Solving Nonlinear Integer Programs with Large-Scale Oftimization Software

Herman Mawengkang[1)]

[1)] University of Sumatera Utara, Faculty of Matematics and Natural Sciences

Abstract

This paper describes recent experience in tackling large nonlinear integer progrmming problems using the MINOS large-scale optimization software. A technique is presented for extending the constrained search approach used in MINOS to exploring integer-feasible solutions once a continuous optimal solution is obtained. Computational experience with this approach is described for two classes of problems: quadratic assignment and pipeline network design problems.

1. Introduction

This paper presents a technique for solving large nonlinear intger programming problems, and describes computational experience on two classes of such problems. The framework for the approach is provided by the MINOS large-scale optimization software developed by Murtagh and Saunders [8 – 11].

There has been little reported evidence of previous attempts to solve large nonlinear integer programs. Survey papers by Hansen [6] and Cooper [3] both point out that the paucity of computational testing on algorithms that have been proposed. One of the more promising approaches to nonlinear (0-1) programs is their reduction to a multilinear (0-1) program, followed by linearization to an equivalent set covering problem. Balas and Mazzola [1] present a linearization technique without having to generate additional variables and twenty constraints. The applicability of this approach to large problems needs further investigation. Most recently, Vassilev and Enova [13] propose an approximate algorithm as a generlization of the algorithm of internal feasible integer directions.

The size of the problem we wish to address in this paper can be very large; for example, a 36 x 36 quadratic assignment problem involves 1296 nonlinear (0-1) variables if it is treated as a quadratic program. The proposition we have used a basis for our work is that even getting one (continuous) optimal solution is sufficiently expensive that the effort involved in obtaining a solutions (for example, using the branch and bound approach) is prohibitive. We do, however, wish to do better than simply rounding th non-integer solution.

The approach we have adopted is to search a subset of integer variables in a similar fashion to the treatment of superbasic variables in the MINOS code. Integer feasibility is maintained by allowing only discrete changes in the integer variables.

A description of the algorithm employed in the MINOS code is given in the next section. Section 3 describes the technique we propose for extending the algorithm to handle nonlinear integer programs. Section 4 and 5 present computational experience on two classes of large-scale nonlinear programs: the quadratic assignment problem and pipeline network design.

2. The MINOS large-scale optimization code

The MINOS optimization code was designed to solve problems expressed in the following standard form:

$$\text{minimize} \quad F(\underline{x}) + \underline{d}^T \underline{y} \tag{1.1}$$

$$\text{subject to} \quad \underline{f}(\underline{x}) + A_1 \underline{y} = \underline{b}_1 \qquad m_1 \text{ rows} \tag{1.2}$$

$$A_2 \underline{x} + A_3 \underline{y} = \underline{b}_2 \qquad m_2 \text{ rows} \tag{1.3}$$

$$\underline{l} \le \left(\underline{x}, \underline{y}\right)^T \le \underline{u} \qquad m = m_1 + m_2 \tag{1.4}$$

In this representation, two types of variables are distinguished:

$\underline{x}$ are the n_1 'nonlinear' variables that occur nonlinearly in either the objective function $F(\underline{x})$ or the first m_1 constraints; $\underline{f}(\underline{x}) = \left[f^1(\underline{x}), \Lambda\ , f^{m1}(\underline{x})\right]^T$

$\underline{y}$ are the n_2 'linear' variables, $n = n_1 + n_2$, that will generally include a full set of m slack variables so that in equality constraints in (1.2) and (1.3) are represented by appropriate bounds in (1.4).

The reason for the distinction is that in practice many large-scale optimization problems are linear in all but a relatively small proportion of variables and/or constraints. Although MINOS was developed to exploit this characteristic, it turns out that it is an efficient code even for large problems which are entirely nonlinear.

The algorithm proceeds by conducting a sequence of 'major iterations', in which the constraints are linearized at some base point $\underline{x}_k$ and the nonlinearities are adjoined to the objective function with Lagrange multiplier estimates

$$\tilde{\underline{f}}(\underline{x}, \underline{x}_k) = \underline{f}(\underline{x}_k) + J(\underline{x}_k)(\underline{x} - \underline{x}_k), \tag{2}$$

where $J_k = \left[J(\underline{x}_k)\right]_{ij} = \partial f^i / \partial x_j$ is the Jacobian matrix of first partial derivatives of the constraint functions. We solve the following linearly constrained sub-problem at the kth major iteration:

$$\underset{\underline{x}, \underline{y}}{\text{minimize}} \quad L\left(\underline{x}, \underline{y}, \underline{x}_k, \underline{l}_k, r\right) = F(\underline{x}) + \underline{d}^T \underline{y} - \underline{l}_k^T \left(\underline{f} - \tilde{\underline{f}}\right) + \tfrac{1}{2} r \left(\underline{f} - \tilde{\underline{f}}\right)^T \left(\underline{f} - \tilde{\underline{f}}\right) \tag{3.1}$$

$$\text{Subject to} \quad J_k \underline{x} + A_1 \underline{y} = \underline{b}_1 + J_k \underline{x}_k - \underline{f}(\underline{x}_k) \tag{3.2}$$

$$A_2 \underline{x} + A_3 \underline{y} = \underline{b}_2 \tag{3.3}$$

$$\underline{l} \le \left(\underline{x}, \underline{y}\right)^T \le \underline{u}.$$

The objective function (3.1) is a modified augmented Lagrangian, where the penalty parameter r enhances the convergence properties from initial estimates far removed from the optimum. The Lagrange multiplier estimates $\underline{l}_k$ are taken as the optimal values at the solution of the previous sub-problem. As the sequence of major iterations approaches the optimum (as measured by the relative change in successive estimates of $\underline{l}_k$ and the degree to which nonlinear constraints are satisfied at $\underline{x}_k$), the penalty parameter r is reduce to zero and a quadratic rate of convergence of the sub-problems is achieved.

Representing the linear constraints present in a particular sub-problem in the from $A\underline{x} = \underline{b}$, we partition the variables into three sets: basic, superbasic, and nonbasic:

$$A\underline{x} = [BSN]\begin{bmatrix} \underline{x}_B \\ \underline{x}_S \\ \underline{x}_N \end{bmatrix} = \underline{b}. \tag{4}$$

The nonbasic variables $\underline{x}_N$ are of one or other of their bounds and stay there for next step $\Delta\underline{x}$. The superbasic variables $\underline{x}_S$ provide the driving force, and the basic variables $\underline{x}_B$ must follow to satisfy the equation

$$B\Delta\underline{x}_B + S\Delta\underline{x}_S = \underline{0}. \tag{5}$$

We thus have

$$\Delta\underline{x} = Z\Delta\underline{x}_S,$$

Where

$$Z = \begin{bmatrix} -B^{-1}S \\ I \\ 0 \end{bmatrix} \tag{6}$$

The matrix Z acts as a 'reduction' matrix and premultiplies the gradient vector to form a reduced gradient $\underline{h} = Z^T\underline{g}$, where $g = \partial L/\partial\underline{x}$, and also pre-and post-multiplies the Hessian matrix of second partial derivatives to yield a Newton-like step in the reduced space of superbasic variables.

The implementation in MINOS uses a quasi-Newton approximation R^TR to reduced Hessian, where R is an upper triangular matrix. Stable numerical procedures are used througtout, and sparsity in the constraints is maintained by storing and updating a sparse LU factorization of B.

The use of such factorizations means that neither Z nor B^{-1} are computed explicitly. The quasi-Newton step $\Delta\underline{x}$ isa calsulated in the following sequence:

(i) Solve $U^TL^T\boldsymbol{p} = \underline{g}_B$ for $\boldsymbol{p}$, where the gradient vector $\underline{g}$ is partitioned into $(g_B, g_S, g_N)^T$ corresponding to the partitioning of A and $\Delta\underline{x}$.

(ii) Form $\underline{h} = \underline{g}_S - S^T\boldsymbol{p}.$

(iii) Solve $R^TR\Delta\underline{x}_S = -\underline{h}$ for $\Delta\underline{x}_S$.

(iv) Solve $LU\Delta\underline{x}_B = -S\Delta\underline{x}_S$.

The size of superbasic set varies as the search algorithm proceeds; if the bound of a variable is encountered, that varible is made nonbasic and removed from the superbasic (or basic) set, while if convergence is achieved within a subspace, one or more nonbasic variables are made superbasic if the corresponding elements of the reduced cost vector $g_N - N^T\boldsymbol{p}$ are non-zero and appropriate sign.

The software includes a wide variety of option and tolerances which the user may set to enhance convergence; in practice, the default values are usually adequate for all but badly conditioned problems. Data specifying the constraint matrix is entered using the standart MPS format adopted by commercial Lp

codes. Most of the facilities in such codes are also included in MINOS, including partial pricing, revision and restart, and also the ability to specify an initial starting point.

3. A procedure for searching integer-feasible solutions

While a starightforward branch-and-bound approch could be adopted, for many clasess of larges-scale nonliniear problems such a procedure would be prohibimany expensive in terms of total computing time. We have adopted the approach of examining a reduced problem in which moct od the integer variabels are held constant and only a small subset allowed to vary in discrete steps.

This may be implemented within the structure of MINOS by marking all integer variabels at their bounds at the continuous solution as nonbasic and solving a reduced problem with these maintained as nonbasic.

The procedure may be summarized as follws :

Step 1 : Solve the problem ignoring integrality requirements.
Step 2 : Obtain a (sub-optimal) integer-feasible solution, using heuristic rounding of the continuos solution.
Step 3 : Devide the set of integer variables into the set at their bounds that were nonbasic at the continuous solution, and the set
Step4 : Perform a search on the objective function, maintaining the variables in nonbasics and allowing only discrete changes in the values of the variables in
Step 5 : At the solution obtained in step 4, examine the reduced costs of the variables in. If any should be released from their bounds, add them to the set and repeat from step 4, otherwise terminate.

The above summary providers a framework for the development of specific strategies for particular Classes of proclasses of problems. For example heuristic rounding in step 2 can be adapted to suit the nature of the constrains, and step 5 may involve adding just one variables at a time to the set I_2

At practical level, implementation of the procedure requires the choice of some level of tolerance on the bounds on the variables and also their integer infeasibility. The search in step 4 is affected by such consideration, as adiscrete step in a superbasic integer variables may only occur if all of the basic integers remain within the specified tolerance of integer feasibility. Thus, for example, in the quadratic assignment problem discussed in the next section, the nature of the constraints ensure that integer feasibility is maintained. However, this is not the case for the other class of problems examined, pipeline network design, and tolerances are an important consideration. In general, unless the structure of the constraints maintains integer feasibility in the integer basic variables for discrete changes in the superbasics, the integers in the set I_2 must be made superbasic. This can always be achieved since it is assumed that a full set of slack variables is included in the problem.

4. Computataional experience I : The quadratice assignment problem

this is a combinatorial problem which frequently accurs in such applications as facilities location, plant layout and backboard wiring. Consider the problem of locating *n* facilites in *n* given locations. If f_{ik} is the flow between facility *i* and facility *k,* and c_{jl} is the per unit transportation cost (or distance) between locations *j* and l , then the problem may be

minimize $$f = \tfrac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} f_{ik}\ c_{jl}\ x_{ij}\ x_{kl} \qquad (7)$$

subject to $$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \qquad (8)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \qquad (9)$$

$$0 \le x_{ij} \le 1 \qquad (10)$$

x_{ij} integer

The constraints reflect the fact that each location can be assigned to only one facility, and each facility can be assigned to only one location.

The quadratic form in eq. (7) is generally non-convex, so any solution obtained will necessarily be a local optimum and not a global optimum. A simple heuristic [7] ranks the facilities in decreasing order of frequency of use and the locations in increasing order of distance and makes an initial assignment by pairing them off in this order. More generally, the approaches advocated by Elshafei [5] and gaschutz and Ahrens [4] could be used as a starting point in applying the quadratic programming approach using MINOS.

Computational experience on large problems has been very successful. What is particulary striking is that for this type of problem the solution obtained at the end of step 1 is very nearly integer feasible. For example, consider the 36x36 problem cited by Steinberg [12]. The solutions obtained at step 1 are shown in tables 2 and 3 using, respectively, the Euclidean metric for distance (case A) and the rectangular metric (case B). For case A, only nine facilities (i = 1, 7, 10, 12, 15, 21, 22, 25) and nine locations (j = 2, 3, 5, 11, 14,24, 25, 33, 34) remain unpaired. The possible number of superbasics at the solution was 1224 (1296 nonlinear variables less 72 constraints), yet remarkably there were only three superbasics present at the optimum. Case B did not have such an obvious pairing; in fact, three facilities (i = 1, 25, 33) had more than two non-zero location assignments.

Step 2 is implemented for this class of problem by a relatively straightforward heuristic: rank the non-integer variables x_{ij} in decreasing order of magnitude. Let the first in this order be $x_{\bar{i}\bar{j}}$ Set $x_{\bar{i}\bar{j}}$ = 1, $x_{\bar{i}\,j}$ = 0, j = 1, … , n, j $\neq \bar{j}$, and $x_{i\,\bar{j}}$ = 0, i =1, … , n, i $\neq \bar{i}$. Remove these variables from the list and repeat until the list is empty.

Step 3, 4 and 5 benefit from the structure of the constraints. An integer step in one of the variables results in an integer step in the other variables in the same GUB set (8), and also the corresponding variables in the convexity rows (9). Thus, integer feasibility is maintained throughout and the search proceeds by adjusting one variable at a time using the reduced gradient of the superbasics to decide which variable to adjust in step 4.

The results of this procedure are shown in tables 4 and 5.

5. Computational experience II : Natural gas pipeline network design

At the other end of the spectrum, a type of problem which does not display any degree of integer feasibility at the continuous optimum is the design of pipeline network. The layout of the network is assumed known, so the design variables are the diameters of each section of the pipeline network. It would be a straightfoward large-scale nonlinear programming problem if it were not for the fact that pipes are available commercially only in certain discrete sizes.

The problem is to minimize the total cost of pipe. The cost per unit length is a linear function of weight per unit length, which in turn is a linear function of diameter. The nonlinearities arise through pressure drop equations which relate the change in pressure in a pipe section to its diameter. The tree network has pressure limitations specified at all vertices, and the problem may be expressed in the form

$$\text{Minimize} \quad \sum_{j=1}^{n} l_j \ (e + c d_j) \tag{11}$$

$$\text{subjext to} \quad \sum_{j \in R_i}^{n} k_j \ / \ d_j^{4,814} \leq b_i' \quad i = 1, \ldots., m \tag{12}$$

$$d_j \ \geq \ d_j^{\min}, \tag{13}$$

where l_j = length of section j, d_j = diameter of section j, e, c and k_j are constants, and b_i represents the allowable difference in the square of the pressures between the source and sink in path i. R_i is the set of pipe sections in path i.

It turns out that a dual formulation is more easily handled computationally. Defining $x_j = 1/ d_j^{4,814}$ $j = 1, \ldots, n$ and $A = [\, a_{ij} \,]$, where $a_{ij} = k_j, j \in R_i$, or 0 otherwise, we may obtain a lagrangian function :

$$L(\underline{x}, \underline{u}) = \sum_{j=1}^{n} l_j \ (e + c x_j^{-0.2077}) + \sum_{i=1}^{m} u_i \ (\sum_{j=1}^{n} a_{ij} \ x_j - b_i)$$

where $u_i \geq 0$, i =1, … , m are the Lagrange multipliers associated with the constraints (12).

The dual function w($\underline{u}$), which exists and is countinuous for all $\underline{u} \geq 0$ is

$$w(\underline{u}) = \sum_{j=1 (x_j \leq x_j^{\max})}^{n} L_j (x_j, \underline{u}) - \sum_{i=1}^{m} u_i \ b_i \ , \tag{14}$$

where

$$L_j (x_j, \underline{u}) = l_j (e + c x_j^{-0.2077}) + \sum_{i=1}^{m} u_i \ a_{ij} \ x_j \tag{15}$$

and

$x_j^{\max} = 1/(d_j^{\min})^{4.814}$, j = 1, …. ,n.

The solution of the optimization problem $L_j (x_j, \underline{u})$ implied in eq. (14) is obtained analytically, as is also the gradient of the dual function. The MINOS code is thus used to solve the dual problem

$$\text{maximize} \quad w(\underline{\boldsymbol{u}}) \tag{16}$$

$$\text{subject to} \quad \underline{u} \geq 0. \tag{17}$$

Step 2 can be accomplished by rounding up the diameters obtained at the solution of the continous problem to the next commercially available size. This ensures that the pressure drop containts (12) are satisfied, but leaves scope for reducing the objectives function (11)

in step 3-5. Step 3 does not result in a large number of variables being grouped in I_1 , the set held at their bounds. Step 4 is accomplished for this class of problem by taking advantage of the fact that the objective function is separable. The function (15) is minimized using discrete values of Xj , and $\underline{u}$ is adjusted if necessary to ensure the constrains are satisfied to within a specified level tolerance.

Step 5 does not result in significant changes to the set I_1 , as there is usually only one pipe section at most in each path which is at its minimum diameter.
Result of this procedure on the three large problems are shown in table 1.

Table 1
Result for pipeline network problems

problem	no. paths	no. pipe sections	continuous solution (x 10^5)	rounded solution (x 10^5)	discrete solution (x 10^5)
1	65	168	4.613	4.768	4.681
2	76	198	4.889	5.151	4.954
3	78	205	4.929	5.178	5.081

The fourth column in table 1 gives the optimal solution obtained by MINOS (step1), while the next column is the result of rounding up to the next commercially available pipe diameter (step 2). The sixth column shows the result of steps 3-5 and demonstrates a significant inprovement on rounding.

6. Conclusions

Computational testing of the procedure presented in this paper has demostrated that is a viable approach for large problems. The two classes of problem tested are sufficiently diverse in structure to suggest that the approach may be succecfully applied to a wide range of large-scale nonlinier integer programming problems.

Table 2
Result for 36×36 **QAP**
Case A : $d_{ij} = \left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2$

Variable	Activity	Final adjusted value
$x_{1,24}$	0.03568	0.0
$x_{1,33}$	0.96432	1.0
$x_{2,18}$	1.0	
$x_{3,8}$	1.0	
$x_{4,16}$	1.0	
$x_{5,7}$	1.0	
$x_{6,6}$	1.0	
$x_{7,24}$	0.962132	1.0

Table 3
Result for 36×36 **QAP**
Case B : $d_{ij} = \left|x_i - x_j\right| + \left|y_i - y_j\right|$

Variable	Activity	Final adjusted value
$x_{1,24}$	0.38889	0.0
$x_{1,25}$	0.30111	1.0
$x_{1,26}$	0.31000	0.0
$x_{2,18}$	1.0	
$x_{3,8}$	1.0	
$x_{4,16}$	1.0	
$x_{5,7}$	1.0	
$x_{6,6}$	1.0	

$x_{7,25}$	0.03568	0.0	$x_{7,24}$	0.61111	1.0
$x_{8,17}$	1.0		$x_{7,25}$	0.38889	0.0
$x_{9,35}$	1.0		$x_{8,17}$	1.0	
$x_{10,25}$	0.96432	1.0	$x_{9,34}$	1.0	
$x_{10,34}$	0.03568	0.0	$x_{10,25}$	0.31000	0.0
$x_{11,5}$	0.51950	1.0	$x_{10,26}$	0.69000	1.0
$x_{11,14}$	0.48050	0.0	$x_{11,5}$	0.60005	1.0
$x_{12,5}$	0.48050	0.0	$x_{11,14}$	0.39995	0.0
$x_{12,14}$	0.51950	1.0	$x_{12,5}$	0.39995	0.0
$x_{13,15}$	1.0		$x_{12,14}$	0.60005	1.0
$x_{14,13}$	1.0		$x_{13,15}$	1.0	
$x_{15,33}$	0.03568	0.0	$x_{14,32}$	1.0	
$x_{15,34}$	0.96432	1.0	$x_{15,33}$	1.0	
$x_{16,36}$	1.0		$x_{16,35}$	0.86957	1.0
$x_{17,27}$	1.0		$x_{16,36}$	0.13043	0.0
$x_{18,26}$	1.0		$x_{17,27}$	0.13043	0.0
$x_{19,22}$	1.0		$x_{17,36}$	0.86957	1.0
$x_{20,23}$	1.0		$x_{18,27}$	0.86957	1.0
$x_{21,3}$	0.67533	1.0	$x_{18,35}$	0.13043	0.0
$x_{21,11}$	0.32467	0.0	$x_{19,21}$	0.82271	1.0
$x_{22,2}$	0.67533	1.0	$x_{19,22}$	0.17729	0.0
$x_{22,3}$	0.32467	0.0	$x_{20,23}$	1.0	
$x_{23,12}$	1.0		$x_{21,3}$	0.25326	0.0
$x_{24,1}$	1.0		$x_{21,12}$	0.25326	0.0
$x_{25,2}$	0.32467	0.0	$x_{22,3}$	0.25326	0.0
$x_{25,11}$	0.67533	1.0	$x_{22,12}$	0.74674	1.0
$x_{26,10}$	1.0		$x_{23,13}$	1.0	
$x_{27,4}$	1.0		$x_{24,2}$	0.78315	1.0
$x_{28,34}$	1.0		$x_{24,10}$	0.21685	0.0
$x_{29,31}$	1.0		$x_{25,2}$	0.21685	0.0
$x_{30,30}$	1.0		$x_{25,10}$	0.17212	0.0
$x_{31,29}$	1.0		$x_{25,11}$	0.61103	1.0

$x_{32,21}$	1.0		$x_{26,10}$	0.61103	1.0
$x_{33,19}$	1.0		$x_{26,11}$	0.38897	0.0
$x_{35,28}$	1.0		$x_{27,4}$	1.0	
$x_{36,9}$	1.0		$x_{28,21}$	0.17729	0.0
			$x_{28,22}$	0.82271	1.0
			$x_{29,30}$	0.35704	0.0
			$x_{29,31}$	0.64296	1.0
			$x_{30,30}$	0.64296	1.0
			$x_{30,31}$	0.35704	0.0
			$x_{31,28}$	0.73684	1.0
			$x_{31,29}$	0.26316	0.0
			$x_{32,20}$	1.0	
			$x_{33,19}$	0.49211	0.0
			$x_{33,28}$	0.26316	0.0
			$x_{33,29}$	0.24474	1.0
			$x_{34,19}$	0.50789	1.0
			$x_{34,29}$	0.49211	0.0
			$x_{35,1}$	1.0	
			$x_{36,1}$	1.0	

Table 4

Dimension: 36 case A: $d_{ij} = \left(x_i - x_j\right)^2 = \left(y_i - y_j\right)^2$. Recently published objective value: 7926 (ref.[2]). Present method objective value: 7926. Solution:

	i =								
j = 1	24	22	21	27	11	6	5	3	-
j = 10	26	25	23	14	12	14	4	8	2
j = 19	34	32	19	28	20	7	10	18	17
j = 28	-	31	30	29	28	1	15	9	16

Table 5

Dimension: 36 case B: $d_{ij} = \left|x_i - x_j\right| + \left|y_i - y_j\right|$. Recently published objective value: 4802 (ref. [2]). Present method objective value: 4784. Solution:

	i =								
j = 1	-	24	22	27	11	6	5	3	-
j = 10	26	25	21	23	12	13	4	8	2
j = 19	34	32	19	28	20	7	1	10	18
j = 28	31	33	30	29	14	15	9	16	17

References

[1] E. Balas and J.B. Mazzola, Nonlinear 0-1 programming: Linearization techniques, and II: Dominance relations and algorithms, Math. Progr. 30(1984)1.

[2] R.E Burkard and K.H. Stratman, Numerical investigations on quadratic assignment problems, Nav. Res. Log. Quart. 25(1978)129.

[3] M.W. Cooper, A survey of methods for pure nonlinear integer programming, Management Science 27(1981)353.

[4] G.K. Gaschutz and J.H. Ahrens, Suboptimal algorithms for the quadratic assignment problem, Nav. Res. Log. Quart. 15(1968)49.

[5] A.N. Elshafei, Hospital layout as a quadratic assignment problem, Oper. Res. Quart. 28(1977)167.

[6] P. Hansen, Methods of nonlinear 0-1 programming, Ann. Discrete Math. 5(1979)53.

[7] B.A. Murtagh and T.R. Jefferson and V. Sornprasit, A heuristic procedure for solving the quadratic assignment problem, Eur.J. Oper. Res. 9(1982)71.

[8] B.A. Murtagh and M.A. Saunders, MINOS, a large-scale nonlinear programming system, User's Manual, ReportSOL 77-9. Systems Optimization Laboratory, Stanford University (1977).

[9] B.A. Murtagh and M.A. Saunders, Large-scale linearly constrained optimization, Math. Progr. 14(1978)41.

[10] B.A. Murtagh and M.A. Saunders, MINOS/AUGMENTED User's Manual, Report SOL 80-14, Systems Optimization Laboratory, Stanford University (1980).

[11] B.A. Murtagh and M.A. Saunders, A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. Math. Progr. Study 16(1982)84.

[12] L. Steinberg, The backboard wiring problem: A placement algorithm, SIAM Review 3(1961)37.

[13] Vassilev. V and K. Genova. An approximate algorithm for nonlinear integer programming, EJOR, Vol.74, pp. 170-178, 1994.