ELSEVIER

Information Sciences 124 (2000) 241–272

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Improving the performance of weighted Lagrange-multiplier methods for nonlinear constrained optimization

Benjamin W. Wah [a], Tao Wang [a], Yi Shang [b,*], Zhe Wu [a]

[a] *Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*

[b] *Department of Computer Engineering and Computer Science, University of Missouri, Columbia, MO 65211, USA*

Received 2 December 1998; accepted 19 April 1999

Abstract

Nonlinear constrained optimization problems in discrete and continuous spaces are an important class of problems studied extensively in artificial intelligence and operations research. These problems can be solved by a Lagrange-multiplier method in continuous space and by an extended discrete Lagrange-multiplier method in discrete space. When constraints are satisfied, these methods rely on gradient descents in the objective space to find high-quality solutions. On the other hand, when constraints are violated, these methods rely on gradient ascents in the Lagrange-multiplier space in order to increase the penalties on unsatisfied constraints and to force the constraints into satisfaction. The balance between gradient descents and gradient ascents depends on the relative weights between the objective function and the constraints, which indirectly control the convergence speed and solution quality of the Lagrangian method. To improve convergence speed without degrading solution quality, we propose an algorithm to dynamically control the relative weights between the objective and the constraints. Starting from an initial weight, the algorithm automatically adjusts the weights based on the behavior of the search progress. With this strategy, we are able to eliminate

* Corresponding author. Fax: +1-573 882-8318.

E-mail addresses: wah@manip.crhc.uiuc.edu (B.W. Wah), wangtao@manip.crhc.uiuc.edu (T. Wang), yshang@risc1.ecn.missouri.eduhttp://manip.crhc.uiuc.edu (Y. Shang), zhewu@manip.crhc.uiuc.edu (Z. Wu).

0020-0255/00/$ - see front matter © 2000 Elsevier Science Inc. All rights reserved.
PII: S0020-0255(99)00081-X

divergence, reduce oscillation, and speed up convergence. We show improved convergence behavior of our proposed algorithm on both nonlinear continuous and discrete problems. © 2000 Elsevier Science Inc. All rights reserved.

Keywords: Adaptive weights; Lagrange-multiplier method; Multiplierless filter-bank design; Nonlinear constrained optimization; Nonlinear integer programming; Nonlinear programming; QMF filter-bank design

1. Introduction

Many applications in engineering, decision science and operations research are formulated as optimization problems. These applications include digital signal processing, structure optimization, engineering design, neural-network learning, computer-aided design for VLSI, and chemical control processing. High-quality solutions in these applications may have significant economical impacts, leading to lower implementation and maintenance costs while improving solution quality of outputs.

The *nonlinear constrained optimization problems* studied in this paper take the following form:

$$\begin{aligned} \text{Min} \quad & f(X), \\ \text{s.t.} \quad & g(X) \leqslant 0, \quad X = (x_1, x_2, \ldots, x_n), \\ & h(X) = 0, \end{aligned} \tag{1}$$

where X is a vector of real variables in continuous problems or a vector of discrete numbers in discrete problems, $f(X)$ is an objective function, $g(X) = [g_1(X), \ldots, g_k(X)]^{\mathrm{T}}$ is a set of k inequality constraints, and $h(X) = [h_1(X), \ldots, h_m(X)]^{\mathrm{T}}$ is a set of m equality constraints. Note that f, g, and h can be either continuous or discrete functions.

The problem defined in (1) can be solved by a large number of existing approaches [4], which are classified into local- and global-search. Local-search algorithms include gradient descent, Newton's method, conjugate-gradient method, and Lagrange-multiplier method. Starting from some initial point, they stop at a local minimum. Since the local minima found by local-search methods may be much worse than the global minimum for nonlinear problems, global-search methods have been studied to perform both global-exploration and local-refinement. The global-exploration component goes through the possible search space and identifies some promising points for regions that may have good solutions, whereas the local-refinement component uses these promising points as starting points and employs a local-search algorithm to find local minima.

In this paper, we focus on the Lagrange-multiplier method as a local-search method to find satisfiable solutions and show how its convergence speed can be improved. In Section 2, we describe the Lagrange-multiplier method in continuous space and show its extension in discrete space. In Section 3, we show that the convergence speed and solution quality can be affected by adjusting the weight of the objective function, and that it is difficult to select a proper static weight for each problem instance. In Section 4, we describe the algorithm to dynamically adjust the weight of the objective function and tailor it to continuous and discrete problems. We illustrate in Section 5 the application of the algorithm on continuous and discrete problems and show in Section 6 further experimental results. (All our experiments were run on 200 MHz Pentium Pro processors with the Linux operating system and *gcc* and *f77* compliers.) Finally, Section 7 concludes the paper.

2. Lagrange-multiplier methods

Lagrange-multiplier methods introduce Lagrange-multipliers to gradually resolve constraints iteratively. It is an exact method that optimizes the objective $f(X)$ to meet the Kuhn–Tucker conditions [4].

2.1. Lagrangian methods in the continuous space (CLM)

For continuous problems, we assume that in (1), every variable x_i $(i = 1, 2, \ldots, n)$ takes a value from R, and that $f(X), g(X)$ and $h(X)$, as well as their derivatives are continuous functions.

Since Lagrangian methods cannot directly deal with inequality constraints $g_i(X) \leqslant 0$ in general cases, we transform inequality constraints into equality constraints by adding slack variables $z_i(X)$, which results in $p_i(X) = g_i(X) + z_i^2(X) = 0$. The corresponding *Lagrangian function* and *augmented Lagrangian function* are defined as follows:

$$\mathscr{L}_c(X, \lambda, \mu) = f(X) + \lambda^T h(X) + \mu^T p(X), \tag{2}$$

$$L_c(X, \lambda, \mu) = f(X) + \lambda^T h(X) + \frac{d_1}{2}\|h(X)\|_2^2 + \mu^T p(X) + \frac{d_2}{2}\|p(X)\|_2^2, \tag{3}$$

where $\lambda = [\lambda_1, \ldots, \lambda_m]^T$ and $\mu = [\mu_1, \ldots, \mu_k]^T$ are two sets of Lagrange multipliers, and $p(X) = [p_1(X), \ldots, p_k(X)]^T$. We use the augmented Lagrangian function in this paper since it provides better numerical stability. After simplification [4], the augmented Lagrangian function becomes:

$$L_c(X, \lambda, \mu) = f(X) + \lambda^T h(X) + \frac{d_1}{2}\|h(X)\|_2^2 + \frac{1}{2d_2}\sum_{i=1}^{k}\left[\max^2(0, d_2\mu_i + g_i(X)) - \mu_i^2\right]. \tag{4}$$

According to classical optimization theory [4], all the (local and global) extrema of (4) that satisfy the constraints and that are regular points are roots of the following set of first-order necessary conditions:

$$\nabla_X L_c(X,\lambda,\mu)=0,\quad \nabla_\lambda L_c(X,\lambda,\mu)=0,\quad \nabla_\mu L_c(X,\lambda,\mu)=0. \tag{5}$$

These conditions are necessary to guarantee the (local) optimality of the solution to (2) and (3). [1]

The set of points satisfying the necessary conditions can be found by a *first-order search method* that can be expressed in a dynamic system of equations

$$\frac{\mathrm{d}}{\mathrm{d}t}X(t)=-\nabla_X L_c(X(t),\lambda(t),\mu(t)),\quad \frac{\mathrm{d}}{\mathrm{d}t}\lambda(t)=\nabla_\lambda L_c(X(t),\lambda(t),\mu(t)), \tag{6}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\mu(t)=\nabla_\mu L_c(X(t),\lambda(t),\mu(t)),$$

which perform descents in the original-variable space of X and ascents in the Lagrange-multiplier space of λ and μ. The dynamic system evolves over time t, and reaches a feasible local extremum when it stops at an *equilibrium point* where all gradients vanish. In this sense, first-order methods can be considered as *local-search methods* that perform gradient descents in the original-variable space and gradient ascents in the Lagrange-multiplier space to reach equilibrium.

2.2. Lagrangian methods in discrete space (DLM)

For discrete optimization problems, variable x_i $(i=1,2,\ldots,n)$ takes discrete values (e.g., integers). Little work has been done in applying Lagrangian methods to solve discrete constrained combinatorial optimization problems [2]. The difficulty in traditional Lagrangian methods lies in the lack of a differentiable continuous space to find an equilibrium point. In this section, we describe an extension of the Lagrange-multiplier method in discrete space [5].

For nonlinear discrete problems with inequality constraints (1), where X is a vector of discrete variables, we first transform inequality constraint $g_i(X)\leqslant 0$ into an equality constraint $\max(g_i(X),0)=0$. This transformation does not use a slack variable as in the continuous case because searches in the discrete Lagrangian space does not require the existence of gradients when $g_i(X)=0$.

After transforming inequality constraints into equality constraints, the resulting optimization problem can be considered as one with equality

[1] There are second-order conditions to guarantee that the extremum found is a local minimum [4].

constraints only. For simplicity in representation, we show the corresponding Lagrangian funtion with only equality constrains

$$L_{\mathrm{d}}(X,\lambda,\mu) = f(X) + \lambda^{\mathrm{T}} h(X). \tag{7}$$

Given the discrete Lagrangian function, we define the *discrete gradient* of the Lagrangian function $L_{\mathrm{d}}(X,\lambda,\mu)$ in the original-variable subspace of X as follows: [2]

$$\nabla_X L_{\mathrm{d}}(X,\lambda,\mu) = \delta_X = Y \ominus X, \tag{8}$$

where $\ominus$ is an operator for changing point X in discrete space to one of its "user-defined" neighborhood points $N(X)$, an example of $\ominus$ is the exclusive-OR operator. Intuitively, δ_X is a vector pointing from X to Y, the point with the minimum value of L_{d} among all neighboring points of X, including X itself. That is,

$$L_{\mathrm{d}}(Y,\lambda,\mu) = \min_{\substack{X' \in N(X) \\ \cup \{X\}}} L_{\mathrm{d}}(X',\lambda,\mu).$$

When X itself has the minimum L_{d}, then $\delta_X = \vec{0}$.

Having defined $\nabla_X L_{\mathrm{d}}(X,\lambda,\mu)$ in the X space, we seek discrete equilibrium points similar to those of continuous problems. The iterative equations are as follows:

$$\begin{aligned} X(k+1) &= X(k) \ominus \nabla_X L_{\mathrm{d}}(X(k),\lambda(k),\mu(k)), \\ \lambda(k+1) &= \lambda(k) + c_1 h(X(k)), \end{aligned} \tag{9}$$

where c_1 is a positive real number controlling how fast the Lagrange-multipliers change. These points are actually saddle points in discrete space that satisfy the following condition:

$$L(X^*,\lambda,\mu) \leqslant L(X^*,\lambda^*,\mu^*) \leqslant L(X,\lambda^*,\mu^*) \tag{10}$$

for all (X^*,λ,μ) and all (X,λ^*,μ^*) sufficiently close to (X^*,λ^*,μ^*). For brevity, the proofs showing the correctness of DLM and (9) are omitted here [5].

3. Convergence speed of Lagrangian methods

Lagrangian methods rely on two counteracting forces to resolve constraints and find high-quality solutions. When constraints are satisfied, Lagrangian

[2] We do not need to define gradients in Lagrange-multiplier space because descents in that space is done differently, as shown in (9).

methods rely on gradient descents in the objective space to find high-quality solutions. On the other hand, when constraints are violated, they rely on gradient ascents in the Lagrange-multiplier space in order to increase the penalties on the unsatisfied constraints and to force the constraints into satisfaction. The balance between gradient descents and gradient ascents depends on the relative magnitudes of the Lagrange multipliers λ and μ with respect to the objective value, which play a role in balancing the objective $f(X)$ and constraints $h(X)$ and $g(X)$ and in controlling indirectly the convergence speed and solution quality of the Lagrangian method. At an equilibrium point, the forces due to descent and ascent reach a balance through appropriate Lagrange-multiplier values.

We show in this section that the convergence speed and/or the solution quality can be affected by an additional weight w in the objective part of the Lagrangian function. After abstracting weights d_1 and d_2 in (4) into w, we have a new Lagrangian function in continuous space as follows:

$$L_{c'}(X,\lambda,\mu) = wf(X) + \lambda^{\mathrm{T}} h(X) + \|h(X)\|_2^2 + \sum_{i=1}^{k} \Big[\max^2(0, \mu_i + g_i(X)) - \mu_i^2 \Big], \tag{11}$$

where $w > 0$ is a static weight on the objective. When $w = 1$, $L_{c'}(X,\lambda,\mu) = L_c(X,\lambda,\mu)$, which is the original Lagrangian function.

The corresponding weighted Lagrangian function in discrete space is as follows:

$$L_{d'}(X,\lambda,\mu) = wf(X) + \lambda^{\mathrm{T}} h(X). \tag{12}$$

In general, adding a weight to the objective changes the Lagrangian function, which in turn may cause the dynamic system to settle at a different equilibrium point with different solution quality. This is especially true when the equilibrium point is on the boundary of a feasible region. In this section, we show that the solution quality and convergence time can be influenced greatly by the choice of the initial static weight.

3.1. Nonlinear continuous optimization

Starting from an initial point $(X(0), \lambda(0), \mu(0))$, the dynamic system to find equilibrium points is based on (6) in which L_c is replaced by $L_{c'}$

$$\frac{\mathrm{d}}{\mathrm{d}t} X(t) = -\nabla_X L_{c'}(X(t),\lambda(t),\mu(t)), \quad \frac{\mathrm{d}}{\mathrm{d}t}\lambda(t) = \nabla_\lambda L_{c'}(X(t),\lambda(t),\mu(t)),$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\mu(t) = \nabla_\mu L_{c'}(X(t),\lambda(t),\mu(t)). \tag{13}$$

We solve this dynamic system using an ordinary differential equation solver LSODE [3] and observe a search trajectory $(X(t), \lambda(t), \mu(t))$. When an equilibrium point is on the boundary of the feasible region (which is true for most problems studied), the dynamic equation approaches it from both the inside and outside of the feasible region. We observe four possible behaviors of the trajectory:

- The trajectory converges without oscillations.
- The trajectory gradually reduces its oscillations and eventually converges.
- The trajectory oscillates within some range but never converges.
- The magnitude of oscillations increases, and the trajectory eventually diverges.

Obviously, the first two cases are desirable, and the other two are not. Moreover, we would like to reduce the amount of oscillations and improve the convergence time by proper control of w.

To illustrate the first behavior, consider the following simple example.

$$\begin{aligned} \text{Min.} \quad & x^2, \\ \text{s.t.} \quad & x \leqslant -10. \end{aligned} \tag{14}$$

When we start from the initial point $X(t=0) = -20$ with $\mu(t=0) = 0$ using $w = 1$, we obtain a trajectory without oscillations, as shown in Fig. 1.

To illustrate the last three behaviors (divergence, oscillations without convergence, and reduction of oscillations until convergence), consider [1, Problem 2.3]. This problem is used as a running example in this paper and defined as follows:

$$\begin{aligned} \text{Min.} \quad & 5x_2 + 5x_3 + 5x_4 + 5x_5 - x_6 - x_7 - x_8 - x_9 - x_{10} - 8x_{11} - 8x_{12} \\ & \quad - 8x_{13} - x_{14} - 5x_2^2 - 5x_3^2 - 5x_4^2 - 5x_5^2, \\ \text{s.t.} \quad & 2x_2 + 2x_3 + 8x_{11} + 8x_{12} \leqslant 10.0, \\ & 0 \leqslant x_i \leqslant 1.0, \quad i = 2, 3, \ldots, 14, \\ & 2x_2 + 2x_4 + 8x_{11} + 8x_{13} \leqslant 10.0, \\ & 2x_3 + 2x_4 + 8x_{12} + 8x_{13} \leqslant 10.0, \\ & 8x_{11} - 8x_2 \leqslant 0.0, \quad 8x_{12} - 8x_3 \leqslant 0.0, \\ & 8x_{13} - 8x_4 \leqslant 0.0, \quad -2x_5 - x6 + 8x_{11} \leqslant 0.0, \\ & -2x_7 - x8 + 8x_{12} \leqslant 0.0, \quad -2x_9 - x10 + 8x_{13} \leqslant 0.0, \end{aligned} \tag{15}$$

where the middle point of the search space is $x_i = 0.5,\ i = 2, 3, \ldots, 14$.

[3] LSODE is a solver for first-order ordinary differential equations, a public-domain package available from http://www.netlib.org.

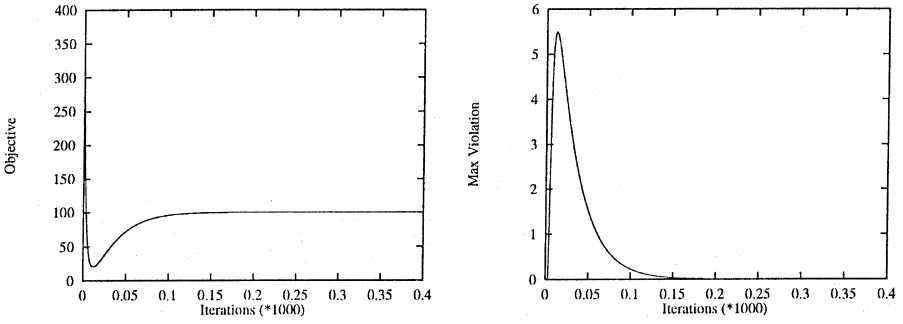


Fig. 1. The objective function and maximum violation converge without oscillations.

We set the initial point at $t = 0$ as follows: $X(t = 0)$ is set at the middle of the search space, and $\lambda(t = 0) = \mu(t = 0) = 0$. The total time used for LSODE is $t_{\max} = 10^5$, which is divided into small units of $\Delta t = 1.0$, resulting in a maximum of 10^5 iterations ($= t_{\max}/\Delta t$). The stopping condition for (13) is when

$$\|\mathrm{d}X(t)/\mathrm{d}t\|^2 + \|\mathrm{d}\lambda(t)/\mathrm{d}t\|^2 + \|\mathrm{d}\mu(t)/\mathrm{d}t\|^2 \leqslant \delta = 10^{-25}. \tag{16}$$

The dynamic system stops when it converges or when it reaches the maximum number of iterations.

When $w = 1$, (13) diverges quickly into infinity, meaning that the original Lagrangian method governed by (6) will diverge. If we scale the objective by 10 (i.e., $w = 1/10$), then the objective value $f(X(t))$ oscillates within the range -17 and -10, while the maximum violation $v_{\max}(t)$ is between 0 and 0.4, as shown in Fig. 2. Here, $v_{\max}(t)$ at time t is defined as

$$v_{\max}(t) = \max_{\substack{1 \leqslant i \leqslant m, \\ 1 \leqslant j \leqslant k}} \left\{ |h_i(X(t))|, \max\left[0, g_j(X(t))\right] \right\}. \tag{17}$$

If we further reduce w to $1/15$, then the oscillations subside, and the trajectory eventually converges (see Fig. 3).

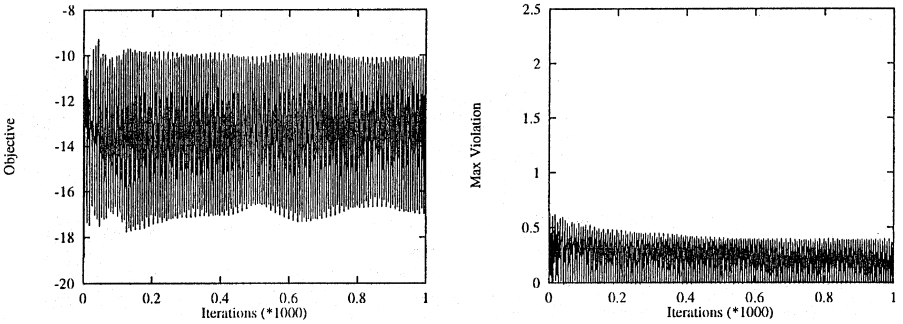


Fig. 2. The objective function and maximum violation oscillate.

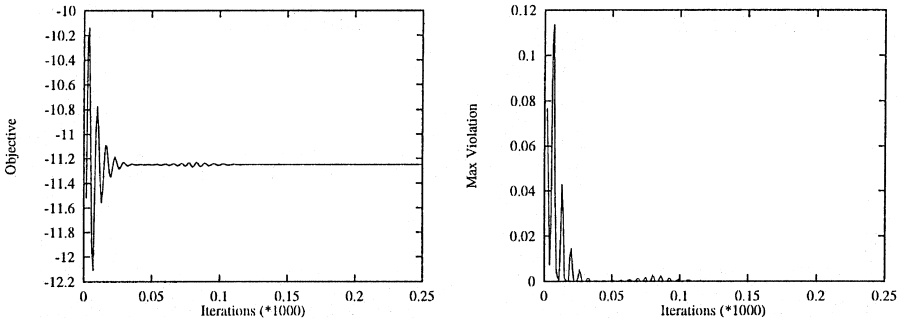


Fig. 3. The objective function and maximum violation converge after oscillations subsided.

Intuitively, the occurrence of oscillations can be explained as follows. Suppose we start from an infeasible point initially ($t = 0$) where an inequality constraint $g_i(X(t = 0))$ is violated, i.e., $g_i(X(t = 0)) > 0$. As the search progresses, the corresponding $\mu_i(t)$ increases and pushes the trajectory towards a feasible region. At some time t, the inequality constraint $g_i(X(t)) \leqslant 0$ is satisfied for the current point $X(t)$. At this point, $\mathrm{d}\mu_i(t)/\mathrm{d}t = \max(0, g_i(X) + \mu_i) - \mu_i$, and is negative when $g_i(X) < 0$. Hence, the trajectory decelerates but continues to move into the feasible region even when the corresponding constraint, $g_i(X(t))$, is satisfied. The movement of the trajectory inside the feasible region eventually stops because the local minimum is on the boundary, and the corresponding force due to descents in the objective space pushes the trajectory outside the feasible region. Likewise, when the trajectory is outside the feasible region, a force due to the constraints pushes the trajectory inside the feasible region. If these two forces are not well balanced, the search may diverge or oscillate without convergence.

To understand the effect of w on convergence, we randomly generated 20 starting points $X(t = 0)$ uniformly distributed in the search space of Problem 2.3 with all Lagrange-multipliers $\lambda(t = 0) = \mu(t = 0) = 0$. For each starting point, we tried different static weight w. In addition, for cases that converged, we report the average convergence time and the solution quality.

Table 1 shows the results. When $w \leqslant 1/3$, the trajectory diverges for every starting point. When w is between $1/5$ and $1/9$, the trajectories always oscillate without convergence. For $w = 1/11$, half of the trajectories finally converge, and the remaining oscillate. When $w \leqslant 1/15$, all the trajectories converge. For cases that converge, the number of oscillations gradually reduces to zero as w is reduced. Hence, it is possible to eliminate all the oscillations at the expense of longer convergence time (e.g., 2875.1 s). The convergence behavior when $w = 1/30$ seems to be good because both the number of oscillations and the convergence time are small.

Table 1
Effect of w on convergence time and solution quality for [1, Problem 2.3][a]

Weight w	# Convergence behavior	Average time (s)	Best solution	Average solution
1	20/0/0	–	–	–
1/3	20/0/0	–	–	–
1/5	0/20/0	–	–	–
1/7	0/20/0	–	–	–
1/9	0/20/0	–	–	–
1/11	0/10/10	–	–	–
1/15	0/0/20	2.88	−15.0	−12.67
1/30	0/0/20	1.72	1−5.0	−12.67
1/60	0/0/20	1.76	−15.0	−12.67
1/150	0/0/20	1.87	−15.0	−12.67
1/500	0/0/20	1.98	−15.0	−12.67
1/1000	0/0/20	2.20	−15.0	−12.67
1/10000	0/0/20	3.90	−15.0	−12.67
1/100000	0/0/20	2875.1	1−5.0	−12.67

[a] The convergence behavior is measured by three integers: the number of diverged trajectories, the number of oscillating trajectories, and the number of converged trajectories.

3.2. Nonlinear discrete optimization

For discrete problems, the iterative equation to seek discrete equilibrium points of (12) is given by (9) where L_{d} is replaced by $L_{\mathrm{d}'}$.

$$X(k+1) = X(k) \ominus \nabla_X L_{\mathrm{d}'}(X(k), \lambda(k), \mu(k)), \qquad (18)$$
$$\lambda(k+1) = \lambda(k) + c_1 h(X(k)).$$

Although discrete problems have discrete variables and require decrete gradients in DLM, their convergence behaviors are very similar to those of continuous problems and exhibit the same four behaviors discussed in Section 3.1. For efficiency reasons, we implement our discrete gradient defined in (8) with L_{d} replaced by $L_{\mathrm{d}'}$ using hill climbing (by finding the first direction rather than the best direction that improves the value of the Lagrangian function).

To illustrate the different behaviors, we generate a discrete problem from a continuous version [1, Problem 2.6]. The original continuous problem has variables defined in $[0, 1]$. We multiply the range by 1000 and restrict the values of variables to be integers in $[0, 1000]$. Note that the discretized problem may not have the same optimal solutions as the original continuous problem.

Fig. 4 shows the first behavior when the trajectory converges without oscillations ($w = 10^{-5}$). The trajectory starts from an infeasible point and moves into a feasible region, while the maximum violation decreases. Fig. 5 shows the third behavior when the trajectory oscillates within some range without convergence ($w = 10^3$). Finally, when w is 10^5 or larger, the trajectory diverges. The second behavior in which oscillations will subside gradually may also occur but cannot be demonstrated by this problem.

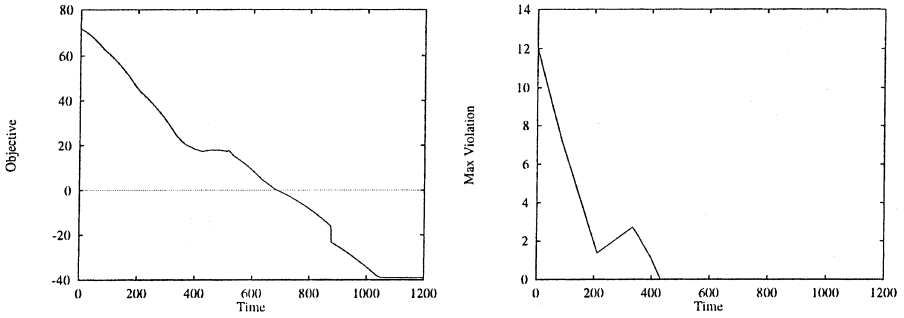


Fig. 4. The objective and maximum violation converge without oscillations when $w = 10^{-5}$.

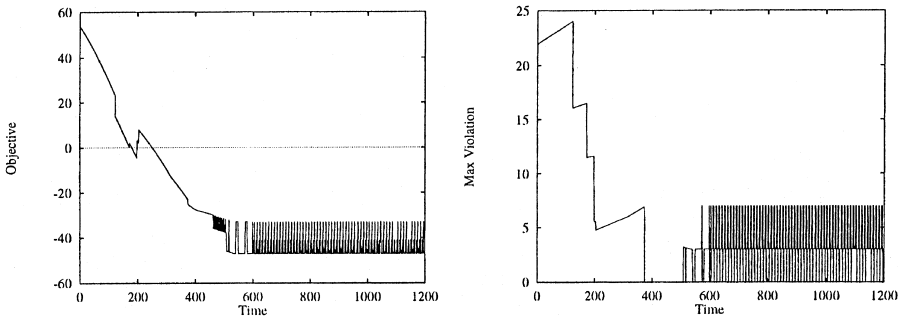


Fig. 5. The objective and maximum violation oscillate when $w = 10^{3}$.

Table 2 shows the detailed results of the discretized Problem 2.6 [1] under different static weights. For each weight, we ran DLM from 20 randomly generated starting points. When w is small, all the trajectories converge but may have worse average solution quality. As w is increased, the number of converged trajectories drop gradually, and eventually none converges when w is 10^4 or larger. Note that the average solution quality improves as w is increased to 100 and 1000.

The examples in this and the last sections illustrate that the convergence can be affected dramatically by the choice of w. Moreover, the best choice is problem-instance dependent and cannot be selected a priori. Hence, we propose next a scheme to adapt w dynamically so that the convergence behavior is robust and predictable, irrespective of the choice of the static w.

4. Dynamic weight adaptation

In this section, we propose a strategy to adapt weight w based on the behavior of the continuous version (13) and the discrete version (18) of the

Table 2
Effect of w on convergence time and solution quality from 20 randomly generated starting points for the discretized Problem 2.6 defined in [1]

Weight w	# Converged trajectories	Average time (s)	Best solution	Average solution
100000	0	–	–	–
10000	0	–	–	–
1000	0	–	–	–
100	8	6310	−39.0	−34.8
10	15	2777	−39.0	−27.8
1	18	1757	−39.0	−27.4
0.1	20	1267	−39.0	−28.3
0.01	20	1254	−39.0	−29.4
0.001	20	1249	−39.0	−29.3
0.0001	20	1247	−39.0	−29.4
0.00001	20	1099	−39.0	−24.7

dynamic system in order to obtain high-quality solutions with short convergence time. In general, before a trajectory reaches an equilibrium point, changing the weight of the objective may speed up or delay convergence. Further, when the trajectory is at an equilibrium point, changing the weight of the objective may bring the trajectory out of the equilibrium point. In this paper, we exploit the first property by designing weight-adaptation algorithms so that we can speed up convergence without affecting solution quality. We plan to exploit the second property to bring a trajectory out from an equilibrium point in our future work. In this section, we present a general strategy, followed by the refinements of the algorithm for continuous and discrete problems.

4.1. General weight-adaptation strategy

Fig. 6 outlines the algorithm. Its basic idea is to first estimate the initial weight $w(t=0)$ (Step 1), measure the performance metrics of the search trajectory $(X(t), \lambda(t), \mu(t))$ periodically, and adapt $w(t)$ to improve convergence time or solution quality.

Let $t_{\max}$ be the total (logical) time for the search, and $t_{\max}$ be divided into small units of time Δt so that the maximum number of iterations is $t_{\max}/\Delta t$. Further, assume a stopping condition if the search were to stop before $t_{\max}$ (Step 4). Given a starting point $X(t=0)$, we set the initial values of the Lagrange multipliers to be zero, i.e., $\lambda(t=0) = \mu(t=0) = 0$. Let (X_i, λ_i, μ_i) be the point of the ith iteration, and $v_{\max}(i)$ be its maximum violation defined in (17).

To monitor the progress of the search trajectory, we divide time into non-overlapping windows of size N_u iterations each (Step 2). In each window, we compute some metrics to measure the progress of the search relative to that of

1. Set control parameters:
   time interval $\triangle t$
   initial weight $w(t = 0)$
   maximum number of iterations i_{max}
2. Set window size $N_u = 100$ or 10
3. $j := 1$ /* j is the iteration number */
4. **while** $j \leq i_{max}$ **and** stopping condition is not satisfied **do**
5. advance search trajectory by $\triangle t$ to get to (X_j, λ_j, μ_j)
6. **if** trajectory diverges **then**
   reduce w; restart the algorithm by going to Step 2
   end if
7. record useful information for calculating performance metrics
8. **if** $((j \ mod \ N_u) == 0)$ **then**
   /* Test whether w should be modified at the end of a window */
9. compute performance metrics based on data collected
10. change w when the conditions defined in Table 3 are satisfied
    end if
11. **end while**

Fig. 6. Framework of a new dynamic weight-adaption algorithm.

previous windows. For the uth window ($u = 1, 2, \ldots$), we calculate the average (or median) of $v_{\max}(i)$ over all the iterations in the window,

$$\bar{v}_u = \frac{1}{N_u} \sum_{j=(u-1)N_u+1}^{uN_u} v_{\max}(j) \quad \text{or} \quad \bar{v}_u = \underset{(u-1)N_u+1 \leqslant j \leqslant uN_u}{\text{median}} \{v_{\max}(j)\}, \tag{19}$$

and the average (or median) of the objective $f_i(X)$.

$$\bar{f}_u = \frac{1}{N_u} \sum_{j=(u-1)N_u+1}^{uN_u} f_j(X) \quad \text{or} \quad \bar{f}_u = \underset{(u-1)N_u+1 \leqslant j \leqslant uN_u}{\text{median}} \{f_j(X)\}. \tag{20}$$

During the search, we apply an algorithm to solve the dynamic system (13) and (18), and advance the trajectory by time interval Δt in each iteration in order to arrive at point (X_j, λ_j, μ_j) (Step 5).

At this point, we test whether the trajectory diverges or not (Step 6). Divergence happens when the maximum violation $v_{\max}(j)$ is larger than an extremely large value (e.g., 10^{20}). If it happens, we reduce w by a large amount, say $w \leftarrow w/10$, and restart the algorithm. In each iteration, we also record some statistics, such as $v_{\max}(j)$ and $f_j(X)$, that will be used to calculate the performance metrics for each window (Step 7).

At the end of each window or every N_u iterations (Step 8), we decide whether to update w based on the performance metrics (19) and (20) (Step 9). In our current implementation, we use the averages (or medians) of maximum violation $v_{\max}(i)$ and objective $f_j(X)$. In general, other application-specific metrics can be used, such as the number of oscillations of the trajectory in nonlinear continuous problems. Based on these measurements, we adjust w accordingly (Step 10).

In the next section, we discuss the specific weight-adaptation algorithm in Step 10 for continuous and discrete problems.

4.2. Dynamic weight adaptation for continuous and discrete problems

To understand how weights should be updated in Step 10, we examine all the possible behaviors of the resulting search trajectory in successive windows. We have identified four possible cases.

First, the trajectory does not stay within a feasible region, but goes from one feasible region to another through an infeasible region. During this time, the maximum violation $v_{\max}(i)$ is zero when the trajectory is in the first feasible region, increased when it travels from the feasible region to an infeasible region, and decreased when going from the infeasible region to the second feasible region. No oscillations will be observed because oscillations normally occur around an equilibrium point in one feasible region, as explained in Section 3. In this case, no change of w is required.

Second, the search trajectory oscillates around an equilibrium point of a feasible region. This can be detected when the number of oscillations in each window is larger than some threshold. Figs. 2 and 3 show two typical types of oscillations. To determine whether the oscillations will subside eventually, we compute $\bar{v}_u - \bar{v}_{u+1}$, the difference of the average values of the maximum violation $v_{\max}(i)$, for two successive windows u and $u+1$. If the difference is not reduced reasonably, then we assume that the trajectory does not converge, and decrease w accordingly.

Third, the search trajectory moves very slowly within a feasible region. This happens when w is very small, and the constraints dominate the search process. As a result, the objective value is improved very slowly and may eventually converge to a poor value. This situation can be identified when the trajectory remains within a feasible region in two successive windows, and there is little improvement in the objective. Obviously, we need to increase w in order to speed up the improvement of the objective. If the objective remains unchanged, then the trajectory has converged, and no further modification of w is necessary.

Finally, the trajectory does not oscillate when it is started from a point in a feasible region, but rather goes outside the feasible region and then converges to a point on the boundary of the feasible region (see Fig. 1 for example). In

this case, a large w on the objective makes it more difficult to satisfy the constraints, causing the trajectory to move slowly back to the feasible region. At this time, it is desirable to reduce w to accelerate convergence. This situation happens frequently when we solve the QMF filter-bank design problem (to be discussed in Section 5.2), and appropriate decrease of w will greatly shorten the convergence time.

Given the four convergence behaviors, Table 3 shows a comprehensive list of conditions to change w for both continuous and discrete problems. Scaling factors $0 < \alpha_0,\ \alpha_1 < 1$ represent how fast w is updated. Because we use numerical methods to solve the dynamic system defined in (13), a trajectory in window u is said to satisfy all the constraints when $v_u < \delta$, where δ is related to the convergence condition and the required precision. Parameters $0 < \beta_0$, $\beta_1 < 1$ control, respectively, the degrees of improvement over the objective and the reduction of the maximum violation. Note that when comparing values between two successive windows $u-1$ and u, both must use the same weight w; otherwise, the comparison is not meaningful because the terrain may be totally different. Hence, after adapting w, we should wait at least two windows before changing it again.

Weight w should be increased when we observe the third convergence behavior. At this time, the trajectory is within a feasible region, and the objective is improved in successive windows (Condition a_1 in Table 3). Further, the improvement of the objective in the feasible region is not fast enough and is below an upper bound (Condition a_2).

Weight w should be decreased when we observe the second convergence behavior (the trajectory oscillating around an equilibrium point) or the fourth convergence behavior (the trajectory moving slowly back to the feasible region). In this case, the trajectory is either oscillating or not oscillating (Condition b_3), is not in a feasible region (Condition b_1), and the trend of the maximum violation does not decrease (Condition b_2).

Table 3
Conjunctive conditions under which weights will be changed in Step 10 of Fig. 6, given performance measurements in the uth window: $\bar{v}_u, \bar{f}_u$, and NO_u (number of oscillations) and application-specific constants $\alpha_0, \alpha_1, \beta_0, \beta_1, \delta$, and ϵ

ID	Conditions to increase w to w/α_0	ID	Condition to decrease w to $w \times \alpha_1$
α_1	$\bar{v}_{u-1}\bar{v}_u < \delta$	b_1	$\bar{v}_u \geqslant \delta$
α_2	$\beta_0 \mid \bar{f}_{u-1} \mid > \bar{f}_{u-1} - \bar{f}_u > \beta_1 \mid \bar{f}_{u-1} \mid$	b_2	$\bar{v} + u - 1 - \bar{v}_u \leqslant \beta_0 \bar{v}_{u-1}$
		b_3	$\begin{cases} \mathrm{NO}_u < \epsilon & \text{(for continuopus QMF design problems)} \\ \mathrm{NO}_u \geqslant \epsilon & \text{(for other continuous and discrete problems)} \end{cases}$

5. Illustrations of the dynamic weight-adaptation strategy

In this section, we illustrate our weight-adaptation algorithm in solving both nonlinear continuous and discrete optimization problems. The continuous problems include a set of benchmark problems [1,6] and the design of QMF filter banks [7], the discrete problems include nonlinear integer programming problems [1] and the design of multiplierless QMF filter banks [8]. (Although we have studied these application problems before, we did not apply weight adaptation there.) These problems cover cases in which variables can be continuous or discrete, and the objectives and constraints can be in closed form or have to be evaluated numerically.

5.1. Nonlinear continuous benchmark problems

In these problems [1], all the variables are real, and the objectives and constraints, as well as their derivatives, are continuous closed-form functions. After some experimentation, we pick time unit $\Delta t = 1$ in LSODE, and set the window size to be $N_u = 100$.

Recall in Fig. 2 that the trajectory oscillates when $w = 1/10$ for [1, Problem 2.3]. Fig. 7 shows the resulting trajectory and the maximum violation when the dynamic weight-adaptation algorithm is applied on the same problem. We started with $w(t = 0) = 1/5, \alpha_0 = \alpha_1 = 1/2, \delta = 10^{-8}, \beta_0 = 10^{-2}$, and $\beta_1 = 10^{-3}$. In the first window (first $N_u = 100$ iterations), the average $\bar{v}_1 = 4.11$, which increases slightly to $\bar{v}_2 = 4.2$ in the second window. In addition, $\bar{v}_2 \geqslant \delta$ and $NO_2 \geqslant 5$. According to the conditions in Table 3, w is updated to $1/10$. This change in w leads to significant reduction in the maximum violation in the third window. Weight w remains unchanged between the third and the fifth windows. At the end of the fifth window, the maximum violation changes very little. Hence, the algorithm reduces w to $1/20$, causing the trajectory to converge quickly.

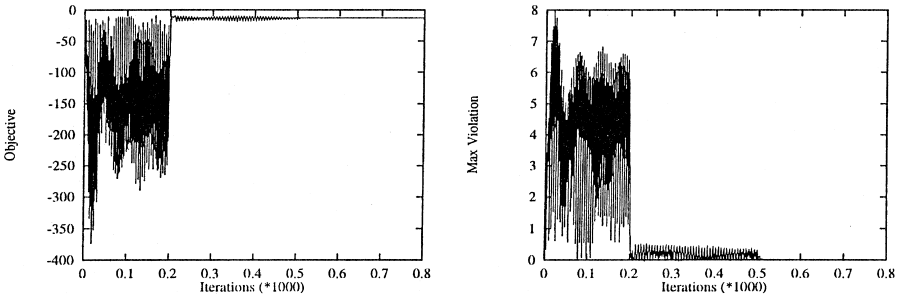


Fig. 7. The objective function and maximum violation first oscillate and then converge using dynamic weight-adaptation.

To illustrate the improvement due to dynamic weight-adaptation, we apply the algorithm on the same problem studied in Table 1 from 20 randomly generated starting points. Table 4 shows the results, indicating the convergence of the search trajectory for each initial weight w. For cases that converge using static weights, dynamic weight adaptation can reduce the convergence time, especially when the choice of the initial weight is poor. For instance, when the initial weight is $1/100000$, the search converges with the final weight in the range $[1/6250, 1/781.25]$ and an average convergence time of 3.85 s, much less than the 2875.1 sec. required before. If the initial weight chosen is good (e.g. $w = 1/60$), then the algorithm is able to maintain the original weight during the search and gets good convergence behavior. This shows the stability and robustness of our dynamic weight-adaptation algorithm, irrespective of the choice of the initial weight.

5.2. Design of QMF filter banks as nonlinear continuous optimization

In the second set of experiments, we consider the optimization of QMF filter banks. Filter banks are important components in digital signal processing and have been applied in modems, data transmission, digital audio broadcasting, speech and audio coding, and image and video coding. Without loss of generality, we choose to study QMF filter banks because they are amongst the simplest filter banks with defined benchmarks. They are also different from benchmarks studied in the last section because some constraints in the design problem are not in closed forms and need to be evaluated numerically.

Two-band QMF filter banks decompose input signals into two frequency subbands. Due to the particular way of selecting filter pairs, the design of a

Table 4
Effects of weight adaption on convergence time and solution quality for Problem 2.3 in [1]

Starting weight w	Final weight range $[w_1, w_2]$	Average convergence time (s)	Best solution	Average solution
1	1/20	19.14	−15.0	13.55
1/3	1/15	10.70	−15.0	−12.67
1/5	1/20	43.36	−15.0	−12.67
1/7	[1/28, 1/14]	27.43	−15.0	−12.67
1/9	1/18	19.97	−15.0	−12.67
1/11	1/22	4.22	−15.0	−12.67
1/15	1/15	2.88	−15.0	−12.67
1/30	1/30	1.72	−15.0	−12.67
1/60	1/60	1.76	−15.0	−12.67
1/150	1/150	1.87	−15.0	−12.67
1/5000	[1/500, 1/1000]	1.95	−15.0	−12.67
1/1000	[1/1000, 1/500]	2.15	−15.0	−12.67
1/10000	[1/5000, 1/625]	3.34	−15.0	−12.67
1/100000	[1/6250, 1/781.25]	3.85	−15.0	−12.67

two-band QMF filter bank becomes the design of only one prototype filter X. This design problem is a multi-objective nonlinear optimization problem, whose objectives consist of performance metrics of both the overall filter bank and the single prototype low-pass filter. The performance metrics of the overall filter-bank include amplitude distortion $E_r(X)$, aliasing distortion $E_a(X)$, and phase distortion $E_f(X)$, whereas the performance metrics of the single low-pass filter include stopband ripple $\delta_s(X)$, passband ripple $\delta_p(X)$, transition bandwidth $T_t(X)$, stopband energy $E_s(X)$, and passband flatness $E_p(X)$. A two-band QMF filter bank with a symmetric prototype lowpass filter has the nice properties of linear phase and no aliasing distortion.

In designing a QMF filter bank, we formulate it as a constrained optimization problem

$$\begin{aligned} \min \quad & E_r(X)/\theta_{E_r}, \\ \text{s.t.} \quad & E_p(X)/\theta_{E_p} \leqslant 1, \quad E_s(X)/\theta_{E_s} \leqslant 1, \\ & \delta_p(X)/\theta_{\delta_p} \leqslant 1, \quad \delta_s(X)/\theta_{\delta_s} \leqslant 1, \\ & T_t(X)/\theta_{T_t} \leqslant 1, \end{aligned} \tag{21}$$

where θ_{E_r}, θ_{E_p}, θ_{E_s}, θ_{δ_p}, θ_{δ_s}, and θ_{T_t} are performance values of the baseline design. Reconstruction error $E_r(X)$ is the objective to be minimized, and all other metrics of a single filter are used as constraints. This formulation allows us to improve on the best existing design (such as designs reported by Johnston [3]) with respect to all performance metrics. Existing methods generally optimize a subset of the performance metrics in constrained form or a weighted sum of the metrics.

The constrained optimization in (21) has nonlinear objective and constraints. Some constraints, such as stopband and passband ripples and transition bandwidth, do not have closed-form formulas. In our experiments, we use numerical methods (Newton's method) to evaluate these function values, and use finite-difference methods to approximate the derivatives and gradients. Errors introduced in function evaluation force LSODE to choose very small step sizes, e.g., 10^{-5}, in order to keep the results within a certain error tolerance.

Depending on the weights between the objective and the constraints, the convergence speeds of Lagrangian methods vary significantly in solving a QMF design problem, ranging from minutes to hours, even days. There is no good method to select the appropriate weights for a given problem. Our proposed dynamic weight-adaptation algorithm adjusts the weights between the objective and the constraints dynamically based on the search profile. It avoids the need to select appropriate weights in the beginning, and achieves faster and robust convergence than using static weights.

We use the 48e QMF filter-bank design problem [3] to illustrate the improvement of our dynamic weight-adaptation method as compared to that of

using static weights. We use Johnston's solution as our starting point. This point is feasible as we use its performance measures as our constraints. However, it is not a local minimum of the objective function. To avoid the difficulty in choosing the initial weight, we choose the starting weight $w(t=0)$ using a two-step method.

1. Set the initial weight based on the maximum-gradient component of the starting point and the number of variables, $w(t=0) = 1/(n \max_{1 \leqslant i \leqslant n} \{(\mathrm{d}L_c/\mathrm{d}x_i)(X(t=0),\ \lambda = 0,\ \mu = 0)\})$.
2. Perform the Lagrangian search for a small amount of time, e.g., $\delta_t = 10^{-5}$. Adjust the weight $w(t=0)$ based on the decrement Δf of the objective-function value: $w(t=0) \leftarrow (w(t=0)\ \delta_t/\Delta f)$.

Experimentally, using static w of $10^{-4}, 10^{-5}$, and 10^{-6} illustrates the three convergence behaviors: objective over-weighted, balanced objective and constraints, and constraints over-weighted.

For static weight $w = 10^{-4}$, Fig. 8 shows the dynamic changes of the objective, the Lagrangian-function value, and the maximum violation as the search progresses. Note that the trajectories of the objective and the Lagrangian-function values are overlapped because constraint violations are small. As the starting point is not a local minimum of the objective, the search descends in the original-variable X space as the objective value decreases. In the meantime, the constraints are getting violated. As constraint violations become large, the Lagrangian part slowly gains ground and pushes the search back towards the feasible region, leading to increases in the objective value and decreases in constraint violations. Eventually, all constraint violations become 0, and the objective value stabilizes. The overall convergence speed to the equilibrium point is slow (949.0 CPU min at $t = 2.819$). Note that fluctuations of the maximum violation as well as the objective are due to inaccuracy in numerical estimation of the function values and gradients.

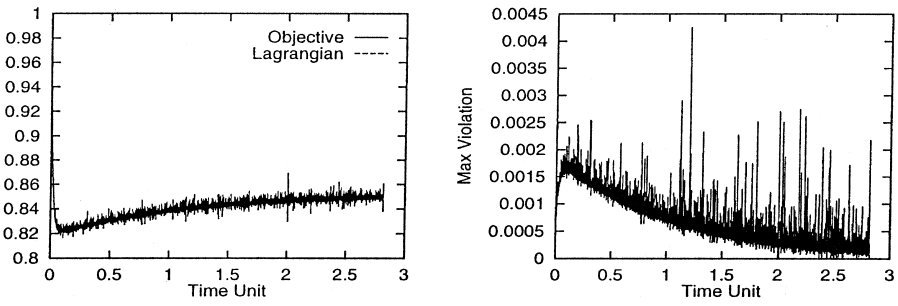


Fig. 8. Search profile with static weight $w = 10^{-4}$ in designing a 48e QMF filter bank. The objective is over-weighted, and its values are reduced quickly in the beginning. Later, the Lagrangian part slowly gains ground and pushes the search back into the feasible region. The convergence time to the equilibrium point is long (949.0 CPU min at $t = 2.819$).

Fig. 9 shows the search profile using static weight $w = 10^{-5}$. The objective and the constraints are more balanced in this case, and the convergence time to the equilibrium point is shorter (125.7 CPU min at $t = 1.577$ time units).

Fig. 10 shows the search profile using static weight $w = 10^{-6}$. The constraints are over-weighted in this case, and constraint satisfaction dominates the search process. The trajectory is kept inside or very close to the feasible region. However, due to the small weight on the objective, improvements of the objective is slow, causing slow overall convergence to the equilibrium point (293.8 CPU min at $t = 7.608$).

Fig. 11 shows the search profile of our Lagrangian method with adaptive weight control in solving the 48e QMF filter-bank problem. We have used a time unit $\Delta t = 10^{-4}$, window size $N_u = 10, \alpha_0 = \alpha_1 = 0.9, \delta = 10^{-5}, \beta_0 = 10^{-2}$, and $\beta_1 = 10^{-3}$. As shown in the search profiles, the objective value was improved fairly quickly in the beginning, and the trajectory was then pushed into

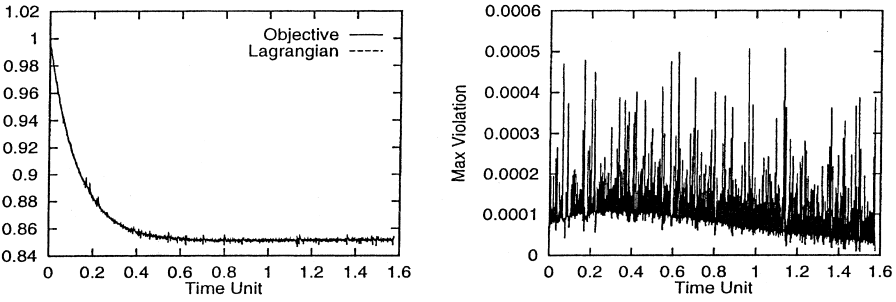


Fig. 9. Search profile with static weight $w = 10^{-5}$ in designing a 48e QMF filter bank. The objective and the constraints are balanced, and the convergence speed to the equilibrium point is faster (125.7 CPU min at $t = 1.577$).

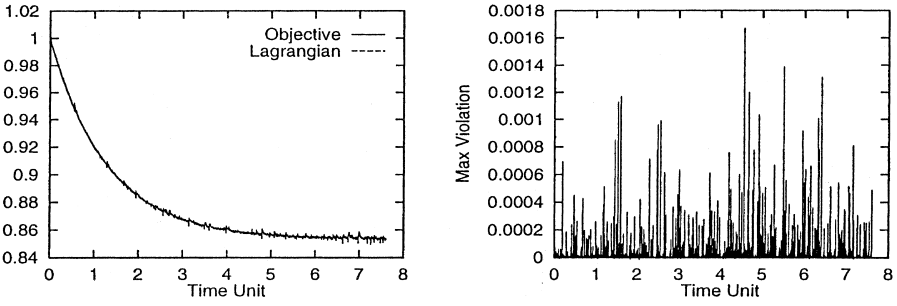


Fig. 10. Search profile with static weight $w = 10^{-6}$ in designing a 48e QMF filter bank. The constraints are over-weighted, and constraint satisfaction dominates the search process. The trajectory is kept inside or very close to the feasible region. However, the improvement of the objective value is slow, causing slow overall convergence to the equilibrium point (293.8 CPU min at $t = 7.608$).

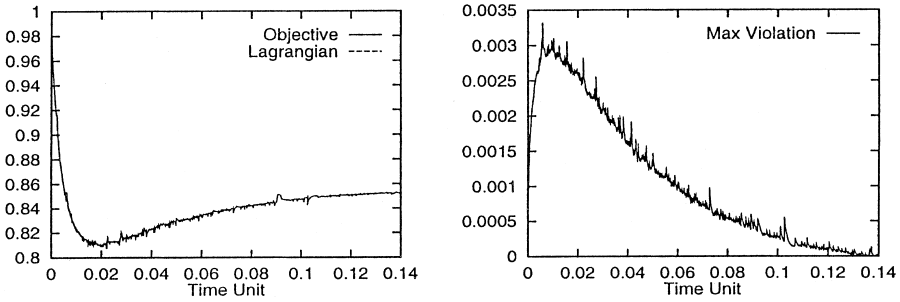


Fig. 11. Profile with adaptive changes of w in designing a 48e QMF filter bank. The search converges much faster (35.1 CPU min in $t = 0.142$) than those using static weights.

a feasible region quickly. Our adaptive method converges at $t = 0.142$ (35.1 CPU min), much faster than the Lagrangian method with static weight.

5.3. Nonlinear discrete integer programming problems

In this section, we apply our adaptive DLM on discretized constrained optimization problems. The discretization process was discussed in Section 3.2 in which we expand each variable into range $[0, N]$ and restrict each variable to integers in the range, where N is a positive integer.

Fig. 12 shows the search profile in solving the discretized Problem 2.6 discussed in Section 3.2 when dynamic weight adaptation is applied. Using an initial weight $w = 10^4$, oscillations finally die down, and the trajectory converges to an equilibrium point. In comparison, if w is statically set to 10^4, the search trajectory will diverge (see Table 2).

Table 5 shows the results corresponding to those in Table 2 when dynamic weight adaptation is applied. Using the same 20 randomly generated starting

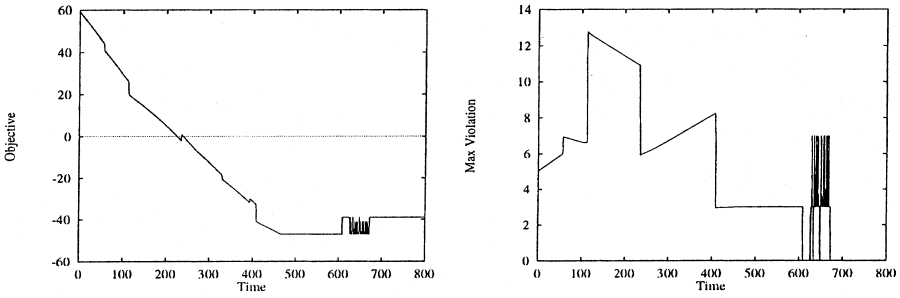


Fig. 12. The objective and maximum violation oscillate but eventually converge for discretized Problem 2.6 [1].

Table 5
Improved convergence and solution quality using dynamic weight adaptation from 20 randomly generated starting points for the discretized Problem 2.6 defined in [1]

Initial w	# Converged trajectories	Average time (s)	Best solution	Average solution
100000	20	2289	−39.0	−33.3
10000	20	2182	−39.0	−38.4
1000	20	1409	−39.0	−38.5
100	20	1902	−39.0	−29.3
10	20	1377	−39.0	−28.1
1	20	1289	−39.0	−27.4
0.1	20	1265	−39.0	−28.8
0.01	20	1249	−39.0	−29.3
0.001	20	1248	−39.0	−29.3
0.0001	20	1249	−39.0	−29.0
0.00001	20	1249	−39.0	−29.0

points and the same set of initial w, our results show that all the searches converge with similar or better average solutions.

5.4. Multiplierless QMF filter-bank design as nonlinear discrete optimization

In the last set of experiments, we study the design of multiplierless QMF filter banks. These filter banks are similar to those studied in Section 5.2, except that filter coefficients are powers-of-two (PO2) numbers (also called Canonical-Signed-Digit (CSD)). The filter coefficients are represented as sums or differences of powers of two so that multiplications can be carried out by additions, subtractions and shifting.

We formulate the design problem as a nonlinear discrete constrained optimization problem, using the reconstruction error as the objective, and other performance metrics as constraints (21). The frequency response of a PO2 filter, $H(z)$, is:

$$\begin{aligned} H(z) &= \sum_{i=0}^{\gamma-1} x_i z^{-i} \\ &= \sum_{i=0}^{\gamma-1} \left(\sum_{j=0}^{d-1} e_{i,j} 2^j \right) z^{-i} \text{ where } \sum_{j=0}^{d-1} |e_{i,j}| \leqslant l \text{ for all } i, \ e_{i,j} = -1, 0, 1. \end{aligned} \tag{22}$$

Here, γ is the length of the PO2 filter, l is the maximum number of ONE bits used in each coefficient, and d is the number of bits in each coefficient.

In formulating the problem and in applying DLM to solve it, we must first transform the real coefficients of the best-known design to PO2 forms using a CSD representation. Experiments show that if each coefficient is scaled

properly before the search starts (based on a heuristic objective), the quality of the final design can be improved significantly. Hence, we enumerate over different scaling constants and scale all the coefficients by a common constant before the search begins. Experiments also show that it is possible to find good designs without requiring the PO2 coefficients to have the same degree of precision as that of continuous coefficients. For instance, in our experiments, we restrict the minimum exponent of the ONE bits in each coefficient (in the range $[-1, 1]$) to be -22, even though the real coefficients have a minimum exponent of -31.

In our DLM implementation, we process all the bits in the coefficients in a predefined order. We perturb the sign and the exponent of each ONE bit to see if they would reduce the Lagrangian-function value (2), and accept the changes if they do. We update the Lagrange multipliers after processing each coefficient.

As an illustration, consider the design of a PO2 QMF filter bank [8] based on Johnston's 32d design [3] as our constraints. Assuming a minimum exponent of -22 in each ONE bit, we enumerate and find the best scaling factor for all the coefficients to be 0.9474. After multiplying each coefficient by this scaling factor and restricting each coefficient to a PO2 form with a maximum of 6 ONE bits, we apply DLM with both static and dynamic weights to find the best PO2 designs.

Table 6 compares the objectives of the designs found and the corresponding convergence times of DLM. Using the adaptive DLM, all the searches converge, and most designs have better reconstruction errors. Fig. 13 illustrates the convergence behavior when the search starts with $w = 1.0$. Using weight adaptation, the search converged after some oscillations and found a PO2 design with reconstruction error of 83.6% of Johnston's original 32d design (with real

Table 6
Multiplierless 32d QMF filter banks found by DLM with static and adaptive weights[a]

Weight w	Static weights		Adaptive weights	
	Objective	Time (min)	Objective	Time (min)
10000.0	–	–	0.998	195.9
1000.0	–	–	0.998	168.3
100.0	–	–	0.885	277.2
10.00	–	–	0.883	119.3
1.00	–	–	0.836	190.8
0.1	0.87	197.1	0.837	161.4
0.01	0.87	115.2	0.87	124.3
0.001	0.87	4.8	0.87	34.5
0.0001	0.88	12.0	0.878	10.8
0.00001	0.9	23.7	0.924	12.0

[a] The objective is the reconstruction error E_r.

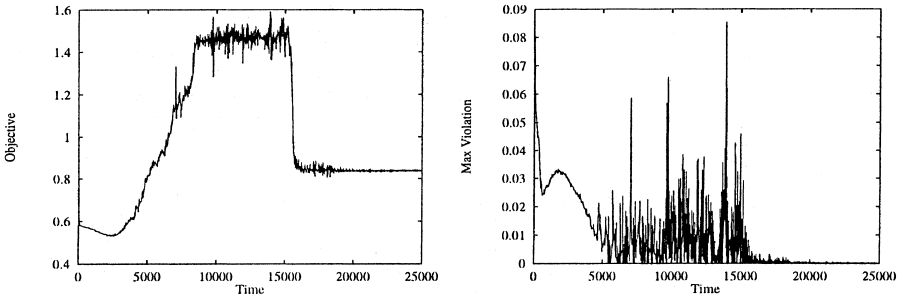


Fig. 13. The objective and maximum violation oscillate but eventually converge in the design of a multiplierless QMF filter bank.

coefficients), while all the other performance metrics are either better than or equal to those of Johnston's.

6. Experimental results

In this section we show further experimental results in applying our adaptive algorithm to the four nonlinear continuous and discrete optimization problems described in the last section. We also compare our results to those obtained by Lagrangian methods with static weights.

6.1. Nonlinear continuous benchmark problems

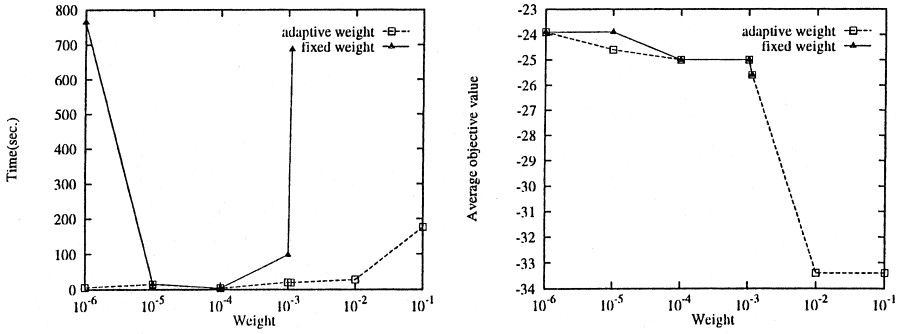
These constrained benchmark problems [1] were derived from a variety of engineering applications. We selected five problems (with identifiers 2.6, 3.4, 4.6, 5.4 and 6.4) from different classes to test our algorithm. In each problem, we randomly generated 20 starting points $X(t=0)$ uniformly distributed in its search range, and set the initial values for the Lagrange-multipliers to be zero ($\lambda(t=0)=\mu(t=0)=0$). For each starting point, we applied the Lagrangian method using both static weights and dynamically changed weights under the same convergence condition defined in (16). Hence, the Lagrangian method stops when it either reaches the maximum iteration count $i_{\max}$ or satisfies the convergence condition.

We chose our static weights in the range between 10^{-6} and 10^{2}, which were also used as initial weights $w(t=0)$ in the adaptive algorithm. The weights were chosen from a wide range in order to test the robustness of our weight-adaptation algorithm. We would like our algorithm to adjust the weight so that the search will converge faster with a solution that is at least as good as that with static weights. In our experiments, we used the following control

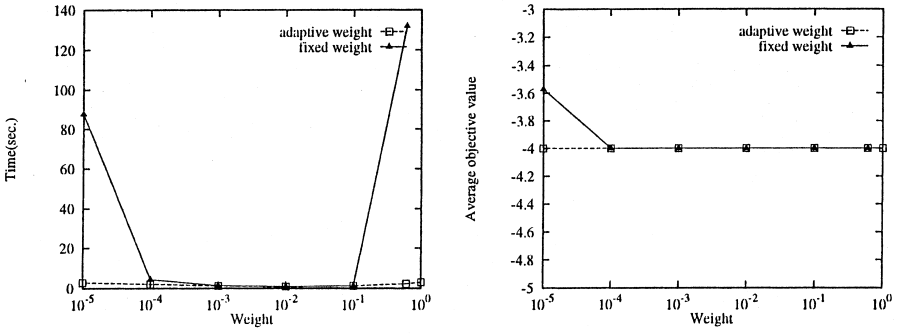
parameters: time unit $\triangle t = 1$, window size $N_u = 100, \alpha_0 = \alpha_1 = 1/2$, $\delta = 10^{-8}, \beta_0 = 10^{-2}$, and $\beta_1 = 10^{-3}$.

We used two performance metrics, convergence time and solution quality, in our experiments, where convergence time was measured by the average convergence time (in seconds) from 20 starting points, and solution quality was measured by the average objective value when the search converged.

From the experimental results shown in Fig. 14, we have the following observations. First, different problems require different ranges of static weights in order to get fast convergence. For example, Problem 2.6 (Fig. 14(a)) converges the fastest using $w = 10^{-4}$, whereas Problem 5.4 (Fig. 14(d)) converges the fastest using $w = 1$. These weights differ by four orders of magnitude. Hence, it is difficult to choose a good static weight for a given problem instance in advance.



(a) Problem 2.6



(b) Problem 3.4

Fig. 14. Comparison of the Lagrangian method with static weights and the adaptive Lagrangian method in terms of convergence time and solution quality.

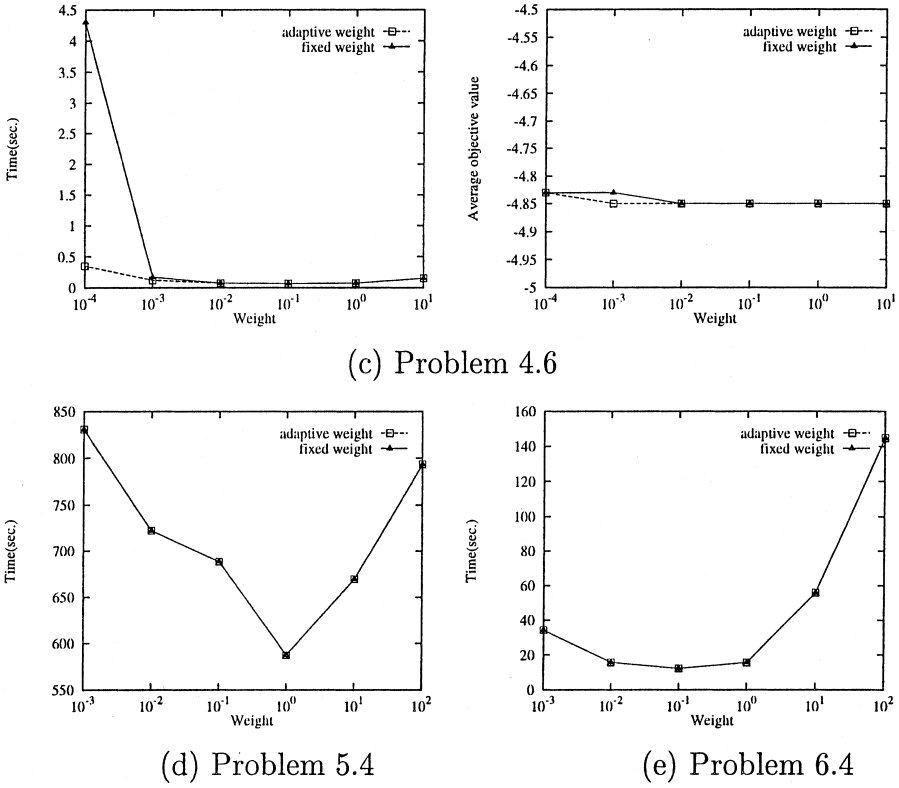


Fig. 14. (continued).

Second, our adaptive algorithm outperforms the Lagrangian method with static weights. For example, in Problem 2.6 (Fig. 14(a)), the Lagrangian method with a static weight is unable to converge (either diverge or oscillate forever) when $w = 10^{-2}$ or larger. However, our adaptive algorithm can detect this misbehavior and adjust w to allow the search to converge in 27.62 seconds on the average. Likewise, the Lagrangian method with static weight $w = 10^{-6}$ converges using an average of 765.0 s, but the adaptive algorithm converges using an average of 5.64 s and with the same solution quality. These results show that our weight-adaptation algorithm is robust and insensitive to the initial weights $w(t = 0)$ in comparison to the Lagrangian method with a static weight. In fact, our adaptive algorithm can get even better solutions in shorter time when the initial weight is small.

Third, when all the constraints are equality constraints, w is unchanged in the adaptive algorithm, resulting in the same performance for both methods. This is demonstrated in Problem 5.4 (Fig. 14(d)). In some cases, w is unchanged in the adaptive algorithm when there are both inequality and equality

constraints, such as Problem 6.4 (Fig. 14(e)). This happens when the Lagrange multipliers already take care of the balance between the objective and the constraints.

6.2. QMF filter-bank design

We have applied our adaptive Lagrangian method to solve the 14 QMF filter-bank design problems formulated by Johnston [3]. These include 16a, 16b, 16c, 24b, 24c, 24d, 32c, 32d, 32e, 48c, 48d, 48e, 64d, and 64e, where the integer in each identifier represents the number of filter taps, and types "a" to "e" represent prototype filters with increasingly sharper (shorter) transition bands.

Our goal is to find designs that are better than Johnston's results across all six performance measures. Hence, we use (21) with the constraint bounds defined by those of Johnston's designs. In our experiments, we started our search from Johnston's solutions as starting points. Note that Johnston used sampling in computing energy values, whereas we use closed-form integration. As a result, Johnston's designs may not be locally optimal in a continuous formulation.

Fig. 15 compares the performance of our adaptive Lagrangian method with that of the Lagrangian method with static weight. It shows the convergence times of the Lagrangian method with static weight normalized with respect to those of the adaptive method in solving the 24d and 48e design problems. For the method with static weights, we have used, respectively, weight values of $10^{-4}, 5\times10^{-5}, 10^{-5}, 5\times10^{-6}$, and 10^{-6} in our experiments. For the method with adaptive weights, we have used the weight-initialization algorithm described in Section 5.2 to set the initial weights. In solving the 24d problem, our adaptive Lagrangian method takes 6.6 min to converge to an equilibrium point with an objective value of 0.789, whereas the Lagrangian method with static

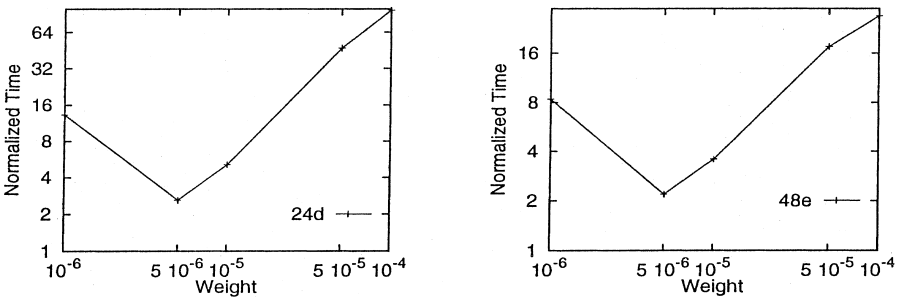


Fig. 15. Comparison of the average convergence speeds of Lagrangian methods with and without adaptive weight control for 24d (left) and 48e (right) QMF filter-bank design problems. The average convergence times of the Lagrangian method using different static weights are normalized with respect to the time spent by the adaptive Lagrangian method.

weights converges to the same equilibrium point most of the time, but with vastly different convergence times. Similarly, the adaptive method takes 35.1 min in solving the 48e design problem, but the static method takes vastly different and longer times for different initial weights.

Table 7 shows the experimental results for all the design problems found using the adaptive method, normalized with respect to Johnston's solutions. A value less than 1 for a performance metric means that our design is better on this metric as compared to Johnston's design. The table shows that the adaptive method improves the objective value (second column), while satisfying all the design constraints (columns 3–7). The execution times to find the solutions vary significantly from a few minutes to several hours.

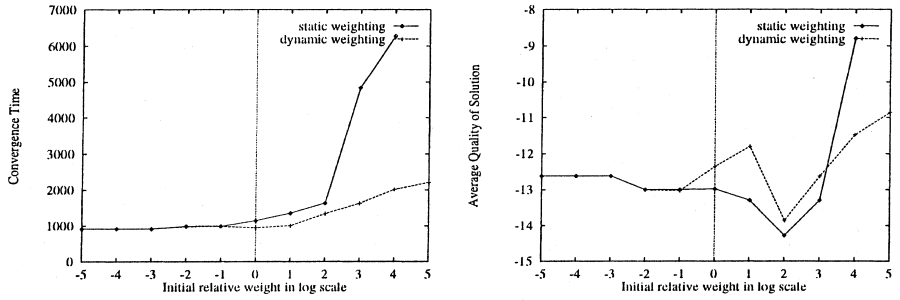
6.3. Nonlinear integer programming

To test the adaptive strategy, we selected three problems 2.3, 2.6 and 3.4 from the benchmark collection [1]. We first transformed them to discrete integer programming problems, generated 20 random discrete starting points, and set the initial value of the Lagrange-multipliers to be zero. We further set the initial weight w in the range $[10^{-5}, 10^{5}]$, which was large enough to test the robustness of our strategy. The parameters for our algorithm were as follows: $N_u = 320, \alpha_0 = 0.8, \alpha_1 = 0.5, \delta = 10^{-8}, \beta_0 = 10^{-3}$, and $\beta_1 = 10^{-4}$. From each starting point, we applied the Lagrangian method using both static weights and adaptive weights, and stopped the search when either the convergence condition was satisfied or the maximal number of iterations was reached.

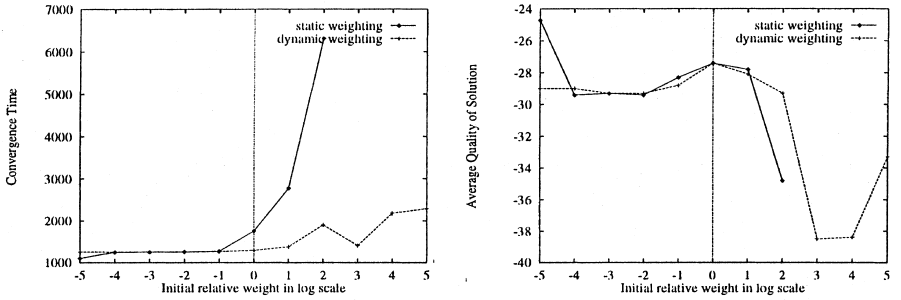
Table 7
Experimental results of the adaptive Lagrangian method in solving QMF filter-bank design problems, using Johnston's solutions as starting points

Filter type	E_r	δ_p	E_p	δ_s	E_s	T_r	CPU time (min)	Time unit
16a	0.990	0.997	0.785	0.999	1.000	1.000	89.8	0.156
16b	0.995	0.962	0.806	1.000	1.000	1.000	16.5	0.278
16c	0.826	1.000	0.916	1.000	1.000	1.000	16.3	0.213
24b	0.966	0.994	0.820	0.999	1.000	1.000	119.0	0.107
24c	0.910	1.000	0.768	1.000	1.000	1.000	11.4	0.148
24d	0.789	1.000	0.835	1.000	1.000	1.000	6.6	0.175
32c	0.959	0.999	0.735	0.999	1.000	1.000	111.8	0.115
32d	0.870	0.999	0.800	1.000	1.000	1.000	15.4	0.136
32e	0.735	1.000	0.924	1.000	1.000	1.000	17.5	0.194
48c	0.793	1.000	0.810	0.945	0.999	1.000	2899.1	0.123
48d	0.948	0.999	0.752	0.999	1.000	1.000	250.5	0.106
48e	0.852	1.000	0.840	1.000	1.000	1.000	35.1	0.142
64d	0.784	0.991	0.787	0.591	0.997	1.000	3143.5	0.021
64e	0.845	1.000	0.764	1.000	1.000	1.000	118.9	0.138

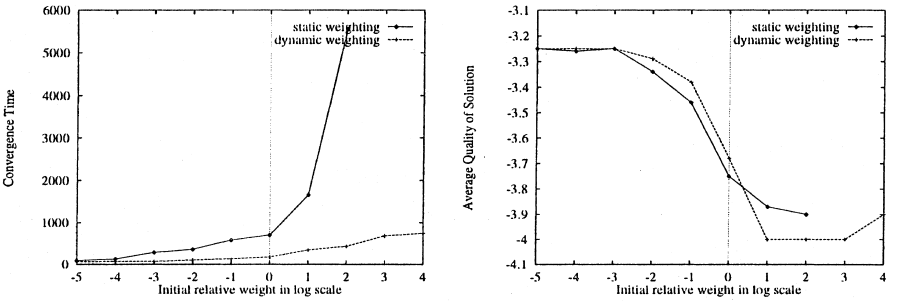
As in continuous problems, we use as performance metrics the average convergence time from 20 starting points and the average objective value when the search converges. Fig. 16 summaries the results, showing that the adaptive method improves a lot in convergence times over the static method. For



(a) Problem 2.3



(b) Problem 2.6



(c) Problem 3.4

Fig. 16. Comparison of Lagrangian methods with static weights and adaptive weights in terms of the average convergence time and average solution quality for discrete benchmark problems.

instance, for Problem 2.6, when w was larger than 100.0, the static method was unable to converge after 10 h, whereas the adaptive method converged in less than one hour for all different initial weights.

6.4. Multiplierless QMF filter-bank design

In our experiments on designing multiplierless QMF filter banks, we set the maximum number of ONE bits to be 6 and the minimum exponent to be -22 for each filter coefficient. We further set our starting points of the search based on Johnston's design, and assigned the same control parameters as those in the previous subsection except that N_u was 10.

Figs. 17 and 18 compare the average convergence times and average solution quality in terms of reconstruction error in designing the 32d and 48e multiplierless QMF filter banks. Our results show that the adaptive algorithm converged in less than 300 (resp. 510) CPU min for the 32d (resp. 48e) problem, even though the initial weights were in a very large range $[10^{-5}, 10^{4}]$ (resp. $[10^{-4}, 10^{2}]$). In contrast, the Lagrangian method with static weights could not

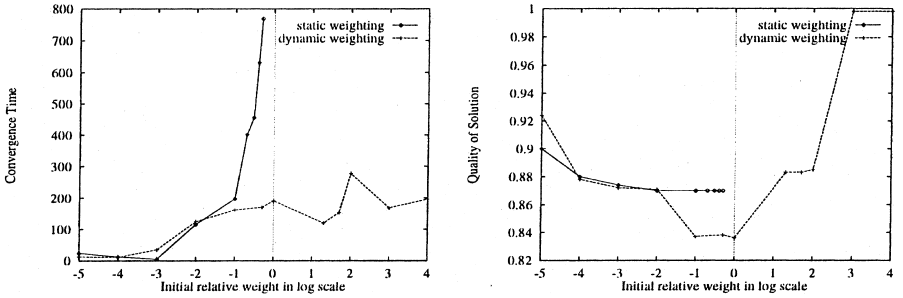


Fig. 17. Comparison of average convergence times and average solution quality between using static weights and adaptive weights in designing multiplierless QMF filter bank 32d.

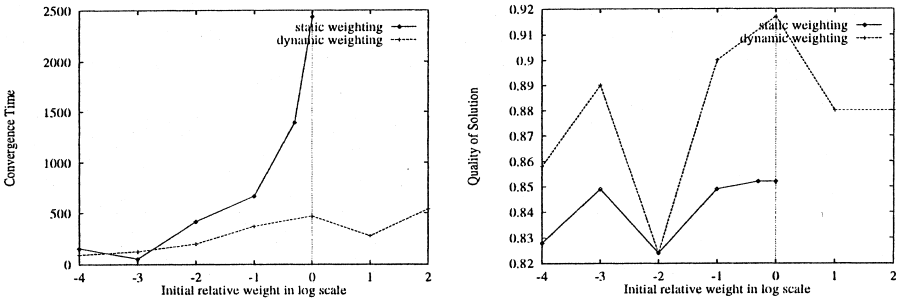


Fig. 18. Comparison of average convergence times and average solution quality between using static weights and adaptive weights in designing multiplierless QMF filter bank 48e.

converge in 15 (resp. 32) h in designing the 32d (resp. 48e) multiplierless filter bank when w was larger than 1.0.

Note that in designing the 48e filter banks, the average solutions of the static method are slightly better than those of the adaptive method under some initial weights. This happens because the latter may change the terrain during the search and finds different solutions.

7. Conclusions

In this paper, we have studied and evaluated efficient weight-adaptation algorithms for Lagrangian methods. We have shown that Lagrangian methods are useful for solving continuous as well as discrete constrined optimization problems. These methods rely on the balance between descents in the objective space and ascents in the Lagrange-multiplier space in order to arrive at an equiliblirum point. We have illustrated through numerous examples that such a balance is very sensitive to the relative weights between the objective and the Lagrangian part in the Lagrangian function. Without a good balance, the search trajectory may converge very slowly, oscillate forever, or diverge. To cope with these problems, we propose a dynamic weight-adaptation algorithm that adjusts the relative weight of the objective based on statistics collected during the search. We have applied the adaptive Lagrangian method on two classes of continuous constrained optimization problems and two classes of discrete constrained problems. In each case, we have shown that the adaptive method always converge quickly with either the same or better solutions. Our results show that our proposed adaptive Lagrangian method is roburst, works for both continuous and discrete constrained optimization problems, and exhibits consistent and good convergence behavior.

Acknowledgements

Research supported by National Science Foundation Grant MIP 96-32316 and National Aeronautics and Space Administration Contract NAG 1-613. A preliminary version of this paper was presented at the 1997 IEEE International Conference on Tools with Artificial Intelligence [9].

References

[1] C.A. Floudas, P.M. Pardalos, A collection of test problems for constrained global optimization algorithms, Lecture Notes in Computer Science, vol. 455. Springer, 1990.

[2] A.M. Geoffrion, Lagrangian relaxation and its uses in integer programming, Mathematical Programming Study 2 (1974) 82–114.
[3] J.D. Johnston, A filter family designed for use in quadrature mirror filter banks, in: Proceedings of International Conference on ASSP, 1980, pp. 291–294.
[4] D.G. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
[5] Y. Shang, B.W. Wah, A discrete Lagrangian based global search method for solving satisfiability problems, J. Global Optim. 12 (1) (1998) 61–99.
[6] B.W. Wah, Y.-J. Chang, Trace-based methods for solving nonlinear global optimization problems, J. Global Optim. 10 (2) (1997) 107–141.
[7] B.W. Wah, Y. Shang, T. Wang, T. Yu, Global optimization of QMF filter-bank design using NOVEL, in: Proceedings of International Conference on Acoustics, Speech and Signal Processing, vol. 3. IEEE, Munich, Germany, 1997, pp. 2081–2084.
[8] B.W. Wah, Y. Shang, Z. Wu, Discrete Lagrangian method for optimizing the design of multiplierless QMF filter banks, in: Proceedings of International Conference on Application Specific Array Processors, IEEE, 1997, pp. 529–538.
[9] B.W. Wah, T. Wang, Y. Shang, Z. Wu, Improving the performance of weighted Lagrange-multiple methods for constrained nonlinear optimization, in: Proceedings of Ninth International Conference on Tools for Artificial Intelligence, IEEE, 1997, pp. 224–231.