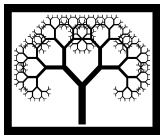
Paper 59



Proceedings of the Eleventh International Conference
on Computational Structures Technology,
B.H.V. Topping, (Editor),
Civil-Comp Press, Stirlingshire, Scotland

Handling Linear Constraints in Genetic Algorithms

H.L. Pina[1] and J.F.A. Madeira[1,2]
[1]IDMEC/IST-Instituto Superior Técnico, Lisbon, Portugal
[2]ISEL-Instituto Superior de Engenharia de Lisboa
Lisbon, Portugal

Abstract

Genetic algorithms lack a general methodology to handle constraints and several approaches have been tried. The best method is certainly to generate by proper encoding of legal individuals only, that is, individuals obeying all the constraints. This may be possible for simple constraints on the upper and lower bounds of design variables but becomes increasingly more difficult or 'next to impossible' for more complicated constraints. Two remedies can be attempted: development of repair mechanisms that modify infeasible individuals into feasible ones, an approach that tends to disrupt the normal performance of the genetic operators; or the design of suitable genetic operators (crossover and mutation) that maintain feasibility. The 'next to best' method relies on the transformation of the constrained problem into an unconstrained one by modifying the fitness function to include terms penalizing the violation of constraints; in this case the presence of illegal individuals can "pollute" the population and thereby slow the convergence process.

Equality constraints pose additional and specific difficulties. As they effectively reduce the dimensionality of the problem, the penalization approach is inefficient since working in the full search space leads to much more fitness function evaluations then otherwise required. Therefore the best technique is to split the problem variables in two sets; the free or independent variables and the dependent variables. These are then expressed in terms of the free variables which are the ones exclusively "seen" by the genetic algorithm, that is, they are effectively "eliminated" from the genetic algorithm.

In the present paper we focus on the problem of proper handling of equality constraints adopting the elimination approach bearing in mind that in many real life applications the linear set of constraints is generated automatically by some user program and one is not sure *a priori* if this set is linearly independent a fact that has to be checked and dealt with. Additionally, it may happen that the set of constraints is 'almost' linearly dependent thus making the problem ill-conditioned a difficulty that

must be addressed in order to obtain meaningful results. We assess three elimination methods: the Gauss-Jordan elimination method, the orthogonal factorization method and the singular value decomposition as tools to identify the independent and dependent variables, examining both the well-conditioned (linear independence) and the ill-conditioned ('almost' linear dependence) cases. Some representative examples are presented and discussed.

Keywords: genetic algorithms, linear constraints, elimination methods, ill-conditioning.

1 Introduction

Genetic Algorithms lack a general methodology to handle constraints and several approaches have been tried (see [11], [3], [2] and [4] and references therein).

1. The best method is certainly to generate by proper encoding only legal individuals, that is, individuals obeying all the constraints. This may be possible for simple upper and lower bounds type constraints but becomes increasingly more difficult or next to impossible for more complicated constraints. Two remedies can be attempted:

   - Development of repair mechanisms that modify infeasible individuals into feasible ones, an approach that tends to disrupt the normal performance of the genetic operators;
   - Designing suitable genetic operators (crossover and mutation) that maintain feasibility;

2. Transformation of the constrained problem into an unconstrained one by modifying the fitness function to include terms penalizing the violation of constraints; in this case the presence of illegal individuals can "pollute" the population and thereby slow the convergence process.

Equality constraints pose additional and specific difficulties. As they reduce the dimensionality of the problem, approach 2 above is inefficient since working in the full search space leads to much more fitness function evaluations then otherwise needed. Therefore the best technique is to eliminate the constrained variables by expressing them in terms of the free variables which are the ones exclusively "seen" by the Genetic Algorithm.

In the present paper we focus on the problem of proper handling of equality constraints adopting the elimination approach bearing in mind that in many real life applications the linear set of constraints is generated automatically by some user program and one is not sure a priori if this set is linearly independent a fact that has to be checked and dealt with. Additionally, it may happen that the set of constraints is "almost" linearly dependent thus making the problem ill-conditioned a difficulty

that provided some motivation for the present work and must be addressed in order to obtain meaningful results. We study the use of three methods of elimination of variables through the set of linear constraints and address both the well-conditioned (linear independence) and the ill-conditioned ("almost" linearly dependency) cases.

2 Problem Formulation

Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ be the vector of design variables and $f : \mathbf{x} \mapsto \mathbb{R}$ be the fitness or objective function. The optimization problem we intend to study is given by

$$\mathbf{x}^* = \operatorname{argmin} f(\mathbf{x}) \tag{1}$$

subject to the following set of constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{A} \in \mathbb{R}^{m \times n}, \quad \mathbf{b} \in \mathbb{R}^m \tag{2}$$

$$\mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \tag{3}$$

Thus we have a set of $m < n$ linear equality constraints (2) together with with lower and upper bounds constraints (3) indicated by the superscripts L and U, respectively.

The constraints (2) and (3) define the domain of feasibility of the problem and we must insure by proper specification of these constraints that it is not empty.

Remark 1 *It is acceptable for the vectors $\mathbf{x}^L$ and $\mathbf{x}^U$ to have components equal to $-\infty$ and $+\infty$, respectively.*

Remark 2 *If there are other inequality constraints other than the bounds* (3) *we assume that they have been incorporated in the fitness function f by adequate penalty terms so we will not refer to them explicitly in the sequel.*

3 Methodology

As we pointed out we deal with linear constraints by expressing dependent or constrained variables in terms of independent or free variables which are the ones to be codified in the genome in the Genetic Algorithm.

To this end we will consider three approaches: (i) the Gauss-Jordan elimination method (ii) the orthogonal factorization and (iii) the singular value decomposition.

3.1 Gauss-Jordan Elimination Method (GJE)

Gauss-Jordan elimination method with partial pivoting applied to the linear constraints (2) yields the system (see [10] or [12]):

$$\begin{pmatrix} \mathbf{I}_r & -\mathbf{C} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}_r \\ \mathbf{x}_s \end{pmatrix} = \begin{pmatrix} \mathbf{d}_r \\ \mathbf{d}_s \end{pmatrix} \tag{4}$$

where $\mathbf{I}_r \in \mathbb{R}^{r\times r}$ is the identity matrix of order r with $r = \operatorname{rank}\mathbf{A}$ and $\mathbf{C} \in \mathbb{R}^{r\times s}$, where $s = n - r$, $\mathbf{x}_r \in \mathbb{R}^r$, $\mathbf{x}_s \in \mathbb{R}^s$, $\mathbf{d}_r \in \mathbb{R}^r$, $\mathbf{d}_s \in \mathbb{R}^s$.

Remark 3 *For consistency we should have* $\mathbf{d}_s = \mathbf{0}$.

Therefore the constrained variables $\mathbf{x}_r$ can be expressed in terms of the free variables $\mathbf{x}_s$ by the relation

$$\mathbf{x}_r = \mathbf{d}_r + \mathbf{C}\mathbf{x}_s \tag{5}$$

The fitness function thus becomes a function of $\mathbf{x}_s$ only.

Remark 4 *Relation* (5) *entails that the bounds* (3) *on the* $\mathbf{x}_r$ *variables have to be enforced by penalization of the fitness function.*

There are some possible drawbacks with this approach that can be better perceived with the following example. Let the extended matrix of the linear constraints (2) be ($n = 3$, $m = 2$)

$$\begin{pmatrix}\mathbf{A} & \mathbf{b}\end{pmatrix} = \begin{pmatrix}1 & 1 & 1 & 1\\ 1 & 1+\epsilon & 1+2\epsilon & 1+\epsilon\end{pmatrix} \tag{6}$$

where $\epsilon \neq 0$ is a "small" number. As we see the two constraints are "almost" (contingent on ϵ) linear dependent. Performing the Gaussian elimination we obtain

$$\begin{pmatrix}\mathbf{A} & \mathbf{b}\end{pmatrix} \rightarrow \begin{pmatrix}1 & 1 & 1 & 1\\ 0 & \epsilon & 2\epsilon & \epsilon\end{pmatrix} \tag{7}$$

Examining this system it seems reasonable that we should drop the second constraint and retain the first. However had we presented the constraints in reverse order the same reasoning would lead us to make a different choice. Summing up, the Gauss-Jordan elimination method is contingent to the order the constraints are presented, eliminating some constraints while keeping others despite the fact that all constraints might be equally contaminated by noise (either experimental or roundoff errors).

As an advantage of the e Gauss-Jordan method we point out to the fact that the optimization remains in a subspace of the original design variables $\mathbf{x}$ space.

3.2 Rectangular Orthogonal Factorization (QRF)

An alternative to the GJE is the $\mathbf{QR}$ decomposition of matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ (see [10] or [12]): there exists an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{m\times m}$, an upper triangular matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$ and a permutation matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{AP} = \mathbf{QR} \tag{8}$$

with $\operatorname{diag}\mathbf{R} = (\rho_1, \ldots, \rho_r, 0, \ldots, 0)$, $\rho_1 \geq \cdots \geq \rho_r > 0$ and thus $\operatorname{rank} A = r$. The permutation matrix $\mathbf{P}$ corresponds to a column pivoting of $\mathbf{A}$.

In this case system (2) can be written as

$$\mathbf{R}\mathbf{y} = \mathbf{c} \tag{9}$$

$$\mathbf{y} = \mathbf{P}^{\mathrm{T}}\mathbf{x} \tag{10}$$

$$\mathbf{c} = \mathbf{Q}^{\mathrm{T}}\mathbf{b} \tag{11}$$

Equation (9) can be partitioned as follows

$$\begin{pmatrix} \mathbf{R}_{rr} & \mathbf{R}_{rs} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{y}_r \\ \mathbf{y}_s \end{pmatrix} = \begin{pmatrix} \mathbf{c}_r \\ \mathbf{c}_s \end{pmatrix} \tag{12}$$

where, putting $s = n - r$, $\mathbf{R}_{rm} \in \mathbb{R}^{r\times m}$, $\mathbf{R}_{rs} \in \mathbb{R}^{r\times s}$, $\mathbf{y}_r, \mathbf{c}_r \in \mathbb{R}^r$ and $\mathbf{y}_s, \mathbf{c}_s \in \mathbb{R}^s$.

Remark 5 *For consistency we should have* $\mathbf{c}_s = \mathbf{0}$.

The constrained variables $\mathbf{y}_r$ can be expressed in terms of the free variables $\mathbf{y}_s$

$$\mathbf{y}_r = \mathbf{d} + \mathbf{C}\mathbf{y}_s \tag{13}$$

with

$$\mathbf{d} = \mathbf{R}_{rr}^{-1}\mathbf{c}_r, \qquad \mathbf{C} = -\mathbf{R}_{rr}^{-1}\mathbf{R}_{rs} \tag{14}$$

The original variables $\mathbf{x}$ can be retrieved using (10) and the property that $\mathbf{P}^{\mathrm{T}} = \mathbf{P}^{-1}$ holds for permutation matrices

$$\mathbf{x} = \mathbf{P}\mathbf{y} \tag{15}$$

Remark 6 *The optimization has now been switched to the* $\mathbf{y}_s$ *variables which through* (15) *are but reshuffled versions of the original* $\mathbf{x}_s$ *design variables. The previous remark about bound constraints applies here as well.*

3.3 Singular Value Decomposition (SVD)

As we pointed out above the aim is to express the dependent variables in terms of the independent ones. Due to its robustness, we assess the singular value decomposition (SVD) of matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ (see [10] or [12] for the proofs): there exists an orthogonal matrix $\mathbf{U} \in \mathbb{R}^{m\times m}$, an orthogonal matrix $\mathbf{V} \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\mathbf{S} \in \mathbb{R}^{m\times n}$ with $\mathbf{S} = \mathrm{diag}(s_1, \ldots, s_r, 0, \ldots, 0)$ whose elements are the the singular values $s_1 \geq s_2 \geq \cdots \geq s_r > 0$ in such a way that

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{16}$$

Remark 7 *We observe (see [12]) that* $\mathrm{rank}\, A = r$ *and* $r \leq m$, *the first* r *columns of* $\mathbf{U}$ *are a basis for the column space of* $\mathbf{A}$ *and the last* $n - r$ *columns of* $\mathbf{V}$ *are a basis for the nullspace of* $\mathbf{A}$.

Invoking (16), equation (2) can be written as

$$\mathbf{S}\mathbf{y} = \mathbf{c} \tag{17}$$

$$\mathbf{y} = \mathbf{V}^{\mathrm{T}}\mathbf{x} \iff \mathbf{x} = \mathbf{V}\mathbf{y} \tag{18}$$

$$\mathbf{c} = \mathbf{U}^{\mathrm{T}}\mathbf{b} \tag{19}$$

For the purpose we have in mind, (17) is partitioned as follows:

$$\begin{pmatrix}\mathbf{S}_r & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{y}_r \\ \mathbf{y}_s\end{pmatrix} = \begin{pmatrix}\mathbf{c}_r \\ \mathbf{c}_s\end{pmatrix} \tag{20}$$

where, putting $s = n - r$,

$$\begin{aligned}\mathbf{S}_r \in \mathbb{R}^{r\times r}, \quad & \mathbf{S}_r = \mathrm{diag}(s_1, \ldots, s_r) \\ \mathbf{y}_r, \mathbf{c}_r \in \mathbb{R}^r, \quad & \mathbf{y}_s, \mathbf{c}_s \in \mathbb{R}^s\end{aligned} \tag{21}$$

Under these conditions, we get

$$\begin{aligned}\mathbf{y}_r &= \mathbf{S}_r^{-1}\mathbf{c}_r \\ \mathbf{0}\mathbf{y}_s &= \mathbf{c}_s\end{aligned} \tag{22}$$

Remark 8 *We find out that, to have a consistent set of constraints, we must have* $\mathbf{c}_s = \mathbf{0}$.

Returning to relation (18) and partitioning $\mathbf{V}$ we can write

$$\begin{pmatrix}\mathbf{x}_r \\ \mathbf{x}_s\end{pmatrix} = \begin{pmatrix}\mathbf{V}_{rr} & \mathbf{V}_{rs} \\ \mathbf{V}_{sr} & \mathbf{V}_{ss}\end{pmatrix}\begin{pmatrix}\mathbf{y}_r \\ \mathbf{y}_s\end{pmatrix} \tag{23}$$

From here we conclude that

$$\mathbf{x}_r = \mathbf{d}_r + \mathbf{C}\mathbf{y}_s \tag{24}$$

where

$$\mathbf{d}_r = \mathbf{V}_{rr}\mathbf{y}_r, \qquad \mathbf{C} = \mathbf{V}_{rs} \tag{25}$$

Relation (23) allows one to express the constrained variables $\mathbf{x}_r$ in terms of the free parameters $\mathbf{y}_s$ which are the ones going to be codified in the Genetic Algorithm chromosome.

Remark 9 *Now the* $\mathbf{y}_s$ *variables are related variables to the* $\mathbf{x}$ *through* (18). *Bound constraints* (3) *have still to be enforced by penalization of the fitness function.*

If we insist in employing the original variables, we can proceed as follows. From (23) we obtain

$$\mathbf{x}_s = \mathbf{V}_{sr}\mathbf{y}_r + \mathbf{V}_{ss}\mathbf{y}_s \tag{26}$$

As matrix $\mathbf{V}_{ss}$ being a principal submatrix of the invertible matrix $\mathbf{V}$ is itself invertible we get upon substitution in (23)

$$\mathbf{x}_r = \mathbf{e}_r + \mathbf{E}\mathbf{x}_s, \qquad \mathbf{e}_r = \mathbf{d}_r - \mathbf{V}_{rs}\mathbf{V}_{ss}^{-1}\mathbf{d}_s, \quad \mathbf{E} = \mathbf{V}_{rs}\mathbf{V}_{ss}^{-1} \tag{27}$$

Remark 10 *Matrix* $\mathbf{V}_{ss}$ *albeit invertible may be ill conditioned so the procedure above is not recommended in general and it is preferable to stick to the* $\mathbf{y}$ *variables.*

Now we recover the example in subsection 3.1 and redo it applying the above SVD based procedure. Taking, for the sake of definiteness, $\epsilon = 10^{-6}$ we get that

$$\mathbf{S} = \mathrm{diag}(2.449490967528560, 9.99999499693279310^{-7}) \tag{28}$$

If we set the second singular value to zero and apply equation (27) we find that the effective linear constraint that is retained is

$$x_1 = 1.000000500000250 - 1.000000500000251x_2 - 1.000001000000501x_3 \tag{29}$$

The same result (apart from roundoff errors) would be obtained had rows 1 and 2 be swapped in (7). Thus the robustness of the SVD as compared to Gaussian elimination seems to justify its extra cost.

4 Examples

The examples below were run using the Genetic Algorithms Toolbox of MATLAB® ([1]) frontended with a routine to perform the computations required to express the constrained variables in terms of the free variables as described above.

Example 1

This is Test Problem 2 in [5, Ch. 12] which arises in the design of planar rectangular spaces, as, for instance, in minimization of the area of electronic packages or building floors. We solve this problem for the particular data in [9] consisting of six rectangles. The vector $\mathbf{x} \in \mathbb{R}^n$ of $n = 12$ designs variables is defined as follows

$$\begin{aligned} x_i &= X_i \\ x_{6+i} &= Y_i \end{aligned}, \qquad i = 1, \ldots, 6 \tag{30}$$

where the X_i and Y_i denote the width and heights of the rectangles. The objective function (the area) to minimize is

$$f(\mathbf{x}) = \sum_{i=1}^{6} X_i Y_i \tag{31}$$

The 12 bounds constraints are

$$\begin{aligned} &5 \le X_1, \quad 5 \le X_2, \quad 2 \le X_3, \quad 4 \le X_2, \quad 4 \le X_5, \quad 5 \le X_6 \\ &5 \le Y_1, \quad 2 \le Y_2, \quad 5 \le Y_3, \quad 4 \le Y_4, \quad 5 \le Y_5, \quad 5 \le Y_6 \end{aligned} \tag{32}$$

Moreover we have a set of $m = 5$ linear constraints

$$\begin{aligned} -X_4 + Y_5 &= 0 \\ -X_1 + X_2 - X_4 + X_5 &= 0 \\ -X_1 + X_3 - X_4 + X_5 &= 0 \\ -Y_3 + Y_6 &= 0 \\ -Y_1 - Y_2 - Y_3 + Y_4 + Y_5 + Y_6 &= 0 \end{aligned} \tag{33}$$

and a set of 8 nonlinear constraints

$$\begin{aligned} X_2 - X_3 &\geq 1 \\ Y_1 - Y_4 &\geq 1 \\ X_1 Y_1 &\geq 30 \\ X_2 Y_2 &\geq 20 \\ X_3 Y_3 &\geq 20 \\ X_4 Y_4 &\geq 25 \\ X_5 Y_5 &\geq 15 \\ X_6 Y_6 &\geq 20 \end{aligned} \tag{34}$$

Since it happens that the set of linear equations is linearly independent we have $n - m = 12 - 5 = 7$ free variables, that is, the search space is $\mathbb{R}^7$ instead of $\mathbb{R}^{12}$. The remaining of the constraints, (32) and (34), are treated through penalty terms added to the fitness function in the standard way (see [6]).

The results for 50 independent runs of a Genetic Algorithm with a population of 500 individuals and a maximum of 500 generations per run, are presented in table 1. We observe that the solution obtained with GJE is worst than that obtained with either QRF or SVD.

	Fitness			
	Best	Average	$\mathbf{x}^* = \begin{pmatrix} X^* \\ Y^* \end{pmatrix}$	Time (s)
[9]	146.25	$--$	$--$	$--$
GJE	165.00	173.62	6.0000, 6.0000, 5.0000, 5.0000, 5.0000, 6.0000, 6.0000, 4.0000, 5.0000, 5.0000, 5.0000, 5.0000	635
QRF	146.52	166.03	5.0004, 5.0004, 3.9999, 4.1102, 4.1102, 5.1107, 7.0830, 3.9997, 5.0001, 6.0827, 5.0001, 5.0001	1270
SVD	146.30	159.36	5.0001, 5.0001, 4.0000, 4.0187, 4.0187, 5.0188, 7.2211, 4.0000, 5.0000, 6.2210, 5.0001, 5.0000	1012

Table 1: Results for Example 1

As we can see QRF and SVD are both able to find the minimum whereas GJE falls far behind (see Figure 1 for the actual layouts).

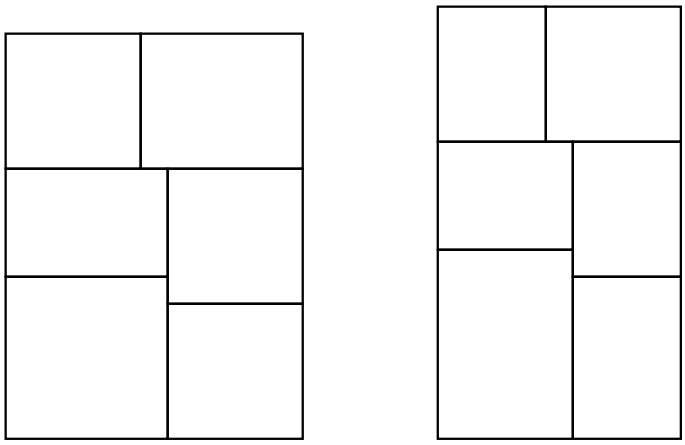


Figure 1: Example 1: layouts for GJE (left) and QRF, SVD (right)

Example 2

This is the "abel" problem in the GLOBALLib collection arising in the solution of a macroeconomic model (see [8]). It has $n = 30$ variables and $m = 14$ linear constraints and bound constraints $\mathbf{x}^L = 0$ and $\mathbf{x}^U = \infty$.

The results for 10 independent runs of a Genetic Algorithm with a population of 1000 individuals and a maximum of 5000 generations per run, are summarized in table 2.

	Fitness			
	Best	Average	$\mathbf{x}^*$	Time (s)
[8]	225.19	$--$	387.90, 389.59, ..., 145.20, 143.10	$--$
GJE	340.01	544.60	397.61, 396.98, ..., 139.21, 147.60	7640
QRF	143.78	143.78	406.23, 404.72, ..., 144.14, 142.91	1062
SVD	143.78	143.78	406.19, 404.69, ..., 144.14, 142.92	4631

Table 2: Results for Example 2

As we can see QRF and SVD are both able to find a better the minimum then the one reported in [8] whereas GJE falls far behind.

Example 3

This is problem 119 in the [7] collection. It has $n = 16$ variables and $m = 8$ linear constraints plus bound constraints.

$$f(\mathbf{x}) = \sum_{i,j=1}^{n} c_{ij}(x_i^2 + x_i + 1)(x_j^2 + x_j + 1) \tag{35}$$

subject to the following set of constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{A} \in \mathbb{R}^{m \times n}, \quad \mathbf{b} \in \mathbb{R}^{m} \tag{36}$$

$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{5} \tag{37}$$

where the c_{ij}, $\mathbf{A}$ and $\mathbf{b}$ are given in [7].

The results for 10 independent runs of a Genetic Algorithm with a population of 1000 individuals and a maximum of 5000 generations per run, are shown in 3.

	Fitness			
	Best	Average	$\mathbf{x}^*$	Time (s)
[7]	244.90	$--$	$(3.985)10^{-2}$, ..., $(6.743)10^{-1}$, 0.0	$--$
GJE	244.90	244.91	$(3.986)10^{-2}$, ..., $(6.743)10^{-1}$, $(5.230)10^{-6}$	290
QRF	245.28	255.51	$(6.166)10^{-2}$, ..., $(6.565)10^{-1}$, $(1.072)10^{-6}$	1754
SVD	256.21	267.79	$(1.180)10^{-1}$, ..., $(6.101)10^{-1}$, $(3.820)10^{-6}$	274

Table 3: Results for Example 3

Example 4

This example is intended to assess the behavior of the different methods in the presence of highly ill-conditioned linear constraints. The problem to solve is as follows:

$$f(\mathbf{x}) = \|\mathbf{x}\|^2 \tag{38}$$

subject to the following set of constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{A} \in \mathbb{R}^{m \times n}, \quad \mathbf{b} = \mathbf{A}\mathbf{e} \in \mathbb{R}^{m} \tag{39}$$

$$-\infty \leq \mathbf{x} \leq +\infty \tag{40}$$

where $\mathbf{A}$ is the segment comprising of m rows and n columns of the Hilbert matrix, $\mathbf{e} = (1 \ \ldots \ 1)^{\mathrm{T}}$ and $n = 100$, $m = 60$. This is a highly constrained problem and given the well known "almost" linear dependence of the rows and columns of the Hilbert matrix the set of constraints is also ill-conditioned.

To find the effective rank r_e of $\mathbf{A}$ we set a cutoff tolerance τ for the pivots in GJE, QRF and for the singular values in SVD, that is, all these quantities were set to zero when below this tolerance and dealt with accordingly.

The results for 10 independent runs of a Genetic Algorithm with a population of 100 individuals and a maximum of 5000 generations per run and $\tau = 10^{-10}$, are as follows:

The results summarized in tables 4 and 5 provide some evidence for QRF and SVD superiority over GJE to detect near linear dependency and provide more realistic results. Also, SVD is the best in terms of computer time as a result of employing a orthonormal base for the null space of the constraint matrix $\mathbf{A}$ thus enhancing the searchability of the genetic algorithm.

	Fitness			
	Best	Average	r_e	Time (s)
GJE	$1.1891\,10^5$	$6.1843\,10^6$	17	393
QRF	$1.0031\,10^2$	$1.0170\,10^2$	14	426
SVD	$1.0052\,10^2$	$1.0259\,10^2$	13	228

Table 4: Results for Example 4: $\tau = 10^{-10}$, the Time column is the total run time.

	Fitness			
	Best	Average	r_e	Time (s)
GJE	$8.4180\,10^2$	$6.2243\,10^3$	26	595
QRF	$1.0032\,10^2$	$1.0102\,10^2$	17	419
SVD	$1.0023\,10^2$	$1.0058\,10^2$	17	285

Table 5: Results for Example 4: $\tau = 10^{-14}$, the Time column is the total run time.

5 Conclusions

We presented three methods to deal with linear constraints: the Gauss-Jordan Elimination (GJE) method, the Rectangular Orthogonal Decomposition (QRF) and the Singular Value Decomposition (SVD), all allowing the identification of the dependent (free) variables and the dependent ones. As the free variables are the ones to be coded in the genome this procedure entails a reduction of the dimension from n of the search space to m entailing a substantial computational economy.

There however are some important differences to note. The SVD is more robust whenever the set of linear constraints is linearly dependent or "almost" linearly dependent a feature that can be useful if the linear constraint happen to be generated automatically. Since it provides an orthonormal base for the null space of $\mathbf{A}$ it has superior searchability as compared with QRF or GJE leading to better results but not necessarily to less computational effort as this seems to be very problem dependent. One disadvantage of all elimination methods (GJE, SVD and QRF) is that the bound constraints expressed in the original $\mathbf{x}$ variables do not translate into bound constraints in the $\mathbf{y}$ variables requiring the respective violation to be treated through a penalty term.

References

[1] *MATLAB*, The MathWorks Inc., 2011.

[2] C.A.C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art", *Comput. Methods Appl. Mech. Engrg.*, 121: 1245–1287, 2002.

[3] K. Deb, "An effcient constraint handling method for genetic algorithms", *Comput. Methods Appl. Mech. Engrg.*, 186: 311–338, 2000.

[4] A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[5] C.A. Floudas, P.M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer, 1990.

[6] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[7] W. Hock, K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Springer, 1981.

[8] D. Kendrick, "Caution and Probing in a Macroeconomic Model", *Journal of Economic Dynamics and Control*, 4(2), 1982.

[9] K. Maling, S.H. Mueller, W.R. Heller, "On Finding Most Optimal Rectangular Packages Plans", in *Proceedings of the 19th Design Automation Conference*, pages 663–670. IEEE, 1982.

[10] C.D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, 2000.

[11] Z. Michalewicz, "A Survey of Constraint Handling Techniques in Evolutionary Computation Methods", in J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Editors), *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155. MIT Press, 1995.

[12] G. Strang, *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 2003.