ANALYSIS OF A NONHIERARCHICAL DECOMPOSITION ALGORITHM

by

Jayashree Shankar

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

APPROVED:

Layne J. Watson

Layne T. Watson

Raphael T. Haftka

Calvin J. Ribbens

August, 1992

Blacksburg, Virginia
ANALYSIS OF A NONHIERARCHICAL DECOMPOSITION
ALGORITHM FOR LARGE SCALE OPTIMIZATION

by

Jayashree Shankar

Committee Chairman: Layne T. Watson
Computer Science

(ABSTRACT)

Large scale optimization problems are tractable only if they are somehow decomposed. Hierarchical decompositions are inappropriate for some types of problems and do not parallelize well. Sobieszczanski-Sobieski has proposed a nonhierarchical decomposition strategy for nonlinear constrained optimization that is naturally parallel. Despite some successes on engineering problems, the algorithm as originally proposed fails on simple two dimensional quadratic programs.

Here, the algorithm is carefully analyzed by testing it on simple quadratic programs, thereby recognizing the problems with the algorithm. Different modifications are made to improve its robustness and the best version is tested on a larger dimensional example. Some of the changes made are very fundamental, affecting the updating of the various tuning parameters present in the original algorithm.

The algorithm involves solving a given problem by dividing it into subproblems and a final coordination phase. The results indicate good success with small problems. On testing it with a larger dimensional example, it was discovered that there is a basic flaw in the coordination phase which needs to be rectified.
ACKNOWLEDGEMENTS.

At the outset I would like to thank Dr. Layne T. Watson for taking me in as his student and guiding me through my Masters program. His willingness to share his knowledge is quite unmatched. His wisdom and experience of technically sound publication has helped me immensely in the preparation of this thesis.

I am very grateful to Dr. Raphael T. Haftka for serving on my committee and for being a source of constant encouragement. I also acknowledge his experience and knowledge which were instrumental in guiding us in our research.

My thanks are also due to Dr. Calvin J. Ribbens for serving on my committee and for all his suggestions regarding my research and thesis.

I would like to thank my husband Shankar for being by my side and helping me during the rough times.

Last but not the least I thank my daughter, little Nikhila, for being so patient and understanding and letting “mommy” work those long and late hours.

It is my greatest pleasure to dedicate this thesis to my loving parents whose dream it was to see me accomplish this and whose love helped me fulfill it.
# TABLE OF CONTENTS

1. Introduction ........................................................................................................ 1

2. Problem Statement and Simple Sequential Decomposition ................................. 4
   2.1. Problem Statement .......................................................................................... 4
   2.2. Simple Sequential Decomposition .................................................................. 5

3. Decomposition with Approximate Coupling ....................................................... 6
   3.1. Cumulative Constraints .................................................................................. 6
   3.2. Subspace Optimization Problem .................................................................... 6
   3.3. Coordination Optimization Phase .................................................................... 8

4. Pseudocode for Algorithm .................................................................................... 10

5. Initial Tests .......................................................................................................... 12
   5.1. 2x2 Example ................................................................................................. 12

6. Modifications to the Original Algorithm ............................................................. 14

7. Further Tests ........................................................................................................ 18
   7.1. Modified algorithm on 2x2 Example .............................................................. 18
   7.2. 3x3 Example ................................................................................................. 19
   7.3. 6x6 Example ................................................................................................. 24

8. Quadratic Programming Algorithms .................................................................... 29
   8.1. Active set strategy - Fletcher ......................................................................... 29
   8.2. QPSOL .......................................................................................................... 33
   8.3. MINOS .......................................................................................................... 34

9. Historical Development of Present Algorithm .................................................... 35
   9.1. The switch coefficient s ................................................................................ 35
   9.2. An infeasible subproblem .............................................................................. 36
   9.3. Bounds on the t-coefficients ......................................................................... 37
   9.4. A description of the different versions .......................................................... 38
LIST OF FIGURES

Figure 1. Trace of the solution iterates for Example 1, corresponding to $\beta = 0.1$,
with starting point (2,3) and solution (0.198,1.98) .................. 20

Figure 2. Blown up view of the region of convergence of the plot shown in Fig. 1. 21

Figure 3. Trace of the solution iterates for Example 2, corresponding to $\beta = 0.5$,
with starting point (0,1,-3) and solution (0.88,0.88,0.44) ................. 23
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table I</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table II</td>
<td>18</td>
</tr>
<tr>
<td>Table III</td>
<td>19</td>
</tr>
<tr>
<td>Table IV</td>
<td>22</td>
</tr>
<tr>
<td>Table V</td>
<td>27</td>
</tr>
<tr>
<td>Table VI</td>
<td>27</td>
</tr>
<tr>
<td>Table VII</td>
<td>28</td>
</tr>
<tr>
<td>Table VIII</td>
<td>40</td>
</tr>
<tr>
<td>Table IX</td>
<td>40</td>
</tr>
<tr>
<td>Table X</td>
<td>40</td>
</tr>
<tr>
<td>Table XI</td>
<td>40</td>
</tr>
<tr>
<td>Table XII</td>
<td>41</td>
</tr>
<tr>
<td>Table XIII</td>
<td>41</td>
</tr>
<tr>
<td>Table XIV</td>
<td>41</td>
</tr>
<tr>
<td>Table XV</td>
<td>42</td>
</tr>
</tbody>
</table>
1. INTRODUCTION.

Many engineering problems involve large scale optimization over many different disciplines. As is the case with many large scale problems, a decomposition of the problem into subproblems helps reduce the time and complexity of solution. The strategy governing the decomposition of a large scale problem can directly affect the ease and accuracy of the solution. The concept of a linear decomposition strategy by Sobieski [33] has been used with good results in a number of cases. This method works very well in the case of a system that is amenable to such a decomposition, i.e., when subsystems can be laid out clearly in a hierarchical fashion.

For a system with many interdependencies between the probable subproblems, using a linear decomposition strategy implies choosing one subsystem before another, thereby establishing an artificial hierarchy. The order chosen will affect the solution iterates, making this strategy ill-suited or even nonconvergent for such nonhierarchic problems.

These considerations led Sobieski [32] to propose a new nonhierarchic decomposition strategy. The algorithm involves introducing various coefficients to represent the interdependencies (responsibility and trade-off coefficients) between the subsystems and solving the subsystems concurrently. The coordination phase for these subproblems involves updating these coefficients based on sensitivity information. This algorithm has been tested with success on engineering problems by Bloebaum/Hajela/Sobieski [7]. Also, heuristics [6] are employed to help with the solution of each subsystem, where human intervention is possible.

In this thesis we carry out a detailed analysis of this algorithm by Sobieski [32] and its feasibility. Since nonlinear optimization can be reduced to a series of quadratic programs, it is appropriate to study this new algorithm first on quadratic programs. Thus, this thesis first studies the various tuning parameters occurring in this algorithm, using a model quadratic programming problem. A series of experiments shows that modifications to the algorithm as originally proposed by Sobieski [32] are necessary for convergence. This modified algorithm is then used to solve problems involving a number of subsystems, each with a varying number of design variables.
The tests are carried out on quadratic programming (QP) problems of different dimensions. The decomposition then yields subproblems which are also QP problems. The method employed to solve these smaller QP problems is an active set strategy described in Fletcher [12]. Also optimization packages such as MINOS [23] and QPSOL [16] were used to verify the answers.

QPSOL is an optimization package that solves quadratic and linear programming problems. The coordination phase of the algorithm presented in this thesis is a linear programming problem that is solved in order to update the coefficients that mark the coupling of the subsystems. LPSOL (part of QPSOL) is used to solve this linear programming problem.

An alternative to the coordination phase by Renaud/Gabriele is given in [27]. Here, instead of having the coordination phase give an updated vector of the interdependency coefficients, an approximation to the global problem is solved to get an updated design vector. In [37] Sobieski gives two alternative ways to solve the coordination optimization phase as is used in [33].

An algorithm for optimization based design of non-hierarchic systems is given by Wu/Azarm in [48]. It uses decomposition techniques to solve the given problem in two stages. Details for handling multidisciplinary systems are not included. In [24] Pan/Diaz give another method for optimization by non-hierarchic decomposition. Here, the subspace optimizations are solved sequentially and not concurrently and global convergence is guaranteed. This is as a result of the subsystems being solved sequentially and by a method given in [24] for moving away from pseudo optimal points (points that are optimal for every subsystem, but not optimal for the global problem) and towards the optimal solution. An obvious disadvantage is the sequential nature of the subspace optimizations.

The fact that the subsystems are solved concurrently in the algorithm given in this thesis makes it naturally parallel. The sensitivity derivatives can also be obtained in parallel too. Future work could involve parallelization of the best serial algorithm given in this thesis after the coordination optimization phase is modified to solve the problems described in
Chapter 7. Towards that end the papers given in [8], [40], [41], [19], [38], and [20] may be useful. They may help in the efficient parallelization of the algorithm.

Chapter 2 includes a detailed description of the problem and the notation employed, and also a description of the simple decomposition technique that originally introduced the concept of subspace optimization. Chapter 3 explains the algorithm with approximate coupling in detail. The pseudocode for the algorithm is given in Chapter 4. The results of some initial tests of the algorithm as it was proposed by Sobieski in [32] are tabulated in Chapter 5. Chapter 6 describes the modifications that were made to the original algorithm. Chapter 7 records the examples that the algorithm was tested on and the results obtained. The quadratic programming algorithm used to solve the individual subproblems is described in Chapter 8. Also, the algorithms used in the optimization packages QPSOL and MINOS are given there. Chapter 9 gives an account of the historical development of the current version of the algorithm from the original one. Chapter 10 concludes.
2. PROBLEM STATEMENT AND SIMPLE SEQUENTIAL DECOMPOSITION.

2.1 Problem Statement.

Although in this thesis we are only dealing with quadratic programming problems, this algorithm [32] is meant to solve any nonlinear programming problem. Therefore the algorithmic details are for any nonlinear programming problem.

Consider the following nonlinear programming problem (NLP),

\[
\begin{align*}
\min_z \theta(x) \\
\text{subject to} \quad g(x, y) &\leq 0, \\
h(x, y) &\quad = 0,
\end{align*}
\]  

(2.1)

where \( x \in \mathbb{E}^m \), \( y \in \mathbb{E}^p \), \( g \) is an \( m \)-dimensional vector function and \( h \) is a \( p \)-dimensional vector function. \( x \) is the set of design variables and \( y \) is the set of behavior variables which are the unknowns in each subsystem.

Subspace optimization involves solving a problem by solving a set of subproblems. The global problem is divided into subproblems according to some logical way of partitioning the problem. In the scheme that originally introduced subspace optimization, which we shall refer to as simple sequential decomposition, there is not much more to the solution of the global problem than the solution of these subproblems iteratively until the same solution vector solves each of the individual subproblems.

To outline the differences between the current scheme and simple decomposition, we introduce the following terminology:

\[
x = (X^1, X^2, \ldots, X^N), \quad X^i \in \mathbb{E}^{n_i}, \quad n_1 + n_2 + \ldots + n_N = n, \\
y = (Y^1, Y^2, \ldots, Y^N), \quad Y^i \in \mathbb{E}^{p_i}, \quad p_1 + p_2 + \ldots + p_N = p, \\
g = \begin{pmatrix} g^1 \\ \vdots \\ g^N \end{pmatrix}, \quad h = \begin{pmatrix} h^1 \\ \vdots \\ h^N \end{pmatrix}, \\
h^i(x, y) \in \mathbb{E}^{p_i}, \quad g^i(x, y) \in \mathbb{E}^{n_i}, \quad m_1 + \ldots + m_N = m, \\
h^i(x, y) = Y^i - h^i(x, Y^1, \ldots, Y^{i-1}, Y^{i+1}, \ldots, Y^N).
\]  

(2.2)
The subvector $X^i$ is the set of design variables corresponding to the $i$th subsystem. Similarly the subvector $Y^i$ is the set of behavior variables of the $i$th subsystem. For any vector function $f(x, y)$, let $\hat{f}(X^i, Y^i)$ denote $f$ with all the components $X^1, \ldots, X^{i-1}, X^{i+1}, \ldots, X^N, Y^1, \ldots, Y^{i-1}, Y^{i+1}, \ldots, Y^N$ fixed except for $X^i$ and $Y^i$. Note the assumption that each $Y^i$ can be explicitly determined in terms of $x$ and the other subvectors $Y^j$.

2.2 Simple Sequential Decomposition.

The approach is to first divide the given large problem into a set of independent subproblems, corresponding naturally to the subsystems comprising the larger system. The $i$th subsystem would be

$$\min_{X^i} \hat{\Theta}(X^i)$$

subject to

$$\hat{g}^i(X^i, Y^i) \leq 0, \quad (2.3)$$
$$\hat{h}^i(X^i, Y^i) = 0,$$

where the system of equalities $\hat{h}^i = 0$ is used to eliminate $Y^i$ from $\hat{g}^i$. The subproblems are solved sequentially for $i = 1, \ldots, N$, with one pass through all the subsystems constituting one outer iteration. The outer iterations are repeated until the same point $(\bar{x}, \bar{y})$ solves all $N$ subproblems. While solving the $i$th subsystem the values of $X^1, \ldots, X^{i-1}, X^{i+1}, \ldots, X^N, Y^1, \ldots, Y^{i-1}, Y^{i+1}, \ldots, Y^N$ are fixed. They can be chosen in a Gauss-Seidel manner where the first $i - 1$ $X$ and $Y$ subvectors used have their latest values from solving the first $i - 1$ subproblems. A parallel algorithm, solving the subproblems concurrently, would use a Jacobi scheme where the values of all the $X^j$ and $Y^j$ vectors are updated only at the end of each major outer iteration. The ensuing discussion of the current scheme assumes a Jacobi scheme.
3. DECOMPOSITION WITH APPROXIMATE COUPLING.

3.1 Cumulative Constraints.

In the scheme proposed by Sobieski [32], a measure of the constraints in each of the other subsystems is also brought into the ith subsystem in the form of one cumulative constraint $C_i^k$ per subsystem. The approximate cumulative constraint $C_i^k$ of the kth subsystem in the ith subsystem is obtained from the corresponding constraints $g^k \in E^{m_k}$ as a linearization of the Kreisselmeier-Steinhauser cumulative constraint

$$K_k(x, y) = \frac{1}{\rho} \ln \left( \sum_{j=1}^{m_k} e^{\rho g^k_j(x, y)} \right). \quad (3.1)$$

The $\rho$ in the Kreisselmeier-Steinhauser function is a constant used to control the accuracy of the cumulative constraint approximation. The linearization of this cumulative constraint of the kth subsystem with respect to the variables of the ith subsystem is

$$C_i^k(X^i, Y^i) = K_k(X_0^i, Y_0^i) + \sum_{j=1}^{m_i} \frac{\partial K_k}{\partial X_j^i}(X_0^i, Y_0^i)(X_j^i - (X_0^i)_j). \quad (3.2)$$

3.2 Subspace Optimization Problem.

In the ith subsystem the cumulative constraints of the other subsystems are brought in as constraints. Therefore, a violated cumulative constraint of one subsystem may be satisfied by decisions taken in every one of the other subsystems. Therefore, we introduce coefficients $r_i^p$ to represent the fractional “responsibility” assigned to the ith subsystem for reducing the violation of the cumulative constraint of the pth subsystem, for each $p = 1, \ldots, N$. Thus we have $N^2$ r-coefficients. The $r_i^p$’s are defined in such a way that

$$\sum_{i=1}^{N} r_i^p = 1, \quad (3.3)$$

Sobieski [32] suggested the initialization of the r-coefficients in such a way that they are proportional to the degree of influence exerted by the ith subsystem on the pth cumulative constraint. This initialization is discussed in Appendix A.
To further reduce the objective function we allow cumulative constraints to be violated in one subsystem, provided that the violation will be offset by oversatisfaction of that constraint in another subsystem. To account for such tradeoffs, we introduce the $N^2$ coefficients $t^p_i$, corresponding to the cumulative constraint of the $p$th subsystem when present in the $i$th subsystem. For the $p$th cumulative constraint,

$$\sum_{i=1}^{N} t^p_i = 0,$$

maintains the constraint at a value of zero. This condition and the condition on the $r$-coefficients are enforced in what is called the coordination optimization phase, which is solved to update the values of the $r$'s and the $t$'s at the end of every outer iteration. The $t^p_i$'s are initialized at the beginning of the algorithm to zero.

As has been described above, the $r^p_i$'s are needed only in the case of a violation and the $t^p_i$'s only when the constraints are critical, therefore only one of the two is needed at a time. Therefore we introduce $N$ coefficients $s^p$ which act as switches, one for each of the cumulative constraints of the subsystems. $s^p$ is set to one (activating the $r$-coefficients) if the corresponding constraint $K_p \leq 0$ is violated at the outset of the system optimization procedure and stays at one until the $K_p$ is driven to a critical status (zero value). Once $K_p$ becomes critical, $s^p$ is reset to zero (activating the $t$-coefficients) and stays at zero until the system optimization procedure terminates. The switch $s^p$ is applied selectively to the natural constraints $g^i$ of the $i$th subsystem (i.e., the constraints that are assigned to the $i$th subsystem) by multiplying the $r$-coefficient $r^i$ by a factor of $\max\{g^i(X_0^i, Y_0^i), 0\}$, so that constraints which are already satisfied are not taken into consideration.

Thus, the $i$th subsystem optimization problem is

$$\min_{X^i} \tilde{\Theta}(X^i)$$

subject to

$$\hat{g}^i(X^i, Y^i) \leq s^i \max\{\hat{g}^i(X_0^i, Y_0^i), 0\}(1 - r^i) + (1 - s^i)t^i,$$

$$C^i_p(X^i, Y^i) \leq \hat{K}_p(X_0^i, Y_0^i) s^p(1 - r^p) + (1 - s^p)t^p,$$

$$p = 1, \ldots, i - 1, i + 1, \ldots, N,$$

$$\hat{h}^i(X^i, Y^i) = 0.$$
3.3 Coordination Optimization Phase.

The constrained minimum of $\Theta$ obtained from each subsystem optimization is a function of the constants $r_i^p$ and $t_i^p$, and its partial derivatives with respect to $r_i^p$ and $t_i^p$ (assuming they exist) can be computed from the expressions given in Appendix A using gradient information for the $\Theta$ and $C$ functions. These derivatives are used for a linear approximation of $\Theta$ that is the objective function for the coordination optimization phase, the last (and synchronizing) step of an outer iteration.

The coordination optimization phase (COP) solves a linear program to adjust the coefficients $r_i^p$ and $t_i^p$, so that the objective function $\Theta$ will be further reduced (if possible) at the end of the next outer iteration. The linear program uses a linear extrapolation of $\Theta$ based on the partial derivatives $\partial \Theta / \partial z$ described above. Here $z$ represents either an $r$ or a $t$ coefficient. Move limits (upper and lower bounds $U_r^p$, $U_t^p$, $L_r^p$ and $L_t^p$ for $r_i^p$ and $t_i^p$, respectively) are needed to prevent large changes in the $r$- and $t$-coefficients caused by the nonlinearity of the original problem. For the first COP execution, the $r_i^p$'s may be initialized as already suggested and the $t_i^p$'s are initialized to zero. For every subsequent execution, the $r_i^p$'s and the $t_i^p$'s are initialized to the terminal values from the previous COP execution.

The result of the COP execution is a new set of $r_i^p$'s and $t_i^p$'s to be used in the next outer loop of subsystem optimizations. The adjustment of the $r_i^p$'s and $t_i^p$'s to the new values amounts to a reassignment of the responsibility for eliminating the constraint violations among the subsystems and to issuing a new set of instructions about trading the constraint violations/oversatisfactions among these subsystems. Let $(x_0, y_0)$ be the current updated point (the result of the Jacobi outer iteration) and

$$\Theta_1 = \Theta(x_0, y_0) + \sum_{p=1}^{N} \sum_{i=1}^{N} \frac{\partial \Theta}{\partial r_i^p} \Delta r_i^p + \sum_{p=1}^{N} \sum_{i=1}^{N} \frac{\partial \Theta}{\partial t_i^p} \Delta t_i^p, \quad (3.6)$$

where $\Delta r_i^p = (r_i^p - (r_i^p)_0)$ and $\Delta t_i^p = (t_i^p - (t_i^p)_0)$. The partial derivatives $\partial \Theta / \partial r_i^p$ and $\partial \Theta / \partial t_i^p$ are evaluated at the optimal point computed by the $i$th subsystem optimization.

Let

$$R = (r_1^1, r_1^2, \ldots, r_1^N, r_2^1, \ldots, r_2^N, \ldots, r_N^1, \ldots, r_N^N) \quad (3.7)$$
\begin{equation}
T = (t_1^1, t_1^2, \ldots, t_1^N, t_2^1, \ldots, t_2^N, \ldots, t_N^1, \ldots, t_N^N).
\end{equation}

Then $\Theta_1$ is a function of $R$ and $T$. The linear program solved during the coordination optimization phase is:

\begin{equation}
\begin{aligned}
\min_{R,T} & \quad \Theta_1(R, T) \\
\text{subject to} & \quad \sum_{k=1}^{N} r_k^p = 1, \quad p = 1, \ldots, N, \\
& \quad \sum_{k=1}^{N} t_k^p = 0, \quad p = 1, \ldots, N, \\
& \quad 0 \leq r_k^p \leq 1, \quad p = 1, \ldots, N, \quad k = 1, \ldots, N, \\
& \quad L_k^p \leq r_k^p \leq U_k^p, \quad p = 1, \ldots, N, \quad k = 1, \ldots, N, \\
& \quad \tilde{L}_k^p \leq t_k^p \leq \tilde{U}_k^p, \quad p = 1, \ldots, N, \quad k = 1, \ldots, N.
\end{aligned}
\end{equation}
4. PSEUDOCODE FOR ALGORITHM.

An algorithmic description of the whole process in pseudo-code is given next, using the following model quadratic programming problem (without the variables y and equality constraints \( h(x, y) = 0 \)) for specificity:

\[
\begin{align*}
\min_z & \quad x^T A x \\
\text{subject to} & \quad B x \leq d,
\end{align*}
\]

where

\[
A = \begin{pmatrix}
A_{11} & \alpha_{12} A_{12} & \cdots & \alpha_{1N} A_{1N} \\
\alpha_{12} A_{12} & A_{22} & \cdots & \alpha_{2N} A_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_{1N} A_{1N} & \alpha_{2N} A_{2N} & \cdots & A_{NN}
\end{pmatrix},
\]

\[
B = \begin{pmatrix}
B_{11} & \beta_{12} B_{12} & \cdots & \beta_{1N} B_{1N} \\
\beta_{21} B_{21} & B_{22} & \cdots & \beta_{2N} B_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{N1} B_{N1} & \beta_{N2} B_{N2} & \cdots & B_{NN}
\end{pmatrix},
\]

\[
d = \begin{pmatrix}
d_1 \\
d_2 \\
\vdots \\
d_N
\end{pmatrix}, \quad x = \begin{pmatrix}
X^1 \\
X^2 \\
\vdots \\
X^N
\end{pmatrix},
\]

and \( X^i \in E^{n_i}, A_{ij} \in E^{n_i \times n_j}, B_{ij} \in E^{n_i \times n_j}, d_i \in E^{p_i}, \) the \( A_{ii} \) are symmetric and positive definite and the \( \alpha_{ij}, \beta_{ij} \) are fixed “coupling” parameters, for all \( i, j = 1, \ldots, N \). Using notation defined in Appendix A, pseudo-code for the algorithm applied to this quadratic programming problem (QP) is:

Choose an initial estimate \( z \) and initialize the \( r-, s- \) and \( t-\)coefficients;

Repeat until minimum reached

begin

for \( i = 1 \) to \( N \) do

begin

Calculate the linearization \( C_{ij}^j(X^i) \) of the cumulative constraint for the \( j \)th subsystem,

for all \( j \neq i \);

end

end

10
Calculate the $i$th subsystem’s self responsibility

$$\delta^i = s^i \max \{ g^i (X^i_0, Y^i_0), 0 \} (1 - r^i_1) + (1 - s^i) t^i_1;$$

Solve the QP (ith subsystem)

$$\min_{X^i} \hat{\Theta}(X^i) = (X^i)^t A_{ii} X^i + 2 \left( \sum_{j \neq i} a_{ij} (X^i)^t A_{ij} X^j \right)$$

subject to

$$\sum_{j=1}^N \beta_{ij} B_{ij} X^j - d_i \leq \delta^i, \quad (\beta_{ii} = 1)$$

$$C^j_i (X^i) \leq 0, \quad \text{for all } j \neq i;$$

Calculate (if not already available) the Lagrange multipliers $\lambda$ using the method given in Appendix A.

Calculate $\frac{\partial \Theta^j}{\partial r^j_i}$ and $\frac{\partial \Theta^j}{\partial t^j_i}$ for $j = 1, \ldots, N$;

end

Solve the LP (Coordination Optimization Phase)

$$\min_{R, T} \Theta_1(R, T)$$

subject to

$$\sum_{k=1}^N r^p_k = 1, \quad \sum_{k=1}^N t^p_k = 0, \quad p = 1, \ldots, N,$$

$$0 \leq r^p_k \leq 1, \quad L^p_k \leq r^p_k \leq U^p_k, \quad \tilde{L}^p_k \leq t^p_k \leq \tilde{U}^p_k,$$

$$p = 1, \ldots, N, \quad k = 1, \ldots, N;$$

end (repeat)
5. INITIAL TESTS.

5.1 2 x 2 Example.

Testing of this algorithm was first performed on a simple 2 x 2 case:

Example 1.

\[
\min_x x_1^2 + x_2^2
\]

subject to \( x_1 + \beta x_2 \leq 4, \)

\[
\beta x_1 + x_2 \geq 2, \tag{5.1}
\]

where \( x = (x_1, x_2)^T \in \mathbb{R}^2. \)

Here each constraint is taken to be in a subsystem by itself with \( X^1 = (x_1) \) and \( X^2 = (x_2). \)

The results are tabulated in Table I. The column headings are the starting points, the last column gives the solutions for the different values of \( \beta, \) and each entry contains a convergence code and the number of iterations taken. The code IF means an infeasible subproblem is encountered at the very first iteration and the procedure is terminated, R means the solution is reached in the iteration indicated, but subsequently an infeasible subproblem is encountered, WR means a wrong point is reached before an infeasible subproblem causes termination, O means there is oscillation through the number of iterations indicated, WC means there is convergence to a point other than the solution, and NC means there is no convergence even after the number of iterations indicated.

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>R 1</td>
<td>R 1</td>
<td>R 1</td>
<td>R 1</td>
<td>R 1</td>
<td>(0.0,2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>WR 1</td>
<td>IF 1</td>
<td>WC 4</td>
<td>WC 4</td>
<td>IF 1</td>
<td>(0.198,1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>WR 1</td>
<td>IF 1</td>
<td>WC 5</td>
<td>WC 4</td>
<td>IF 1</td>
<td>(0.55,1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>WR 1</td>
<td>IF 1</td>
<td>O 150</td>
<td>NC 150</td>
<td>WR 1</td>
<td>(0.8,1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>O 150</td>
<td>O 150</td>
<td>O 150</td>
<td>O 150</td>
<td>O 150</td>
<td>(1.0,1.0)</td>
</tr>
</tbody>
</table>
As can be seen from the table, the main problem with the algorithm is not being able to deal with infeasibility in a subproblem. This issue of infeasibility was mentioned in the appendix of [33], and is addressed in the next section. Another reason for the algorithm not being successful is the way the s-coefficient is set permanently to zero once a constraint becomes critical. When a constraint that was once critical becomes violated, the r-coefficients cannot be brought in to reduce the violation.
6. MODIFICATIONS TO THE ORIGINAL ALGORITHM.

Several modifications and variations of the original algorithm as described in [32] are discussed next. The order of the topics is not significant.

Natural Constraints.

The original algorithm, while solving a particular subsystem, uses a cumulative constraint to represent the constraints of each of the subsystems, including those of the subsystem being solved. In the modified version we use a cumulative constraint for each of the other subsystems, but the natural constraints of the subsystem being solved are used instead of the cumulative constraint representing them. The algorithm described in Section 4 includes this modification.

Changes in setting of the switch coefficients $s^p$.

When a $K_p$ becomes critical, the corresponding $s^p$ is set to zero and stays at zero until the whole procedure terminates. This means that the term with the $r^p$ coefficient does not contribute any longer to the constraints. If the constraint becomes violated later, then the violation cannot be reduced using the $r$-coefficients. Hence two alternatives to the algorithm were considered. One was to remove the $s$-coefficient from the $r$ term. The other was to set the $s^p$ coefficient at the end of every outer iteration depending on whether the corresponding constraint was satisfied or not. This would make one of the $r$ or $t$-coefficients active all the time. The second alternative performed better in initial tests and therefore was selected.

Handling infeasibility.

Because of the linearization and the allocation of responsibility, some subproblems may be infeasible. The following procedure is employed to recover from infeasibility in a subsystem. A new variable $\omega$ is introduced in each of the constraints and a large multiple of this variable is added to the objective function to be minimized. Thus, the corresponding subproblem would now be

\[
\min_{x_i} \hat{\theta}(x_i) + M\omega
\]

subject to

\[
\hat{g}^i(x^i, y^i) - \omega \leq s^i \max\{\hat{g}^i(x^i_0, y^i_0), 0\}(1 - r^i) + (1 - s^i)t^i_i,
\]

\[
C^p_i(x^i, y^i) - \omega \leq \hat{K}_p(x^i_0, y^i_0) s^p(1 - r^p) + (1 - s^p)t^p_i,
\]

\[p = 1, \ldots, i - 1, i + 1, \ldots, N,\]
where $M$ is a large positive number.

The linear objective function in the Coordination Optimization Phase is formed using the Lagrange multipliers obtained from the subproblems. The introduction of the variable $\omega$ affects these Lagrange multipliers. The sensitivity derivatives are affected considerably because of these Lagrange multipliers, as indicated in Appendix A. A thought as to whether this was justified or not led to two variations of the algorithm. In the first variation, in case of an infeasibility in any subproblem, the Coordination Optimization Phase is omitted at the end of that outer iteration. In the second variation, the COP is included in every outer iteration.

*Limit on t-coefficients.*

Initially tests were performed with the $t$ coefficients left unbounded, but clearly this is unwise. A few variations for the bounds on the $t$'s were considered. One possibility is keeping the bound fixed throughout the procedure, but this may result in nonconvergence to the solution. Also the bounds should not decrease too fast, because this may force convergence to a nonoptimal solution.

The bound is reduced by a factor of $f = 0.8$ at the end of every outer iteration. Thus, if the bound at the first iteration is $t_1$, then the bound at the $m$th iteration is

$$t_m = 0.8^{(m-1)}t_1.$$  \hspace{1cm} (6.2)

A variation of having the bound at the $m$th iteration equal a factor $f^{\log(m-1)}$ or a factor $f^{(m-1)^{1/2}}$ of the bound at the first iteration was also considered. Bloebaum, Hajela, and Sobieski [7] also experimented with variable limits on the $t$-coefficients.

A later modification was to change the move limits on the $t$-coefficients based on information about the corresponding coefficients in the objective function of the COP. To ensure that no subsystem is allowed a violation that cannot be offset by an equivalent oversatisfaction in the other subsystems, a change was made to the move limits on the $t$-coefficients, using information about the corresponding sensitivities. The $a^{pk}$ coefficient (as described in Appendix A) is a measure of the sensitivity of the $p$th cumulative constraint to
the variables of the \( k \)th subsystem. The lower limit of \( t_k^p \) is now \( \max\{-a^{pk}, -t_m\} \), reasoning that the oversatisfaction expected of the \( p \)th cumulative constraint in the \( k \)th subsystem will be restricted to what it can handle. The \( a^{pk} \) coefficients are obtained at every iteration using the current value of the \( x \) vector.

The initial value of the move limit on the \( t \)-coefficient affects the results and the path taken. Various values of this initial limit were tried. Another alternative to starting with a larger limit on \( t \) is to fix the limit on the \( t \)-coefficient to some smaller value for a few iterations, and then perform the reduction as described earlier. This is to make sure that after a certain number of iterations, the \( t \)-coefficient can take on a value large enough to contribute to the solution process.

At the beginning of every major iteration involving a new \( \rho \), the initial value of the move limit on the \( t \)-coefficient may be reset to either the original value or some fraction of it.

**Convergence criterion.**

The convergence criterion initially involved a measure of the difference between three successive iteration values of the design vector. Later, tests revealed that with this criterion the procedure could stop even if there was a chance for further improvement, because of changes in the values of the \( r \) and \( t \)-coefficients. Hence the difference between the values of the \( t \) and \( r \)-coefficients were also included in the convergence criterion. Thus, if \( S_{m-2} \) represents the normalized form of the vector \((x, R, T)\) at the \((m-2)\)th iteration and similarly for \( S_{m-1} \) and \( S_m \), then using the 2-norm the convergence criterion is

\[
\|S_{m-1} - S_{m-2}\| + \|S_m - S_{m-1}\| \leq 0.0001. \tag{6.3}
\]

**Changes to the \( \rho \) coefficient.**

After the required convergence criterion is met the \( \rho \) coefficient is increased and the whole process is repeated again to check if the convergence criterion is still met; if not another major iteration is performed. This is because with increasing \( \rho \) the cumulative constraint is closer to the actual constraints. The process is not started with a large \( \rho \), as the problem is then very ill conditioned.
Cross derivatives.

The cross derivatives are checked to see if one subsystem is at all dependent on the variables of another subsystem, if not the corresponding \( r \) and \( t \)-coefficients are fixed at zero. This is done at the end of every outer iteration.

Changes to the \( r \)-coefficients.

The diagonal \( r \)-coefficients \( r_i^i \) are assigned a minimum value of 0.2 always, reasoning that every subsystem always has some responsibility towards its own constraints. In addition, a 20\% move limit is imposed on all the \( r \)-coefficients. Thus if \( r_k^n \) is the value of an \( r \)-coefficient in the \( n \)th iteration and \( \tilde{r}_k^{n-1} \) is the value of the same \( r \)-coefficient in the \( (n-1) \)st iteration, then

\[
0.8\tilde{r}_k^{n-1} \leq r_k^n \leq 1.2\tilde{r}_k^{n-1} + 0.01
\]  \hspace{1cm} (6.4)

No COP.

If the objective function of the COP is a constant then the COP is skipped for that iteration, which prevents the possibility of the \( t \)-coefficients being assigned arbitrary values.

Resetting the \( t \)-coefficients.

It was observed in one case that \( t \)-coefficients (corresponding to one subsystem's constraint) with equal derivatives in the objective function of the COP assigned extreme values to the corresponding \( t \)-coefficients even though their combined contribution to the objective function was zero. Setting each of these \( t \)-coefficients to zero will also keep their contribution to the objective function at zero. Hence, after the COP a check is performed on the \( t \)-coefficients to see if for a particular \( p \) the sum of the contributions of all the corresponding coefficients to the objective function of the COP is zero. If so all the \( t \) coefficients corresponding to this \( p \) are forced to be zero. This check is performed for all values of \( p \).

Different combinations of these modifications were used on three different test problems and the results are given in Tables II-VII in Chapter 7.
7. FURTHER TESTS.

7.1 Modified algorithm on 2 x 2 Example.

The original algorithm as proposed by Sobieszczanski-Sobieski [32] did not prove to be successful as indicated by Table I. Results of two of the most successful variations to this algorithm tested on the 2 x 2 case are tabulated in Tables II and III. The characteristics of the algorithms used are given above the corresponding tables. "s updated" indicates that the s-coefficient is updated at the end of every outer iteration as indicated in the modifications given in Section 7. "ω used" means that an artificial variable ω was introduced to deal with infeasible subproblems. The inclusion or exclusion of the COP is in the case of an infeasibility in any subproblem. The limit on the magnitude of the t-coefficient is 1 initially and this bound is decreased using a factor of 0.8 as described in Section 7. The most successful version was used for larger test problems like the 3 x 3 case with two subsystems and the 6 x 6 case with three subsystems. The tests were carried out for five different values of β and for five different starting points. The column headings are the starting points, the last column gives the solutions for the different values of β, and each entry contains a convergence code, and the number of iterations until the two-norm of the change in (x, R, T) is less than 0.0001. For Tables II and III the number of iterations the limit on the t-coefficient is held fixed (at 1.0) before being reduced is 10, and 30 for Table IV. The code C means there is convergence to the solution, WC means there is convergence but not to the solution, and NC means there is no convergence even in the specified number of iterations.

<table>
<thead>
<tr>
<th>β</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 6</td>
<td>C 6</td>
<td>C 6</td>
<td>C 6</td>
<td>C 6</td>
<td>(0.0,2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>C 71</td>
<td>WC 12</td>
<td>C 71</td>
<td>C 71</td>
<td>C 71</td>
<td>(0.198,1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>C 67</td>
<td>WC 25</td>
<td>C 64</td>
<td>C 67</td>
<td>C 67</td>
<td>(0.55,1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>C 66</td>
<td>WC 18</td>
<td>C 66</td>
<td>C 65</td>
<td>C 64</td>
<td>(0.8,1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 7</td>
<td>C 8</td>
<td>C 7</td>
<td>C 62</td>
<td>C 8</td>
<td>(1.0,1.0)</td>
</tr>
</tbody>
</table>
TABLE III
s updated, ω used, COP, t bound at 1 and 0.8 update after 10 iterations.

<table>
<thead>
<tr>
<th>β</th>
<th>(2,3)</th>
<th>(4,1)</th>
<th>(1,1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 6 C 6 C 6 C 6 C 6 C 6</td>
<td>(0.0,2.0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>C 71 C 67 C 71 C 71 C 71 C 71</td>
<td>(0.198,1.98)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>C 67 C 72 C 64 C 67 C 67 C 67</td>
<td>(0.55,1.835)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>C 66 C 65 C 66 C 65 C 64 C 64</td>
<td>(0.8,1.6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>C 7 C 8 C 7 C 62 C 8</td>
<td>(1.0,1.0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figures 1 and 2 are provided for a better understanding of the path taken from the starting point to the solution. The pictures correspond to Example 1 with β = 0.1. Figure 1 includes all the iterate values (except for the starting point) up to the solution. The first few segments are numbered 1 — 8 at their midpoints. Figure 2 is a blown up view of the region of convergence that is marked in Figure 1. Some of the intermediate segments are numbered at their midpoints here. The solution is indicated by a *.

7.2 3 × 3 Example.

Example 2.

\[
\begin{align*}
\min_x & \quad x_1^2 + x_2^2 + x_3^2 \\
\text{subject to} & \quad x_1 + x_2 + \beta x_3 \leq 4, \\
& \quad -x_1 - x_2 - \beta x_3 \leq -2, \\
& \quad -\beta x_1 - \beta x_2 - 5x_3 \leq -2,
\end{align*}
\]

(7.1)

Here, the first two constraints belong to one subsystem and the third constraint to another subsystem, \(X^1 = (x_1, x_2)\) and \(X^2 = (x_3)\).

In Tables II–IV for \(β = 0\), the 6 iterations corresponds to 3 iterations for the convergence test to be satisfied with 2 consecutive values of \(ρ\); thus 6 actually is the smallest possible
Figure 1. Trace of the solution iterates for Example 1, corresponding to
$\beta = 0.1$, with starting point (2,3) and solution (0.198,1.98).
FIGURE 2. Blown up view of the region of convergence of the plot shown in Figure 1.
number of iterations, and corresponds to reaching the solution in one step. For the WC entries in Table IV, performing a cold start at the point they converge to results in convergence to the correct solution. Figure 3 corresponds to Example 2 and gives a trace of the iterates for $\beta = 0.5$ and starting point $(0,1,-3)$ (not included in the picture).

*Move limits on the $x$ vector.*

Introducing move limits on the $x$ variables will help prevent oscillation of the iterates. The size of the move limits permitted can be varied depending on the problem. Let $m$ be the move limit permitted and $\bar{x}_i$ the current value of $x_i$. Then the constraint

$$\bar{x}_i - m|\bar{x}_i| - 0.1 \leq x_i \leq \bar{x}_i + m|\bar{x}_i| + 0.1$$

(7.2)

is added for every component $x_i$ of the $x$ vector.

Tests were performed on the following $6 \times 6$ example using different move limits and the results are tabulated accordingly.
FIGURE 3. Trace of the solution iterates for Example 2, corresponding to \( \beta = 0.5 \), with starting point \((0,1,3)\) and solution \((0.88,0.88,0.44)\).
Example 3.

\[ \begin{align*}
\min_{x} & \quad x_1^2 + x_2^2 + x_3^2 + 2.5x_4^2 + 2.5x_5^2 + 10x_6^2 \\
\text{subject to} & \quad x_1 + x_2 + x_3 + 0 - \beta x_5 - 2\beta x_6 \leq 4, \\
& \quad -x_1 - x_2 - x_3 - \beta x_4 + 0 + 0 \leq -2, \\
& \quad -x_1 - x_2 - 5x_3 + 0 + 0 \leq -2, \\
& \quad 0 + 0 + x_4 + x_5 - \beta x_6 \leq -4, \\
& \quad \beta x_1 + \beta x_2 + 0 - 5x_4 - 4x_5 - \beta x_6 \leq 20, \\
& \quad \beta x_1 + \beta x_2 - \beta x_3 + 0 + 0 - x_6 \leq -6,
\end{align*} \]

where \( x = (x_1, x_2, x_3, x_4, x_5, x_6)^t \in \mathbb{R}^6 \)

Here there are three subsystems with \( n_1 = 3 \), \( n_2 = 2 \) and \( n_3 = 1 \).

If the algorithm as described thus far is used, the results are not encouraging as indicated by Table V. In Table V the number in each entry gives the number of iterations taken. The entries in this table correspond to a move limit of \( m = 0.3 \) on each component of the \( x \) vector. The limit on the \( t \)-coefficients is held fixed for 30 iterations before being reduced. There is convergence to the solution only in the case of \( \beta = 0.0 \); for larger \( \beta \)s the solution is not obtained for any of the starting points. An attempt to use the actual solution vector as the starting point also did not lead to convergence to the solution – the process actually diverged away from the solution. Clearly, this is a fundamental flaw.

The cause of the divergence was traced to the sensitivity information used in the COP. This would therefore affect the \( r \)- and \( t \)-coefficients that are updated in the COP. The coefficients in the linear objective function of the COP are the partial derivatives of the main objective function (of the global problem) \( \Theta \) with respect to each of the \( r \)- and the \( t \)-coefficients. Consider one such derivative \( \partial \Theta / \partial z \), where \( z \) represents any one of the \( r \)- or the \( t \)-coefficients. It is calculated using the Lagrange multipliers obtained from the subsystem optimizations (as given in Appendix A) via

\[ \frac{\partial \Theta}{\partial z} = \lambda^r \frac{\partial G^i_A}{\partial z}. \]  

(7.4)
The Lagrange multipliers obtained from the subsystems correspond to different values of the design vector \( x \) by necessity, because the subsystem optimizations must be done independently and concurrently. On completion of the subsystem optimizations, the \( x \) vector is updated as a composite of the optimal subvectors obtained from the subsystems. Thus, the sensitivity derivatives obtained using these Lagrange multipliers no longer reflect the actual sensitivity of the objective function at the updated \( x \) to changes in the \( r \)- and the \( t \)-coefficients.

To illustrate this point, consider the following step in the optimization process for Example 3 with \( \beta = 0.1 \) and the solution vector \((-2.4, -2.4, 7.0, -1.7, -1.8, 4.8)\) as the starting point \( x_0 \). At the end of the first outer iteration the value of the design vector is

\[
x_1 = (-2.4, -2.4, 6.99, -1.7, -1.7, 4.8).
\]

The derivatives of the objective function \( \Theta \) with respect to the \( t \)-coefficients corresponding to the 2nd constraint (i.e., \( t_1^2 \), \( t_2^2 \) and \( t_3^2 \)) are calculated to be 0.0, -8.7, and 0.0 respectively, and at the end of the COP these \( t \)-coefficients are assigned the values 0.0, 0.1, and -0.1 respectively.

If the sensitivity information were correct, the derivative of \( \Theta \) with respect to \( t_3^2 \) being zero should ensure that the objective function is not sensitive to the value of \( t_3^2 \). In the third subsystem optimization, the cumulative constraint \( \tilde{C}_3^2 \leq 0 \) corresponding to the second subsystem is active at the solution, thus determining the value of the variable \( x_6 \). This constraint is \( 0.1x_6 \geq 0.58 \). Therefore \( x_6 = 5.8 \), which considerably increases the value of the objective function. If the value of \( t_3^2 \) had been 0.0 instead of the -0.1 it was assigned, the constraint \( \tilde{C}_3^2 \leq 0 \) would have been 0.1\( x_6 \geq 0.48 \) and \( x_6 \) would have been unaltered from its value 4.8 at the end of the previous iteration. Thus, the objective function is clearly very sensitive to the value of \( t_3^2 \). The objective function of the COP and the third subproblem in the second iteration are given in full below for a better understanding of the problem:

The objective function of the COP at the end of the first iteration is

\[
\min_x \quad 0.027r_1^1 - 94t_1^3 - 8.7t_2^3 - 96.3t_3^3
\]
The third subsystem is

$$\begin{align*}
\min_{x_6} & \quad x_6 \\
\text{subject to} & \quad x_6 \geq 4.7, \\
& \quad 0.0x_6 \geq 0.0, \\
& \quad 0.1x_6 \geq 0.58, \\
& \quad 6.36 \geq x_6 \geq 3.27,
\end{align*}$$

(7.5)

Here, there is a move limit of \((m = 0.3)\) on \(x_6\), which gives the last constraint. The first constraint is \(\bar{g}_1^3 \leq 0\), the second constraint corresponds to \(\bar{C}_1^3 \leq 0\), and the third constraint corresponds to \(\bar{C}_3^3 \leq 0\).

Table VI gives the results for the 6 x 6 problem if the sensitivities used in the COP are the average of the sensitivities obtained over two iterations. In each entry in Tables VI-VII, the number to the right of the convergence code indicates the number of iterations taken, the number below gives the number of iterations the limit on the \(t\)-coefficient is held fixed (at 1.0) before being reduced, and the third number gives the move limit permitted on each component of the \(x\) vector. While smoothed sensitivities clearly help (compare Tables V and VI), they are not a cure for the larger \(\beta\) cases. We remark that simply changing the move limits on \(x\), \(r\), and \(t\) is not the remedy, since hundreds of move limit variations were tried with results no better than those in Table VI.

As a final illustration of the erratic behavior induced by the COP, we describe an irrational scheme resulting from a programming error. In this version, the \(a_k^p\) coefficients and the cross derivatives are not updated at the end of every outer iteration as described in the list of modifications given earlier in Section 7. Also, the \(a_k^p\) coefficients that are obtained initially are assigned incorrectly (i.e., \(a_k^p\) is taken to be \(a_k^p\)). The absolute value of 0.1 is not added to the move limit on the \(r\)-coefficients. For the move limit on the components of \(x\) the absolute value added is equal to the move limit. This irrational scheme applied to Example 3 yields Table VII, which is comparable to the results obtained by any of the more plausible versions on Example 3.
### Table V

$s$ updated, $\omega$ used, $COP$,  
$t$ bounded at 1 and 0.8 update.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>(0,0,0,0,0)</th>
<th>(1,2,3,-1,1,5)</th>
<th>(-10,4,4,0,8,0,1,1)</th>
<th>(1,1,1,1,1,1)</th>
<th>(-4,2,2,0,1,1)</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 17</td>
<td>C 19</td>
<td>C 17</td>
<td>C 17</td>
<td>(0.6,0.6,0.6,-2.0,-2.0,6.0)</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>NC 154</td>
<td>WC 113</td>
<td>NC 154</td>
<td>NC 151</td>
<td>(-2.4,-2.4,7.0,-1.7,1.8,4.8)</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>WC 105</td>
<td>WC 106</td>
<td>WC 105</td>
<td>NC 150</td>
<td>(-2.7,-2.7,8.0,-1.5,1.8,1.9)</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>WC 102</td>
<td>WC 102</td>
<td>WC 102</td>
<td>WC 105</td>
<td>(-1.7,-1.7,6.3,-1.5,1.9,1.0)</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>WC 102</td>
<td>WC 102</td>
<td>WC 102</td>
<td>WC 105</td>
<td>(-0.5,-0.5,4.2,-1.2,2.0,0.7)</td>
<td></td>
</tr>
</tbody>
</table>

### Table VI

$s$ updated, $\omega$ used, $COP$,  
$t$ bounded at 1 and 0.8 update.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>(0,0,0,0,0)</th>
<th>(1,2,3,-1,1,5)</th>
<th>(-10,4,4,0,8,0,1,1)</th>
<th>(1,1,1,1,1,1)</th>
<th>(-4,2,2,0,1,1)</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 17</td>
<td>C 19</td>
<td>C 17</td>
<td>C 17</td>
<td>(0.6,0.6,0.6,-2.0,-2.0,6.0)</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>C 110</td>
<td>C 113</td>
<td>C 72</td>
<td>WC 109</td>
<td>(-2.4,-2.4,7.0,-1.7,1.8,4.8)</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>WC 105</td>
<td>C 103</td>
<td>WC 103</td>
<td>WC 103</td>
<td>(-2.7,-2.7,8.0,-1.5,1.8,1.9)</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>WC 103</td>
<td>WC 104</td>
<td>WC 107</td>
<td>WC 107</td>
<td>(-1.7,-1.7,6.3,-1.5,1.9,1.0)</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>WC 103</td>
<td>WC 104</td>
<td>WC 102</td>
<td>WC 102</td>
<td>(-0.5,-0.5,4.2,-1.2,2.0,0.7)</td>
<td></td>
</tr>
</tbody>
</table>

27
TABLE VII
$s$ updated, $\omega$ used, COP,
t bound at 1 and 0.8 update.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$(0,0,0,0,0)$</th>
<th>$(1,2,3,-1,1,5)$</th>
<th>$(-10,4,4,0.8,0.1,1)$</th>
<th>$(1,1,1,1,1,1)$</th>
<th>$(-4,2,2,0,1,1)$</th>
<th>solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 16</td>
<td>C 14</td>
<td>C 19</td>
<td>C 14</td>
<td>C 15</td>
<td>(0.6,0.6,0.6,-2.0,-2.0,6.0)</td>
</tr>
<tr>
<td></td>
<td>1 0.2</td>
<td>1 0.2</td>
<td>1 0.2</td>
<td>1 0.2</td>
<td>1 0.2</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>WC 82</td>
<td>C 102</td>
<td>WC 23</td>
<td>C 102</td>
<td>WC 26</td>
<td>(-2.4,-2.4,7.0,-1.7,-1.8,4.8)</td>
</tr>
<tr>
<td></td>
<td>30 0.3</td>
<td>30 0.1</td>
<td>30 0.3</td>
<td>30 0.1</td>
<td>30 0.3</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>WC 34</td>
<td>C 199</td>
<td>WC 81</td>
<td>WC 101</td>
<td>WC 35</td>
<td>(-2.7,-2.7,8.0,-1.5,-1.8,1.9)</td>
</tr>
<tr>
<td></td>
<td>30 0.3</td>
<td>80 0.5</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>WC 26</td>
<td>C 200</td>
<td>WC 24</td>
<td>WC 110</td>
<td>WC 12</td>
<td>(-1.7,-1.7,6.3,-1.5,-1.9,1.0)</td>
</tr>
<tr>
<td></td>
<td>30 0.3</td>
<td>80 0.8</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>WC 20</td>
<td>WC 8</td>
<td>WC 20</td>
<td>WC 99</td>
<td>WC 15</td>
<td>(-0.5,-0.5,4.2,-1.2,-2.0,0.7)</td>
</tr>
<tr>
<td></td>
<td>30 0.3</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td>30 0.3</td>
<td></td>
</tr>
</tbody>
</table>
8. QUADRATIC PROGRAMMING ALGORITHMS.

The proposed algorithm was tested on simple quadratic programming (QP) problems. When a QP problem is divided into subsystems, each subproblem is also a QP problem. This chapter gives the algorithm used to solve a QP subproblem. This algorithm by Fletcher [12] uses an active set strategy.

As mentioned before, an optimization package called QPSOL [16] was used extensively to verify results and the LPSOL part of this package was used to solve the linear programming problem that arises in the COP.

Also, another optimization package, MINOS, was used during the initial tests to verify the results obtained. The algorithms used in QPSOL and MINOS are given in Sections 8.2 and 8.3.

8.1 Active set strategy - Fletcher.

Let \( x \in \mathbb{R}^n \) and \( G \in \mathbb{R}^{n \times n} \) be positive definite. We need to find a solution \( x^* \) to a quadratic programming problem of the following type:

\[
\begin{align*}
\text{minimize} & \quad q(x) = \frac{1}{2} x^T G x + g^T x, \\
\text{subject to} & \quad a_i^T x = b_i, \quad i \in E, \\
& \quad a_i^T x \geq b_i, \quad i \in I.
\end{align*}
\]  

(8.1)

The implemented algorithm is as follows: First, the QP problem with just equality constraints is solved by one of two different methods. Next these methods are generalized to handle inequality constraints also by means of an active set method.

In the active set method [12], certain constraints, indexed by the active set \( \mathcal{A} \), are regarded as equalities while the rest are temporarily disregarded, and the method adjusts this set in order to identify the correct active constraints at the solution to the given QP problem.

On iteration \( k \), a feasible point \( x^{(k)} \) is known which satisfies the active constraints as equalities, that is, \( a_i^T x^{(k)} = b_i, \ i \in \mathcal{A} \). Each iteration attempts to locate the solution to an
equality problem (EP) in which only the active constraints occur. This is done by shifting the origin to \( x^{(k)} \) and looking for a correction \( \delta^{(k)} \) which solves

\[
\begin{align*}
\min_{\delta} & \quad \frac{1}{2} \delta^T G \delta + \delta^T g^{(k)}, \\
\text{subject to} & \quad a_i^T \delta = 0, \quad i \in \mathcal{A},
\end{align*}
\]

where \( g^{(k)} \) is defined by \( g^{(k)} = g + \nabla q(x^{(k)}) \) for the function \( q(x) \) defined before. This is in the form of an EP and can be solved by one of the methods given below.

If the step \( \delta^{(k)} \) is feasible with regard to the constraints not in \( \mathcal{A} \), then the next iterate is taken as \( x^{(k+1)} = x^{(k)} + \delta^{(k)} \). If not, a line search is made in the direction of \( \delta^{(k)} \) to find the best feasible point. This is done by first defining the solution of the EP for the correction to be \( s^{(k)} \), and choosing the step \( \alpha^{(k)} \) by

\[
\alpha = \min \left( 1, \min_{i \in \mathcal{A}, s^{(k)}_i < 0} \frac{b_i - a_i^T x^{(k)}}{a_i^T s^{(k)}} \right)
\]

so that \( x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)} \). If \( \alpha^{(k)} < 1 \), then a new constraint becomes active, defined by the index \( p \) which achieves the minimum and this index is added to the active set \( \mathcal{A} \). If there are more than one constraint with the minimum value, all of them are added to the active set.

If \( x^{(k)} \) (that is, \( \delta^k = 0 \)) solves the current EP, then it is possible to compute multipliers \( \lambda^{(k)} \) for the active constraints as described in the two methods given below. The vectors \( x^{(k)} \) and \( \lambda^{(k)} \) then satisfy all the first order optimality conditions for the inequality constraint problem except possibly that \( \lambda_i \geq 0, i \in I \). Thus, a test is made to determine whether \( \lambda_i^{(k)} \geq 0 \forall i \in \mathcal{A} \cap I \); if so, then the first order optimality conditions are satisfied and these are sufficient to ensure a global solution since \( g(x) \) is convex. Otherwise there exists an index, \( r \) say, \( r \in \mathcal{A} \cap I \) such that \( \lambda_r^{(k)} < 0 \). In this case, it is possible to reduce \( g(x) \) by allowing the constraint \( r \) to become inactive. Thus \( r \) is removed from \( \mathcal{A} \) and the algorithm continues as before by solving the resulting EP. If there is more than one index \( r \) for which \( \lambda_r^{(k)} < 0 \) then \( r \) is selected to solve

\[
\min_{i \in \mathcal{A} \cap I} \lambda_i^{(k)}.
\]
In the case of a tie, all the constraints that correspond to the minimum of the negative \( \lambda a \) are made inactive.

**Algorithms to solve an EP**

The aim is to find a solution \( z^* \) to the equality constrained problem

\[
\begin{align*}
\text{minimize} \quad & q(x) = \frac{1}{2} x^T G x + q^T x, \\
\text{subject to} \quad & A^T x = b.
\end{align*}
\]

(8.5)

It is assumed that there are \( m \leq n \) constraints, so that \( b \in \mathbb{R}^m \), and \( A \) is \( n \times m \) with the column vectors \( a_i, i \in E \). It is assumed that \( A \) has rank \( m \).

**Method 1**

One way of solving the EP is to use the constraints to eliminate variables. If the partitions

\[
x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad g = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}, \quad G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}
\]

(8.6)

are defined, where \( x_1 \in \mathbb{R}^m \) and \( x_2 \in \mathbb{R}^{n-m} \), etc., and rank \( A_1 = m \), then the constraints become \( A_1^T x_1 + A_2^T x_2 = b \) and are readily solved (by Gaussian Elimination) for \( x_1 \) in terms of \( x_2 \):

\[
x_1 = A_1^{-T} (b - A_2^T x_2).
\]

Substituting into \( q(x) \) gives the problem: minimize \( \psi(x_2), x_2 \in \mathbb{R}^{n-m} \) where \( \psi(x_2) \) is the quadratic function

\[
\begin{align*}
\psi(x_2) &= \frac{1}{2} x_2^T (G_{22} - G_{21}A_1^{-T}A_2^T - A_2A_1^{-1}G_{12} + A_2A_1^{-1}G_{11}A_1^{-1}A_2^T) x_2 \\
&\quad + x_2^T (G_{21} - A_2A_1^{-1}G_{11})A_1^{-T} b + \frac{1}{2} b^T A_1^{-1}G_{11} A_1^{-T} b \\
&\quad + x_2^T (g_2 - A_2A_1^{-1} g_1) + g_1^T A_1^{-T} b.
\end{align*}
\]

(8.7)

The Lagrange multiplier vector \( \lambda^* \) is defined by \( g^* = \nabla \psi(x^*) = A\lambda^* \) and can be calculated by solving \( g^* = \nabla \psi(x^*) \) the first partition \( g_1^* = A_1 \lambda^* \). By the definition of \( q(x) \), \( g^* = g + Gx^* \), so an explicit expression for \( \lambda^* \) is

\[
\lambda^* = A_1^{-1} (g_1 + G_{11} x_1^* + G_{12} x_2^*).
\]

(8.8)
Method 2

A generalized elimination method is possible in which essentially a linear transformation of variables is made initially. Let $S$ and $Z$ be $n \times m$ and $n \times (n - m)$ matrices respectively such that $[S : Z]$ is nonsingular, and in addition let $A^T S = I$ and $A^T Z = 0$. $S^T$ can be regarded as a left generalized inverse for $A$ so that a solution of $A^T x = b$ is given by $x = Sb$. However this solution is non-unique in general and other feasible points are given by $x = Sb + \delta$ where $\delta$ is in the null space of $A^T$.

\[ \{ \delta \mid A^T \delta = 0 \}, \]

which has dimension $n - m$. The columns $z_1, z_2, \ldots, z_{n-m}$ of $Z$ are a basis for this null space. That is to say, any feasible correction $\delta$ can be written as

\[ \delta = Zy = \sum_{i=1}^{n-m} z_i y_i, \quad (8.9) \]

where $y_1, y_2, \ldots, y_{n-m}$ are the components in each reduced coordinate direction. Thus, any feasible point $x$ can be written as

\[ x = Sb + Zy. \]

Substituting into $q(x)$ gives

\[ \psi(y) = \frac{1}{2} y^T Z^T GZ y + (g + GSb)^T Z y + \frac{1}{2} (g + GSb)^T Sb. \quad (8.10) \]

Since $Z^T GZ$ is positive definite a unique minimizer $y^*$ exists which solves the linear system

\[ (Z^T GZ) y = -Z^T (g + GSb). \]

Now $x^*$ is obtained by substitution,

\[ x^* = Sb - Z(Z^T GZ)^{-1} Z^T (g + GSb) \]

$g^* = A\lambda^*$ gives $\lambda^* = S^T g^*$ which can be used to calculate Lagrange multipliers,

\[ \lambda^* = S^T (g + Gx^*) = (S^T - S^T GZ(Z^T GZ)^{-1} Z^T) g + S^T (G - GZ(Z^T GZ)^{-1} Z^T) Sb. \]

32
Depending on the choice of $S$ and $Z$, a number of methods exist, but one numerically stable way uses the QR factorization of the matrix $A$.

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 \ Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R,$$  \hspace{1cm} (8.11)

where $Q$ is $n \times n$ and orthogonal, $R$ is $m \times m$ and upper triangular and $Q_1$ and $Q_2$ are $n \times m$ and $n \times (n - m)$, respectively. $S$ and $Z$ are chosen as follows:

$$S = A^+ T = Q_1 R^{-T}, \quad Z = Q_2,$$  \hspace{1cm} (8.12)

where $A^+$ denotes the Moore Penrose inverse of $A$.

Method 2 was used in the implementation with $S$ and $Z$ chosen using the QR factorization of the matrix $A$.

8.2 QPSOL.

QPSOL [16] is a FORTRAN package for solving quadratic programming problems. It is designed to solve a QP of the following form:

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad l \leq \begin{bmatrix} x \\ Ax \end{bmatrix} \leq u,
\end{align*}$$  \hspace{1cm} (8.13)

where $A$ is $m_L \times n$ ($m_L$ may be zero) and $f(x)$ is either a linear or a quadratic objective function. It can be used to find a feasible point for a set of constraints, the corresponding objective function is then set to zero.

QPSOL is based on a type of active set strategy called an inertia-controlling method, due to Gill and Murray [16]. Briefly, the algorithm used in QPSOL has two phases, a feasibility phase and an optimality phase.

The feasibility phase deals with finding an initial feasible point and the optimality phase concentrates on minimizing the quadratic objective function within the feasible region. The two phases are performed by the same routines, simply changing the function to be minimized from the sum of the infeasibilities to the quadratic objective function.
8.3 MINOS.

MINOS [23] is an optimization package designed to solve problems in

- linear programming,
- unconstrained optimization,
- linearly constrained optimization, and
- nonlinearly constrained optimization.

Thus, the optimization problem solved can be expressed as:

\[
\begin{align*}
\text{minimize} & \quad F(x) + c^T x + d^T y \\
\text{subject to} & \quad f(x) + A_1 y = b_1, \\
& \quad A_2 x + A_3 y = b_2, \\
& \quad l \leq \begin{pmatrix} x \\ y \end{pmatrix} \leq u,
\end{align*}
\]

where the vectors \(c, d, b_1, b_2, l, u\) and the matrices \(A_1, A_2, A_3\) are constant, \(F(x)\) is a smooth scalar function, and \(f(x)\) is a vector of smooth functions. The \(n_1\) components of \(x\) are called the nonlinear variables, and the \(n_2\) components of \(y\) are the linear variables. The first set of equations are the nonlinear constraints, the second set the linear constraints, and the third set the upper and lower bounds on all the variables. There is a provision for introducing upper and lower bounds on the constraints as well. Such constraints would be of the form

\[
\begin{align*}
l_1 & \leq f(x) + A_1 \leq u_1, \\
l_2 & \leq A_2 x + A_3 y \leq u_2,
\end{align*}
\]

but are actually specified in terms of a right-hand side \(b_i\) and a range \(u_i - l_i\).

The algorithms that MINOS is based on are

- the simplex method, (Dantzig [9]),
- a quasi-Newton method, (Davidon [10]),
- the reduced-gradient method, (Wolfe [46]), and
- a projected Lagrangian method (Robinson [28]; Rosen and Kreuser [29]).
9. HISTORICAL DEVELOPMENT OF PRESENT ALGORITHM.

The original algorithm presented by Sobieski [33] was used without any modifications initially on a 2 x 2 test problem. This chapter gives a description of the main problems encountered at the beginning and the modifications that were made to the original algorithm to resolve these problems, with the results tabulated for every stage. The tables are given together at the end for ease of comparison.

Also, a brief description of some of the later modifications attempted that were not successful is given in the end.

9.1. The switch coefficient $s$.

A simple 2 x 2 quadratic programming problem (5.1) was used in the beginning to better understand any problems encountered with either the algorithm or the implementation. The results of testing the original algorithm on this example for 5 different starting points and $\beta$s are tabulated in Table I in Chapter 5. There the $t$-coefficients are bounded in magnitude by 1.0. The original algorithm with the $t$-coefficients left unbounded (see Section 9.3) led to Table VIII. On examining the algorithm [33] closely, one major problem was detected. The $s$-coefficient, which is a switch, turns one of the $r$ or the $t$-coefficients on, depending on whether the corresponding constraint is violated or satisfied, respectively. But once the $s$-coefficient is set to zero, it stays at zero until the entire procedure terminates. Consider the following cumulative constraint of the $p$th subsystem present in the $i$th subsystem:

$$C_i^p(X_i^i, Y_i^i) \leq \hat{K}_p(X_0^i, Y_0^i) s^p(1 - r_i^p) + (1 - s^p)t_i^p. \quad (9.1)$$

If the $s$-coefficient $s^p$ activates $t_i^p$, then $r_i^p$ can never be activated again, even in the event of a violation. This is not right, as the purpose of the $r$-coefficient is to help in reducing constraint violation.

Two different solutions were suggested to this problem. One was to switch the $s$-coefficient on and off depending on whether the constraint is currently violated or not. Thus, at the end of every outer iteration, the $s$-coefficients are updated after checking to see if the corresponding constraints are violated or not.
The original algorithm with this change was tried out on the 2 x 2 example (with the t-coefficients left unbounded, see Section 9.3) and the results are given in Table IX.

The other solution was to remove the s-coefficient from the r term. Thus, the r-coefficient is always active, both in the case of a violation and satisfaction. Equation (9.1) would then become

\[ C_i^p (X^i, Y^i) \leq \hat{K}_p (X_0^i, Y_0^i) (1 - r_i^p) + (1 - s^p) t_i^p. \] (9.2)

This modification was also tried on the 2 x 2 example and the results (no bounds on the t-coefficients, see Section 9.3) are given in Table X. Since the results corresponding to the first solution were better, this was used for further testing.


As is evident from the Tables VIII, IX, and X corresponding to the above mentioned changes, when an infeasible subproblem was encountered, the original algorithm had no way of dealing with it. The procedure had to terminate abnormally at this point. Thus one of the first few modifications was to deal with infeasibility in a subproblem.

This is done by introducing a new variable \( \omega \) in the corresponding subsystem. The variable \( \omega \) is added to each of the constraints and a large multiple of this variable is added to the objective function. The infeasible subsystem is now solved once again with \( \omega \). Precisely, the modified subsystem is:

\[
\min_{X^i} \hat{\Theta}(X^i) + M\omega \\
\text{subject to } \hat{g}_i^i(X^i, Y^i) - \omega \leq s^i \max\{\hat{g}_i^i(X_0^i, Y_0^i), 0\}(1 - r_i^p) + (1 - s^i)t_i^p, \\
C_p^i (X^i, Y^i) - \omega \leq \hat{K}_p (X_0^i, Y_0^i) s^p(1 - r_i^p) + (1 - s^p)t_i^p, \\
p = 1, \ldots, i - 1, i + 1, \ldots, N,
\] (9.3)

where \( M \) is a large positive number.

The coefficients of the variables in the objective function of the COP involve calculation of the sensitivity derivatives according to the method given in Appendix A. The Lagrange multipliers obtained from solving the individual subproblems are used to obtain these derivatives.
When \( \omega \) is introduced in a subsystem to deal with infeasibility, the corresponding Lagrange multipliers are also affected. A thought as to whether these Lagrange multipliers can be rightfully allowed to affect the COP led to two different versions. In one version the COP is performed at the end of every outer iteration, and in the other whenever there is an infeasibility in any subproblem and \( \omega \) is used to recover from the infeasibility, the COP is not performed in the corresponding outer iteration. This leads to the two Tables XI and XII. Both of them correspond to the \( s \)-coefficient being updated at the end of every outer iteration. Table XII corresponds to the COP being included and Table XI corresponds to the COP not being present in the case of an infeasibility. Both these Tables correspond to the \( \varepsilon \) being unbounded.

9.3. Bounds on the \( t \)-coefficients.

The \( t \)-coefficients allow a violation of a constraint in one subsystem, provided it is compensated for by an oversatisfaction in another subsystem. Initially tests were conducted with these \( t \)-coefficients left unbounded, and the results are given in Tables XI and XII. This implies that the \( t \)-coefficient can be any large value, which means that a constraint can be violated by a large amount in a particular subsystem. This is not right. Therefore bounds were introduced for the \( t \)-coefficients, which were a feature of the original algorithm also.

Different ways of handling these bounds were tried. One approach was to fix the bound at a certain value initially, and then have this bound decrease by a certain factor after a specified number of iterations.

Tables XI and XII (Section 9.4) had the \( t \)-coefficients unbounded. This modified version but with the \( t \) being bounded in magnitude by 1.0 initially and having these bounds reduced by a factor of 0.8 at the end of every outer iteration, starting from the first, yielded Tables XIII and XIV. Having the bound always fixed at 1.0 gives Table XV.

The bound on the \( t \) needs to be reduced in some fashion, especially when close to the solution. Thus, other ways of reducing these bounds were tried. One was to reduce the bound by a logarithmic function (see Chapter 6). The geometric update yielded the best results, although the logarithmic update also was good enough.
9.4. A description of the different versions.

The following list gives the characteristics of the different versions of the algorithm that are tabulated at the end.

1. The original algorithm with the $s$-coefficient set to zero when the constraint is satisfied and held at zero thereafter. Exits with a message in the case of an infeasibility. The value of the $t$-coefficient is unbounded.

2. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. Exits with a message in the case of an infeasibility. The value of the $t$-coefficient is unbounded.

3. The $s$-coefficient is removed from the $r$ term in each of the constraints in the subsystem. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. Exits with a message in the case of an infeasibility. The value of the $t$-coefficient is unbounded.

4. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable $w$ is introduced in every constraint to deal with infeasibility. A large multiple of $w$ is added to the objective function. There is no Coordination Optimization Phase in the case of an infeasibility. The value of the $t$-coefficient is unbounded.

5. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable $w$ is introduced in every constraint to deal with infeasibility. A large multiple of $w$ is added to the objective function. There is a Coordination Optimization Phase at the end of every outer iteration. The value of the $t$-coefficient is unbounded.

6. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is
satisfied. A new variable $\omega$ is introduced in every constraint to deal with infeasibility. A large multiple of $\omega$ is added to the objective function. There is no Coordination Optimization Phase in the case of an infeasibility. The value of the $t$-coefficient is bounded initially between $-1$ and $1$ and these bounds are multiplied by a factor of $0.8$ at the end of every outer iteration starting from the first.

7. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to $1$ if the constraint is not satisfied and to $0$ if it is satisfied. A new variable $\omega$ is introduced in every constraint to deal with infeasibility. A large multiple of $\omega$ is added to the objective function. There is a Coordination Optimization Phase at the end of every outer iteration. The value of the $t$-coefficient is bounded initially between $-1$ and $1$ and these bounds are multiplied by a factor of $0.8$ at the end of every outer iteration starting from the first.

8. At the end of every outer iteration the $s$-coefficient corresponding to each of the cumulative constraints is set to $1$ if the constraint is not satisfied and to $0$ if it is satisfied. A new variable $\omega$ is introduced in every constraint to deal with infeasibility. A large multiple of $\omega$ is added to the objective function. There is no Coordination Optimization Phase. The value of the $t$-coefficient is bounded initially between $-1$ and $1$ and these bounds are held fixed.

KEY TO THE TABLES

1. C—Converges to the right value.
2. NC—No Convergence.
3. WC—Converges to a wrong value.
4. AC—Converges to approximately the right value.
5. CD—Reaches the correct value and then diverges.
6. IF—Problem is infeasible.
7. ACD—Reaches approximately the right value and then diverges.
8. ACO—Reaches approximately the right value and then oscillates.
### Table VIII

**Original Algorithm**

\[ t \text{ unbounded.} \]

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>O 30</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>O 30</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>O 30</td>
<td>IF 1</td>
<td>WC 1</td>
<td>AC 1</td>
<td>IF 2</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>O 30</td>
<td>WC 2</td>
<td>WC 1</td>
<td>O 30</td>
<td>O 30</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

### Table IX

**s updated and \( t \) unbounded.**

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>AC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>AC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>AC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>AC 1</td>
<td>WC 2</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 2</td>
<td>WC 2</td>
<td>WC 1</td>
<td>WC 2</td>
<td>WC 2</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

### Table X

**s removed and \( t \) unbounded.**

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>WC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>WC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>WC 1</td>
<td>IF 1</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>WC 2</td>
<td>IF 1</td>
<td>WC 1</td>
<td>AC 1</td>
<td>IF 1</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>WC 2</td>
<td>WC 2</td>
<td>WC 1</td>
<td>WC 2</td>
<td>WC 2</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

### Table XI

**s updated, \( \omega \) used, no COP**

\[ t \text{ unbounded.} \]

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,-1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>AC 2</td>
<td>WC 2</td>
<td>WC 1</td>
<td>WC 1</td>
<td>WC 2</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>AC 2</td>
<td>NC 20</td>
<td>WC 1</td>
<td>WC 1</td>
<td>WC 2</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>AC 2</td>
<td>NC 20</td>
<td>WC 1</td>
<td>AC 1</td>
<td>WC 2</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 2</td>
<td>WC 2</td>
<td>WC 1</td>
<td>WC 2</td>
<td>WC 2</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>
### Table XII

$s$ updated, $\omega$ used, COP

$t$ unbounded.

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>C 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>AC 2</td>
<td>WC 2</td>
<td>WC 1</td>
<td>WC 1</td>
<td>WC 2</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>AC 2</td>
<td>NC 20</td>
<td>NC 100</td>
<td>NC 100</td>
<td>WC 2</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>AC 2</td>
<td>WC 3</td>
<td>WC 1</td>
<td>AC 1</td>
<td>WC 2</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 2</td>
<td>WC 2</td>
<td>WC 2</td>
<td>WC 2</td>
<td>WC 2</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

### Table XIII

$s$ updated, $\omega$ used, no COP

$t$ bound at 1 and 0.8 update.

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>CD 1</td>
<td>CD 3</td>
<td>CD 3</td>
<td>CD 3</td>
<td>CD 1</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>ACD 3</td>
<td>WC 2</td>
<td>NC 100</td>
<td>NC 100</td>
<td>WC 2</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>NC 100</td>
<td>NC 100</td>
<td>NC 100</td>
<td>NC 100</td>
<td>WC 2</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>AC 16</td>
<td>NC 20</td>
<td>AC 10</td>
<td>AC 10</td>
<td>AC 10</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 2</td>
<td>AC 13</td>
<td>AC 15</td>
<td>C 15</td>
<td>AC 17</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>

### Table XIV

$s$ updated, $\omega$ used, COP

$t$ bound at 1 and 0.8 update.

<table>
<thead>
<tr>
<th>*</th>
<th>(2,3)</th>
<th>(4,1)</th>
<th>(1,-1)</th>
<th>(0.8,1.5)</th>
<th>(10,3)</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>CD 4</td>
<td>CD 3</td>
<td>CD 3</td>
<td>CD 3</td>
<td>CD 4</td>
<td>(0.0, 2.0)</td>
</tr>
<tr>
<td>0.1</td>
<td>ACD 2</td>
<td>NC 20</td>
<td>NC 20</td>
<td>NC 20</td>
<td>NC 20</td>
<td>(0.198, 1.98)</td>
</tr>
<tr>
<td>0.3</td>
<td>NC 20</td>
<td>NC 20</td>
<td>NC 20</td>
<td>NC 20</td>
<td>CD 18</td>
<td>(0.55, 1.835)</td>
</tr>
<tr>
<td>0.5</td>
<td>AC 16</td>
<td>AC 16</td>
<td>AC 10</td>
<td>AC 10</td>
<td>AC 10</td>
<td>(0.8, 1.6)</td>
</tr>
<tr>
<td>1.0</td>
<td>C 2</td>
<td>AC 13</td>
<td>AC 15</td>
<td>AC 15</td>
<td>AC 17</td>
<td>(1.0, 1.0)</td>
</tr>
</tbody>
</table>
9.5. Some later modifications that were tested.

The current version of the algorithm evolved from the original one after a number of modifications were made along the way. Some of them worked and some of them did not work as they had been expected to and so were dropped along the way.

This section gives some of the later modifications attempted that were not successful.

1. In the last phase of the work, before it was discovered that there could be a problem with the sensitivity information used in the COP (see Chapter 7), a number of attempts were made to arrive at the right set of t-coefficients at every iteration. It was observed that a particular t-coefficient could jump from one extreme of the t range to the other over two successive iterations. Move limits were introduced on the t-coefficients to prevent this jump, but this did not help in obtaining the right set of t-coefficients.

2. The algorithm that works for the 2 x 2 and 3 x 3 examples does not work well for the 6 x 6 case. An attempt was made to check if with a small bound on the t-coefficients there was convergence to the solution after a certain number of iterations. If the process converged to a wrong point, then a cold start was performed at this point (i.e., the whole process is restarted with this wrong point as the starting point). This did not help in obtaining any better results, which should be so, as this version of the algorithm diverges away from the solution even when the correct solution is used as the starting point.
10. Conclusions.

Despite the success reported by [7] with the original algorithm of Sobieski [32], we have required major modifications even for a $2 \times 2$ quadratic program. For larger QPs with weak subsystem coupling, the modified algorithm described in Chapter 7 works reasonably well. For larger QPs with strong coupling, the sensitivities obtained from the parallel subsystem optimizations were too unreliable for the COP to produce rational changes in the $r$- and $t$-coefficients. One possibility for correcting this problem may be to compute subsystem sensitivities with a global (and hence serial) step, as advocated in [27].
REFERENCES


Appendix A.

Initialization of the r-coefficients.

The coefficients may be initialized on the basis of sensitivity information so as to assign a greater responsibility for a cumulative constraint satisfaction (of the ith subsystem say) to those subsystems that have a greater influence on that constraint. Let

\[(K_{pk})_i = \frac{\partial K_p}{\partial X^k_i}(x_0, y_0).\]

Since 1 ≤ p ≤ N, 1 ≤ k ≤ N, and 1 ≤ i ≤ n_k, there are N n_k such partial derivatives \((K_{pk})_i\), for every k. Now define

\[a_{pk} = \max_{1 \leq i \leq n_k} |(K_{pk})_i|, \quad 1 \leq p \leq N, \quad 1 \leq k \leq N,
\]

which measures the influence of the kth subsystem's variables \(X^k\) on the pth subsystem's constraints, as represented by \(K_p\). Normalizing these \(N^2\) influence coefficients gives the r-coefficients

\[r_{pk} = \frac{a_{pk}}{\sum_{j=1}^{N} a_{pj}}, \quad 1 \leq p \leq N, \quad 1 \leq k \leq N.
\]

Optimum Sensitivity Analysis.

Let \(z\) denote either of \(r_{pk}\) or \(t_{pk}\), and define the modified constraint functions

\[\bar{g}^i(X^i, Y^i) = \hat{g}^i(X^i, Y^i) - \left[ s^i \max\{\hat{g}^i(X^i_0, Y^i_0), 0\}(1 - r^i_d) + (1 - s^i)\right],
\]

\[\bar{C}_i(X^i, Y^i) = \bar{C}_i(X^i, Y^i) - \left[ K_p(x_0, y_0) s^p(1 - r^i_d) + (1 - s^p)\right],
\]

\[i = 1, \ldots, N, \quad p = 1, \ldots, i - 1, i + 1, \ldots, N.
\]

Let \(\nabla_i = \left( \frac{\partial}{\partial X^1_i}, \ldots, \frac{\partial}{\partial X^n_i} \right),
\]

\[G^i = \begin{pmatrix}
\bar{g}^i \\
\bar{C}_i \\
\vdots \\
\bar{C}_i^{i-1} \\
\bar{C}_i^{i+1} \\
\vdots \\
\bar{C}_i^N
\end{pmatrix},
\]

49
and $G_{A}^{i}$ denote the subvector of $G^{i}$ corresponding to the active constraints at the current point. It is assumed that the dimension of $G_{A}^{i}$ is less than or equal to $n_{i}$, and that the Jacobian matrix $\nabla_{i}G_{A}^{i}$ has full rank. Then the sensitivities of the minimum of $\Theta$ with respect to the constraints $\bar{g}^{i} \leq 0$, $\bar{\Theta}_{i}^{p} \leq 0$ are given by the Lagrange multipliers

$$
\lambda = - \left[ (\nabla_{i}G_{A}^{i})(\nabla_{i}G_{A}^{i})^{t} \right]^{-1} (\nabla_{i}G_{A}^{i})(\nabla_{i}\Theta)^{t},
$$

where everything is evaluated at the current point—the result of the $N$th subsystem optimization. Now from this the sensitivities of the minimum of $\Theta$ with respect to the $r_{i}^{p}$ and $t_{i}^{p}$ are given by

$$
\frac{\partial \Theta}{\partial z} = \lambda^{t} \frac{\partial G_{A}^{i}}{\partial z}.
$$

Observe that from the form of $G^{i}$, the partials $\partial G_{A}^{i}/\partial z$ are trivial to compute. $\lambda$ would not be computed explicitly from the projection operator as described above, but rather from a QR factorization of $(\nabla_{i}G_{A}^{i})^{t}$, as described in Fletcher [12].
This routine solves an optimization problem with a quadratic objective function and linear constraints, by dividing the system into \( k \) different subsystems and solving each independently. There is a final coordination phase that coordinates the results obtained from the different subsystems. The process is carried on iteratively until the desired convergence criterion is met.

```
subroutine nhdec(maxsub,maxvar,maxcon,amat,cmat,alph,beta,
&  d,x,delx,tlim,rho,CONTOL,sum,
&  k,n,np,itfix,itval)
maxsub gives the maximum number of subsystems
maxvar gives the maximum number of variables in any subsystem
maxcon should be the max no of var + 1 to account for omega,
a variable introduced to deal with infeasibility
amat the objective function matrices \( A_{ij} \) are given row by row using
the array amat
cmat the constraint matrices are given using the array cmat
alph and beta give the alpha and beta coefficients
d gives the rhs of the constraints
x gives a starting point on input and the final design vector value on output
delx is the move limit on x
tlim is the initial bound on t
rho is the value of rho
CONTOL is the desired convergence value
sum gives the value of the objective function on output
k gives the number of subsystems,
and np give the number of variables and
constraints in each subsystem respectively
itfix is the number of iterations that the limit on t is held fixed
at tlim before being reduced
itval is the limit on the total number of outer iterations
```

integer maxsub,maxvar,maxcon,k,n(maxsub),np(maxsub),itfix,itval
double precision amat(maxsub**2,maxvar,maxvar),
&  cmat(maxsub**2,maxcon,maxvar),
&  d(maxsub,maxcon),x(maxsub,maxvar),
&  alph(maxsub,maxsub),beta(maxsub,maxsub),
&  delx,rho,tlim,CONTOL,sum

double precision prevx(maxsub,maxvar),kpo(maxsub),
&  apk(maxsub,maxsub),r0x(maxsub,maxvar),
&  newx(maxsub,maxvar),min(maxsub,maxcon),
&  rpk(maxsub,maxsub),prrk(maxsub,maxsub),
&  tpk(maxsub,maxsub),tprk(maxsub,maxsub),
&  thet(maxsub,maxsub),thett(maxsub,maxsub),
&  dfvec(maxsub**2*maxvar+2*maxsub**2),
&  dfjx(maxsub**var+2*maxsub**2),
&  movelo(maxsub,maxsub),
&  oldrho,dscp,dscn,lomax
integer iref(maxsub,maxsub),is(maxsub),mmark(maxsub,maxcon),
& icheck,ictr,ictr2,itcnt,idcnt,i,j,ii,jj

do i ii = 1,k
  alph(ii,ii) = 1.0
  beta(ii,ii) = 1.0
continue

do 2 ii = 1,k
  do 2 jj = 1,n(ii)
    prevx(ii,jj) = x(ii,jj)
continue

oldrho = rho
calculation of values of KS FW.
call kpocal(k,rho,x,kpo,np,n,cmat,beta,d)
do 3 i=1,k
do 3 j = 1,n(i)
rhox(i,j) = 0.0
continue
calculation of apk AND iref
iref is an array that is used to indicate zero
cross derivatives and is used to set the lower and upper
limits on the r and t coefficients
call apkcal(np,n,cmat,beta,x,rho,k,d,apk,iref)
calculation of r,t and s coefficients
call rtsinit(kpo,np,n,cmat,beta,x,rho,apk,k,d,apk,tpk,is,
& prpk,ptpk)
&icheck = 0
dscp = 0.5
dscn = 0.5
repeat the process for the specified number of iterations
do 13 itcnt = 1,itval
calculation of derivatives of theta W.R.T r and t coefficients
do 6 i = 1,k
call calthet(i,cmat,d,neux,n,np,k,beta,thetr,thett,is,mlam,
& mmark,ictr,ictr2,kpo)
do 7 ii = 1,k
  do 7 jj = 1,n(ii)
    dfvec(idcnt) = neux(ii,jj)-x(ii,jj)
    dfjx(idcnt) = x(ii,jj)
    idcnt = idcnt+1
continue
do 8 i = 1,k
   do 8 jj = 1,n(i)
      x(i, jj) = newx(i, jj)
      if (dabs(x(i, jj)).le.1.0e-15) x(i, jj) = 0.0
   continue
8 continue

c TO CALCULATE THE OBJECTIVE FUNCTION VALUE
   call objcal(amat, x, k, n, sum)

c ictr AND ictr2 INDICATE WHETHER THE COP NEEDS TO BE
   SKIPPED i.e., IN THE CASE OF A CONSTANT OBJECTIVE FUNCTION
   if ((ictr.ne.k*.5).or. (ictr2.ne.k*.5)) then
      call apkcal(np, n, cmat, beta, newx, rho, k, d, apk, iref)
      do 10 ii = 1, k
         do 9 jj = 1, k
            if (apk(ii, jj).gt.tlim) then
               lomax = -tlim
            else
               lomax = -apk(ii, jj)
            endif
            movelo(ii, jj) = lomax
         continue
      9 continue
53
VITA.

Jayashree Shankar was born on 2nd October, 1966 in Madras, India. She graduated from Good Shepherd Higher Secondary School in Madras, in April 1983. She received her first Bachelors Degree in Mathematics from Stella Maris College, Madras, affiliated to the University of Madras, in April 1986. Her second Bachelors Degree was in Computer Science from the Indian Institute of Science, Bangalore, in June 1989. In August 1992 she earned an MS degree in Computer Science from Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Jayashree Shankar