method for underdetermined systems. The test results indicate that this technique works well.

keywords: constrained optimization, underdetermined systems, generalized inverse, feasibility

1 Introduction

feasible region is $O(L^{n \epsilon p}$

DO for I=1 to Ntries

1. Randomly generate a point in the search region.
2. Take the point in step 1 as an initial guess and call
   a routine that iteratively finds an approximate feasible point.

3. If $\|\tilde{X} - X\| \le \varepsilon_{\text{domain}} \max\{\|X\|, 1\}$, then
   Return $\tilde{X}$; STOP
   Else
   $X \leftarrow \tilde{X}$; Go to step 1.
   End if

End Algorithm 3

This technique has disadvantages. Inactive inequality constraints can be ignored. But if inequality constraints are transformed to equality constraints, they will always be present in the system. Transforming to equality constraints increases the number of independent variables and number of bound constraints, so each step is more costly. Also, the entire approximate feasible point scheme is often embedded into a global optimization algorithm, and such algorithms are sometimes less efficient when the number of bound constraints is too large.

Next, we present our algorithm that handles inequality constraints without slack variables.

Algorithm 4 (For the step 2 of Algorithm 2)

1. If X is approximately feasible, then
   Return X; STOP
   End if
2. If $\max\{g$ [illegible] Return

2f) If the present system of inequality constraints is not empty and $\max\{g_j(\tilde{X})\,|\,j = 1, 2, \ldots, q'\} > \bullet$, then

Go to 2c

End if

End if

3. If $\|\tilde{X} - X\| < \bullet_{\text{domain}} \max\{\|X\|, 1\}$, then

Return $\tilde{X}$; STOP

Else

$X \leftarrow \tilde{X}$; Go to step 1.

End if

End Algorithm 4

Comparing with transforming to equality constraints, the technique of -31.23-14.e4-14.ehasIf theX

5 Test Problems and Test Environment

5.1 The Test Set

The set of test problems is the same as that in [5]. Five problems were taken from [3]. They were selected to be non-trivial problems with a variety of constraint types, as well as di□ering numbers of variables, inequality constraints, equality constraints and bound constraints. The remaining three problems were taken from [8]. They are relatively simpler. Each problem is identified with a mnemonic, given below.

fphe1 is the first heat exchanger network test problem in [3, pages 63-66].

fpnlp3 is the third nonlinear programming test problem in [3, page 30].

fpnlp6 is the sixth nonlinear programming test problem in [3, page 30].

fppb1 is the first pooling-blending test problem in [3, page 59].

fpqp3 is the third quadratic programming test problem in [3, pages 8-9].

gould is the first test problem in [8].

bracken is the second test problem in [8].

wolfe3 is the third test problem in [8].

5.2 Implementation Environment

The algorithms in *x*2, *x*3 and *x*4 were programmed in the Fortran 90 envi-
xonment developed and described in [6]. Similar Finscribedfr

The Sun Fortran 90 compiler version 1.2 was used on a Sparc Ultra model 140. Execution times were measured using the routine `DSECND`. All times are given in terms of Standard Times Units (STU's), defined in [2], pp 12–14.

Table 1: Results of the Three Methods

[redacted]

For all tests, the error tolerance for both equality and inequality constraints is 10^{-6}.

With pure random search, we found no approximate feasible points for problems fphe1, fppb1 and wolfe3 when we used 10^6 randomly generated