOS. In Table 2, HEPTS dramatically improved the objective values found

Table 2
Comparison of average objective values found by four methods: Frank–Wolfe, EPTS, HEPTS, and MINOS on randomly generated, separable three-decision problems

Dimension	Number of points	Method			
		FW	EPTS	HEPTS	MINOS
5	200	57.4	23.4	**0.0**	24.4
10	200	**0.0**	10.8	**0.0**	29.2
20	200	50.7	16.9	**0.0**	77.1
5	500	259.9	287.9	**73.6**	168.4
10	500	388.7	**165.1**	255.0	301.2
20	500	186.1	211.5	160.1	**69.2**

Table 3
Comparison of average classification errors found by four methods: Frank–Wolfe, EPTS, HEPTS, and MINOS on randomly generated, separable three-decision problems

Data	Number of points	Set	Method			
			FW	EPTS	HEPTS	MINOS
5 Dim	200	Train	3.4	0.9	**0.0**	1.3
		Test	5.4	4.3	**3.4**	5.8
10 Dim	200	Train	**0.0**	0.4	**0.0**	1.4
		Test	13.4	**12.9**	13.4	13.8
20 Dim	200	Train	2.8	0.7	**0.0**	3.9
		Test	31.6	**29.7**	31.4	30.1
5 Dim	500	Train	3.0	5.1	**1.3**	2.7
		Test	4.0	6.2	**2.8**	4.0
10 Dim	500	Train	6.5	**2.4**	3.0	4.2
		Test	10.1	**6.0**	7.7	8.3
20 Dim	500	Train	3.3	3.8	2.9	**0.8**
		Test	11.7	11.8	11.6	**10.7**

by FW except, in the cases where FW found the global optimal solutions. In every case but one, HEPTS also found better solutions than EPTS in terms of objective values. HEPTS also found better solutions than MINOS on five of the six problems. Since EPTS proved to be a relatively slow algorithm and HEPTS outperformed EPTS more often than not, only results for HEPTS are compared with FW and MINOS for the remaining data sets. For the seven-decision problem (Table 4) HEPTS outperformed FW and MINOS.

To assess the practicality of these methods for real-world problems we experimented with four data sets available via anonymous FTP from the Machine Learning Repository at the University of California at Irvine [16]. The data sets are: the BUPA Liver Disease data set (Liver); the PIMA Indians Diabetes data set (Diabetes), the Wisconsin Breast Cancer Database (Cancer) [26], and the Cleveland Heart Disease Database (Heart) [8]. We used five-fold cross validation. Each data set was divided into five parts. The decision tree was constructed using 4/5 of the data and tested on the remaining 1/5. This was repeated for each of the five parts and the results were averaged. Again, since HEPTS consistently outperformed EPTS, we only present results for HEPTS, FW, and MINOS in Tables 5 and 6.

In every case HEPTS found better objective values than either FW or MINOS. In the Diabetes data set, the difference in the objective values found by FW and HEPTS was less than 0.1 when averaged for the five-fold cross validation, thus the numbers reported in the table do not reflect that HEPTS did slightly better than FW. This improvement of the objective value by HEPTS did not

Table 4
Comparison of average classification errors found by three methods: Frank–Wolfe, HEPTS, and MINOS, on randomly generated, separable seven-decision problems consisting of 200 points in five dimensions

	Method		
	FW	HEPTS	MINOS
Objective Value	5445.8	**1467.6**	6327.0
Training error (%)	16.1	**9.5**	41.5
Testing error (%)	20.8	**18.5**	45.4

Table 5
Comparison of average objective values found by three methods on four real-world data sets: Frank–Wolfe, HEPTS, and MINOS

Data	Dimension	Total pts	Method		
			FW	HEPTS	MINOS
Heart	13	297	206.8	**205.1**	284.2
Cancer	9	682	64.9	**32.6**	94.0
Diabetes	8	768	770.4	**770.4**	1132.9
Liver	6	345	492.2	**486.7**	492.4

Table 6
Comparison of average classification errors found by three methods on real-world data sets: Frank–Wolfe, HEPTS, and MINOS

Data	Set	Method		
		FW	HEPTS	MINOS
Heart	Train	16.0	**15.7**	16.8
	Test	**17.8**	18.0	22.5
Cancer	Train	1.9	**1.1**	2.2
	Test	**4.0**	4.4	4.4
Diabetes	Train	22.1	**22.0**	24.2
	Test	**22.8**	23.2	26.6
Liver	Train	**28.0**	31.2	29.4
	Test	**32.2**	33.9	32.8

always lead to a reduction in the training and testing set errors. The increased errors were probably caused by the fact that a three-decision tree may not reflect the underlying structure of the data set. Fitting a poor model more accurately will not necessarily lead to better results. GTO must be used in the context of a larger algorithm to help select the appropriate structure of the tree. Also, alternate objective functions may produce better generalization. One benefit of tabu search methods is that we have the flexibility to use alternate objective functions. For example, we could use an objective that counts the number of points misclassified during the EPTS cycle of the HEPTS algorithm [5].

A comparison of the CPU times each algorithm took to run is given in Table 7. The times for the best solution found by HEPTS (HEPTS-BEST) and the total time HEPTS was allowed to run (HEPTS-TOTAL) are given in this table. The times for EPTS are not included since the algorithm is relatively slow compared to MINOS or HEPTS. While the FW algorithm is quite fast, it stops as soon as it reaches a local minimum. Similarly, the MINOS algorithm stops when it reaches a local minimum. MINOS was quite slow due to the large number of gradient evaluations necessary. The total run times of HEPTS are comparable to those of MINOS. The times reported are for IBM RS6000 model 590 workstations with 128 MB RAM.

7. Conclusions

We have developed an HEPTS algorithm which combines tabu search with gradient descent methods. The method is applicable to objective functions minimized subject to linear constraints where extreme point solutions are known to exist or are desired. We implemented two versions, one using UBPA and one with FW. We obtained excellent computational results using HEPTS on two practical problems: the QCP in robotics and the GTO problem in machine learning.

Table 7
Comparison of average CPU times (in seconds) of Frank–Wolfe, HEPTS, and MINOS on IBM RS6000 model 590 workstations with 128 MB RAM. Best and total CPU times are reported for HEPTS

Data	Method			
	FW	MINOS	HEPTS	
			BEST	TOTAL
5 Dim,200 pts	28	584	554	554
10 Dim,200 pts	53	969	52	52
20 Dim,200 pts	93	1133	1174	1174
5 Dim,500 pts	219	10 041	3027	6711
10 Dim,500 pts	281	14 410	8820	21 386
20 Dim,500 pts	786	26 413	3600	16 271
7 Decision	326	5219	9325	10 452
Heart	76	1420	1488	3743
Cancer	185	5231	6444	12 795
Diabetes	585	28 747	19 555	32 899
Liver	122	3305	1317	3612

We believe this is a very promising line of research. This work could be expanded in several directions. The descent method could be more tightly integrated into the tabu search. In this work, the descent method was used for intensification within the tabu search but no memory information was used or collected during the intensification phase. Both EPTS and the descent methods use the same neighborhood structure. The pivots chosen by the descent algorithm could be added into the tabu search memory. Tabu search could also be used to enhance the descent algorithm. During intensification, the descent method acts as a candidate list strategy – cheaply picking good possible moves using the linearized objective. Tabu search could be used to make the final move selection. There are many other applications and global optimization problems that could be addressed using HEPTS, for example concave minimization, complementarity problems, bilinear programming, multiplicative programming, parametric bilinear programming, mathematical programming problems with equilibrium constraints, or optimization problems in which a vertex of a polyhedron is desired.

Acknowledgements

This material is based on research supported by National Science Foundation Grant IRI-9409427. The authors wish to thank Fred Glover for suggesting the use of tabu search for the global tree optimization problem.

References

[1] R. Aboudi, K. Jörnsten, Tabu search for general zero-one integer programs using the pivot and complement heuristic, ORSA Journal on Computing 6 (1) (1994) 82–936.

[2] M. Bazaraa, H. Sherali, C. Shetty, Nonlinear Programming Theory and Algorithms, Wiley, New York, 1993.

[3] K.P. Bennett, Decision tree construction via linear programming, in: M. Evans (Ed.), Proceedings of the Fourth Midwest Artificial Intelligence and Cognitive Science Society Conference, Utica, Illinois, 1992, pp. 97–101.

[4] K.P. Bennett, Global tree optimization: A non-greedy decision tree algorithm, Computing Science and Statistics 26 (1994) 156–160.

[5] K.P. Bennett, J.A. Blue, Optimal decision trees, R.P.I. Math Report No. 214 (Revised), Rensselaer Polytechnic Institute, Troy, NY, 1996.

[6] K.P. Bennett, O.L. Mangasarian, Bilinear separation of two sets in n-space, Computational Optimization and Applications 2 (1993) 207–227.

[7] R. Cottle, J. Pang, R. Stone, The Linear Complementarity Problem, Academic Press, San Diego, CA, 1992.

[8] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, V. Froelicher, International application of a new probability algorithm for the diagnosis of coronary artery disease, American Journal of Cardiology 64 (1989) 304–310.

[9] M. Frank, P. Wolfe, An algorithm for quadratic programming, Naval Research Logistics Quarterly 3 (1956) 95–110.

[10] F. Glover, Tabu search fundamentals and uses, Technical report, School of Business, University of Colorado, Boulder, Colorado (1995).

[11] F. Glover, A. Løkketangen, Probabilistic tabu search for zero-one mixed integer programming problems, Manuscript, School of Business, University of Colorado (1994).

[12] R. Horst, P. Pardalos (Eds.), Global Optimization, Kluwer Academic Publishers, New York, 1995.

[13] A. Løkketangen, K. Jörnsten, S. Storøy, Tabu search within a pivot and complement framework, International Transactions of Operations Research 1 (3) (1994) 305–316.

[14] P.G. McKeown, A branch-and-bound algorithm for solving fixed charge problems, Naval Research Logistics 28 (1981) 607–617.

[15] N. Megiddo, On the complexity of polyhedral separability, Discrete and Computational Geometry 3 (1988) 325–337.

[16] P.M. Murphy, D.W. Aha, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, California (1992).

[17] B.A. Murtagh, M.A. Saunders, MINOS 5.4 user's guide, Technical Report SOL 83.20, Stanford University (1993).

[18] K.G. Murty, Linear Programming, Wiley, New York, 1983.

[19] J. Pang, J. Trinkle, G. Lo, A complementarity approach to a quasistatic multi-rigid-body contact problem, Computational Optimization and Applications 5 (1996) 139–154.

[20] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, CA, 1993.

[21] B. De Schutter, B. De Moor, The extended linear complementarity problem, Mathematical Programming 71 (1995) 289–325.

[22] M. Sun, A tabu search heuristic procedure for solving the transportation problem with exclusionary side constraints, in: Institute for studies in business vol. 18, no. 1, College of Business, The University of Texas at San Antonio, San Antonio, Texas, 1996.

[23] M. Sun, J.E. Aronson, P.G. McKeown, D. Drinka, A tabu search heuristic procedure for the fixed charge transportation problem, Manuscript, The University of Texas at San Antonio, Texas, 1996.

[24] M. Sun, P.G. McKeown, Tabu search applied to the general fixed charge problem, Annals of Operations Research 41 (1993) 405–420.

[25] J. Trinkle, D. Zeng, Planar quasistatic motion of a contacted rigid body, IEEE Transactions on Robotics and Automation 11 (1995) 229–246.

[26] W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, in: Proceedings of the National Academy of Sciences, USA, vol. 87, 1990, pp. 9193–9196.