Integrated Design and Process Technology, IDPT-Vol. 1
Printed in the United States of America, December 1996

HANDLING INEQUALITY CONSTRAINTS IN CONTINUOUS NONLINEAR GLOBAL OPTIMIZATION

Tao Wang and Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

ABSTRACT

In this paper, we present a new method to handle inequality constraints and apply it in *NOVEL* (Nonlinear Optimization via External Lead), a system we have developed for solving constrained continuous nonlinear optimization problems. In general, in applying Lagrange-multiplier methods to solve these problems, inequality constraints are first converted into equivalent equality constraints. One such conversion method adds a slack variable to each inequality constraint in order to convert it into an equality constraint. The disadvantage of this conversion is that when the search is inside a feasible region, some satisfied constraints may still pose a non-zero weight in the Lagrangian function, leading to possible oscillations and divergence when a local optimum lies on the boundary of a feasible region. We propose a new conversion method called the *MaxQ* method such that all satisfied constraints in a feasible region always carry zero weight in the Lagrange function; hence, minimizing the Lagrange function in a feasible region always leads to local minima of the objective function. We demonstrate that oscillations do not happen in our method. We also propose methods to speed up convergence when a local optimum lies on the boundary of a feasible region. Finally, we show improved experimental results in applying our proposed method in *NOVEL* on some existing benchmark problems and compare them to those obtained by applying the method based on slack variables.

1. INTRODUCTION

In this paper, we study methods to handle inequality constraints in Lagrangian formulations to solve constrained global optimization problems over continuous variables. Here global minimization looks for a solution that satisfies all the constraints and is no larger than any other local minimum. This is a challenging problem as there may not be enough time to find a feasible solution, or even if a feasible solution is found, there is no way to show that it is optimal.. In practice, one only seeks as many local optima as possible that satisfy the constraints, and pick the best local optimum.

The *constrained nonlinear global optimization problems* that we study take the following form

$$\begin{aligned} &\textit{Minimize} && f(X) \\ &\textit{Subject To} && g(X) \leq 0 \qquad X = (x_1, \dots, x_n) \in R^n \quad (1) \\ & && h(X) = 0 \end{aligned}$$

*RESEARCH SUPPORTED BY NATIONAL SCIENCE FOUNDATION GRANTS MIP 92-18715 AND MIP 96-32316.
PROC. SOCIETY FOR DESIGN AND PROCESS SCIENCE CONFERENCE, DECEMBER 1996.

where $f(X)$ is an objective function, $g(X) = [g_1(X), \cdots, g_k(X)]^T$ is a set of k inequality constraints, and $h(X) = [h_1(X), \cdots, h_m(X)]^T$ is a set of m equality constraints. All $f(X)$, $g(X)$, and $h(X)$ are assumed to be differentiable real-valued functions.

In this paper, we propose a new method to handle inequality constraints in constrained global optimization and compare it with an existing method based on slack variables [9]. We first summarize previous work in the area in the next section. In Section 3., we present our proposed *MaxQ* method and discuss strategies to control its convergence. In Section 4., we apply the MaxQ method in *NOVEL*, a system we have developed to solve constrained [14] as well as unconstrained [12] nonlinear optimization problems. Finally, we present experimental results of the *MaxQ* method in Section 5. and compare them to those based on slack variables.

2. PREVIOUS WORK ON CONSTRAINED NONLINEAR OPTIMIZATION

2.1. Lagrange Multiplier Method

Active research in the past two decades has produced a variety of methods to find solutions to constrained nonconvex nonlinear continuous optimization problems [13, 6, 4, 5, 11, 10]. In general, they are divided into transformational and non-transformational methods.

Non-transformational approaches include discarding methods, back-to-feasible-region methods, and enumerative methods. Discarding methods [7, 10] drop solutions once they were found to be infeasible, and back-to-feasible-region methods [8] attempt to maintain feasibility by reflecting moves from boundaries if such moves went off the current feasible region. Both of these methods have been combined with global search and do not involve transformation to relax constraints. Last, enumerative methods [6] are generally too expensive to apply except for problems with linear objectives and constraints, and for bilinear programming problems [1].

Transformational approaches, on the other hand, convert a problem into another before solving it. Well known methods include penalty, barrier, and Lagrange-multiplier methods [9]. *Penalty methods* transform constraints into part of the objective function and require tuning penalty coefficients either before or during the search. *Barrier methods* are similar except that barriers are set up to avoid solutions from going out of feasible regions. Both methods have difficulties when they start from an infeasible region and when feasible solutions are hard to find. However, they can be combined with other methods to improve their quality.

Lagrange-multiplier methods introduce Lagrange variables to gradually resolve constraints through iterative updates. They are exact methods that optimize the objective using Lagrange multipliers to meet the Kuhn-Tucker conditions [9]. In view of their advantages, we use them for constraint relaxation in this paper. Given an optimization problem with equality constraints,

$$\begin{array}{ll} \textit{Minimize} & f(X) \\ \textit{Subject To} & h(X) = 0 \end{array} \tag{2}$$

the corresponding *Lagrangian function* and *augmented Lagrangian function* are defined as

$$\mathcal{L}(X,\lambda) = f(X) + \lambda^T h(X) \tag{3}$$

$$L(X,\lambda) = f(X) + ||h(X)||_2^2 + \lambda^T h(X) \tag{4}$$

where $\lambda = [\lambda_1, \cdots, \lambda_m]$ is the set of Lagrange multipliers. We use the augmented Lagrangian function in this paper since it provides better numerical stability.

According to classical optimization theory [9], all the extrema of (4), whether local or global, are roots of the following sets of equations.

$$\nabla_X L(X,\lambda) = 0 \tag{5}$$

$$\nabla_\lambda L(X,\lambda) = 0 \tag{6}$$

These conditions are necessary to guarantee the (local) optimality to the solution of (2).

The roots in (5) and (6) can be solved by forming the following dynamic system of equations to seek equilibrium points.

$$\frac{d}{dt}X(t) = -\nabla_X L(X(t), \lambda(t)) \tag{7}$$

$$\frac{d}{dt}\lambda(t) = \nabla_\lambda L(X(t), \lambda(t)) \tag{8}$$

These equilibrium points are called *saddle-points* of (5) and (6), which correspond to the constrained minima of (2). Eq's (7) and (8) perform descent in the original-variable space of X and ascent in the Lagrangian space of λ.

2.2. Converting Inequality Constraints By Slack Variables

Lagrange-multiplier methods work well with equality constraints. However, they have difficulties in dealing with inequality constraints (2) directly, except in some simple cases in which one can directly solve the first-order condition. When there are inequality constraints, they must first be converted into equality constraints before Lagrange-multiplier methods can be applied.

One possible transformation [9] for $g_i(X) \le 0$ ($i = 1, \cdots, k$) is to have $g_i(X) + z_i^2 = 0$, where z_i is a slack variable. After simplification [9], the resulting augmented Lagrangian function becomes

$$\begin{aligned} L_z(X,\lambda,\mu) &= f(x) + \lambda^T h(X) + ||h(X)||_2^2 \\ &+ \sum_{i=1}^{k} \left[max^2(0, \mu_i + g_i(X)) - \mu_i^2 \right] \end{aligned} \tag{9}$$

where λ and μ are Lagrange multipliers. In the same way as (7) and (8), (9) can be implemented by the dynamic system that performs descent in the original-variable space of X and ascent in the Lagrangian space of both λ and μ.

$$\begin{aligned} \frac{d}{dt}X(t) &= -\nabla_X L_z(X(t), \lambda(t), \mu(t)) \\ \frac{d}{dt}\lambda(t) &= \nabla_\lambda L_z(X(t), \lambda(t), \mu(t)) \\ \frac{d}{dt}\mu(t) &= \nabla_\mu L_z(X(t), \lambda(t), \mu(t)) \end{aligned} \tag{10}$$

We have used a differential equation solver *LSODE* to solve the above dynamic system and have observed oscillations in some cases before local minima are found. There are two types of oscillations.

- The trajectory's oscillations decay, and the search converges to a local minimum.
- The oscillations never decay, and the search does not converge.

The second type of oscillations may happen when a local minimum is on the boundary of a feasible region. To understand this, suppose that, at time t, an inequality constraint $g_i(X(t)) \le 0$ is satisfied but $\mu(t) > 0$ is large, and thus $g_i(X(t)) + \mu(t) > 0$. This means that when the search trajectory is inside the feasible region (the corresponding constraint is satisfied), $g_i(X(t))$ also appears in the Lagrangian function (9). Since the local minimum is on the boundary, the force from inside the feasible region pushes the trajectory to outside the feasible region. Likewise, when the trajectory is outside the feasible region, the force due to the Lagrange multipliers pushes the trajectory inside the feasible region, causing an oscillation. Depending on the relative magnitude of the objective-function value and the weighted sum of the Lagrange multipliers and the constraints in (9). The oscillations may amplify and cause divergence.

It is possible to eliminate some of these oscillations by careful scaling of the objective function and some constraints. For instance, consider Problem 2.3.1 in Table 1. If we do not scale the objective and constraints and do descents starting from the midpoint of the search space, the search diverges into infinity. Next, when we scale the objective by a factor 7.5, the search oscillates within some range but never converges. (See Figure 1.) Last, when we scale the objective by a factor 15, convergence can be obtained shown. (See Figure 2.) Clearly, such scaling is problem-instance dependent and is generally undesirable.

3. PROPOSED MAXQ METHOD

To avoid oscillations in the method based on slack variables, we would like a search to converge to a local minimum directly when it is on the boundary of a feasible region and when the trajectory is outside the feasible region. This is done by our proposed *MaxQ* method.

3.1. Converting Inequality Constraints by MaxQ

Without loss of generality, we ignore equality constraints in the following discussion, knowing that equality constraints are handled in the way described in Section 2.1.. The *MaxQ* method converts an inequality constraint as follows.

$$g_i(X) \le 0 \iff Q_i(X) = max^{q_i}(0, g_i(X)) = 0 \tag{11}$$

where $q_i > 1$ ($i = 1, \cdots, k$) are control parameters. This means that $Q_i(X)$ carries zero weight when the constraint $g_i(X) \le 0$ is satisfied.

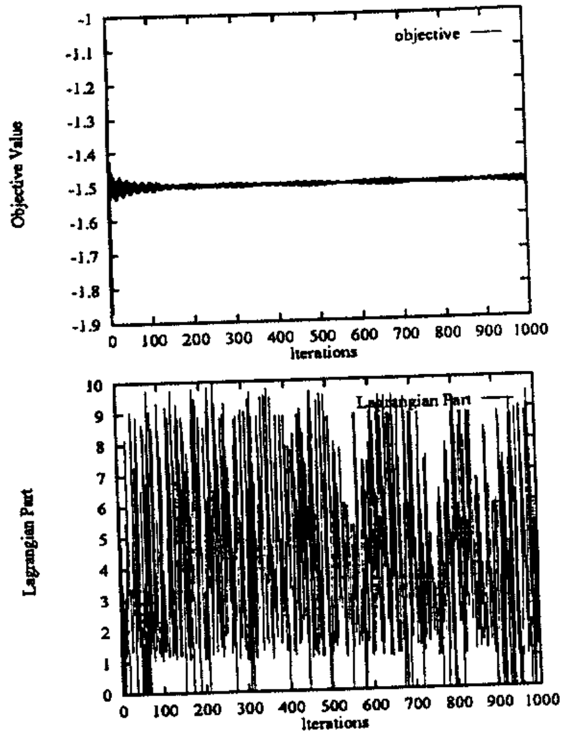

Figure 1. Objective function and Lagrangian part oscillate, when the Lagrangian part is the last three terms in (9).

The augmented Lagrangian function is

$$L_q(X,\mu) = f(X) + \sum_{i=1}^{k} [\mu_i max^{q_i}(0, g_i(X)) + max^{2q_i}(0, g_i(X))] \quad (12)$$

Assuming that q_i is constant, the corresponding dynamic system is

$$\begin{aligned} \frac{d}{dt}X(t) &= -\nabla_X L_q(X(t),\mu(t)) \\ &= -\nabla_X f(X) - \sum_{i=1}^{k} [\mu_i q_i max^{q_i-1}(0, g_i(X)) \\ &\quad + 2q_i max^{2q_i-1}(0, g_i(X))] \nabla_X g_i(X) \\ \frac{d}{dt}\mu_i(t) &= \nabla_\mu L_q(X(t),\mu(t)) = max^{q_i}(0, g_i(X)) \end{aligned} \quad (13)$$

Note that (12) is similar to (9) in the sense that both use the *max* function. However, (12) avoids the case in (9) in which an inequality constraint $g_i(X(t)) \leq 0$ is satisfied at time t, and $g_i(X(t))$ also appears in the Lagrange function (12). When $g_i(X(t))$ is satisfied, it is meaningful to minimize $f(X)$ independent of the value of $g_i(X(t))$.

3.2. Control Parameters

It is important to choose suitable control parameters q_i $(i = 1, \cdots, k)$ because they are related to the convergence speed and precision of our algorithm. One can easily

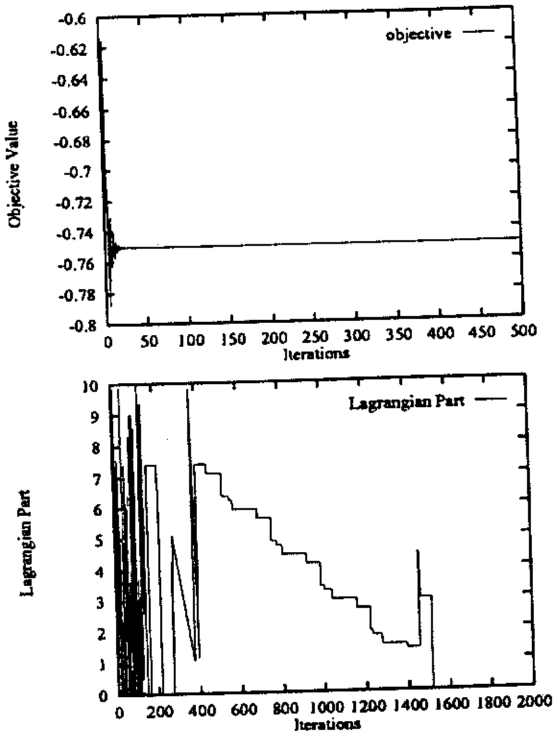

Figure 2. Objective function and Lagrangian part converge.

show that, when $q_i \geq 1$, the inequality constraint $g_i(X) \leq 0$ is equivalent to the equality constraint $max^q(0, g_i(X)) = 0$. When applying (10) to handle the transformed inequality constraints, we need to use $\nabla_X Q_i(X)$, *i.e.*,

$$\begin{aligned} \nabla_X Q_i(X) &= q_i max^{q_i-1}(0, g(X)) \nabla_X g(X) \\ &= Q_i'(X) \nabla_X g(X) \end{aligned} \quad (14)$$

If $q_i \leq 1$, $Q_i'(X)$ is not continuous, as the derivative of $L(X,\mu)$ is not continuous when $g_i(X) = 0$. However, continuity of derivatives is required by most differential-equation solvers, such as *LSODE*. This is the reason why we require $q_i > 1$ in (11).

Due to the way we handle inequality constraints (11), it can be shown that (13) will converge exactly to a saddle point if it is within a feasible region. According to (13), its equilibrium point is given by

$$\frac{d}{dt}X(t) = 0 \quad \text{and} \quad \frac{d}{dt}\mu_i(t) = 0 \quad i = 1, \cdots, k \quad (15)$$

which means there is no change of the original variables X and Lagrange multipliers μ_i. At this time, $max^{q_i}(0, g_i(X)) = 0$, and $\nabla_X f(X) = 0$ is solved. Hence, the saddle point is within the feasible region.

The transformation (11), however, has some difficulty when the saddle point is on the boundary of a feasible region, *i.e.*, $\exists i, g_i(X) = 0$. Suppose $\nabla_X f(X) \neq 0$ for every point X in the feasible region. In this case, no equilibrium point in theory will exist; that is, there is at least one inequality constraint that is violated, $g_i(X) > 0$, for example.

Otherwise, based on (13), we will get $\nabla_X f(X) = 0$. To exactly satisfy such a constraint, q_i in (11) must be set to one in theory. But this will cause the derivative $\nabla_X L_q(X)$ to be discontinuous at the point where $g_i(X) = 0$, making it impossible for the differential-equation solver *LSODE* to converge to a saddle point at $g_i(X) = 0$.

One way to tackle this problem is to choose q_i to be very close to 1. But this may cause slow convergence and implementation difficulties. When $q_i \rightarrow 1$, $Q_i'(X) \simeq 1$ if $g_i(X) > 0$, but $Q_i'(X) = 0$ when $g_i(X) \leq 0$. Hence, $Q_i'(X)$ changes quickly from 1 to 0 near the saddle point as $g_i(X) \rightarrow 0$, making it difficult for *LSODE* to find a suitable step size in order to reach the saddle point. In addition, the system approaches the feasible region slowly since $Q_i'(X) \simeq 1$ if $g_i(X) > 0$, which is independent of how far the current point X is away from the feasible region. Thus, larger control parameters q_i are needed for fast convergence if the current iteration point X is far from the feasible region. In contrast, if we choose $q_i \gg 1$, then $Q'(X) \simeq 0$ as $g_i(X) \rightarrow 0$, meaning that *LSODE* converges very slowly towards the boundary saddle point and may not reach it within the precision specified in the differential-equation solver.

Taking these facts into account, in order to have fast convergence and to reach the precision specified in *LSODE*, we should change q_i dynamically as the search goes to a saddle point on the boundary. Since different inequality constraints may have different convergence rates to the boundary of a feasible region, we associate each inequality constraint $g_i(X) \leq 0$ with its own control parameter q_i that will be updated dynamically based on the value of $g_i(X)$: q_i is large if $g_i(X) \gg 1$, and q_i is gradually reduced to a value close to 1 when the search approaches a saddle point on the boundary where $g_i(X) = 0$. In our implementation, q_i is defined as a continuous function of $g_i(X)$.

$$q_i(g_i(X)) = \frac{s_0}{1 + exp(-s_1 g_i(X))} \qquad (16)$$

where s_0 is the parameter that determines what convergence precision is needed, since $q_i = s_0/2$ if $g_i(X) = 0$. Parameter s_1 is chosen to satisfy so that $q_i(g_i(X)) = 2$ when $g_i(X) = 1$. Thus, $s_1 = -Ln[s_0/2 - 1]$. In our experiments, we use $s_0 = 2.5$ or $s_0 = 3.0$, respectively.

Since every control parameter $q_i(g_i(X))$ is now a function of X instead of a constant, we need to modify the dynamic system (13) into

$$\begin{aligned}
\frac{d}{dt}X(t) &= -\nabla_X L_q(X(t), \mu(t)) \\
&= -\nabla_X f(X) - \sum_{i=1}^{k} \left[\mu_i max^{q_i - 1}(0, g_i(X)) \right. \qquad (17) \\
&\quad \left. + 2max^{2q_i - 1}(0, g_i(X))\right] \left[q_i + q_i'(X) \right. \\
&\quad \left. \times max(0, g(X)) Ln\, max(0, g(X))\right] \nabla_X g_i(X) \\
\frac{d}{dt}\mu_i(t) &= \nabla_\mu L_q(X(t), \mu(t)) = max^{q_i}(0, g_i(X)) \qquad (18)
\end{aligned}$$

We like to point out that the strategy to dynamically change q_i is also suitable when the saddle point is in a feasible region. In this case, the only effect is to speed up the convergence rate without compromising the precision of the solution when convergence is reached, since its saddle point can be reached exactly according to the argument above.

3.3. Periodic Reduction of Lagrangian Variables

In the *MaxQ* method, the Lagrange multipliers μ_i are non-decreasing according to (13). This may cause μ_i to be unbounded in theory. In practice, this is undesirable as some of the Lagrange multipliers may be large and some may be small, and large Lagrange multipliers may lead to stiffness of the Lagrangian function $L_q(X, \mu)$ in the search.

We tackle this problem by adding decay terms $r \times \mu_i \times sign[-g_i(x)]$ to the second dynamic system (18) to result in the following equation.

$$\frac{d}{dt}\mu_i(t) = \begin{cases} max^{q_i}(0, g_i(X)) & g(X) \leq 0 \\ \quad -r \times \mu_i \times sign[-g_i(X)] & \\ max^{q_i}(0, g_i(X)) & \text{Otherwise} \end{cases} \qquad (19)$$

where $i = 1, \cdots, k$. Here, $sign(x) = 1$ if $x \geq 0$, and 0 otherwise. The decay term $r \times \mu_i \times sign[-g_i(x)]$ takes effect only when the corresponding constraint $g_i(X) < 0$ is satisfied. Parameter r is positive and controls how fast μ_i is reduced. (In our experiments, $r = 1$.)

Our approach of introducing decay terms is reasonable. When an inequality constraint $g_i(X) \leq 0$ is violated, there will be no decay term $r \times \mu_i \times sign[-g_i(X)]$, and Lagrange multiplier μ_i will increase. This places more weight on the constraint, and forces it into a feasible region. When the constraint is satisfied, the decay term reduces the value of μ_i. Other unsatisfied constraints will gain more weights at the same time, making them easier to be satisfied.

3.4. Dynamic Conversion to Equality Constraints

As we have discussed above, if solution X^* is on the boundary, *i.e.*, when some $g_i(X^*)$ equals zero, the dynamic system (13) cannot reach this point exactly, but the point can be approached as close as possible if control parameter q_i is also very close to 1.

Suppose we know that some $g_j(X^*) = 0$ for a give solution. In this case, faster convergence and higher precision can be obtained if we consider $g_j(X)$ as an equality constraint, $g_j(X) = 0$, instead of an inequality constraint, $g_j(X) \leq 0$. The difficulty, however, is that it is impossible to know in advance which inequality constraints $g_j(X) \leq 0$ will satisfy the boundary condition (*i.e.*, $g_j(X) = 0$) if a solution is on the boundary.

In order to improve convergence rate and precision, we dynamically convert inequality constraints that are very close to the boundary into equality constraints. Since we solve the dynamic system (13) using *LSODE*, let X and X_0 be the points of two successive iterations. The conversion for inequality constraint $g_j(X) \leq 0$ occurs when the following conditions are satisfied.

- The dynamic system converges to some point X when it changes very little for a large number of iterations. We define the current point to be stable when $\#\{max_i|x_i - x_{0i}|/max_j|x_j| < \delta\} \geq 5000$ ($\delta = 10^{-4}$ in our experiments).
- The dynamic system converges to the boundary when $g_j(X) \leq 0$ is very close to zero; that is, $0 < g_j(X) < \epsilon$ ($\epsilon = 10^{-4}$ in our experiments).

Note that dynamic conversion can happen to many inequality constraints at the same time as long as they satisfy those two conditions.

If dynamic conversion is performed on (13), then the terrain of the Lagrangian function $L_q(X, \mu)$ will be totally different. To maintain the search direction in the original-variable space X, we have to adjust the Lagrange multiplier μ_j. Let the current point be (X, μ_j) just before the conversion. The Lagrangian term associated with inequality constraint $g_j(X) \leq 0$ is

$$\begin{aligned} L_j(X, \mu_j) &= \mu_j max^{q_j}(0, g_j(X)) + max^{2q_j}(0, g_j(X)) \\ &= \mu_j g_j^{q_j}(X) + g_j^{2q_j}(X) \end{aligned}$$

according to the conversion conditions. The derivative of $L_j(X, \mu_j)$ with respect to X and μ_j are

$$\begin{aligned} \nabla_X L_j(X, \mu_j) &= \left[\mu_j g_j^{q_j-1}(X) + 2g_j^{2q_j-1}(X)\right] * [q_j \\ &\quad + q_j'(X) g_j(X) Ln\ g_j(X)] \nabla_X\, g_j(X) \\ \nabla_{\mu_j} L_j(X, \mu_j) &= g_j^{q_j}(X) \end{aligned}$$

Let $(X, \hat{\mu}_j)$ be the point after the conversion. This means that we consider the inequality constraint $g_j(X) \leq 0$ in the boundary where the corresponding equality constraint $g_j(X) = 0$ can be applied. Then the Lagrangian term related to $g_j(X) = 0$ is

$$\hat{L}_j(X, \hat{\mu}_j) = \hat{\mu}_j g_j(X) + g_j^2(X)$$

and the derivative of $\hat{L}_j(X, \hat{\mu}_j)$ with respect to X and $\hat{\mu}_j$ are

$$\begin{aligned} \nabla_X \hat{L}_j(X, \hat{\mu}_j) &= (\hat{\mu}_j + 2g_j(X)) \nabla_X\, g_j(X) \\ \nabla_{\hat{\mu}_j} \hat{L}_j(X, \hat{\mu}_j) &= g_j(X) \end{aligned}$$

Since the control parameter q_j is close to 1, the search direction in the Lagrange-multiplier space changes very little, meaning that $\nabla_{\mu_j} L_j(X, \mu_j) \simeq \nabla_{\hat{\mu}_j} \hat{L}_j(X, \hat{\mu}_j)$, independent of the value μ_j. To retain the search direction in the original-variable space X, we have to set $\nabla_X L_j(X, \mu_j) = \nabla_X \hat{L}_j(X, \hat{\mu}_j)$ and get

$$\begin{aligned} \hat{\mu}_j &= \left[\mu_j g_j^{q_j-1}(X) + 2g_j^{2q_j-1}(X)\right] * \\ &\quad \left[q_j + q_j'(X) g_j(X) Ln\ g_j(X)\right] - 2g_j(X) \qquad (20) \end{aligned}$$

4. NOVEL: A SYSTEM FOR GLOBAL OPTIMIZATION

After converting inequality constraints into equality constraints and by formulating the optimization problem (2) using the Lagrange-multiplier method (3) or (4), the Lagrangian dynamic system (19) can be considered as an unconstrained nonlinear optimization problem.

4.1. NOVEL: A Trace-Based Search Method

We have developed *NOVEL* [14, 12], a hybrid global- and local-search method for constrained and unconstrained global optimization. It is a trajectory-based method that relies on an *external* force to pull the search out of local minima, and employs local searches to locate local minima. *NOVEL* has three components: exploring the search space, locating promising regions, and finding local minima. In exploring the search space, the search is guided by a continuous terrain-independent *trace* that does not get trapped in local minima. In locating promising regions, *NOVEL* uses local gradients to attract the search to a local minimum but also relies on the trace to pull it out once little improvement can be found. Finally, *NOVEL* selects one initial point for each region containing promising local minima, and uses it as an initial point for a local search algorithm to find local minima.

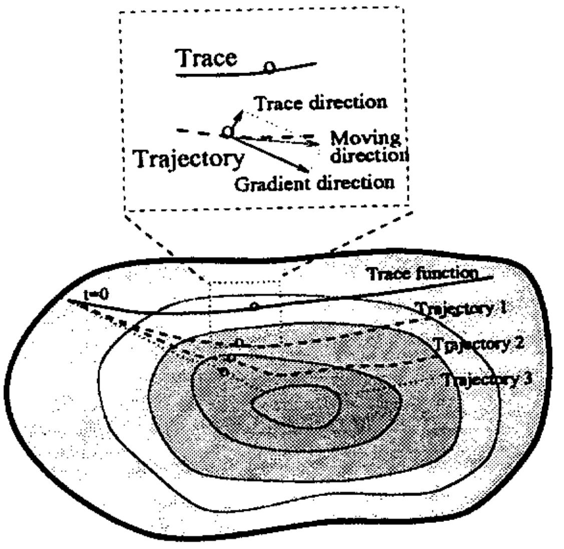

Figure 3. *NOVEL* has two phases: global search and local refinement. In the global search phase, the trajectory shows a combined effect of gradient descents and pull exerted by the moving trace. In the local search phase, the trajectory is sampled to collect starting points for pure local descents.

In exploring the search space, the trace plays an important role in uncovering regions with new local minima. A *trace* is a continuous aperiodic function of (logical) time that generates a *trajectory*. At time 0, both the trace and the trajectory start at the same point. As the trace moves from point x_1 to point x_2, the trajectory moves from point y_1 to y_2, where y_2 is a function of the local gradient at y_1 and the distance between x_1 and y_1 (see Figure 3). These two counteracting forces (descents into local minima and attraction exerted by the trace) form a composite vector that represents the route taken by the trajectory.

When dealing with constrained problems formulated using Lagrangian functions, there are two different sets of variables, the original variables and the Lagrange multipliers. Intuitively, there is no need of the trace function in the Lagrangian space as the Lagrange multipliers actually indicate the degree to which inequalities or equalities are satisfied. As the trace pulls the search out of local saddle points and enters an infeasible region, the corresponding Lagrange multipliers will automatically increase and then attract the trajectory back into the feasible region. In this sense, the Lagrange multipliers are passive because they change with the constraints. If one also uses a trace function in the Lagrangian space, then the force imposed by the trace and that by the Lagrange multipliers may contradict. (In the original *NOVEL* implementation [14], a trace was also used in the Lagrangian space.)

The overall dynamic system for constrained optimization is described as follows.

$$\begin{aligned} \frac{d}{dt} X(t) &= -\eta_l \nabla_X L_q(X(t), \lambda(t), \mu(t)) - \eta_g * (X(t) \\ &\quad - T_X(t)) \\ \frac{d}{dt} \lambda(t) &= \nabla_\lambda L_q(X(t), \lambda(t), \mu(t)) \qquad (21) \end{aligned}$$

$$\frac{d}{dt}\mu(t) = \begin{cases} \nabla_{\mu} L_q(X(t),\lambda(t),\mu(t)) & g(X) \leq 0 \\ -r \times \mu \times sign[-g(X)] & \\ \nabla_{\mu} L_q(X(t),\lambda(t),\mu(t)) & \text{Otherwise} \end{cases}$$

where η_l and η_g are constant controlling the relative weights between local search and global exploration.

The dynamic system is used in both the global-search and the local- search phases. There are three stages in the global-search phase, each of which outputs a trajectory based on (21). In the first stage of the global search, the user-defined trace function $T_X(t)$ leads the trajectory to form Trajectory 1 in Figure 3. In the second and third stages of the global search, the trace function $T_X(t)$ is defined as the trajectory resulted from the previous stage. According to the trajectory output from the three stages, we identify a set of promising starting points and perform local searches from them. The final result is the best solution among all these local searches.

NOVEL uses a continuous trace to travel through a problem space in order to produce a terrain-specific trajectory of $(X(t),\lambda(t),\mu(t))$. Thus, designing a good initial trace function $T_X(t)$ is a very important problem. Four criteria have been considered up to now. First, the trace should be aperiodic so that it does not return to the same starting points and regenerates possibly the same trajectory. Second, the trace needs to be continuous in order to be differentiable. This allows the generated trajectory to follow the terrain in a continuous manner without restarting to new starting points. Third, the trace should be bounded so that it will not explore unwanted regions. Last, the trace should be designed to travel from coarse to fine so that it examines the search space in greater details when more time is allowed.

Since the design of a good trace function is an intractable functional programming problem, we have studied a number of heuristic functions and fine-tuned them. Based on substantial experiments, we have designed a non-periodic, analytic trace function as follows.

$$T_i(t) = \rho \, sin\left[2\pi\left(\frac{t}{2}\right)^{0.95+0.45\frac{i-1}{n}} + 2\pi\frac{(i-1)}{n}\right] \quad (22)$$

where i represents the i'th dimension, ρ is a coefficient specifying the range, and n is the dimension of the original variables X.

4.2. Discontinuities in the Function Space

In some problems, there may exist points with infinite gradients or function values. A continuous differential-equation solver will fail to work when such points are approached because the corresponding gradients become extremely large, and no suitable step size can be used to lead to convergence.

One way to solve this problem is to scale the Lagrangian function (12), *i.e.*, $L_f(X,\lambda,\mu) = L_g(X,\lambda,\mu)/S$ by a common constant S that is dynamically changed. This strategy does not work well in most cases because not all variables will cause very large derivatives of the Lagrangian function, and scaling some with small derivatives may lose accuracy in convergence. Our strategy is to scale each variable independently to result in the following scaled Lagrangian function:

$$L_s(X,\lambda,\mu) = L_q(X/A,\lambda/B,\mu/C) \quad (23)$$

where $A = [a_1,\cdots,a_n]^T$ are the scaling variables for X, and X/A is carried out for each variable individually, namely, $X/A = [x_1/a_1,\cdots,x_n/a_n]^T$. Similarly, B and C are the scaling variables associated with Lagrange multipliers λ and μ, respectively.

We require all scaling variables $A, B, C \geq 1$. The goal of scaling is to expand only those variable axes whose derivatives become very large while preserving other variables. Initially, all scaling variables are set to one ($A = B = C = 1$), and the scaled Lagrangian function is the same as the original one.

Suppose the current point is (X,λ,μ). If the derivative $dx_j = \nabla_{x_j} L_s(X,\lambda,\mu) > \gamma$ where γ is a prespecified constant ($\gamma = 100$ in our experiments), we increase the value of a_j to $\hat{a}_j = a_j * dx_j$ and also the value of variable $\hat{x}_j = x_j * dx_j$. It is easy to show that the derivative $\nabla_{\hat{x}_j} L_s(X,\lambda,\mu) = 1$, and other derivatives remain the same as before. When the value of $a_j > \beta$ ($\beta = 10^{15}$ in our experiments), we consider the current point to be where its derivative approaches infinity and fix this variable to this value for the duration of the search. This means that this variable is excluded from the dynamic system. At this time, the local search will be in the subspace of the original space with variable x_j fixed. On the other hand, when the derivative dx_j of x_j decreases after scaling ($a_j > 1$) to be less than a threshold (10^{-3} in our experiments), we scale variable x_j back to its original scale.

5. EXPERIMENTAL RESULTS

In this section we describe our experimental results on some existing benchmarks [3]. These benchmarks are challenging because they model practical applications that have been studied extensively in the past. As a result, improvements are generally difficult.

We use the same set of parameters, such as step size and trace function, to solve all the problems. The reason for not tuning the parameters is to avoid any bias, as good solutions can always be obtained by sufficient tuning. We further set the starting points of *NOVEL* as those suggested in the benchmark set. The only exception is the problem-specific search range, which we set manually based on the solutions reported in the benchmarks. In practice, this is reasonable as search ranges are generally known. In cases that the search range is not available, we use trial and error, starting from a small range and gradually increasing it until no improvement in solutions can be found.

Table 1 summarizes the results found by *NOVEL*. Column 1 lists the problem identifications that appear in the benchmark collection [3]. Column 2 shows the problem-dependent search range that the trace function covers. Column 3 gives the CPU time limit spent by *NOVEL* on each problem. The time limit is set in an ad hoc fashion because the relationship between solution quality and computation time is unknown for a given problem. Column 4 shows the best known solutions reported in [3], and Column 5, the solutions reported by Epperly [2]. Here, symbol '–' means that the method was not able to find a solution for the corresponding problem. Column 6 shows the results obtained by *NOVEL* using the slack-variable method without any scaling. Due to oscillations described in Section 2.2., many of these problems could not be solved.

We overcome these oscillations by trial and error scaling of the objective functions, and Column 7 gives the results. Column 8 shows the results obtained by *NOVEL* with the *MaxQ* method. Results in bold font are improved by *NOVEL* over the best known results, with improvements of up to 10%. Our results indicate that *NOVEL* is robust in discovering new regions and in escaping from local traps.

Table 1. Results on a collection of constrained global optimization benchmarks [3] comparing *NOVEL* using *MaxQ*, *NOVEL* using the slack-variable method, and Epperly's method [2]. Search times are in seconds on a Sun SS 10/51 computer. Improved solutions found by *MaxQ* are indicated in bold font. Symbol '–' means that the method was not able to find a solution for the corresponding problem.

Problem ID	Search Range	*NOVEL* Search Time Limit	Best Known Solutions	Epperly's Solutions	Slack w/o Scaling Solutions	Slack w/ Scaling Solutions	MaxQ Solutions
2.1.1	1.0	3279	-17.00	-17.00	-	-17.00	-17.00
2.2.1	10.0	5856	-213.00	-213.00	-	-213.00	-213.00
2.3.1	10.0	57404	-15.00	-15.00	-	-15.00	-15.00
2.4.1	10.0	29829	-11.00	-11.00	-11.00	-11.00	-11.00
2.5.1	1.0	2937	-268.00	-268.00	-	-268.00	-268.00
2.6.1	1.0	3608	-39.00	-39.00	-	-39.00	-39.00
2.7.1(1)	40.0	68563	-394.75	-394.75	-	-394.75	-394.75
2.7.1(2)	40.0	51175	-884.75	-884.75	-	-884.75	-884.75
2.7.1(3)	40.0	170751	-8695.00	-8695.00	-	-8695.00	-8695.00
2.7.1(4)	40.0	203	-754.75	-754.75	-	-754.75	-754.75
2.7.1(5)	40.0	97470	-4150.40	-4150.40	-	-4150.40	-4150.40
2.8.1	25.0	158310	15990.00	15990.00	-	**15639.00**	**15639.00**
3.1.1	5000.0	352305	7049.25	-	-	7049.25	7049.25
3.2.1	50.0	47346	-30665.50	-30665.50	-	-30665.50	-30665.50
3.3.1	10.0	803	-310.00	-310.00	-	-310.00	-310.00
3.4.1	5.0	199	-4.00	-4.00	-	-4.00	-4.00
4.3.1	5.0	20890	-4.51	-4.51	-4.51	-4.51	-4.51
4.4.1	5.0	73	-2.217	-2.217	-2.217	-2.217	-2.217
4.5.1	5.0	16372	-11.96	**-13.40**	-	**-13.40**	**-13.40**
4.6.1	5.0	4435	-5.51	-5.51	-5.51	-5.51	-5.51
4.7.1	5.0	423	-16.74	-16.74	-16.75	-16.75	-16.75
5.2.1	50.0	240829	1.567	-	1.567	1.567	1.567
5.4.1	50.0	374850	1.86	-	1.86	1.86	1.86
6.2.1	100.0	3017	400.00	400.00	400.00	400.00	400.00
6.3.1	100.0	2756	600.00	600.00	600.00	600.00	600.00
6.4.1	100.0	3340	750.00	750.00	750.00	750.00	750.00
7.2.1	100.0	162643	56825.00	-	56825.00	56825.00	56825.00
7.3.1	150.0	228320	46266.00	-	-	46266.00	**44903.00**
7.4.1	150.0	631029	35920.00	-	-	35920.00	35920.00

6. CONCLUSIONS

In this paper we have proposed and studied *MaxQ*, a method to handle inequality constraints in nonlinear constrained optimization. We have applied *MaxQ* in *NOVEL* [14, 12], a global optimization system we have developed to solve constrained as well as unconstrained optimization problems. *NOVEL* generates information-bearing trajectories in its global search phase based on a user-defined trace function, and samples these trajectories for good starting points in its local search phase. We have tested a number of benchmark problems derived from manufacturing, computed aided design, and other engineering applications and have compared *MaxQ* to a method based on slack variables [9]. Our results show that *MaxQ* is more robust in convergence and has found solutions that are either better than or the same as existing solutions. Our future work in this area will be on finding better trace functions, parallelizing the execution on massively parallel computers, and studying many challenging applications in neural network learning and signal processing.

REFERENCES

[1] A. Ben-Tal, G. Eiger, and V. Gershovitz. Global minimization by reducing the duality gap. *Mathematical Programming*, 63:193–212, 1994.

[2] T. Epperly. *Global Optimization of Nonconvex Nonolinear Programs Using Parallel Branch And Bound.* PhD thesis, University of Wisconsin-Madison, 1995.

[3] C. A. Floudas and P. M. Pardalos. *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[4] C. A. Floudas and P. M. Pardalos, editors. *Recent Advances in GLobal Optimization.* Princeton University Press, 1992.

[5] E. R. Hansen. *Global optimization using interval analysis.* M. Dekker, New York, 1992.

[6] R. Horst and H. Tuy. *Global optimization: Deterministic approaches.* Springer-Verlag, Berlin, 1993.

[7] L. Ingber. *Adaptive Simulated Annealing (ASA).* Lester Ingber Research, 1995.

[8] A. E. W. Jones and G. W. Forbes. An adaptive simulated annealing algorithm for global optimisation over continuous variables. *Journal of Optimization Theory and Applications*, 6:1–37, 1995.

[9] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, 1984.

[10] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs.* Springer-Verlag, 1994.

[11] P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications*, volume 268 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[12] Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29:45–54, March 1996.

[13] A. Törn and A. Žilinskas. *Global Optimization*. Springer-Verlag, 1989.

[14] B. W. Wah and Y.-J. Chang. *Trace-Based Methods for Solving Nonlinear Global Optimization Problems*. J. of Global Optimization, (accepted to appear in special issue on SAT problems) 1996.