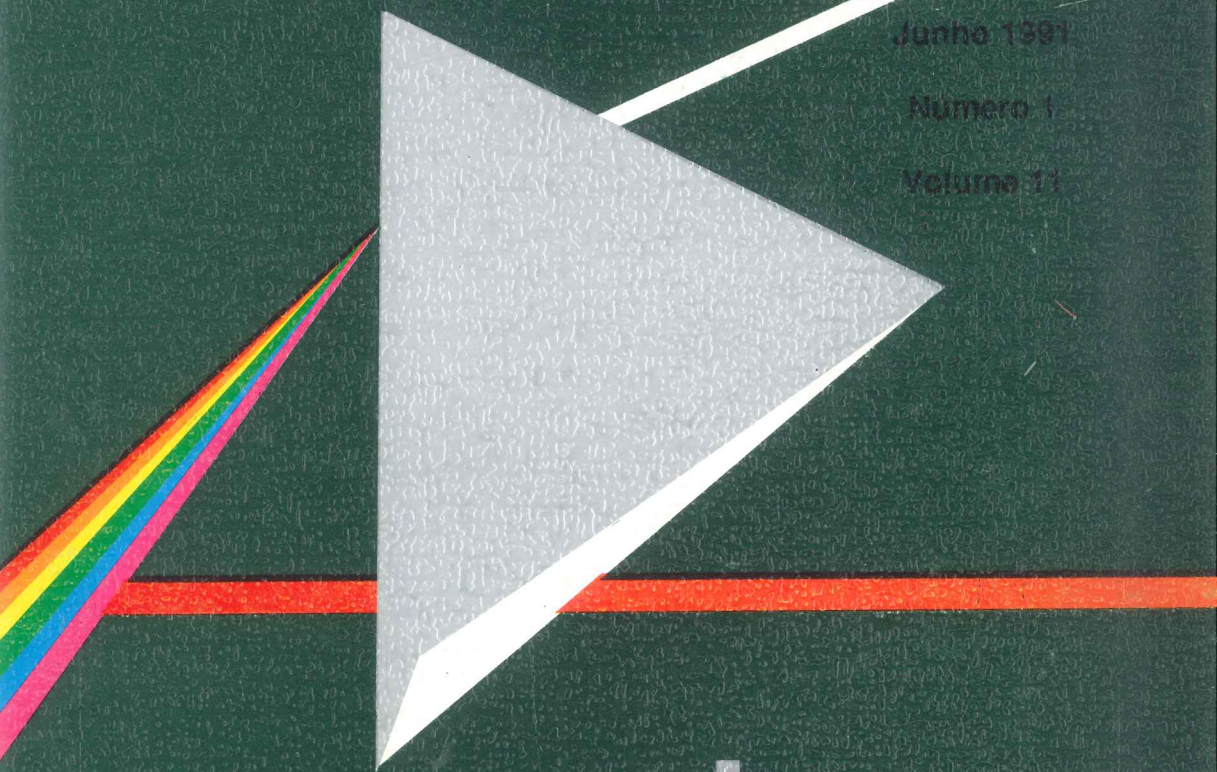
INVESTIGAÇÃO OPERACIONAL

Junho 1991

Número 1

Volume 11



Publicação Científica da

Associação Portuguesa para o Desenvolvimento da Investigação Operacional

INVESTIGAÇÃO OPERACIONAL

Propriedade:

APDIO — Associação Portuguesa para o Desenvolvimento da Investigação Operacional

ESTATUTO EDITORIAL

«Investigação Operacional», órgão oficial da APDIO cobre uma larga gama de assuntos reflectindo assim a grande diversidade de profissões e interesses dos sócios da Associação, bem como as muitas áreas de aplicação da I. O. O seu objectivo primordial é promover a aplicação do método e técnicas da I.O. aos problemas da Sociedade Portuguesa.

A publicação acolhe contribuições nos campos da metodologia, técnicas, e áreas de aplicação e software de I. O. sendo no entanto dada prioridade a bons casos de estudo de carácter eminentemente prático.

Distribuição gratuita aos sócios da APDIO

INVESTIGAÇÃO OPERACIONAL

Volume 11 - nº 1 - Junho 1991 Publicação semestral

A Revista "INVESTIGAÇÃO OPERACIONAL" está registada na Secretaria de Estado da Comunicação Social sob o nº 108335.
Esta Revista é distribuida gratuitamente aos sócios da APDIO. As informações sobre inscrições na Associação, assim como a correspondência para a Revista devem ser enviadas para a sede da APDIO – Associação Portuguesa para o Desenvolvimento da Investigação Operacional – CESUR, Instituto Superior Técnico, Av. Rovisco Pais, 1000 Lisboa.

Este Volume foi subsidiado por :

Instituto Nacional de Investigação Científica (INIC)

Junta Nacional de Investigação Científica e Tecnológica (JNICT)

Fundação Calouste Gulbenkian

Para efeitos de dactilografia e composição, foram utilizados equipamentos gentilmente postos à disposição pelo CEAUL (DEIO - Faculdade de Ciências de Lisboa).

Assinatura : 5000$00

INVESTIGAÇÃO OPERACIONAL, Vol.11, nº1 (Junho 1991) 3-30

COMPUTATIONAL TECHNIQUES FOR SOLVING GLOBAL OPTIMIZATION PROBLEMS

Panos M. Pardalos
Department of Industrial and Systems Engineering
303 Weil Hall, University of Florida
Gainesville, FL 32611

Abstract

Global optimization has been the subject of active research in the last two decades. The outcome of this research includes results on the complexity of global optimization problems, modeling of applications, and the development and implementation of numerical algorithms. We will discuss recent developments in global optimization with emphasis on computational algorithms for solving some structured global optimization problems.

Resumo

A Optimização Global tem sido nas últimas duas décadas uma área de investigação bastante activa. Nessa investigação há a distinguir resultados de complexidade computacional, modelos de aplicação e desenvolvimento e implementação de algoritmos. Neste trabalho são discutidos alguns desenvolvimentos recentes da optimização global, sendo dado especial enfase aos algoritmos para a resolução de alguns problemas de optimização global com estruturas especiais.

Keywords: Global optimization, Interior point methods, Concave-cost network flow problems, NP-hard.

1. Problem Formulation and Complexity

The general problem to be considered is to find the function value f* and an associated feasible point x* such that

$$f^* = f(x^*) = \text{global} \min_{x \in S} (\text{or} \max_{x \in S}) f(x)$$

where S is some convex compact set and f is a continuous function. Problems of this form have a wide range of applications in operations research, economics, engineering and computer science [10], [41].

If the objective function is convex, then every local minimum is global, and many classical local optimization techniques can be used to solve the above problem. Therefore convex minimization problems are considered as computationally "easy" problems. However, convexity of a function cannot be easily recognized. For example, there is no known algorithm to decide whether a fourth degree polynomial (in R^n) is a convex function or not. In addition, it is not clear that convexity is the key property that separates "difficult problems" from "easy problems". There are several classes of

nonconvex optimization problems that can be solved in polynomial time [37], [57]. Hence, in general we prefer to use the term "global optimization" than "nonconvex optimization".

The task of determining the global (or even a local) optimum of f(x) is, in fact, a NP-hard problem. In [44] it is shown that the following problem

$$\text{global} \quad \min_{x \in S} f(x) = x_1 - x_2^2$$

where S is a polytope in R^n is a NP-hard problem. Furthermore, checking if a feasible point is a local minimum of the quadratic programming problem

$$\min_{x \in P} f(x) = -x_1^2 - \ldots - x_n^2 + x_{n+1}^2$$

where P is a polytope in R^{n+1}, remains a NP-hard problem.

Most of the traditional optimization methods are designed to compute Kuhn-Tucker points. It can be shown, that the problem of deciding existence of a Kuhn-Tucker point is also NP-hard (even when the function is quadratic). For example, the problem of finding (or providing existence of) a Kuhn-Tucker point for the quadratic program

$$\min_{x \geq 0} f(x) = c^T x + \frac{1}{2} x^T Q x$$

where Q is an n×n symmetric matrix and $x \in R^n$, is equivalent to solving a corresponding symmetric Linear Complementarity Problem with data Q and c, which is NP-hard. Most of these complexity results characterize worst-case instances of the problems. Nevertheless, they are indicative of the difficulty of the general global optimization problem.

In the last three decades a variety of deterministic algorithms have been proposed. In order to solve for the global optimum of large scale problems in engineering and business applications, it is necessary to design algorithms which take advantage of the special structures usually present in such applications. Often this involves functions of a particular type and special types of constraints. Such types of problems include quadratic programming, bilinear and linear complementarity problems, polynomial optimization, separable programming, problems with Lipschitz functions, reverse convex programming, and problems with a dc (difference of convex) objective function. On the other hand we have problems with network constraints, sparse problems, and black-box type optimization problems where the objective function is not given in analytic form.

Constrained global optimization algorithms (both parallel and serial) can be categorized into two groups: stochastic and deterministic. The stochastic approaches tend to be faster but unfortunately do not, in general, provide bounds on the global optimum function value nor do they guarantee that the global optimum will be obtained. On the other hand, the deterministic approaches, although somewhat slower, usually provide bounds and guarantee that the global optimum will be determined. Regarding stochastic methods see [33], [5], [50], [52] and [54]. In this paper, we will consider only deterministic approaches for solving some classes of structured global optimization problems.

2. Separable Programming

The most extensively tested parallel deterministic approaches are those discussed in [40], [41], [46] and [47]. The general problem considered is to find

$$\Psi^* = \text{global} \min_{(x,y,z)\in S} \Psi(x, y, z) = \emptyset(x) + d^T y + s(z)$$

where S is nonempty bounded polyhedral set in R^{n+k+p}. The nonlinear term $\emptyset(x)$ is assumed to be separable and concave whereas the nonlinear term s(z) is assumed only to be convex (but not necessarily separable). A large number of functions can be expressed in this form including the very important classes of quadratic and synomial functions. Quadratic programming can be easily reduced to a separable problem using the spectral decomposition theorem [41], or an LU factorization of the corresponding symmetric matrix.

Many practical problems in engineering design can be formulated as global optimization problems. For example, the class of posynomial optimization problems, for which the theory of geometric programming was originally developed, can be transformed into partially separable problems. In an engineering design problem, the function to be minimized can often be expressed as the sum of component costs of the form

$$u_i = c_i t_1^{a_{i1}} t_2^{a_{i2}} \dots t_m^{a_{im}}$$

where $c_i > 0$ and a_{ij} are specified (possibly negative) real constants, and the design parameters t_i are assumed to be positive variables. Hence, the objective function can be expressed as the synomial function

$$g(t) = \sum_{i=1}^{n} c_i \, t_1^{a_{i1}} \, t_2^{a_{i2}} \ldots t_m^{a_{im}}.$$

The general constrained synomial optimization problem with p inequality constraints can then be expressed as

$$\min \; g(t) = \sum_{i=1}^{n} c_i \prod_{j=1}^{m} t_j^{a_{ij}}.$$

subject to

$$g_k(t) = \sum_{j=1}^{m} b_{kj} \, \ln(t_j) \leq b_k, \; k = 1,\ldots, p$$

where $t_j > 0$, $j = 1,\ldots, m$, is assumed, and the coefficients b_{kj} and right hand sides b_k are specified real constants. By defining

$$x_i = \sum_{j=1}^{m} a_{ij} \, \ln(t_j), \; i = 1,\ldots, n \quad \text{and}$$

$$z_i = \ln(t_i), \; i = 1,\ldots, m,$$

then the general synomial optimization problem can be transformed into a partially separable optimization problem. See [47] for more details on this transformation.

The general (parallel) algorithm for solving the partially separable global optimization problem is of the branch and bound type and is as follows:

(1) Compute a hyperrectangle which encloses the feasible region S by solving the multiple cost row linear program (in parallel)

$$\max_{(x,y,z) \in S} \; x_i$$

for $i = 1,\ldots, n$ $(x \in R^n)$.

(2) For each hyperrectangle currently available, do (in parallel):

(2.1) Use some heuristic method to compute a lower bound (LB) and an upper bound (UB) for Ψ^*, and use these bounds to possibly eliminate parts of the hyperrectangle from further consideration.

(2.2) If UB – LB $\leq \varepsilon$ (ε is a specified tolerance) then eliminate this hyperrectangle from any further consideration, and save UB (and its associated feasible point) as a possible global optimum function value.

(3) For each hyperrectangle remaining after step 2, pick some coordinate direction and bisect the hyperrectangle into two new sub-hyperrectangles and go back to step 2.

(4) If no hyperrectangles remain after step 2, then set Ψ^* to be the smallest of all the upper bounds (UB) obtained and x* to be the corresponding feasible point.

Step 2.1, computing the upper and lower bounds, is very important computationally since it is the heuristic step which will allow the method to be efficient. A method for underestimating the concave part $\emptyset(x)$ of the objective function over various subregions of the hyperrectangle is described in detail in [41] and [35]. This procedure reduces the problem to the simpler case of convex minimization. In addition, a procedure is developed which enable parts of the feasible region to be eliminated at each step.

Given the hyperrectangle defined by:

$$R = \{x : \beta_{i1} \le x_i \le \beta_{i2},\ i = 1,\ldots, n\}$$

a convex underestimator $\gamma(x) + d^T y + s(z)$ can easily be constructed to agree with the nonconvex function $\emptyset(x) + d^T y + s(z)$ at all vertices of R. That is, since $\emptyset(x)$ is separable, then $\gamma(x)$ will just be a linear function which interpolates $\emptyset(x)$ at each vertex of R. Furthermore, the solution of the convex program

$$\min_{(x,y,z)\in S} \gamma(x) + d^T y + s(z)$$

will provide not only a candidate solution (upper bound) UB but also (a lower bound) LB on the global optimum function value Ψ^*. If UB – LB $\le \varepsilon$ then an ε-approximate solution (i.e. the relative error in the objective function is bounded by the user specified tolerance $\varepsilon > 0$) over this hyperrectangle has been found and the procedure may be terminated (as stated in step 2.2).

If this tolerance has not been achieved, then the hyperrectangle may be divided into any number of smaller hyperrectangles. One such choice is to generate 2n sub-hyperrectangles by alternately bisecting each coordinate direction (see [47], [41]). That is, for

$$R = \{x : \beta_{i1} \le x_i \le \beta_{i2},\ i = 1,\ldots, n\}$$

and each coordinate direction e_j, the two sub-hyperrectangles

$$R_{j1} = \{x : \beta_{i1} \le x_i \le \beta_{i2},\ i = 1,\ldots, n\ ,\ x_j \le (\beta_{j1} + \beta_{j2})/2\}$$

and

$$R_{j2} = \{x : \beta_{i1} \le x_i \le \beta_{i2},\ i = 1,\ldots, n\ ,\ (\beta_{j1} + \beta_{j2})/2 \le x_j\}$$

are constructed. Again, the convex underestimators which agree with the original nonconvex function at every vertex of the smaller sub-hyperrectangles may be easily computed and minimized over the original feasible region. For each of these convex programs, another, possibly better, upper bound will be obtained, but the resulting optimal function value does not immediately provide a new lower bound. If γ_{i1} and γ_{i2} denote the optimal function values obtained from the 2n convex problems and Ψ_{i1} and Ψ_{i2} denote the corresponding candidate solutions then the new lower bound may be obtained from

$$LB = \max_{i=1,\ldots,n} \min\{\gamma_{i1}, \gamma_{i2}\}.$$

Additionally, an elimination procedure can be used such that if $\gamma_{i1} > \Psi_{i1}$ then the "first" half of the hyperrectangle R_{i1} may be eliminated from further consideration [47], [41]. Likewise, if $\gamma_{i2} > \Psi_{i2}$ then the "second" half of the hyperrectangle R_{i2} may be eliminated from further consideration. Naturally, it is possible that neither of the conditions may be met. If it is the case that no subregions are eliminated after the 2n convex optimizations, then the heuristic step must halt with the best available upper bound UB and the best available lower bound LB. If any of the subregions are eliminated, then the same procedure may be repeated on the resulting hyperrectangle.

It should be noted that even though this procedure is highly parallel (all 2n convex programs may be solved in parallel), the procedure is not guaranteed to result in an ε-approximate solution until the hyperrectangle is sufficiently small. But, according to the extensive computational experience cited in [46] on indefinite quadratic problems, the heuristic is extremely efficient in practice (see also [46]).

In order to guarantee that the algorithm will terminate at an ε-approximate solution, the direction in which to perform the bisection in step 4 must be selected in an appropriate way.

A large amount of computational testing has been performed for the case when the nonlinear terms are quadratic. Numerous results are cited in [46] for this case and it is observed that when the algorithm is implemented in parallel, super-linear speedups may be obtained in certain instances (in this case, the

serial algorithm used was just the parallel algorithm limited to one processor but with no communication costs). Similar results are sited in [47] for the more general case of synomial functions.

Parallel branch and bound algorithm have been used successfully for solving difficult combinatorial optimization problems such as 0-1 integer programming and the quadratic assignment problem. For different issues related to these approaches see the recent article by Pardalos and Li [38].

3. Large Scale Block Structured Problems

Optimization problems with a block-structure occur in many important large-scale applications and have been the subject of study for many years. Many of these larger problems can be broken into subproblems of lower dimensionality. In many cases these subproblems can be solved independently and the union (or some appropriate adjustment) of their solution is the solution of the original problem. For a specific decomposition algorithm to be computationally effective, the subproblems must have fewer variables and/or constraints than the original problem, and the algorithm should take advantage of the structural characteristics of the problem. The idea of decomposition is not new. In fact, ideas about distributed computing techniques were proposed by G. Dantzig and P. Wolfe as early as 1960 for the solution of large-scale problems using decomposition methods [4].

One of the most common practical types of global optimization involves problems of a very large scale with a special structure. Specifically, consider the block structured problem

$$\text{global min } w(y) + \sum_{i=1}^{N} f^{(i)}(x^{(i)}, y)$$

subject to the constraints

$$g_i^{(j)}(x^{(j)}, y) \geq 0,\ i \in I^{(j)},\ j = 1,\ldots, N$$

where $x^{(1)}, x^{(2)},\ldots, x^{(N)}$, and y denote vectors (possibly of different dimensions), and the index sets $I^{(1)}, I^{(2)},\ldots, I^{(N)}$ are disjoint. Clearly, y is the only coupling variable, the absence of which would provide N independent optimization problems available to be performed in parallel. In fact, if the optimal value of $\hat{y}$ were known, then by fixing y at that value, N independent subproblems are available to be solved in parallel.

The general idea then is to decompose the solution into two phases: a master problem involving the linking variable y only, and N independent subproblems involving the remaining variables. In particular, the master problem can be stated as

$$\min w(y) + \sum_{i=1}^{N} v_i(y)$$

subject to the constraints

$$y \in V^{(i)},\ i = 1,\ldots, N$$

where the function $v_i(y)$ is the i^{th} independent subproblem

$$v_i(y) = \min f^{(i)}(x^{(i)}, y)$$

subject to

$$g_j^{(i)}(x^{(i)}, y) \geq 0 \quad \text{for } j \in I^{(i)}$$

and $V^{(i)}$ denotes the definition area of $v_i(y)$. Clearly, this sort of decomposition scheme permits the solution of the N subproblems in parallel at every iteration. For more details, applications to engineering problems, and related references see [2], [9], [29], [30] and [40].

4. Interior Point Methods

During the last decade, a number of interior point approaches have been considered for solving linear and convex quadratic programming problems. Some of these approaches have been extended to solve nonconvex quadratic problems ([21], [27], [57]). In [21], computational aspects of an interior-point algorithm (see also [58]) for indefinite quadratic programming problems with box constraints are presented. The interior point (IPA) algorithm finds a stationary point that satisfies the Kuhn-Tucker conditions, by successively solving indefinite quadratic problems with an ellipsoid constraint. The proposed algorithm has been implemented on an IBM 3090 computer and tested on a variety of dense test problems, including problems with a known global optimum. The problem has the following special form:

$$\min f(x) = \frac{1}{2} x^T Q x + c^T x \qquad (4.1)$$

$$\text{s.t. } x \in S = \{x \in R^n \mid 0 \leq x \leq e\},$$

where e is a vector of all 1's. In the IPA algorithm, we solve an indefinite quadratic program (IQP) subject to an ellipsoid constraint at each step. It is

known that the IQP with an ellipsoid constraint can be solved in polynomial time (e.g., [58]). First, we describe the main algorithm, IPA, and then present the procedure, IQE, which is used to solve an indefinite quadratic program with an ellipsoid constraint.

Initially, we have a starting point x^0 that is interior to the feasible region. We consider an ellipsoid E_1 with center x^0 that is inscribed in the feasible region. Then, we solve the IQP problem with an ellipsoid E_1 constraint using the procedure IQE. Let x^1 be a solution of the problem. Again we consider an ellipsoid E_2 with center x^1 that is inscribed in the feasible region. By repeating this process, we compute a sequence of interior points $x^0, x^1, x^2, \ldots$ After sufficiently many steps, we obtain a stationary point of problem (4.1). The following summarizes the algorithm IPA.

Algorithm IPA

1. $k = 1$; $x^0 = 1/2e$; $D_1 = \text{diag}\,(\frac{1}{2}, \ldots, \frac{1}{2})$.
2. Consider an ellipsoid $E_k \subseteq [0, 1]^n$ with center x^{k-1} such that
   $E_k = \{x \mid \| D_k^{-1}(x - x^{k-1}) \| \le r < 1\}$.
3. Solve the following indefinite quadratic problems with an ellipsoid constraint using the procedure IQE (that will be discussed later).

$$\min\ f(x) = \frac{1}{2} x^T Q x + c^T x \qquad (4.2)$$

   s.t. $x \in E_k$.

   Let x^k be a global minimum of (4.2).
4. Checking stopping criterion:
   If x^k does not satisfy the stopping criterion, then compute D_{k+1}, where
   $D_{k+1} = \text{diag}(d_1, \ldots, d_n)$ and $d_i = \min\{x_i^k, 1 - x_i^k\}$, $i = 1, \ldots, n$.
   $k = k + 1$; goto step 2,
   Else x^k is an approximate stationary point of (4.1).

Now we describe details for solving problems of the form (4.2). Consider the following indefinite quadratic problem with an ellipsoid constraint:

$$\min f(x) = \frac{1}{2} x^T Q x + c^T x \qquad (4.3)$$

s.t. $\| D^{-1}(x - a) \| \le r$,

where Q is indefinite and $D = \text{diag}(d_i)$, $d_i > 0$, $i = 1,\ldots, n$. Problem (4.3) has a global optimum solution x^* iff (e.g., [34], [53])

$$(Q + \mu D^{-2})\Delta x = -(Qa + c), \tag{4.4}$$

$$\| D^{-1} \Delta x \| = r, \tag{4.5}$$

and $Q + \mu D^{-2}$ is positive semidefinite, (4.6)

where $\Delta x = x^* - a$. The equation (4.4) can be written as

$$(DQD + \mu I)D^{-1}\Delta x = -D(Qa + c). \tag{4.7}$$

Let

$$DQD = \bar{Q},\ D^{-1}\Delta x = \Delta\bar{x},\ \text{and} -D(Qa + c) = -\bar{c}.$$

The, equations (4.4)–(4.6) become

$$(\bar{Q} + \mu I)\Delta\bar{x} = -\bar{c}, \tag{4.8}$$

$$\| \Delta\bar{x} \| = r, \tag{4.9}$$

$$\mu \geq | \underline{\lambda}(\bar{Q}) |, \tag{4.10}$$

where $\underline{\lambda}(\bar{Q})$ is the minimum eigenvalue of $\bar{Q}$. If we assume that Q is nonsingular, since DQD is a congruent transformation and Q is an indefinite matrix, $\underline{\lambda}(\bar{Q})$ is negative. For any symmetric matrix $\bar{Q}$, we have an orthogonal matrix $U \in R^{n\times n}$ such that

$$U^T\bar{Q}U = \Lambda \text{ and } UU^T = I,$$

where $\Lambda = \text{diag}(\lambda_i)$, $i = 1,\ldots, n$. Note that the columns of U are eigenvectors of $\bar{Q}$. Since $\bar{Q} = U\Lambda U^T$, from (4.8),

$$U^T(U\Lambda U^T + \mu I)\Delta\bar{x} = -U^T\bar{c}. \tag{4.11}$$

Then, system (4.8)–(4.10) becomes

$$(\Lambda + \mu I)U^T\Delta\bar{x} = -U^T\bar{c}, \tag{4.12}$$

$$\| U^T\Delta\bar{x} \| = r, \tag{4.13}$$

$$\mu \geq | \underline{\lambda}(\Lambda) | = | \underline{\lambda}(\bar{Q}) |. \tag{4.14}$$

Let $\Delta\bar{\bar{x}} = U^T\Delta\bar{x}$ and $\bar{\bar{c}} = U^T\bar{c}$. Then we have

$$(\Lambda + \mu I)\Delta\bar{\bar{x}} = -\bar{\bar{c}}, \tag{4.15}$$

$$\| \Delta\bar{\bar{x}} \| = r, \tag{4.16}$$

$$\mu \geq | \underline{\lambda}(\Lambda) |. \tag{4.17}$$

If we solve the system (4.15) – (4.17) and get $\Delta\bar{\bar{x}}$, then from the relations $\Delta\bar{\bar{x}} = U^T\Delta\bar{x}$ and $\Delta\bar{x} = D^{-1}\Delta x$, we can obtain Δx that is a solution of (4.4) – (4.6). It is clear that the system (4.15) – (4.17) is simpler than the system (4.4) – (4.6) since we have only a diagonal matrix Λ in (4.15) – (4.17).

We will discuss a method to solve the system (4.15) – (4.17). For simplicity call $\lambda_1 = \underline{\lambda}(\bar{Q})$ the smallest (single) eigenvalue of $\bar{Q}$. There are two cases we have to consider:

1. $\bar{\bar{c}}_1 \neq 0$.

   We search for a μ that satisfies

   $$\sum_{i=1}^{n} \left(\frac{-\bar{\bar{c}}_i}{\mu + \lambda_i} \right)^2 = r^2$$

   by using a binary search for $|\lambda_1| + \frac{|\bar{\bar{c}}_i|}{r} < \mu \leq |\lambda_1| + \frac{\sqrt{n}\, \max_i |\bar{\bar{c}}_i|}{r}$

2. $\bar{\bar{c}}_1 = 0$.

   Set $\mu = |\lambda_1|$ and compute $s = \sum_{i=2}^{n} (\Delta\bar{\bar{x}}_i)^2$ where $\Delta\bar{\bar{x}}_1 = 0$ and for $i = 2, \ldots, m$

   $\Delta\bar{\bar{x}}_i = \frac{\bar{\bar{c}}_i}{|\lambda_1| + \lambda_i}$, i.e., $\Delta\bar{\bar{x}}$ is the minimum-norm solution of (4.15).

   If $s \leq r^2$ then

   $(\Delta\bar{\bar{x}}_1)^2 = r^2 - s$.

   do **Choose_a_sign_of_**$\Delta\bar{\bar{x}}_1$.

   else $(s > r^2)$

   find a μ that satisfies $s = \sum_{i=2}^{n} \left(\frac{\bar{\bar{c}}_i}{\mu + \lambda_i} \right)^2 < r^2$ using a binary search for $|\lambda_1| < \mu \leq |\lambda_1| + \frac{\sqrt{n}\, \max_i |\bar{\bar{c}}_i|}{r}$

   $(\Delta\bar{\bar{x}}_1)^2 = r^2 - s$.

   do **Choose_a_sign_of_**$\Delta\bar{\bar{x}}_1$.

   endif.

In the above approach, r is a constant between 0 and 1. In practice, r can be flexibly chosen as long as $x^k + \Delta x$ is an interior feasible point. This will result in a larger step to obtain x^{k+1}. In theory, the sign of $\Delta\bar{\bar{x}}_1$ should not make any difference for $f(x^{k+1})$. However, in the subprocedure **Choose_a_sign_of_**$\Delta\bar{\bar{x}}_1$, we choose the sign of $\Delta\bar{\bar{x}}_1$ to be that which makes $f(x^{k+1})$ smaller in practice.

From $\Delta\bar{\bar{x}}$, we can obtain Δx. Then $a + \Delta x = x^*$ is a global minimum of (4.3) and

$$f(a) - f(x^*) = \frac{1}{2}\mu r^2 + \frac{1}{2}\Delta x^T (Q + \mu D^{-2})\Delta x > 0.$$

Next, we summarize the procedure IQE at step k.

Procedure IQE

1. $\bar{c} = D_k(Qx^{k-1} + c)$;
2. λ_i, i = 1,..., n are the eigenvalues of $\bar{Q} = D_k Q D_k$ and let λ_1 be the minimum eigenvalue and U is a matrix whose columns are eigenvectors of $\bar{Q} = D_k Q D_k$;
3. $\bar{\bar{c}} = U^T \bar{c}$;
4. $\mu_1 = |\lambda_1|$,
   $\mu_3 = |\lambda_1| + \frac{\sqrt{n}\max_i|\bar{\bar{c}}_i|}{r}$;
5. Solve the system (4.15) – (4.17).
   If $\bar{\bar{c}}_1 \neq 0$ then do binary search for $\mu_1 + \frac{|\bar{\bar{c}}_1|}{r} < \mu \leq \mu_3$;
   else fix $\mu = |\lambda_1|$ and let $\Delta\bar{\bar{x}}$ be the minimum-norm solution of (4.15);
   if $\sum_{i=2}^{n} \Delta\bar{\bar{x}}_i^2 > r^2$ then
   do binary search for $\mu_1 < \mu \leq \mu_3$;
   endif.
   $\Delta\bar{\bar{x}}_1^2 = r^2 - \sum_{i=2}^{n} \Delta\bar{\bar{x}}_i^2$;
   do **Choose_a_sign_of_**$\Delta\bar{\bar{x}}_1$;
   endif.
6. $\Delta x = D_k \Delta\bar{x} = D_k U^{-T} \Delta\bar{\bar{x}} = D_k U \Delta\bar{\bar{x}}$.
7. $x^{k+1} = x^k + \Delta x$.

We can verify that the above procedure is a worst-case polynomial-time algorithm. In practice, one can use a very efficient algorithm described in [34].

We implemented the algorithm IPA on an IBM 3090-600S computer with vector facilities using the VS Fortran compiler and double precision accuracy for real variables. The ESSL subroutines were used for matrix and vector operations, random number generation, and the eigensystem solving. $\| x^k - x^{k-1} \|_2 <$ 10E-4 was used for stopping criterion of the main algorithm IPA. We used binary search for the parameter μ in IQE and the binary search

was terminated when "upper_bound-lower_bound" < 10E-8 was satisfied. From the preliminary computational results, we found that when the parameter r is close to 1, the number of iterations needed to find a stationary point becomes small. Therefore, we set r = 0.99 in our implementation. All averages were obtained from 5 problems and CPU times are given in seconds.

We have two parameters for generating the test problems: the number of active constraints (nac) at the optimum solution and the number of negative eigenvalues (neg) of matrix Q. Note that neg ≤ nac. We solved two types of dense test problems: test problems (see [21]) with a known solution (type I) and random problems (type II). For generating type I test problems we have the following:

1. The optimum solution x^* is chosen randomly such that $x_i^* \in (0, 1)$, for i = 1,..., n. Half of the m active constraints are active at the upper bound.
2. The orthogonal matrices, U_1 and U_2, are Householder matrices of random vectors $v \in R^{m \times m}$ and $w \in R^{(n-m)\times(n-m)}$, respectively. Each element of v and w is chosen randomly from (0, 1).
3. The eigenvalues λ_i, i = 1,..., n, of Λ are chosen randomly so that $1 < |\lambda_i| < 2$, i = 1,..., n.
4. The submatrix $A \in R^{m\times(n-m)}$ of W has elements a_{ij}, i = 1,..., m, j = 1,..., n–m, chosen randomly from (0, 1).
5. Choose the diagonal matrix H such that F + H is diagonally dominant.

Table 1 shows the computational results obtained by solving test problems with a known optimal solution. From Table 1, it is observed that IPA takes more steps to converge to a stationary point as the number of active constraints increases. We investigated the relationship between the number of negative eigenvalues of matrix Q and the complexity of the test problems with a known solution. Table 2 shows that the number of iterations is not dependent on the parameter neg. It is interesting to note that the algorithm always found a global minimum for these type of test problems (although many local minima exist).

Next, we solved randomly generated indefinite quadratic problems (type II). The matrix Q was generated by $Q = U^T \Lambda U$, where U is an orthogonal

matrix (Householder matrix of a random vector v, where $0 < v_i < 1$, i = 1,..., n) and L is a diagonal matrix whose diagonal elements are chosen randomly from

$-2 < \lambda_i < -1$, i = 1,..., neg,

$1 < \lambda_i < 2$, i = neg + 1,..., n.

nac	Avg.Itr.	Avg.CPU time
10	40.8	10.4
30	68.8	15.2
50	96.4	19.1
70	100.4	20.7
90	118.8	28.9

Table 1: IPA on varying nac (n = 100, neg = nac)

nac	Avg.Itr.	Avg.CPU time
10	94.6	19.1
20	95.0	19.1
30	105.8	21.1
40	97.4	19.4
50	94.4	19.1
60	104.4	20.9

Table 2: IPA on varying neg (n = 100, nac = 60)

The vector c, $0 < c_i < 1$, i = 1,..., n is generated randomly. Table 3 shows the computational results for type II test problems. Note that all test problems we solved are dense.

nac	Avg.Itr.	Avg.CPU time
10	88.6	19.3
30	94.4	20.7
50	97.2	22.1
70	91.4	22.4
90	86.8	22.0

Table 3: IPA for randomly generated problems (n = 100, neg = nac)

The preliminary computational results (with dense problems) show that the difficulty of the problem is not dependent on the number of negative eigenvalues of Q. However, our algorithm needs more steps to converge to a stationary point as the number of active constraints increases. Although the algorithm is not guarenteed to find a global solution of the problem, it was observed that the algorithm found the global minimum in many cases. Further research is needed for applying the algorithm to solve large sparse problems. The main difficulty for solving large sparse problems is the availability of software for computing the eigenvalues and eigenvectors of large sparse matrices.

5. Minimum Concave-Cost Network Flow Problems

In this section we are going to discuss a special class of concave minimization problems with network constraints. The single-source uncapacitated (SSU) version of the minimum concave-cost network flow problem (MCNFP) requires establishing a minimum cost flow from a single generating source to a set of sinks, through a directed network. All arcs are uncapacitated, indicating that the entire source flow can pass through any arc. The SSU MCNFP can be stated formally as follows:

Given a directed graph $G = (N_G, A_G)$ consisting of a set N_G of n nodes and a set A_G of m ordered pairs of distinct nodes called arcs, coupled with a n-vector (demand vector) $d = (d_i)$ with $d_1 < 0$ and $d_i \geq 0$, $i = 2,\ldots, n$, and a concave cost function for each arc, $c_{ij}(x_{ij})$, then solve

$$\text{global min} \quad \sum_{(i,j)\in A_G} c_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A_G} x_{ki} - \sum_{(i,k)\in A_G} x_{ik} = d_i, \ \forall i \in N_G \tag{5.1}$$

and

$$0 \leq x_{ij}, \ \forall (i, j)\in A_G \tag{5.2}$$

All constraints and demands are assumed to be integral. The requirement that only $d_1 < 0$ corresponds to the single-source case. The lack of an upper bound for the x_{ij} gives rise to the uncapacitated case.

The SSU MCNFP is a concave optimization problem over a polyhedron. Hence, if a finite optimal solution exists, then there exists an extreme point of the feasible domain that is optimal. For the SSU case an extreme flow (corresponding to an extreme point) is a tree. The leaves of the solution tree correspond to a subset of the sink nodes. The integral constraints and demands give rise to extreme flows of integral value.

A SSU MCNFP has a finite optimal solution if it contains no negative cost cycles, and all sinks are reachable from the source (i.e., there exists a directed path from the source to each sink). The latter requirement is necessary for the existence of a feasible flow. The presence of a negative cost cycle would imply an unbounded negative cost solution; the absence of such a cycle guarantees a finite solution [31].

Efficient algorithms for the SSU MCNFP have been found only for a small set of structured problems. This is not surprising as the general global search problem for the SSU MCNFP is known to be NP-hard [13], [14], [31]. An overview of existing techniques for general MCNFP can be found in [15]. Due to the complexity of global search, numerous techniques based on local search have been developed.

Here, we consider cases of the SSU MCNFP with arc flow costs that are non-negative, non-decreasing and concave. This property of objective functions accurately reflects cost functions for models of real world problems in areas such as production planning and transportation analysis. For example, in a production setting, decreasing concave arc cost functions would exclude the influence of demand on production.

A solution X to a SSU MCNFP is locally optimal if no better solution exists in a specified neighborhood of X. Varying the definition of neighborhood results in different conditions for a local optimum. The standard marginal definition of local optimality defines a neighborhood of X to be

$$N_\varepsilon(X) = \{X' \mid X' \text{ satisfies (1) and (2) and } \| X - X' \| < \varepsilon\}$$

for a specified vector norm and $\varepsilon > 0$. Local search based on N_ε for concave optimization is explored by Minoux [32] and Yaged [56]. For the single commodity case with fixed-charge arc costs, all extreme points are local

optima. This led to the development of the following generalized definition of neighborhood by Gallo and Sodini [12]:

$$N_{AEF}(X) = \{X' \mid X' \text{ satisfies (1) and (2) and } X' \text{ is an adjacent extreme flow to } X\}$$

Here, X' is an adjacent extreme flow to an extreme solution X if X' is an extreme flow, and $X \cup X'$ contains a single undirected cycle.

An even more relaxed definition of neighborhood is the following:

$$N_{AF}(X) = \{X' \mid X' \text{ satisfies (1) and (2) and } X' \text{ is an adjacent flow to } X\}$$

Here, X' is an adjacent flow to an extreme solution X if X' results from rerouting a single subpath of flow within X. This concept of neighborhood was developed by Guisewite and Pardalos [14] for single-source problems, and independently by Plasil and Chlebnican [48] for the multicommodity case. We emply both N_{AEF} and N_{AF} neighborhoods in our global heuristic. Extensive computational results (applied to random initial solutions and greedy initial solutions) in [14] indicate that the choice of neighborhood for local search affects the processing time for convergence. The rate of convergence for random initial solutions is significantly slower than for the greedy initial solutions. This is due to an increased number of iterations required for convergence. However, the objective value of the computed solutions generated from local search with random starting solutions is roughly the same (on the average). For details see [14].

Next, an exact global search algorithm is presented for SSU MCNFP. The full algorithm is based on branch-and-bound enumeration of extreme feasible solutions. The basic enumeration subset of the algorithm exploits the fact that extreme feasible solutions correspond to sub-trees in the network. These sub-trees are constructed by establishing a path from each sink to the source node. Bounding the search is achieved using linear underestimation after a path is constructed. This allows the underestimating function to gain efficiency as the search progresses.

Numerous exact algorithms have been proposed for the general MCNFP [15]. Extreme point ranking methods [35] rely on linear underestimation applied to the initial problem. Results in this section demonstrate that the initial underestimation can be quite poor for this class of problems. Some branch-and-bound algorithms for MCNFP perform branching on the arc cost

functions to improve the linear underestimation [3], [28], [51]. For these approaches, worst case processing time can exceed a brute-force enumerative approach. Other branch-and-bound approaches, such as in [8], suffer severe performance penalties when degenerate problems are encountered [15]. Dynamic programming solutions to MCNFP require excessive storage requirements [6]. The proposed algorithm can incorporate dynamic programming to efficiently solve subproblems, while limiting the storage requirements.

The basis of the exact global search algorithm is a procedure that systematically enumerates the extreme feasible solutions of the current SSU MCNFP. As noted earlier, these correspond to sub-trees of $G = (N_G, A_G)$ that establish a path from the source to each sink. A path is constructed from each sink to the source node over reversed arcs. As in the random search algorithm, non-extreme solutions (cycles) are avoided by labelling nodes as follows:

$$\text{visited}(i) = \begin{cases} 0 & \text{if node i is unused} \\ 1 & \text{if node i is in the current sink path} \\ 2 & \text{if node i is in a previous path} \end{cases} \quad (5.3)$$

This labelling defines a partition of the nodes of G

$$N_G = N_G^0 \cup N_G^1 \cup N_G^2 \quad (5.4)$$

where $n_j \in N_G^i$ implies $\text{visited}(n_j) = i$.

The current state of the enumeration process is summarized by a collection of stacks, one for each completed path from a sink to the source, and one for the current active path. Each entry within a stack contains a node in the path defined by the stack. For each node, additional information is maintained, including the visit value, a pointer to the next element in the path, and a pointer to the current in-coming arc under consideration for the node. Processing within a stack is similar to any branch-and-bound procedure. However, additional processing is required to handle cases where a stack becomes empty, or a path is completed.

If a stack becomes empty during the enumerative search, augmenting paths to the previous stack have been fathomed. The current stack is popped, and the previous stack becomes active again at its top node. This implies that

the entries in the previous stack have their visit value reset to 1. When a path is completed, all entries within the current stack have their visit values set to 2. When the last stack is popped, the search is complete.

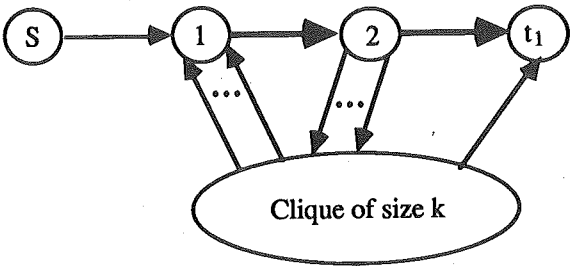


Figure 1: Worst case dead end for enumerative search

The algorithm presented above does not have processing time proportional to the number of extreme feasible solutions for the current problem. A worst case example of the effects of a dead-end is presented in Figure 1. The current search node is 1, and the current path is depicted by the thicker arcs. For this case, the dead-end contains a clique of size k. The enumerative search process would enumerate all paths through the clique before establishing that no feasible path exists to the source or to any node in a previous path.

Fortunately, this can be avoided with some additional processing. When a new node is added to the current search path, all nodes reachable on the network $G' = (N_G^0 \cup N_G^2, A_G')$ are computed, where A_G' is the set of arcs in A_G connecting nodes in $N_G^0 \cup N_G^2$. The reachable nodes can be identified by solving a single-source shortest weighted path problem. For the example, the entire clique would be removed from consideration when node 2 is added to the current search path.

The global search algorithm can be extended to exploit the cost functions assigned to the arcs. For the case with non-decreasing cost functions, the cost incurred by an arc at an intermediate point in the algorithm cannot decrease with the addition of other paths. This can be exploited by maintaining a running cost of the current search at each position of the stack. This corresponds to the cost of the current flow up to the active node being processed. If the current flow cost exceeds the current best solution, then the search can be terminated for the current path, and the next path in the search can be considered.

The arc cost functions can be exploited further by using linear underestimation to project a lower bound on the cost of extending the current path. This is achieved by

- Computing the maximum possible flow through each node i, based on the sinks reachable from node i, node.max(i).
- Maintaining the current flow through each node i, node.current(i).
- Maintaining the current flow through each arc (i,j), arc.current(i,j).

Initially, the maximum flow possible for any arc (i, j) is node.max[j]. The minimum flow through any arc, initially, is 0. During the search process the bounds on arc (i, j) can be tightened:

$$\text{upper.bound(i,j)} = \text{node.max(j)} - (\text{node.current(j)} - \text{arc.current(i,j)})$$
$$\text{lower.bound(i, j)} = \text{arc.current(i, j)} \qquad (5.5)$$

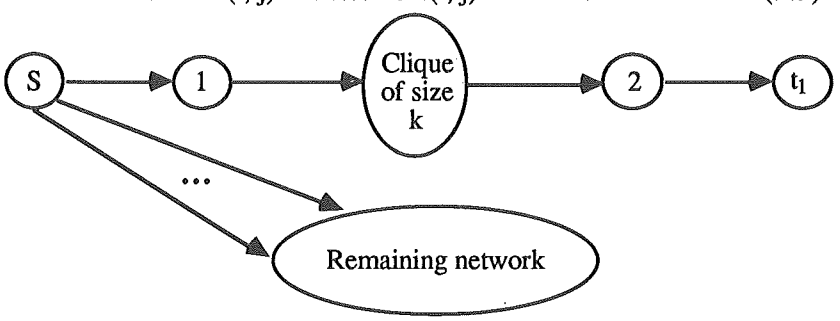


Figure 2: Simplex subproblem example

The linear underestimation process uses the refined bounds and the currently unsatisfied sink nodes to project a lower bound on the final cost of the current search. This is achieved by computing new arc costs corresponding to the linear underestimating cost:

$$\text{linear.cost(i,j)} = \frac{c_{ij}(\text{upper.bound(i,j)}) - c_{ij}(\text{lower.bound(i,j)})}{\text{upper.bound(i,j)} - \text{lower.bound(i,j)}} \qquad (5.6)$$

Then the cost of an augmenting flow on arc (i, j) incurs cost

$$c'_{ij}(x_{ij}) = \text{linear.cost(i, j)} * x_{ij} \qquad (5.7)$$

A single-source shortest weighted path problem can be solved to obtain the shortest path from the source to each currently unsatisfied sink, using the linear.cost vector as the arc weights.

The enumeration process can be further improved by detecting subproblems that can be more efficiently solved using other techniques. For

example, the network in Figure 2 has a subproblem to sink t_i that can be solved using a shortest weighted path algorithm. More general cases that are computationally expensive for enumeration, but more efficiently solvable by dynamic programming include sub-networks that have small total in and out-degree, and contain a large number of nodes.

Some simple subproblem cases can be detected by computing spanning trees for small perturbations of network G. For example, if subproblems with in-degree and out-degree of one are sought, then O(n) spanning trees need to be computed. For each node $n_i \in N_G$, two problems are solved. The first identifies those nodes no longer reachable from the source when n_i is removed from network G, denoted by $SCUT_i$. The second identifies nodes that are disconnected from all sinks when n_i is removed, denoted by $TCUT_i$. The intersection of these sets for each node pair identifies a candidate subproblem, $SUB_{ij} = SCUT_i \cap TCUT_j$. The process can be applied once, and the maximal subproblems can be removed iteratively.

In general, the problem of identifying the best subproblem can be NP-complete. For example, if we require that a subproblem have a specified lower bound on the number of nodes, and search for the node set with minimal total degree, then the problem corresponds to the Graph Partitioning problem [13]. However, heuristics, such as the Kernighan-Lin algorithm, have been successful identifying subgraphs with the desired property, or near to the desired optimum.

The enumeration process can avoid re-solving the detected subproblems by collapsing each subproblem into a single super-node. When a super-node is encountered during the search process, the sub-problem can be solved if it is new, or, its solution can be extracted using a look-up table if it was previously solved.

The overall process of exploiting subproblems can be viewed in relation to the send-and-split algorithm [6]. In send-and-split, the optimal flow for the entire network is solved using dynamic programming. This requires substantial storage, but recomputes all subproblems. The full enumeration algorithm requires little storage, but recomputes all subproblems. By exploiting a select subset of the subproblems, the resulting algorithm gains

efficiency, while reducing the storage requirements based on available resources.

Next we are going to discuss preliminary computational experience. Test networks are generated in a random fashion. Arcs are generated by computing two random integers uniformly distributed in [1, 2,..., n], where n is the number of nodes in the network. Duplicate arcs and arcs of the form (i, i) are discarded. After the specified number of arcs is successfully generated, the resulting network is tested for connectivity by solving a single source shortest path problem. If the connectivity is suitably high, then cost functions are generated for each arc. Each cost function is of the form $\alpha_{ij}x^{i\beta}$, where α_{ij} are uniformly distributed in [1, 2,..., 100] and the β_{ij} are uniformly distributed in [.1, .2,..., 1]. The test generator is implemented so that if two problems contain the same number of nodes, and the same random number seed, but have a different number of arcs then the smaller network generated is a subset of the larger network.

The algorithms were implemented on one or multiple microprocessors. Our processing system consists of one to twenty Transputer T800s. The Transputer is a microprocessor developed under the European Espirit project, and is designed to facilitate parallel processing. Each 20 MHz T800 consists of a 10 MIPS fixed point processor, a 1.5 MFLOPS floating-point co-processor, a 4 KByte cache, and 4 DMA I/O processors, all on a single chip. Our system includes 1 MByte of memory per processor. Experience indicates that 3 to 5 MIPS are achievable by each processor for large general processing applications. Four processors are configured in a pipeline on a single board, with the remaining DMA links connected to the board edge through a cross-bar switch.

Variations of the enumerative search process were applied to randomly generated problems. The problems processed were of moderate density. Analysis of the problems indicated that dead-ends did not have a serious impact on the search. The networks were, also, preprocessed to identify subproblems with in-degree and out-degree equal to one. No significant subproblems of this type were detected. The algorithms used included

- Full enumeration (with no cost bounding)
- Cost bounding

- Cost bounding with linear underestimation
- Initial solutions provided.

In Tables 4 and 5, results are provided for four problem sets, each generated with a different random number seed. The effect of varying the number of sinks while maintaining the network strcuture is investigated. The results indicate that problems of moderate network size and density are solvable using the algorithm with cost bounding. Linear underestimation becomes efficient only for a few difficult problems. This is due to linear underestimation requiring a substantial amount of additional processing at each path completion. This could be avoided by selectively applying linear underestimation based on the current path cost, the previous path cost, and the amount of change between the current path and previous path.

In Table 6, results indicate that the quality of the initial solution has a direct relation to the performance of the search, when cost bounding is employed. These results, also, provide a comparison of the performance of initial solutions computed using linear underestimation, random search, and greedy algorithms coupled with local search. The initial solutions obtained using random search were the best for all test cases. In fact, random search detected the global optimum for all test cases in Tables 4 to 5. The initial solutions computed using local search were, on the average, better than those obtained using linear underestimation. Table 7 compares the number of solutions that were fully processed for enumerative search with cost bounding, and enumerative search with linear underestimation.

The exact global search algorithm, was demonstrated to be useful for problems of moderate size and density. This approach has the benefit of solving "easy" problems quickly, where an easy problem has few extreme feasible solutions, or has few solutions with cost near to the cost of the global optimum. The algorithm gains efficiency by bounding the search based on cost properties, and projected cost based on linear underestimation. Initial solutions obtained using random search and local search provide a good initial approximation to the global optimum in most cases.

TEST SET 1			
Nodes/Arcs/Sinks	Full Enumeration	Cost Bounding	Linear Underestimation
10/40/5	4.64	.13	.36
15/60/5	753.08	.22	.52
15/60/10	7766.13	8.97	18.55
20/80/5	31820.84	1.15	2.50
20/80/10	X	25.84	41.77
20/80/15	X	624.96	1202.03
25/100/5	X	.91	1.27
25/100/10	X	6.63	12.67
25/100/15	X	5224.91	12808.24
25/100/20	X	17908.40	44663.96
30/120/5	X	1.42	3.48
30/120/10	X	23.67	60.88
30/120/15	X	351.29	800.04

Table 4: Global search results – Test set 1

TEST SET 2			
Nodes/Arcs/Sinks	Full Enumeration	Cost Bounding	Linear Underestimation
10/40/5	7.57	.37	.53
15/60/5	311.89	.42	.58
15/60/10	3869.25	.82	1.10
20/80/5	3525.02	12.25	13.69
20/80/10	X	21.36	26.96
20/80/15	X	32.70	47.60
25/100/5	X	.24	.54
25/100/10	X	32.38	61.64
25/100/15	X	138.83	247.56
25/100/20	X	717.58	1254.99
30/120/5	X	1.61	3.58
30/120/10	X	63.43	154.51
30/120/15	X	2942.59	7320.70

Table 5: Global search results – Test set 2

Test Set (25/100/15)	Technique	Initial Solution	Time (seconds)
1	GREEDY	1589.76	5224.91
	LINEAR	1745.48	6351.81
	RANDOM	1506.38	3583.58
2	GREEDY	1007.78	138.83
	LINEAR	1133.64	152.96
	RANDOM	943.46	115.46
3	GREEDY	764.12	8.10
	LINEAR	1111.01	13.40
	RANDOM	764.12	8.10
4	GREEDY	1600.26	3300.59
	LINEAR	1869.02	3783.58
	RANDOM	1558.54	2949.78

Table 6: Global search results – Varying initial solutions

Test Set	Nodes/Arcs/Sinks	Cost Bounding	Linear Underestimation
1	20/80/15	971	598
4	20/80/15	59222	519
1	25/100/15	17512	8927
4	25/100/15	928	162
1	30/120/15	72	27
4	30/120/15	X	11392

Table 7: Global search results – Number of solutions examined

6. Concluding Remarks

In this paper we gave a brief overview of some of the computational approaches that are used to solve global optimization problems. We discussed only a small class of deterministic algorithms. Other promising deterministic approaches that have been used to solve global optimization problems include interval analysis methods. For a recent reference on this subject, see for example [49]. In addition, many approaches, such as genetic algorithms, space filling curve methods, tuneling methods etc have been proposed and used for solving different types of problems [10], [11], [25]. Many other approaches are listed in the references below. Most of the computational results presented in the literature are not directly comparable because different

authors have used different test problems. Recently, Floudas and Pardalos [10] have published a book on test problems for constrained global optimization. It is hoped that the problems in that book could provide the means to compare one algorithm against another.

References

[1] Al-Khayyal, F.A., Horst, R. and Pardalos, P.M., Global Optimization of Concave Functions Subject to Separable Quadratic Constraints: An Application to Bilevel Programming, To appear in Annals of Operations Research (1991).

[2] Aggarwal, A. and Floudas, C.A., A Decomposition Approach for Global Optimum Search in QP, NLP and MINLP Problems, Annals of Operations Research 25 (1990) 119-146.

[3] Bornstein, C.T. and Rust, R., Minimizing a Sum of Staircase Functions Under Linear Constraints, Optimization 19 (1988) 181-190.

[4] Dantzig, G.B. and Wolfe, P., The Decomposition Principle for Linear Programs, Operations Research 8 (1960) 101-111.

[5] Dixon, L.C.W. and Szego, G.P. (Eds), Towards Global Optimization, Vol. 1 (1975) and Vol. 2 (1978), North-Holland, Amsterdam.

[6] Erickson, R.E., Monma, C.L. and Veinott Jr., A.F., Send-and-split Method for Minimum-Concave-Cost Network Flows, Mathematics of Operations Research 12 (1987) 634-664.

[7] Evtushenko, Y.G., Numerical Optimization Techniques, Optimization Software Inc. (1985).

[8] Florian, M. and Robillard, P., An Implicit Enumeration Algorithm for the Concave Set Network Flow Problem, Management Science 18 (1971) 184-193.

[9] Floudas, C.A. and Aggarwal, A., A Decomposition Strategy for Global Optimum Search in the Pooling Problem, ORSA Journal on Computing 2 (1990) 225-235.

[10] Floudas, C.A. and Pardalos, P.M., A Collection of Test Problems for Constrained Global Optimization Algorithms, Lecture Notes in Computer Science 445, Springer-Verlag (1990).

[11] Floudas, C.A. and Pardalos, P.M. (Editors), Recent Adavances in Global Optimization, Princeton University Press (1991).

[12] Gallo, G. and Sodini, C., Adjacent Extreme Flows and Application to Min Concave-Cost Flow Problems, Networks 9 (1979) 95-121.

[13] Garey, M.R. and Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H.Freeman and Company, San Francisco, CA (1979).

[14] Guisewite, G.M. and Pardalos, P.M., Algorithms for the Uncapacitated Single-Source Minimum Concave-Cost Network Flow Problem, Operational Research '90 (Edited by H.E.Bradley), Pergamon Press (1990) 703-713.

[15] Guisewite, G.M. and Pardalos, P.M., Minimum Concave-Cost Network Flow Problems: Applications, Complexity and Algorithms, Annals of Operations Research 25 (1990) 75-100.

[16] Guisewite, G.M. and Pardalos, P.M., Performance of Local Search in Minimum Concave-Cost Network Flow Problems, In "Recent Advances in Global Optimization" (C.A.Floudas and P.M.Pardalos, Eds), Princeton University Press (1991).

[17] Guisewite, G.M. and Pardalos, P.M., Global Search Algorithms for Minimum Concave-Cost Network Flow Problems, To appear in the Journal of Global Optimization (1991).

[18] Gupta, S. and Pardalos, P.M., A Note on a Quadratic Formulation for Linear Complementarity Problems, JOTA 57 (1988) 197-202.

[19] Hager, W.W., Pardalos, P.M., Roussos, I.M. and Sahinoglou, H.D., Active Constraints, Indefinite Quadratic Test Problems and Complexity, JOTA 68 (1991) 499-511.
[20] Han, C.G., Pardalos, P.M. and Ye, Y., Computational Aspects of an Interior Point Algorithm for Quadratic Programming Problems with Box Constraints, Large-Scale Numerical Optimization (Eds: T.F.Coleman and Y.Li), SIAM (1990) 92-112
[21] Han, C.G., Pardalos, P.M. and Ye, Y., On the Solution of Indefinite Quadratic Problems Using an Interior Point Algorithm, To appear in SIAM J.Scient.Stat. Computing (1991).
[22] Hansen, P. and Jaumard, B., Reduction of Indefinite Quadratic Programs to Bilinear Programs, To appear in the Journal of Global Optimization (1991).
[23] Horst, R., Phong, T.Q. and Thoai, N.V., On Solving General Reverse Convex Programming Problems by a Sequence of Linear Programs and Line Searches, Annals of Operations Research 25 (1990) 1-18.
[24] Horst, R., Thoai, N.V. and Benson, H.P., Concave Minimization via Conical Partitions and Polyhedral Outer Approximation, Mathematical Programming 50 (1991) 259-274.
[25] Horst, R. and Tuy, H., Constrained Global Optimization, Springer-Verlag (1990).
[26] Júdice, J.J. and Mitra, G., Reformulation of Mathematical Programming Problems as Linear Complementarity Problems and Investigation of their Solution Method, JOTA 57 (1988) 123-149.
[27] Karmarkar, N., An Interior-Point Approach to NP-Complete Problems - Part I, Contemporary Mathematics 114 (1990) 297-308.
[28] Lamar, B., An Improved Branch and Bound Algorithm for Minimum Concave Cost Network Flow Problems, Working Paper, Graduate School of Management and Institute of Transportation Studies, University of California, Irvine (1989).
[29] Lootsma, F.A., A Comparative Srudy of Primal and Dual Approaches for Solving Separable and Partially-Separable Nonlinear Optimization Problems, Structural Optimization 1 (1989) 73-79.
[30] Lootsma, F.A., Parallel Nonlinear Optimization, To be published in "Advances in Distributed and Parallel Computing" (ed. H.W.Furer), Ablex Publishing Corp., Norwood, NJ (1990).
[31] Lozovanu, D.D., Properties of Optimal Solutions of a Grid Trnasport Problem with Concave Function of the Flows on the Arcs, Engineering Cybernetics 20 (1983) 34-38.
[32] Minoux, M., Multiflots de Coût Minimal avec Fonctions de Coût Concaves, Annals of Telecommunication 31 (1976) 77-92.
[33] Mockus, J., Bayesian Approach to Global Optimization, Kluwer Academic Publishers (1989).
[34] Moré, J.J. and Sorensen, D.C., Computing a Trust Region Step, SIAM J.Sci.Stat. Comput 4 (1983) 553-572.
[35] Pardalos, P.M., Enumerative Techniques for Solving Some Nonconvex Global Optimization Problems, OR Spectrum 10 (1988) 29-35.
[36] Pardallos, P.M., Parallel Search Algorithms in Global Optimization, Applied Mathematics and Computation 29 (1989), pp.219-229.
[37] Pardalos, P.M., Polynomial Time Algorithms for Some Classes of Constrained Nonconvex Quadratic Probems, Optimization 21 (1990) 843-853.
[38] Pardalos, P.M. and Li, X., Parallel Branch-and-Bound Algorithms for Combinatorial Optimization Problems, Supercomputer 39 (1990) 23-30.
[39] Pardalos, P.M., Ye, Y. and Han, C.-G., Algorithms for the Solution of Quadratic Knapsack Problems, Linear Algebra and its Applications 152 (1991) 69-91.
[40] Pardalos, P.M., Phillips, A.T. and Rosen, J.B., Topics in Parallel Computing in Mathematical Programming, Penn State University (1990).
[41] Pardalos, and Rosen, J.B., Global Optimization: Algorithms and Applications, Springer-Verlag, Lecture Notes in Computer Science 268 (1987).

[42] Pardalos, and Rosen, J.B. (Editors), Computational Methods in Global Optimization, Annals of Operations Research 25 (1990).
[43] Pardalos, P.M., and Schnitger, G., Checking Local Optimality in Constrained Quadratic Programming in NP-hard, Operations Research Letters 7 (1988) 33-35.
[44] Pardalos, P.M., and Vavasis, S.A., Quadratic Programming with One Negative Eigenvalue is NP-hard, Journal of Global Optimization 1 (1991) 15-22.
[45] Phillips, A.T. and Rosen, J.B., A Parallel Algorithm for the Linear Complementarity Problem, Annals of Operations Research (1988) 77-104.
[46] Phillips, A.T. and Rosen, J.B., Guaranteed ε-Approximate Solution for Indefinite Quadratic Global Minimization, Naval Research Logistics Quarterly 37 (1989) 499-514.
[47] Phillips, A.T. and Rosen, J.B., A Parallel Algorithm for Partially Separable Non-Convex Global Minimization, Annals of Operations Research 25 (1990) 101-118.
[48] Plasil, J. and Chlebnican, P., A New Algorithm for the Min Concave Cost Flow Problem, Working Paper, Technical University of Transport and Communications, Czechoslc 'akia (1990).
[49] Ratschek, H. and Rokne, J., New Computer Methods for Global Optimization, Halsted Press (1988).
[50] Rinnooy Kan, A.H.G. and Timmer, G.T., Stochastic Methods for Global Optimization, American Journal of Mathematics and Management Science 4 (1984) 7-40.
[51] Rech, P. and Barton, L.G., A Non-Convex Transportation Algorithm, in: E.M.L. Beale (ed.), Applications of Mathematical Programming Techniques, The English Universities Press LTD, London (1970) 250-260.
[52] Rosen, J.B. and Vliet, M.van, A Parallel Stochastic Method for the Constrained Concave Global Minimization Problem, Technical Report 87-31, Computer Science Department, University of Minnesota (1987).
[53] Sorensen, D.C., Newton's Method with a Model Trust Region Modification, SIAM J. Numer. Anal. 19 (1982) 409-426.
[54] Törn, A and Zilinskas, A., Global Optimization, Springer-Verlag, Lecture Notes in Computer Science 350 (1989).
[55] Tuy, H., Concave Programming Under Linear Constraints, Soviet Mathematics Doklady 5 (1964) 1437-1440.
[56] Yaged Jr., B., Minimum Cost Routing for Static Network Models, Networks 1 (1971) 139-172.
[57] Ye, Y. and Pardalos, P.M., A Class of Linear Complementarity Problems Solvable in Polynomial Time, Linear Algebra and its Applications 152 (1991) 3-17.
[58] Ye, Y., On the Affine Scaling Algorithm for Nonconvex Quadratic Programming, To appear in Mathematical Programming (1991).

INVESTIGAÇÃO OPERACIONAL, Vol. 11, nº1 (Junho de 1991) 31-51

SOLUTION OF LARGE-SCALE STRICTLY CONVEX LINEAR COMPLEMENTARITY PROBLEMS

J.J. Júdice*
Departamento de Matemática
Universidade de Coimbra
3000 Coimbra - Portugal

F.M.Pires*
Universidade do Algarve
Faro, Portugal

Abstract

The Linear Complementarity Problem (LCP) consists of finding vectors $z \in R^n$ and $w \in R^n$ such that $w = q + M z$, $z \geq 0$, $w \geq 0$ and $z^T w = 0$, where $q \in R^n$ and M is a square matrix of order n. The LCP is strictly convex if its matrix M is (symmetric or unsymmetric) positive definite. In this paper the use of (single and block) principal pivoting algorithms for the solution of large-scale strictly convex LCPs is discussed. A computational study indicates that block principal pivoting algorithms are highly recommendable to solve these LCPs and are in general superior over other alternative techniques.

Resumo

O Problema Linear Complementar (LCP) consiste em encontrar vectores $z \in R^n$ e $w \in R^n$ tais que $w = q + M z$, $z \geq 0$, $w \geq 0$ e $z^T w = 0$, onde $q \in R^n$ e M é uma matriz de ordem n. O LCP diz-se estritamente convexo se a matriz M é (simétrica ou não simétrica) positiva definida. Neste artigo é investigado o uso de métodos pivotais principais (simples ou por blocos) na resolução de LCPs estritamente convexos de grande dimensão. Um estudo computacional indica que os algoritmos pivotais principais por blocos são altamente recomendáveis para a resolução deste tipo de LCPs e são em geral superiores a outras técnicas alternativas.

Keywords: Linear Complementarity Problem, Convex Quadratic Programming, Linear Variational Inequalities, Large-Scale Problems, Sparse Matrices.

1 - Introduction

The Linear Complementarity Problem (LCP) consists of finding vectors $z \in R^n$ and $w \in R^n$ such that

$$w = q + Mz, \; z \geq 0, \; w \geq 0, \; z^T w = 0 \qquad (1)$$

where $q \in R^n$ and M is an n by n square matrix of order n. This nonlinear optimization problem has received a great interest during the past twenty years. Several direct, iterative and enumerative algorithms have been designed

* Partially supported by Instituto Nacional de Investigação Científica (INIC) under project 89/EXA/5.

for the solution of the LCP [32]. The LCP is equivalent to the following quadratic programming (QP) problem [6]

$$\underset{z\in K}{\text{Minimize}}\ f(z) = q^T z + \frac{1}{2}\, z^T (M + M^T) z \qquad (2)$$

where

$$K = \{z \in R^n : q + Mz \geq 0,\ z \geq 0\}$$

and three possible cases may occur:

$f(\bar{z}) = \min_{z\in K} f(z) = 0 \Rightarrow \bar{z}$ is solution of the LCP

$f(\bar{z}) = \min_{z\in K} f(z) > 0 \Rightarrow$ LCP is feasible but has no solution

$K = \emptyset \Rightarrow$ LCP is infeasible

Because of the equivalence stated above, the LCP is said to be Strictly Convex if and only if its matrix M is positive definite (PD), that is

$$z^T Mz > 0 \text{ for all } z \in R^n - \{0\} \qquad (3)$$

It is well-known that a strictly convex LCP has a unique solution for each $q \in R^n$ [32]. Furthermore it has been established recently [24] that strictly convex LCPs can be solved in polynomial-time.

The LCP is equivalent to the following Linear Variational Inequality Problem (LVI)

Find $\bar{z} \in R^n_+$ such that

$$(z - \bar{z})^T (q + M\bar{z}) \geq 0 \text{ for all } z \in R^n_+ \qquad (4)$$

where

$$R^n_+ = \{z \in R^n : z \geq 0\} \qquad (5)$$

If M is a symmetric PD matrix then both the LVI and the LCP are equivalent to the following Strictly Convex Quadratic Program (SCQP)

$$\underset{z\in R^n_+}{\text{Minimize}}\ q^T z + \frac{1}{2}\, z^T Mz \qquad (6)$$

To date many applications of the strictly convex LCP have been proposed. These include the solution of partial differential equations arising in Dirichlet problems with obstacles [9, 16] or free-boundary value pro' lems, such as the journal bearing [7, 8] and the elastic beam bending [43] problems. Elastoplastic analysis of structures [34, 37], portfolio selection problems [36, 37] and spatial equilibrium models [38] are also examples of important

applications of the strictly convex LCP. Strictly Convex Nonlinear Programs and Monotone Variational Inequality Problems can be solved by sequential algorithms relying on the solution of strictly convex LCPs [15, 20]. In most of these applications the matrix M is large and sparse.

Single [18, 23, 33] and Block [21, 25] Principal Pivoting algorithms have been recommended to solve strictly convex LCPs. Efficient implementations for these algorithms exist for the symmetric case [5, 22, 35] and it is possible to extend some of these procedures for unsymmetric PD matrices. Recently, a damped-Newton method [19] has been developed for the solution of the strictly convex LCP. A more practical version of this algorithm can be designed in which a block principal pivoting methodology is incorporated. In this paper we discuss all these techniques and corresponding implementations for large-scale strictly convex LCPs.

As stated before, the LCP is equivalent to the SCQP (6) provided M is a symmetric PD matrix. Therefore active-set methods [5, 13] can also be used to solve the strictly convex LCP when M is symmetric. Projected S.O.R. [7, 8] and projected gradient [2, 9, 30, 44] algorithms have been developed for the solution of the SCQP, whence they can handle the LCP when M is a symmetric PD matrix. Computational experience on SCQPs with some well-conditioned structured matrices indicates that these algorithms are efficient in these cases [7, 9, 30, 44].

In this paper the efficiency of the algorithms stated above is investigated by solving a number of large-scale LCPs taken from different sources. This study indicates that Projected Gradient and S.O.R. methods are efficient for some problems but they are not robust. The accuracy of the computed solution is usually small and their behavior is catastrophic in presence of ill-conditioning. Active-set and single principal pivoting methods are slow if the initial and last solutions are quite different, whence they are not too recommendable for large-scale strictly convex LCPs. A finite version of Kostreva block principal pivoting method is shown to be quite efficient for all the test problems and is in general superior to the remaining techniques. Furthermore the damped-Newton method is shown to be usually competitive with the block principal pivoting algorithm, but may face some problems when M is an ill-conditioned matrix.

The organization of this paper is as follows. Principal pivoting algorithms and the damped-Newton method are discussed in sections 2 and 3 respectively. An implementation of these algorithms for large-scale strictly convex LCPs is briefly described in section 4. A computational study of the performance of the algorithms stated above for the solution of a number of strictly convex LCPs is presented in the last section of this paper.

2 - Principal Pivoting Algorithms

Consider the LCP (1) and let F and T be two sets satisfying

$$F \cap T = \emptyset, \; F \cup T = \{1,\ldots, n\}$$

where $\emptyset$ represents the empty set. If M is a PD matrix the same happens to M_{FF} [32], whence this latter matrix is nonsingular [32]. Therefore $w = q+Mz$ is equivalent to

$$\begin{bmatrix} z_F \\ w_T \end{bmatrix} = \begin{bmatrix} \bar{q}_F \\ \bar{q}_T \end{bmatrix} + \begin{bmatrix} \bar{M}_{FF} & \bar{M}_{FT} \\ \bar{M}_{TF} & \bar{M}_{TT} \end{bmatrix} \begin{bmatrix} w_F \\ z_T \end{bmatrix} \tag{7}$$

where

$$\bar{M}_{FF} = M_{FF}^{-1}, \; \bar{M}_{FT} = -M_{FF}^{-1}M_{FT}, \; \bar{M}_{TF} = M_{TF}\, M_{FF}^{-1}, \; \bar{q}_F = -M_{FF}^{-1}q_F \tag{8}$$

$$\bar{M}_{TT} = M_{TT} - M_{TF}M_{FF}^{-1}M_{FT}, \; \bar{q}_T - q_T - M_{TF}\, M_{FF}^{-1}q_F$$

A complementary basic solution for the LCP is a solution of the system (7) in which $w_F = 0$ and $z_T = 0$. The variables z_i, $i \in T$ and w_i, $i \in F$ are called nonbasic while the remaining variables are said to be basic. If follows from (8) that the values of these variables can be found by

Solve	$M_{FF}\; \bar{q}_F = -\, q_F$
Compute	$\bar{q}_T = q_T + M_{TF}\, \bar{q}_F$

(9)

If the vector $\bar{q}$ is nonnegative, then

$$z_F = \bar{q}_F, \; z_T = 0, \; w_F = 0, \; w_T = \bar{q}_T$$

is the unique solution of the LCP. Otherwise the set

$$H = \{i : \bar{q}_i < 0\} \tag{10}$$

is nonempty and the complementary basic solution is said to be infeasible. Any index i satisfying $\bar{q}_i < 0$ is called an infeasibility and H is named the set of infeasibilities.

Principal pivoting algorithms are procedures that use in each iteration complementary basic infeasible solutions until finding a complementary basic feasible solution ($H = \emptyset$). In each iteration a set $H_1 \subseteq H$ is considered and the sets F and T are modified according to the following rules

$$\begin{aligned} F &= F - (F \cap H_1) \cup (T \cap H_1) \\ T &= T - (T \cap H_1) \cup (F \cap H_1) \end{aligned} \tag{11}$$

A principal pivoting algorithm is said to be single if the set H_1 has a single element in each iteration. Otherwise it is called a block principal pivoting algorithm. Next, we discuss the most important procedures of these two types.

(i) Single Principal Pivoting Algorithms

The most important single principal pivoting algorithms are due to Murty [33] and Keller [23] and their steps are presented below.

MURTY'S ALGORITHM

Step 0 - Let $F = \emptyset$ and $T = \{1, \ldots, n\}$.

Step 1 - Compute $\bar{q}$ by (9). If $\bar{q} \geq 0$, $z = (\bar{q}_F, 0)$, $w = (0, \bar{q}_T)$ is the unique solution of the LCP. Otherwise let

$$s = \min \{i \in F \cup T : \bar{q}_i < 0\} \tag{12}$$

Step 2 - Set

$$F = \begin{cases} F - \{s\} & \text{if } s \in F \\ F \cup \{s\} & \text{if } s \notin F \end{cases}$$

and $T = \{1, \ldots, n\} - F$. Go to Step 1.

KELLER'S ALGORITHM

Step 0 - Let $F = \emptyset$ and $T = \{1, \ldots, n\}$.

Step 1 - Compute $\bar{q}$. If $\bar{q}_T \geq 0$, $z = (\bar{q}_F, 0)$, $w = (0, \bar{q}_T)$ is the unique solution of the LCP. Otherwise let

$$s = \min \{i \in T : \bar{q}_i < 0\} \tag{13}$$

Step 2 - Compute $\bar{m}_{ss}$ and $\bar{M}_{Fs}$. Let $\theta_1 = -\dfrac{\bar{q}_s}{\bar{m}_{ss}}$ and

$$\theta_2 = \begin{cases} \dfrac{\bar{q}_r}{-\bar{m}_{rs}} = \min \left\{ \dfrac{\bar{q}_i}{-\bar{m}_{is}} : i \in F \text{ and } \bar{m}_{is} < 0 \right\} \\ +\infty \text{ if } \bar{m}_{is} \geq 0 \text{ for all } i \in F \end{cases}$$

where r is the first index in case of ties.

Step 3 - Let $\theta = \min \{\theta_1, \theta_2\}$.

(i) If $\theta = \theta_1$, set $F = F \cup \{s\}$, $T = T - \{s\}$ and go to Step 1.

(ii) If $\theta = \theta_2$, set $F = F - \{r\}$, $T = T \cup \{r\}$ and go to Step 4.

Step 4 - Compute $\bar{q}_s$ and $\bar{q}_F$. Go to Step 2.

It follows from (8) that the quantity $\bar{m}_{ss}$ and the vector $\bar{M}_{Fs}$ can be computed as follows:

Solve	$M_{FF}\, \bar{M}_{Fs} = -M_{Fs}$
Compute	$\bar{m}_{ss} = m_{ss} + M_{sF}\, \bar{M}_{Fs}$

(14)

Hence a system with the matrix M_{FF} has to be solved. However, no system is required to find the vector $\bar{q}$, since its components can be updated according to the following formulas:

$$\theta = \theta_1 \Rightarrow \begin{cases} \bar{q}_s = \theta_1 \\ \bar{q}_F = \bar{q}_F + \theta_1\, \bar{M}_{Fs} \\ \bar{q}_T = \bar{q}_T + \theta_1\, (M_{TF}\, \bar{M}_{Fs} + M_{Ts}) \end{cases}$$

and

$$\theta = \theta_2 \Rightarrow \begin{cases} \bar{q}_F = \bar{q}_F + \theta_2\, \bar{M}_{Fs} \\ \bar{q}_s = \bar{q}_s + \theta_2\, \bar{m}_{ss} \end{cases}$$

These formulas can be established by linear algebra manipulations.

Murty's method terminates in a finite number of iterations if M is a (symmetric or unsymmetric) PD matrix [33]. The convergence of Keller's method for symmetric and unsymmetric PD matrices has been established in [23] and [4] respectively. If M is a symmetric PD matrix, then Keller's

method reduces to the Fletcher and Jackson active-set algorithm [13] and is also convergent if the criterion (13) is replaced by

$$\bar{q}_s = \min \{ \bar{q}_i : \bar{q}_i < 0 \text{ and } i \in T\} \qquad (15)$$

This criterion leads to a reduction on the number of iterations and is usually recommendable to be incorporated in the algorithm [22]. In case of M being unsymmetric cycling might occur if the criterion (15) is incorporated in the algorithm. However, we have never faced any cycling in all the tests performed so far.

Computational experience with both the algorithms for the solution of large-scale LCPs with a symmetric PD matrix indicates that Keller's method is usually more efficient than Murty's method but the gap is small [22]. Furthermore the algorithms are quite sensitive to the number of initial infeasibilities and the number of variables that are nonbasic initially and become basic at the solution of the LCP [22]. The same conclusions can be stated for the case of unsymmetric PD matrices.

Murty's algorithm may start with any set $F \subseteq \{1,\ldots, n\}$. This is a great advantage over Keller's method in which F has to be chosen in such a way that the system

$$M_{FF}\, \bar{q}_F = - q_F$$

has a nonnegative solution. This important feature of Murty's algorithm is exploited in the design of a finite block principal pivoting algorithm, as is explained later.

Another single principal pivoting algorithm has been developed by Graves [18] and can be seen as a principal pivoting version of the famous Lemke's method [27]. In this paper we do not concentrate on Graves' algorithm, since this procedure is less efficient than the single principal pivoting techniques mentioned before [22]. Furthermore the usual version of Lemke's method requires in each iteration the solution of a system of order n [41], whence it is not competitive with the single principal pivoting methods.

(ii) Block Principal Pivoting Algorithms

As stated before, this type of algorithms allows in each iteration modifications of more than one element in the sets F and T. To our knowledge, the first technique of this type was developed by Kostreva [25]. The algorithm has the following features:

(i) $F = \emptyset$ initially.

(ii) In each iteration the sets F and T are modified according to the formulas (11) with $H_1 = H$, where H is the set of infeasibilities given by (10).

It is, however, possible to show that cycling may occur in this algorithm [32]. The algorithm can start with a set $F \neq \emptyset$ and there are at least two cases in which the procedure terminates in a finite number of iterations:

(i) All the off-diagonal of $M \in PD$ are nonpositive, that is, M is a nonsingular M-matrix ($M \in NSM$), and $F = \emptyset$ initially [3].

(ii) $M^{-1} \in NSM$ and $F = \{1,\ldots, n\}$ initially.

Furthermore the algorithm is polynomial in both cases, since it requires at most n iterations.

Substantial computational experience with Kostreva's algorithm has shown that cycling only occurs when the number of infeasibilities is small. Since Murty's algorithm terminates in a finite number of iterations whatever the initial set F is, then a good strategy is to use Kostreva's method until the number of infeasibilities is smaller than or equal to a quantity atc and apply Murty's method from then on. This hybrid procedure has been suggested by us in [21] and has proven quite successful for solving large-scale strictly convex LCPs [21]. The value atc = 3 is usually the most recommended choice, but slightly larger values for atc ($atc \leq 10$, say) seem to mantain the efficiency of the hybrid algorithm [21].

Despite sharing these good features, the algorithm remains heuristic and at least in theory cycling may occur. A simple way of overcoming this drawback consists of introducing a quantity NMAXPV, which represents the maximum number of iterations that Kostreva's algorithm is allowed to be used. This quantity is defined dynamically and is related with the reduction of the number of infeasibilities. The steps of the algorithm are as follows:

BLOCK PRINCIPAL PIVOTING ALGORITHM

Step 0 - Let atc and p be given positive integer numbers. Set k = 1, ninf = NMAXPV = $+\infty$, F = $\emptyset$ and T = {1, ..., n}.

Step 1 - Compute $\overline{q}$ by (9) and let

$$H = \{ i : \overline{q}_i < 0 \}$$

If H = $\emptyset$, z = $(\overline{q}_F , 0)$, w = $(0, \overline{q}_T)$ is the unique solution of the LCP and stop. If

$|H| \leq$ atc or k > NMAXPV

where |H| is the number of elements of H, go to Step 3. Otherwise go to Step 2.

Step 2 - Set

$$F = F - (H \cap F) \cup (H \cap T)$$

and T = {1,..., n} - F. If |H| < ninf, set ninf = |H| and NMAXPV = k + p.

Set k = k + 1 and go to Step 1.

Step 3 - Set NMAXPV = k. Let

$$r = \min \{ i \in H \}$$

and set

$$F = \begin{cases} F - \{r\} & \text{if } r \in F \\ F \cup \{r\} & \text{if } r \notin F \end{cases}$$

and T = {1, ..., n} - F. Set k = k + 1 and go to Step 1.

As it can be seen from the description of these steps, the idea behind this algorithm is to use Kostreva's method while the number of infeasibilities is reducing. If the number of infeasibilities is smaller than or equal to atc, then Kostreva's method is replaced by Murty's algorithm, which is used until the solution of the LCP is found. If the number of infeasibilities is greater than atc but has not reduced in an iteration to a value smaller than ninf, then (p - 1) iterations of Kostreva's method are allowed. If during these iterations the number of infeasibilities reduces to a value smaller than ninf, then the hybrid procedure continues as before. Otherwise, after these (p - 1) iterations Murty's method is used until the end. Computational experience with this

hybrid algorithm indicates that p should be chosen quite small (p = 3 is usually a good choice).

It is important to add that this block principal pivoting algorithm terminates in exactly k iterations with the unique solution of the LCP. The complexity of this algorithm is an open question. However, the famous problems of Murty [31] and Fathi [12] are solved in a number of iterations of order n for small values of p and atc. These are the problems that have been designed to show that single pivoting algorithms may require an exponential number of iterations to solve strictly convex LCPs.

3 - A Damped-Newton Algorithm

As stated in [29] the LCP is equivalent to the system of nonlinear equations

$$H(z) = \min(z, q + Mz) = 0 \tag{16}$$

The function $H : R^n \rightarrow R^n$ is B-differentiable [19], whence it is possible to develop a damped-Newton method in the spirit of [10] for the solution of this system. This algorithm has been developed by Harker and Pang [19] and incorporates an Armijo Criterion [2] for the function

$$g(z) = \frac{1}{2} H^T(z) H(z) \tag{17}$$

to assure global convergence. By exploiting the nice expression of the B-derivative of H it is possible to present the steps of this damped-Newton algorithm in the following form:

DAMPED-NEWTON ALGORITHM

Step 0 - Let $k = 1$, $z^k = 0$, $w^k = q$ and $\sigma \in]0, \frac{1}{2}[$.

Step 1 - Let

$$F = \{i : w_i^k < z_i^k\}$$
$$E = \{i : w_i^k = z_i^k\}$$

and $T = \{1,\ldots, n\}$ - (FUE). Solve the Mixed LCP

$$0 = q_F + M_{FF} v_F + M_{FE} v_E$$
$$u_E = q_E + M_{EF} v_F + M_{EE} v_E \tag{18}$$
$$v_E \geq 0,\ u_E \geq 0,\ u_E^T v_E = 0$$

Let

$$u_F = 0,\ v_T = 0,\ u_T = q_T + M_{TF}\, v_F + M_{TE}\, v_E$$

If $u \geq 0$ and $v \geq 0$, then $(z = v,\ w = u)$ is the solution of the LCP. Otherwise go to Step 2.

Step 2 - Let $z^{k+1} = v$ and $\lambda_k = 1$. If

$$g(z^{k+1}) \leq (1 - \sigma \lambda_k)\, g(z_k) \tag{19}$$

set $w^{k+1} = u$, $k = k + 1$ and go to Step 1. Otherwise let

$$\rho = \begin{cases} 0 & \text{if } E \neq \emptyset \\ \min\{\rho_1, \rho_2\} & \text{if } E = \emptyset \end{cases}$$

where

$$\rho_1 = \min\left\{ \frac{z_i^k - w_i^k}{z_i^k - w_i^k - v_i} : i \in F,\ v_i < 0 \right\}$$

$$\rho_2 = \min\left\{ \frac{w_i^k - z_i^k}{w_i^k - z_i^k - u_i} : i \in T,\ u_i < 0 \right\}$$

Step 3 - Find $\lambda_k \in]\, \rho,\ 1[$ by using the Armijo Criterion:

Let $m = 1$, $\mu \in]\, 0,\ 1[$

While $g(z^{k+1}) > (1 - \sigma\lambda_k)\, g(z^k)$ do

$\lambda_k = \mu^m$

$z^{k+1} = (1 - \lambda_k)\, z^k + \lambda_k\, v$

Set $w^{k+1} = (1 - \lambda_k)\, w^k + \lambda_k\, u$, $k = k + 1$ and go to Step 1.

It is important to add that if $E = \emptyset$ then the Mixed LCP (18) reduces to the system of linear equations

$$M_{FF}\, v_F = -q_F$$

In this case

$$u_T = q_T + M_{TF}\, v_F$$

Therefore, if $E = \emptyset$ the nonzero components of the vectors u and v are computed in a way similar to the procedure for computing the vector $\bar{q}$ in the principal pivoting methods. We note that if $q \neq 0$ then the set E is always empty. In fact the quantity ρ is used exactly to mantain empty this set E [19]. Finally, it is interesting to notice that the damped-Newton method reduces to Kostreva's algorithm if $q \neq 0$ and the condition (19) is satisfied for $\lambda_k = 1$ in each iteration k.

The damped-Newton method possesses global convergence and local quadratic convergence [19]. This is an advantage over the block pivoting methods, where no convergence rate is known for the time being. However, the algorithm depends on two parameters μ and σ (Harker and Pang have recommended $\mu = \frac{1}{2}$ and $\sigma = 0.2$ to be chosen in practice). Another drawback of this algorithm is the need of solving a Mixed LCP if the set E is not empty. As stated before this is overcome if $q \neq 0$, but the authors do not provide any explanation of how to deal with the case in which some of the components of the vector q are equal to zero.

4. An Implementation for Large-Scale LCP

It follows from the description of all the algorithms presented in this paper that the major computational effort in each iteration relies on the solution of a system of linear equations with a principal submatrix M_{FF} of M. If M is a symmetric PD matrix then M_{FF} also satisfies this property for each set F [32]. Hence the algorithms can be implemented for large-scale LCPs with a symmetric PD matrix by exploiting the ideas presented in [11, 14] for the solution of large systems of linear equations with these matrices. In this type of implementation an Analyse Phase is performed, which consists of finding an ordering for the rows and columns of M by using the minimum-degree algorithm. This phase terminates by providing storage space for the matrices L and D of the LDL^T decomposition of the matrix M_{FF}. In each iteration the matrices L and D are stored in a column oriented data structure in which the diagonal elements of D are stored apart in a dense vector of order n. If the number of modifications on the set F is small, then the LDL^T decomposition of the resulting matrix M_{FF} can be obtained from the LDL^T decomposition of the previous matrix M_{FF} by an efficient procedure that incorporates a sparse version [26] of Bennett's algorithm [1]. If the set F is modified in t elements where t is much larger than one, then it is better to compute the LDL^T decomposition from stack instead of applying t times the updating procedure. Another data structure is necessary to store the matrix M. In order to facilitate the updating of the LDL^T decomposition it is better to store M columnwise as an unsymmetric matrix. We suggest [22] for a full description of this implementation.

Consider now the case in which M is an unsymmetric PD matrix. Then diagonal pivoting is stable under one of the two following hypothesis:

(P): (i) M is diagonally dominant [42].

(ii) The quantity cond(R) ||S|| is not large, where

$$R = \frac{1}{2}(M + M^T), \quad S = \frac{1}{2}(M - M^T)$$

and cond(R) represents the condition number of the matrix R [17].

If one of these conditions is satisfied, then an implementation based on the ideas stated before can be designed for the algorithms discussed in this paper. As before, an Analyse Phase is first performed in which the minimum-degree algorithm is applied to the structure of the matrix $M + M^T$. The LDU decomposition of a matrix M_{FF} is used in each iteration for the solution of the linear system with this matrix. This decomposition is stored in a data structure containing a further real array in which all nonzero elements of the matrix U are stored by rows. When the number of modifications of the set F is small, the LDU decomposition of M_{FF} can be updated from the LDU decomposition of the previous matrix M_{FF} by a simple modification of the procedure for symmetric matrices described in [22]. If the number of modifications is not small, then it is better to compute the LDU decomposition from stack according to the ordering achieved in the Analyse Phase. The matrix M is stored as in the symmetric case.

If the property (P) does not hold or the structure of the matrix M is quite unsymmetric, then this type of implementation is not recommendable. In this case it is better not to use an updating scheme and solve the systems with matrix M_{FF} by using a sparse solver for unsymmetric linear systems [11].

5. Computational Experience

In this section we present some computational experience with the algorithms discussed in this paper for the solution of the following large-scale problems:

TP1 – LCPs with a symmetric PD matrix taken from elastoplastic analysis of structures [34].

TP2 – LCP with a symmetric PD matrix taken from a portfolio selection model [36].

TP3, TP4, TP5 – LCPs with a symmetric matrix that has been randomly generated according to the Stewart's technique described in [44].

TP6, TP7 – LCPs with a 5-point finite difference symmetric PD matrix [35].

TP8, TP9, TP10, TP11 – LCPs with a pentadiagonal symmetric PD matrix taken from [28].

TP12 – LCP in which M is an unsymmetric diagonally dominant PD matrix with a symmetric pattern and has been randomly generated according to the technique described in [39].

TP13 – LCP in which M is an unsymmetric PD matrix with a symmetric pattern and has been randomly generated according to the technique described in [39].

The orders of the matrices of these test problems are presented in tables 1 and 2. In all the test problems but TP11 the vector q has been generated by a technique described in [40] that fixes the number of basic variables z_i in the unique solution of the LCP. The vector q of the test problem has been randomly generated. In both cases all the components of the vector q are nonzero.

The experiences have been performed on a CDC CYBER 180-830 of the University of Porto, whose machine epsilon [10] is 10^{-14}. The following notations are used in the tables containing the results of the experiments:

n = dimension of the LCP = order of the matrix M.

neg = number of negative components of the vector q.

|F| = number of elements of the set F associated with the unique solution of the LCP.

TO = CPU time in seconds for finding an ordering for the rows and columns of M (minimum-degree algorithm [14]).

TF = CPU time in seconds for the symbolic phase [14].

T = CPU time in seconds for the algorithms.

TT = Total CPU time in seconds (=TO+TF+T for the BLOCK and KELLER algorithms).

NO = number of operations (multiplications and divisions) multiplied by 10^{-5}.

RES = Residual of the unique solution $(\bar{z},\bar{w})$ of the LCP. It is given by

$$\| \bar{w} - (q + M\bar{z}) \|_2 = \left[\sum_{i=1}^{n} (\bar{w}_i - q_i - \sum_{j=1}^{n} m_{ij} \bar{z}_j)^2 \right]^{1/2}$$

If α and β are positive integer numbers we use the quantity α–β to represent the residual $\alpha \times 10^{-\beta}$.

NC = algorithm fails to converge after 5000 seconds of CPU time.

In Table 1 we present a comparison among the algorithms BLOCK (p = atc = 3), KELLER, an Active-Set iterative method [35] (ASI), a Projected-Gradient algorithm [9] (PGRAD) and a Projected S.O.R. method [7, 8] (PSOR). The tolerances used in the stopping criteria of these iterative methods have been chosen in order to achieve at least some accuracy in the computed solution. These values are presented under the notation TOL and, as before, we use the quantity β to represent the tolerance $10^{-\beta}$. The efficiency of the algorithms PGRAD and PSOR depends on the values of the parameter η [9] and the relaxation parameter [7, 8] respectively. For each problem we have tested different values for these two parameters. In Table 1 only the results corresponding to the best performances of the algorithms are presented. The corresponding values of these parameters are given under the notation PAR.

The computational results presented in Table 1 lead to the conclusion that the BLOCK algorithm is usually the most efficient procedure to solve large-scale strictly convex LCPs with sparse structure. The algorithm is quite robust and does not seem to be much influenced by the number of negative components of the vector q or by the number |F| of the basic variables z_i in the unique solution of the LCP. The residuals of the computed solutions are quite good. This last property is shared by KELLER method. However, the quantities neg and |F| mentioned above have a strong effect on the efficiency of this last algorithm. This is not surprising, since the number of basic variables z_i is initially equal to zero and exactly one variable z_i changes from basic to nonbasic or vice-versa in each iteration. The iterative methods are efficient for solving the test problem TP2, but in general perform much worse in terms of the number of operations and CPU time than the BLOCK algorithm. Furthermore the residuals of the computed solutions are in certain

cases quite large. The algorithms face a slow convergence for some of the test problems. This is particularly evident for the LCPs with the pentadiagonal matrix stated in [28], where the procedures are able to find a solution of the LCP for small values of n ($n \leq 100$), with a large residual, but fail to converge for larger values of n. It is important to add that the condition number of this pentadiagonal matrix is an increasing function of its order n. We also note that the BLOCK and KELLER algorithms find solutions for these problems with small residuals.

In Table 2 the efficiencies of the algorithms BLOCK and damped-Newton are compared. It is important to add that all the components of the vector q are nonzero, whence exactly a linear system of equations has to be solved in each iteration of the damped-Newton algorithm. The computational study shows that the performance of the two algorithms is quite similar in terms of the number of iterations and CPU time for almost all the test problems. However, the algorithms do not solve the test problem TP11 efficiently. In the BLOCK algorithm we have experienced an increase of the number of infeasibilities quite often. Then the number of infeasibilities reduces to a lower level during the (p-1) next block iterations that are performed after the increase of the number of infeasibilities. This type of situation has occurred several times in the second experiment with the test problem TP11. The damped-Newton method performs much worse for this test problem. It seems that the reason for this bad behavior is that the condition (19) never holds with $\lambda_k = 1$, whence the quadratic convergence is not satisfied for this test problem. It is important to add that the reason for the bad behavior of the algorithm does not rely on the generation of the vector. In fact the other problem in which q was randomly generated was solved efficiently by the algorithms. Furthermore the test problem TP9 was not solved so efficiently by the BLOCK method as usual and the vector q was generated by the technique that fixes the solution of the LCP beforehand.

As a final conclusion of this computational study we claim that the block algorithm is highly recommendable to solve large-scale strictly convex LCPs. The algorithm is in general superior over iterative methods, single principal pivoting algorithms and active-set methods. A damped-Newton developed by Harker and Pang [19] has shown to be usually competitive with the algorithm

BLOCK. However, its performance may be much worse in some cases where the matrix of the LCP is ill-conditioned. The efficiency of the BLOCK algorithm also seems to be influenced by ill-conditioning, but this drawback is not so relevant. Furthermore no difficulty arises in the algorithm BLOCK if some of the components of the vector q are equal to zero. As stated before, the damped-Newton becomes more involved in this case, since it is necessary to solve a Mixed LCP in each iteration instead of a system of linear equations. We are currently investigating how to deal with this case.

TP	n	neg	|F|		BLOCK	KELLER	ASI	PGRAD	PSOR
				NI	6	242	1319	279	287
				NO	2.12	5.77	90.8	45.7	19.2
1	484	151	242	TT	9.2	22.2	138.2	65.2	40.8
				RES	1-11	2-11	1-6	8-4	1-3
				TOL			6	3	4
				PAR				$+\infty$	1.5
				NI	3	300	7	14	614
				NO	0.87	6.44	1.69	2.08	44.3
2	600	303	300	TT	10.1	24.3	3.7	4.9	91.6
				RES	3-12	7-12	7-12	3-12	2-7
				TOL			6	6	6
				PAR				0.03	1.7
				NI	7	550	504	305	175
				NO	1.24	16.8	88.9	87.2	20.9
3	1000	631	500	TT	9.7	61.8	140.8	128.5	43.5
				RES	2-12	2-12	1-6	1-4	1-4
				TOL			6	4	4
				PAR				$+\infty$	1.7

Table 1 – Comparison between direct and iterative methods for the solution of strictly convex LCPs

Cont.(Table 1)

TP	n	neg	|F|		BLOCK	KELLER	ASI	PGRAD	PSOR
				NI	7	832	319	255	139
				NO	0.62	16.2	70.3	55.2	15.3
4	1600	947	800	TT	8.1	85.7	125.7	101.2	29.3
				RES	1-12	1-12	6-7	1-4	7-5
				TOL			6	3	4
				PAR				+∞	1.3
				NI	4	640	98	86	85
				NO	1.5	21.8	10.9	11.3	6.53
6	1280	326	640	TT	7.82	76.2	18.3	24.3	12.1
				RES	1-12	2-12	8-7	9-5	1-9
				TOL			6	3	6
				PAR				+∞	1.7
				NI	4	800	89	88	92
				NO	1.87	28.4	14.3	14.5	8.83
7	1600	407	800	TT	9.85	111.6	23.9	31.2	16.3
				RES	2-12	2-12	8-7	9-5	1-9
				TOL			6	3	6
				PAR				+∞	1.7
				NI	8	50	79	67	1695
				NO	0.03	0.16	2.0	1.51	18.09
8	100	23	50	TT	0.14	0.53	1.57	1.32	16.4
				RES	7-13	1-12	6-7	1-5	1-10
				TOL			6	4	6
				PAR				+∞	1.7
				NI	67	829			
				NO	10.5	149.2			
9	1500	357	750	TT	31.9	271.1	NC	NC	NC
				RES	4-12	5-12			
				TOL					
				PAR					

TP	n	TO	TF	neg	|F|	BLOCK NI	BLOCK T	DAMPED NEWTON NI	DAMPED NEWTON T
1	484	3.83	0.41	268	242	6	5.2	5	4.46
				297	386	5	8.41	5	8.57
2	600	7.2	0.25	314	300	3	2.54	3	2.61
				450	480	3	4.71	3	4.81
5	1500	4.46	0.25	921	750	5	2.05	7	4.93
				1072	1200	10	7.55	10	9.59
10	1000	–	0.18	478	500	6	1.21	5	1.24
				503	800	5	1.6	5	1.87
11	400	–	0.07	145	207	19	1.77	21	3.6
				331	400	223	46.7	696	432.6
12	1000	1.77	0.19	591	500	2	0.33	2	0.39
				843	800	2	0.54	2	0.61
13	1500	1.75	0.17	876	750	2	0.46	2	0.54
				1259	1200	2	0.79	2	0.86

Table 2 – Comparison between BLOCK and Damped-Newton algorithms

References

[1] J.M.Bennett, *Triangular factors of modified matrices*, Numerische Mathematik 7 (1965) 217-221.

[2] D.P.Bertsekas, *Constrained Optimization and Lagrange Multipliers Methods*, Academic Press, New York, 1982.

[3] R.Chandrasekaran, *A special case of the complementary pivot problem*, Opsearch 7 (1970) 263-268.

[4] Y.Y.Chang, *Least-index resolution of degeneracy in linear complementarity problems*, Technical Report 79-14, Department of Operations Research, Stanford University, 1979.

[5] T.F.Coleman and L.A.Hulbert, *A direct active set algorithm for large sparse quadratic programs with bounds*, Mathematical Programming 45 (1989) 373-406.

[6] R.W.Cottle, *Note on a fundamental theorem in quadratic programming*, Journal SIAM 12 (1964) 663-665.

[7] R.W.Cottle, G.H.Golub and R.S.Sacher, *On the solution of large, structured linear complementarity problems: the block partitioned case*, Applied Mathematics and Optimization 4 (1978) 347-363.

[8] C.W.Cryer, *The solution of a quadratic programming problem using systematic overrelaxation*, Journal SIAM Control 9 (1971) 385-392.

[9] R.S.Dembo and U.Tulowitzki, *On the minimization of a qudratic function subject to box constraints*, Working Paper 71, Yale University, School of Organization and Management, 1983.

[10] J.E.Dennis Jr. and R.B.Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[11] I.S.Duff, A.M.Erisman and J.R.Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.

[12] Y.Fathi, *Computational complexity of LCPs associated with positive definite symmetric matrices*, Mathematical Programming 17 (1979) 335-344.

[13] R.Fletcher and M.P.Jackson, *Minimization of a quadratic function subject only to upper and lower bounds*, Journal Institute Mathematics and Applications 14 (1974) 159-174.

[14] A.George and J.W.H.Liu, *Computer Solution of Large Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[15] P.E.Gill, W.Murray and M.H.Wright, *Practical Optimization*, Academic Press, New York, 1981.

[16] R.Glowinski, *Finite elements and variational inequalities*, MRC Technical Report 1885, Mathematics Research Center, University of Wisconsin-Madison, 1978.

[17] G.H.Golub and C.Van Loan, *Unsymmetric positive definite linear systems*, Linear Algebra and its Applications 28 (1979) 85-98.

[18] R.L.Graves, *A principal pivoting simplex algorithm for linear and quadratic programming*, Operations Research 15 (1967) 482-494.

[19] P.T.Harker and J.S.Pang, *A damped-Newton method for the linear complementarity problem*, Lectures in Applied Mathematics 26 (1990) 265-284.

[20] N.H.Josephy, *Newton's method for generalized equations*, Technical Report 1966, Mathematics Research Center, University of Wisconsin-Madison, 1979.

[21] J.J.Júdice and F.M.Pires, *Bard-type methods for the linear complementarity problem with symmetric positive definite matrices*, IMA Journal of Mathematics Applied in Business and Industry 2 (1988/89) 51-68.

[22] J.J.Júdice and F.M.Pires, *Direct methods for convex quadratic programs subject to box constraints*, Investigação Operacional 9 (1989) 23-56.

[23] E.L.Keller, *The general quadratic optimization problem*, Mathematical Programming 5 (1973) 311-337.

[24] M.Kojima, S.Mizuno and A.Yoshise, *A polynomial-time algorithm for a class of linear complementarity problems*, Mathematical Programming 44 (1989) 1-26.

[25] M.Kostreva, *Block pivot methods for solving the complementarity problem*, Linear Algebra and its Applications 21 (1978) 207-215.

[26] K.H.Law, *Sparse matrix modifications in structural reanalysis*, International Journal for Numerical Methods in Engineering 21 (1985) 37-63.

[27] C.E.Lemke, *On complementary pivot theory*, in "Mathematics of Decision Sciences", edited by G.B.Dantzig and A.F.Veinott Jr., American Mathematical Society, Providence, 1986, pp.95-114.

[28] Y.Y.Lin and J.S.Pang, *Iterative methods for large convex programs: a survey*, SIAM Journal on Control and Optimization 25 (1987) 383-411.

[29] O.L.Mangasarian, *Equivalence of the complementarity problem to a system of nonlinear equations*, SIAM Journal of Applied Mathematics 31 (1976) 89-92.

[30] J.J.Moré and G.Toraldo, *On the solution of large quadratic programming problems with bound constraints*, SIAM Journal on Optimization 1 (1991) 93-113.

[31] K.G.Murty, *Computational complexity of complementarity pivot methods*, Mathematical Programming Study 7 (1978) 61-73.

[32] K.G.Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann Verlag, Berlin, 1988.

[33] K.G.Murty, *Note on a Bard-type scheme for solving the complementary problem*, Opsearch 11 (1974) 123-130.

[34] A.S.Neves, A.M.Faustino, F.M.Pires and J.J.Júdice, *Elastoplastic analysis of structures and linear complementarity*, in "OR Models on Microcomputers", edited by J.D.Coelho and L.V.Tavares, North-Holland, Amsterdam, 1986, pp.217-228.

[35] D.P.O'Leary, *A generalized conjugate gradient method for solving a class of quadratic programming problems*, Linear Algebra and its Applications 34 (1980) 371-399.

[36] J.S.Pang, *A new and efficient algorithm for a class of portfolio selection problems*, Operations Research 28 (1980) 754-767.

[37] J.S.Pang, I.Kaneko and W.P.Hallman, *On the solution of some (parametric) linear complementarity problems with applications to portfolio analysis, structural engineering and graduation*, Mathematical Programming 16 (1979) 325-347.

[38] J.S.Pang and S.C.Lee, *A parametric linear complementarity technique for the computation of equilibrium in a single commodity spatial model*, Mathematical Programming 20 (1981) 81-102.

[39] P.M.Pardalos and G.P.Rodgers, *Computational aspects of a branch and bound algorithm for quadratic zero-one programming*, Computing 45 (1990) 131-144.

[40] B.Ramarao and C.M.Shetty, *Application of disjunctive programming to the linear complementarity problem*, Naval Research Logistics Quarterly 31 (1984) 589-600.

[41] J.A.Tomlin, *Robust implementation of Lemke's method for the linear complementarity problem*, Mathematical Programming Study 7 (1978) 55-60.

[42] B.Wendroff, *Theoretical Numerical Analysis*, Academic Press, New York, 1966.

[43] D.R.Westbrook, *Contact problems for the elastic beam*, Computers and Structures 15 (1982) 473-479.

[44] E.K.Yang and J.W.Tolle, *A class of methods for solving large convex quadratic programs subject to box constraints*, Working Paper, Management Sciences Department, University of Massachusetts at Boston, 1985.

INVESTIGAÇÃO OPERACIONAL, Vol. 11, nº1 (Junho de 1991) 52- 69

AN ALGORITHM FOR THE MULTIOBJECTIVE SHORTEST PATH PROBLEM ON ACYCLIC NETWORKS

José A. Azevedo
Escola Superior Agrária
Instituto Politécnico de Coimbra

Ernesto Q.V. Martins
Departamento de Matemática
Universidade de Coimbra

Abstract

In this paper a new algorithm for the determination of the set of nondominated paths in an acyclic network is presented. The algorithm is based on a ranking path procedure and is supported by the Principle of Optimality. Computational experience is reported showing the superiority of this new approach over other alternative techniques, namely in terms of execution time and memory requirements.

Resumo

Neste artigo apresenta-se um novo algoritmo para determinação do conjunto dos caminhos não dominados em redes sem ciclos. O algoritmo é baseado num processo para a determinação ordenada de caminhos e no Princípio da Optimalidade. Descreve-se ainda a experiência computacional realizada, que indica a superioridade deste novo processo sobre outras técnicas alternativas, nomeadamente no que respeita ao tempo de execução e ao espaço de memória necessário.

Keywords: Network, nondominated path, labelling algorithm, paths ranking.

1. Introduction

Let $(\mathcal{N}, \mathcal{A})$ be a directed network, where $\mathcal{N} = \{1,\ldots, n\}$ is a set of n elements called nodes and $\mathcal{A} = \{a_1,\ldots, a_m\}$ a set of m arcs. Each arc a_k is defined by an ordered pair (i, j) of nodes, such that $i \neq j$. Throughout, the arc a_k is also denoted by (i, j).

Let $x, h \in \mathcal{N}$ be two nodes of $(\mathcal{N}, \mathcal{A})$. A path from x to h in $(\mathcal{N}, \mathcal{A})$ is a sequence $\{x \equiv v_0, a_1, v_1, a_2,\ldots, v_{l-1}, a_l, v_l \equiv h\}$, of nodes and arcs, such that $v_k \in \mathcal{N}$ and $a_k = (v_{k-1}, v_k) \in \mathcal{A}$. An elementary path is a path for which:

1. for all $i, j \in \{1,\ldots, l-1\}$, $v_i \neq v_j$, if $i \neq j$; i.e., all intermediate nodes are distinct;
2. $v_i \notin \{x, h\}$ for all $i \in \{1,\ldots, l-1\}$.

An elementary path from a node to itself is called a cycle.

In this paper we assume that the network has no cycles, that is, $(\mathcal{N}, \mathcal{A})$ is an acyclic network. As a consequence, every path in $(\mathcal{N}, \mathcal{A})$ is an elementary path.

We denote by $\mathcal{P}_{xh}$, the set of all paths from x to h in $(\mathcal{N}, \mathcal{A})$, $\forall x, h \in \mathcal{N}$.

For $r \geq 2$, let $[c_{ij}^1, c_{ij}^2, \ldots, c_{ij}^r]$ be a real r-vector associated with each arc (i, j), for all $(i, j) \in \mathcal{A}$. Let $c : \mathcal{P}_{xh} \rightarrow \mathbb{R}^r : p \rightarrow c(p) = [c^1(p), \ldots, c^r(p)] = [\sum_{(i,j)\in p} c_{ij}^1, \ldots, \sum_{(i,j)\in p} c_{ij}^r]$, be a function which assigns a real r-vector to each path $p \in \mathcal{P}_{xh}$ for any $x, h \in \mathcal{N}$.

Let $s, t \in \mathcal{N}$ be a specific pair of given nodes, such that $s \neq t$. We denote $\mathcal{P}_{st}$ as $\mathcal{P}$. For this given pair of nodes, suppose we wish to find a path $\bar{p} \in \mathcal{P}$, which minimizes the r linear functions $c^k(p)$ simultaneously. This problem has no solution in general, since there might exist some sort of conflict among this functions. So our goal is to find a preferred path, which is optimal in some sense. A good measure for optimality is the concept of nondominated path, which is introduced below.

Definition 1.1 Let $p_1, p_2 \in \mathcal{P}_{xh}$ be two paths from x to h in $(\mathcal{N}, \mathcal{A})$ for any pair of nodes x and h. p_1 **dominates** p_2, denoted by $p_1 \, \mathbf{D} \, p_2$, **if and only if** $c^1(p_1) \leq c^1(p_2), \ldots, c^r(p_1) \leq c^r(p_2)$, and the strict inequality holds for some $k \in \{1, \ldots, r\}$.

Definition 1.2 $p_2 \in \mathcal{P}_{xh}$, is said to be a **dominated path, if and only if** there exists a path $p_1 \in \mathcal{P}_{xh}$, such that $p_1 \, \mathbf{D} \, p_2$.

Let $\mathcal{P}_{xh}^D$ be the set of dominated paths from x to h in $(\mathcal{N}, \mathcal{A})$. A path p from x to h is said to be a nondominated path in $\mathcal{P}_{xh}$ if and only if $p \notin \mathcal{P}_{xh}^D$. Let $\mathcal{P}_{xh}^N = \mathcal{P}_{xh} - \mathcal{P}_{xh}^D$ denote the set of nondominated paths from x to h in $(\mathcal{N}, \mathcal{A})$. Notice that $\mathcal{P}_{xh}^N \neq \emptyset$, whenever $\mathcal{P}_{xh} \neq \emptyset$, [5, 10, 11]. We also write for simplicity $\mathcal{P}^N$ and $\mathcal{P}^D$ instead $\mathcal{P}_{st}^N$ and $\mathcal{P}_{st}^D$ respectively.

Whenever there is no path from s to t simultaneously minimizing the r functions $c^k(p)$ in $\mathcal{P}$, $\mathcal{P}^N$ is a suitable set for our purposes. In fact, if $p^* \in \mathcal{P}^N$ then p^* can be considered as an optimal path in the sense that every path p in $\mathcal{P}$ for which a component $c^i(p)$ of $c(p)$ has smaller value, then p^* must satisfy $c^j(p) > c^j(p^*)$ for some $j \neq i$.

In this paper we present an algorithm for determining $\mathcal{P}^N$ in acyclic networks.

2. Mathematical Background

In this section we recall some concepts that are used later.

Definition 2.1 Let $x, y \in \mathbb{R}^r$. x is a lexicographic positive vector, $(x >_L 0,)$ **if and only if** $x_1 > 0$ or $x_1 = \ldots = x_l = 0$ and $x_{l+1} > 0$, for some $l \in \{1,\ldots, r-1\}$. x is lexicographic greater than y, $(x >_L y)$, **if and only if** $(x - y)$ is a lexicographic positive vector.

Definition 2.2 Let $p \in \mathcal{P}^N$ be some nondominated path form s to t in $(\mathcal{N}, \mathcal{A})$. The Principle of Optimality holds for p **if and only if** each intermediate node i of p, the sub-path p_{si} of p is a nondominated path from s to i in $(\mathcal{N}, \mathcal{A})$.

Lemma 2.1 Let $(\mathcal{N}, \mathcal{A})$ be an acyclic network. The Principle of Optimality holds for any nondominated path in $(\mathcal{N}, \mathcal{A})$.

Proof: Let $p \in \mathcal{P}^N$ be a nondominated path for a given pair of nodes s and t in an acyclic network $(\mathcal{N},\mathcal{A})$ such that its sub-path $p_{xh} \in \mathcal{P}^D_{xh}$ is dominated. Thus there is some path $q_{si} \in \mathcal{P}_{si}$ such that $c^k(q_{si}) \leq c^k(p_{si})$ holds for any $k \in \{1,\ldots, r\}$ and $c^j(q_{si}) < c^j(p_{si})$ for some $j \in \{1,\ldots, r\}$.

Let $\bar{p} = q_{si} \cup p_{it}$. Since $(\mathcal{N}, \mathcal{A})$ is an acyclic network $\bar{p}$ is an elementary path from s to t in $(\mathcal{N}, \mathcal{A})$. Furthermore, $c^k(\bar{p}) \leq c^k(p)$ for any $k \in \{1,\ldots, r\}$ and $c^j(\bar{p}) < c^j(p)$. So we may conclude that $\bar{p}\ D\ p$, which is impossible. ♦

To show that the Principle of Optimality may not hold if some cycle exists in $(\mathcal{N}, \mathcal{A})$, let us consider a nerwork $(\mathcal{N}, \mathcal{A})$ in which N={1, 2, 3, 4} and the set of arcs is defined as follows:

$\mathcal{A}$	(1,2)	(1,3)	(2,3)	(3,2)	(2,4)	(3,4)
c^1_{ij}	0	0	0	–1	0	0
c^2_{ij}	0	0	–1	0	0	0

Only two (elementary) nondominated paths can be determined in $(\mathcal{N}, \mathcal{A})$ from s = 1 to t = 4: p = {1, (1,2), 2, (2,3), 3, (3,4), 4} and q = {1, (1,3), 3, (3,2), 2, (2,4), 4}. However $\bar{q}$ = {1, (1,3), 3, (3,2), 2} dominates $\bar{p}$ = {1, (1,2), 2}, since $c(\bar{q})$ = [–1, 0] and $c(\bar{p})$ = [0, 0].

A more general sufficient condition for the Principle of Optimality to hold for general networks was established by one of the authors [10, 11].

A class containing all the algorithms based on the Principle of Optimality can be devised. Hansen [9] seems to be the first one to propose an algorithm in this class. His algorithm is valid for the bicriterion case and it can be viewed as a generalization of the classical shortest path algorithm of Dijkstra [8]. In this algorithm one or more labels are assigned to each node, the set of labels being split in two sub-sets: the set of the permanent labels and the set of the temporary ones. Each label assigned to a node $i \in \mathcal{N}$, is a pair (π_i, ξ_i), where $\pi_i = [c^1(p_{si}), c^2(p_{si})]$ is the value of some path $p_{si} \in \mathcal{P}_{si}$ and ξ_i is the predecessor node of i in the path p_{si}. Permanent labels correspond to nondominated paths, while each temporary label corresponds to some path which can be dominated or not.

At each step of Hansen's algorithm the lexicographic smallest temporary label, assigned to some node $i \in \mathcal{N}$, becomes permanent and is used to assign a new temporary label to all nodes j for which (i, j) is an arc of $(\mathcal{N}, \mathcal{A})$. After this step, a nondominance test is used to remove all the temporary labels corresponding to dominated paths.

A straightforward generalization of Hansen's algorithm for any number of objectives was proposed by Martins [10]. More general conditions were also established in order to ensure that the lexicographic smallest temporary label corresponds to a nondominated path. In fact, while Hansen considers $c_{ij}^k > 0$ for $k \in \{1, 2\}$, Martins proved that we only need to assume that $[c_{ij}^1, \ldots, c_{ij}^r] >_L 0$ or $[c_{ij}^1, \ldots, c_{ij}^r] = 0$ holds for any arc $(i, j) \in \mathcal{A}$. In this paper we take this assumption into account.

Another class comprises the algorithms based on the ranking of a well determined sub-set of paths. Clímaco and Martins [5] were the first to propose an algorithm in this class valid for the bicriterion case. A generalization for any number of objectives was also proposed by the same authors, [6]. The nondominance test is the main difference between the bicriterion shortest path algorithm and its generalization for any nymber of objectives. In fact, in the bicriterion case the value $[c^1(p), c^2(p)]$ of each path $p \in \mathcal{P}$ is compared with the value of the nondominated path just determined. Since paths are found in accordance to the nondecreasing lexicographic order of their values, we may easily decide whether p is a nondominated path.

When we are faced with three or more objectives, the nondominance test for some path $p \in \mathcal{P}$ requires the comparison of $c(p)$ with the value of all the nondominated paths $q \in \mathcal{P}^N$ for which $c(q) <_L c(p)$ holds. This is not necessary if a nondominated path $q \in \mathcal{P}^N$ had been previously found such that $c(p) = c(q)$ or $q \mathbb{D} p$. In the first case $p \in \mathcal{P}^N$ while $p \notin \mathcal{P}^N$ in the second case.

This fact can be easily illustrated with a small example. Let us assume that three nondominated paths p_1, p_2 and p_3 were already determined such that $c(p_1) = [0,10,2]$, $c(p_2) = [0,11,1]$ and $c(p_3) = [5,0,12]$. The determination of a new path p is as follows: if $c(p) = c(p_3)$ then we conclude that p is also a nondominated path; if $c(p) >_L c(p_3)$, for instance, $c(p) = [1000,1000,2]$, the comparison of $c(p)$ with $c(p_2)$ is sufficient to show that p is a dominated path; if for instance, $c(p) = [1000,1000,0]$, we have to compare it with the values $c(p_1)$, $c(p_2)$ and $c(p_3)$ to conclude that p is also a nondominated path.

Clearly, the algorithm is valid since the lexicographic shortest path is nondominated and paths are determined by nondecreasing lexicographic order.

3. The Algorithm

The main idea associated with the algorithm is quite simple, since it uses a ranking paths procedure as in Clímaco and Martins' algorithms. However, since the Principle of Optimality holds for acyclic networks, nondominated paths cannot be made up of dominated sub-paths. So, the nondominance test can be applied to each intermediate node of some candidate path to $\mathcal{P}^N$, to determine whether it is dominated.

Roughly, let $p = p_{sx} \cup p_{xt}$ be some candidate path and let us assume that x is the first intermediate node of p for which $p_{sx} \notin \mathcal{P}^N_{sx}$. Since $p_{sx} \notin \mathcal{P}^N_{sx}$ and p_{sx} is a sub-path of p, we may conclude that $p \notin \mathcal{P}^N$. The next step is an attempt to find a nondominated path $q_{sx} \in \mathcal{P}^N_{sx}$, such that $c(q_{sx}) >_L c(p_{sx})$. As $(\mathcal{N}, \mathcal{A})$ is an acyclic network, $q_{sx} \cup p_{xt}$ is an elementary path candidate to $\mathcal{P}^N$.

Let $\mathcal{P}_{si} = \{p_1, p_2, \ldots, p_K\}$ be the lexicographic ordered set of all the paths from $s \in \mathcal{N}$ to some node $i \in \mathcal{N} - \{s\}$. That is, $c(p_h) <_L c(p_{h+1})$ or $c(p_h) = c(p_{h+1})$, $\forall h \in \{1, \ldots, K-1\}$.

Definition 3.1 $p_{h+1} \in \mathcal{P}_{si}$ is the **alternative** to $p_h \in \mathcal{P}_{si}$, for any $h \in \{1, \ldots, K-1\}$.

It follows from the definition 3.1 that there are no alternatives to p_K, since p_K is the lexicographic greatest path.

Definition 3.2 Let p_h be a path of $\mathcal{P}_{si}$. A node $x \in p_h$ is a **used node** if there exists some path $p_k \in \mathcal{P}_{si}$ such that $x \in p_k$ and $k < h$.

Let $\mathcal{P}_{si}^N = \{p^1, \ldots, p^k\}$ be the lexicographic ordered set of nondominated paths from $s \in \mathcal{N}$ to $i \in \mathcal{N} - \{s\}$, that is:

1. $c(p^h) \prec_L c(p^{h+1})$ or $c(p^h) = c(p^{h+1})$, $\forall h \in \{1, \ldots, k-1\}$,
2. p is a dominated path, $\forall h \in \mathcal{P}_{si} - \mathcal{P}_{si}^N$.

Definition 3.3 $p^{h+1} \in \mathcal{P}_{si}^N$ is the **nondominated alternative** to $p^h \in \mathcal{P}_{si}^N$, for any $h \in \{1, \ldots, k-1\}$.

We may conclude from definition 3.3 that there are no nondominated alternatives to p_k.

Because of its superiority, corroborated by the best known theoretical complexity ($O(K \times |\mathcal{A}|)$ in a worst case analysis) and the reported computational experiments, the ranking shortest paths algorithm of Azevedo et al [2, 3, 4] is used to determine the alternative q_{sx} of p_{sx}. In order to clarify the presentation of the new algorithm, this ranking shortest paths algorithm it is briefly described below.

For any $j \in \mathcal{N}$, let $P(j) = \{i \in \mathcal{N} \mid (i, j) \in \mathcal{A}\}$ be the set of predecessor nodes of j and $B(j) = \{(i, j) \in \mathcal{A} \mid i \in P(j)\}$ be the set of incoming arcs of node $j \in \mathcal{N}$.

The first step of the algorithm is the determination of the lexicographic shortest tree rooted at s, that is, a tree which is formed by a lexicographic shortest path from s to i, for every node $i \in \mathcal{N} - \{s\}$. A labelling algorithm is used in this first step. As a consequence, a label is assigned to each node x being $[\pi^1, \ldots, \pi^r]_{(x)}$ its first field, which denotes the value of p_{sx}, the path from s to x in the tree; that is, $\pi^k = c^k(p_{sx})$, for $k \in \{1, \ldots, r\}$. An important characteristic of this ranking paths algorithm is the enlargement of the set of nodes and the set of arcs, in such a way that the label of each node, with the exception of node t, is associated with the lexicographic shortest path from s to that node. Obviously, a correspondence is established so that the alternative of p_{sx} can be determined as a lexicographic shortest path from s to some node x', corresponding to x. Moreover, all the nodes that do not belong to the initial network are alternatives for well determined nodes. From now on, we

write alternative of some node with the meaning of an alternative to the shortest path from s to that node.

Let $p^i \in \mathcal{P}^N$ be the i^{th} lexicographic shortest path from s to t in $(\mathcal{N}, \mathcal{A})$, and let us assume that $p^i = \{s \equiv s_0, a_1, s_1, \ldots, s_{k-1}, a_k, s_k \equiv t\}$, where $a_x = (s_{x-1}, s_x) \in \mathcal{A}$. In the ranking paths algorithm, for each node in p^i, but the initial node s, the alternative node is determined, labelled and added to the network. An enlarged network is constructed such that each new node and its label are associated with the alternative path from s to the corresponding node in the original network.

Let $s_x \neq s_0 (= s)$ be some unused node of p^i; that is, s_x does not belong to some previously determined path from s_0 to t, and the determination of at least one path from s_0 to s_x is possible. The alternative to $p_{s_0 s_x}$, the sub-path of p^i from s_0 to s_x, is determined according to the procedure use (s_x).

procedure use (s_x)

begin

if $(s_x \neq t)$

then begin s_x is an used node;

$\mathcal{N} \leftarrow \mathcal{N} \cup \{s'_x\}$;

$Ps'_x \leftarrow P(s_x) - \{s_{x-1}\}$;

if $(s'_{x-1} \in \mathcal{N})$ **then** $P(s'_x) \leftarrow P(s'_x) \cup \{s'_{x-1}\}$;

$B(s'_x) \leftarrow \{i, s'_x) \mid i \in P(s'_x)\}$;

compute the label of s'_x;

end

else begin $P(t) \leftarrow P(t) - \{s_{x-1}\}$;

if $(s'_{x-1} \in \mathcal{N})$ **then** $P(t) \leftarrow P(t) \cup \{s'_{x-1}\}$;

$B(t) \leftarrow \{(i, t) \mid i \in P(t)\}$;

compute the label of t;

end

end

A correspondence between arcs, that is, between the elements of $B(s_x)$ and $B(s'_x)$, is established where $c(i, s'_x) = c(i, s_x)$ holds for any $(i, s'_x) \in B(s'_x)$. A lexicographic shortest label of s'_x is determined from the permanent label of the nodes in $P(s'_x)$, as follows:

$$[\pi^1,\ldots, \pi^r]_{(s'_x)} = \underset{h \in P(s'_x)}{\textbf{lexmin}} \{[\pi^1,\ldots, \pi^r]_{(h)} + [c^1(h, s'_x),\ldots, c^r(h, s'_x)]\},$$

where **lexmin** stands for the lexicographic minimization. Since it is obtained from permanent labels, $[\pi^1,\ldots, \pi^r]_{(s'_x)}$ is the value of a lexicographic shortest path from s to s'_x, whence it is permanent. Moreover, it is the value of the $(i+1)^{th}$ lexicographic shortest path from s to s_x.

More details concerning this ranking paths algorithm can be found in [2, 3, 4].

In the algorithm presented in this paper we write **l.s.p.** to mean the lexicographic shortest path from s to t; **l.s.t.** is used to denote a lexicographic shortest tree rooted at s. We also use stack to represent a given set of nodes $x \in \mathcal{N}$ such that the existence of a nondominated alternative path from s to x has to be considered; moreover first denotes the first unused node in stack; by s_{y+1} we mean the node following s_y in stack. delete(x) is a procedure used to delete x and B(x) from $(\mathcal{N}, \mathcal{A})$.

The stopping condition may be stated in such a way that the possible non existence of nondominated paths is assured. Details about the value of the paths bounds in the ranking paths algorithm may be found in [5, 6, 11].

We must remark that some dominated node x' may belong to $(\mathcal{N}, \mathcal{A})$ until a nondominated alternative of node x is determined. In this case, the node x' is followed in stack by a nondominated node x whose nondominated alternative we intend to find, since stack takes the value of {nodes of $q_{sx'}$}$\cup${nodes of p_{xt}}.

```
The algorithm:
begin
    compute the l.s.t. and let p be the l.s.p.;
    P^N ← {p};
    all the nodes but s are unused;
    while (a stop condition is not verified) do
        begin stack ←{nodes of p};
            x ← first;
            while (x ≠ t) do
                begin use (x);
                    q_sx' ← alternative to p_sx;
                    q̄_sx ← correspondent of q_sx' in the original
                            network;
                    if (p̄_sx ∈ P^N_sx)
                    then begin s_y ← node following x in
                                        stack;
                                if (x ≠ s'_y)
                                then x ← s_y
                                else begin delete (x);
                                            x ← s_y+1
                                     end
                         end
                    else begin stack ← {nodes of q_sx'} ∪
                                        ∪{nodes of p_xt};
                                x ← first
                         end;
                end;
            use (t);
            p_st ← alternative to p;
            p̄_st ← correspondent of p_st in the original network;
            if (p̄_st ∈ P)
            then P^N ← P^N ∪ {p̄_st};
            p ← p_st;
        end
end
```

In order to clarify the algorithm, a small example is depicted in Fig. 1. The lexicographic shortest tree as well as the lexicographic shortest label of each node are also shown in the figure.

Since $p^1 = \{1, (1,3), 3, (3,5), 5\}$ is the lexicographic shortest path from s = 1 to t = 5, then stack = {1, 3, 5}, first = 3 and a nondominated alternative to $p_{13} = \{1, (1,3), 3\}$ is determined (if such alternative exists). The node 3' and B(3') = {(2, 3')} are added (perhaps temporarily) to $(\mathcal{N}, \mathcal{A})$ and the lexicographic shortest label $(3, 14, 2)_2$ of 3' is computed. Clearly,

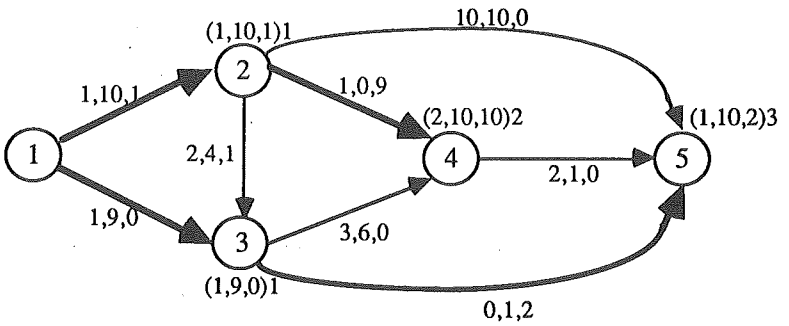


Figure 1

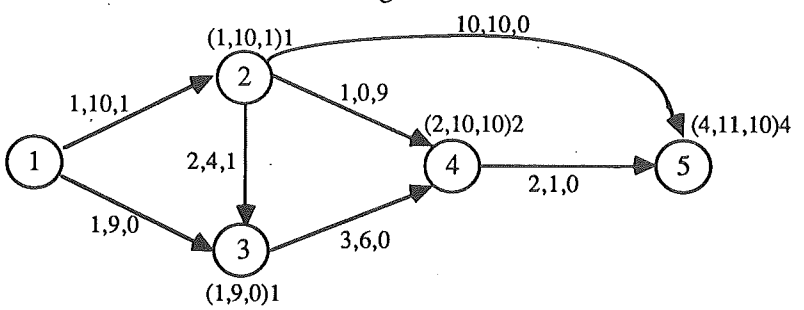


Figure 2

this label is associated to the path $p_{13'} = \{1, (1,2), 2, (2,3'), 3'\}$ which corresponds to the path $\bar{p}_{13} = \{1, (1,2), 2, (2,3), 3\}$ in the original network. Since $p_{13} \mathbf{D} \bar{p}_{13}$, then $\bar{p}_{13} \notin \mathcal{P}_{13}^N$ and $\bar{p}_{13}$ is not a subpath of some nondominated path from s = 1 to t = 5. So, stack = {nodes of $p_{13'}$}$\cup${nodes of p_{35}} = {1, 2, 3', 3, 5}, where node 3' (whose label corresponds to a dominated path) is followed by the nondominated node 3. From this stack, we conclude that there are no alternatives for both nodes 2 and 3'. As a consequence, 3' and B(3') are removed from $(\mathcal{N}, \mathcal{A})$. In fact, from the algorithm, it would not be necessary to add node 3' and B(3') to $(\mathcal{N}, \mathcal{A})$,

once we concluded the non existence of any nondominated alternative for node 3. Since 3' is followed by 3 in stack then first = 5 and x = 5 is the next node to be used. So, as B(5) = {(2,5), (3,5), (4,5)}, (3,5)∈ p and no alternative exists for node 3, the result of use(5) is the removal of (3,5) from $(\mathcal{N}, \mathcal{A})$ and the updating of B(5) as B(5) = {(2,5), (4,5)} – (see Fig. 2).

The new label of node 5 is $(4, 11, 4)_4$, which is dominated by its lexicographic shortest label. As a consequence, stack = {1, 2, 4, 5} and first = 4 since it is the first unused node in stack for which a nondominated alternative may exist. Node 4' is added to $(\mathcal{N}, \mathcal{A})$ as well as B(4') = {(3, 4')} – (see Fig. 3).

Note that B(4') = B(4) – {(2, 4)}, since (2, 4)∈ p_{14} and 2'∉ $\mathcal{N}$. $(4, 15, 0)_3$ is the lexicographic shortest label of node 4' to which corresponds the path $p_{14'}$ = {1, (1,3), 3, (3, 4'), 4'}, that is, {1, (1,3), 3, (3, 4), 4} in the original network which is a nondominated path from s = 1 to 4. So, node 5 is the next node in stack and B(5) is updated to {(4', 5), (2, 5)} (see Fig. 4) and $(6, 16, 10)_{4'}$ is its new label.

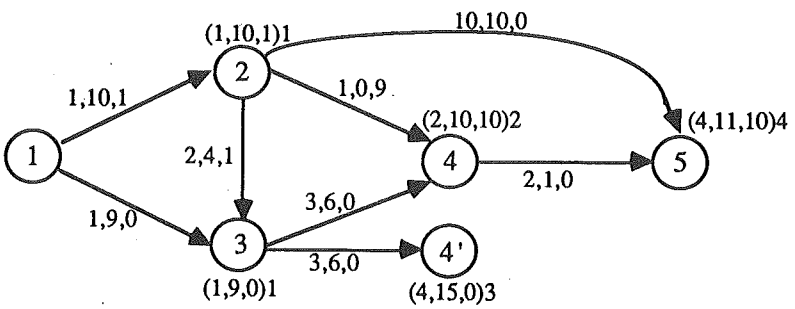


Figure 3

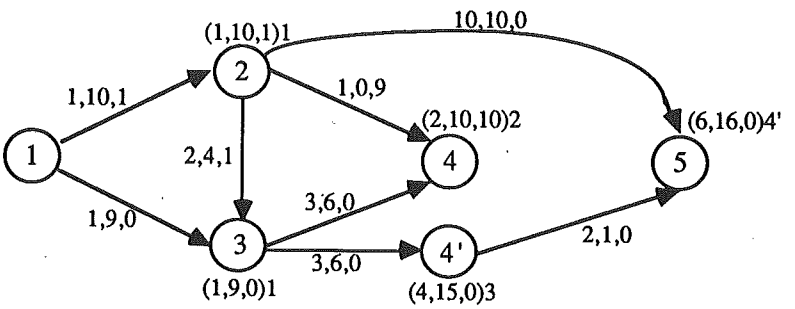


Figure 4

In the actual network p = {1, (1,3), 3, (3, 4'), 4', (4', 5), 5} is the lexicographic shortest path from s = 1 to t = 5. Its corresponding path in the original network is p^2 = {1, (1,3), 3, (3, 4), 4, (4, 5), 5} which is a nondominated path. So, the algorithm follows with stack = {1, 3, 4', 5} and node 5 as the first unused node in stack for which a nondominated alternative may exist. Then B(5) is updated to B(5) = {(2, 5)} and $(11, 20, 1)_2$ is the lexicographic shortest label of node 5 in the current network which is a dominated label. Since no more paths from node s = 1 to node t = 5 can be found in the network, the algorithm stops.

Next we provide an example showing that the algorithm cannot be applied to networks with cycles. Let us consider a network such that N={1,2,3,4,5} and

$\mathcal{A}$	(1,2)	(1,3)	(2,4)	(3,2)	(4,2)	(4,5)
c_{ij}^1	1	2	1	2	1	1
c_{ij}^2	2	1	2	1	2	2

Clearly, stack = {1, 2, 4, 5} since p = {1, (1,2), 2, (2, 4), 4, (4, 5), 5} is the **l.s.p.** for which we need to determine an alternative. So, the node 2' is added to $\mathcal{N}$ together with arcs (3, 2') and (4, 2'). Moreover, $c_{32'}^1 = c_{42'}^2 = 2$ and $c_{32'}^2 = c_{42'}^1 = 1$. Since $(3, 6)_4$ is the shortest label of 2' to which corresponds the path {1, (1,2), 2, (2, 4), 4, (4, 2'), 2'} which is dominated by the shortest path {1, (1, 2), 2} from 1 to 2 , stack is updated to stack = {1, 2, 4, 2', 2, 4, 5}. So, we would have to find again an alternative to node 2 and stack would be updated to stack = {1, 2, 4, 2", 2, 4, 2', 2, 4, 5}. Clearly, the algorithm would not be finite, because an alternative to node 2 would have to be determined again and again.

4. Computational Results

In a previous work [1] the superiority of the algorithm described in this paper over the labelling algorithms of Hansen [9] and Martins [10] was shown. In this section we compare the proposed algorithm with the procedure of Clímaco and Martins for the bicriterion case. The computational experiments were carried out on a Dec System 3100 (14 Mips) computer with

16Mb of RAM. Both algorithms were coded in FORTRAN 77 and the same routines were used whenever possible.

For a given number of nodes and arcs, an acyclic network is defined in such a way that:

1. $(i, i + 1) \in \mathcal{A}$, for any node $i \in \mathcal{N} - \{|\mathcal{N}|\}$,
2. $(i, j) \in \mathcal{A} \Rightarrow i < j$.

For the first objective the arc distances c_{ij}^1 were randomly generated in the range [1, 1000]. For the second objective, the arc distances c_{ij}^2 were also randomly generated in the same range, unless $c_{ij}^1 \leq 250$ or $c_{ij}^1 \geq 750$. In the first case $c_{ij}^2 \in [750, 1000]$ and in the second case $c_{ij}^2 \in [1, 250]$. The uniform probability distribution was used to randomly generate all the required values.

For 1000, 3000 and 5000 nodes and densities (ratio between the number of arcs and the number of nodes) in the set {2, 2.5, 3, 3.5,..., 7.5, 8} one hundred of instances were solved with both algorithms, considering each of the objectives as the first one, that is, considering each objective as the basis for the ranking procedure. So, each instance was solved four times.

All the networks were stored only in the reverse star form [7] since it is more appropriate for the ranking procedure [4]. Furthermore, since $(\mathcal{N}, \mathcal{A})$ is acyclic, the determination of the shortest tree is also possible using this reverse form representation and a straightforward labelling algorithm which requires a single scanning over the set of arcs.

All the problems were executed when the demand for the computer use was comparable and no attempt was made to exploit the hardware characteristics of the computer.

The computational results are shown in tables 1 and 2. Table 1 reports the results when the first objective was used in the ranking procedure, while table 2 reports similar results for the second objective. In both tables we report the number of nondominated paths that were determined by each one of the algorithms, the mean time (in seconds) and the number of problems that were solved completely, i.e., until the determination of all the nondominated paths. The times were obtained with the internal clock of the computer, which is accurate under two milliseconds. They include only the elapsed time after the input of the problem and prior to the output of its solution.

The presented results show the great superiority of the proposed algorithm over the procedure of Clímaco and Martins (CM) for the bicriterion case. In fact, not only the proposed algorithm runs faster, but it also determines more nondominated paths than the CM algorithm. In fact, for all the problems, the proposed algorithm determined the entire set of nondominated paths while the bicriterion algorithm of Clímaco and Martins finishes many times before the complete determination of the set of nondominated paths. This fact is a consequence of the number of nodes and arcs added to the original network by the use of the ranking path procedure. However, this procedure is used in both algorithms. Moreover, the maximum number of nodes and the maximum number of arcs are equal in both algorithms. So, as the number of nodes and the number of arcs have a smaller increase with the proposed algorithm, it is obvious that larger problems can be solved completely.

Another conclusion seems to be pertinent. It concerns the great stability of this new algotrithm regarding the use of either the first or the second objective in the ranking procedure. In fact, we do not observe great differences in the results when the second objective replaces the first one in that procedure. This is not the case of the bicriterion algorithm of Clímaco and Martins for which great differences are obtained either in the mean execution time or in the number of problems that eventually terminated. A significative difference can be seen even in the number of nondominated paths that were found for each class of problems. These facts are graphically depicted in figures 6 and 7, for a new set of computational tests where, for a given density, we report the mean execution times (taken again over 100 problems) for networks with a number of nodes in the set {2000, 3000, 4000,..., 9000, 10000}. We must also remark that the results obtained with AM algorithm seems to be almost linear.

According to the computational experience, we may conclude that this new algorithm is a better alternative for solving this particular class of multicriterion shortest path problems.

d	alg.	1000 Nodes (1)	1000 Nodes (2)	1000 Nodes (3)	3000 Nodes (1)	3000 Nodes (2)	3000 Nodes (3)	5000 Nodes (1)	5000 Nodes (2)	5000 Nodes (3)
2.0	AM	425	0.16	100	405	0.25	100	474	0.35	100
	CM	425	0.79	100	398	1.63	94	468	1.39	98
2.5	AM	462	0.21	100	614	0.44	100	567	0.64	100
	CM	447	2.45	90	574	6.49	73	520	6.64	71
3.0	AM	593	0.30	100	652	0.72	100	689	1.15	100
	CM	573	5.58	82	596	9.35	63	604	11.47	52
3.5	AM	650	0.43	100	741	1.62	100	775	1.88	100
	CM	564	9.84	58	620	13.06	39	630	12.26	42
4.0	AM	754	0.60	100	806	1.57	100	842	2.70	100
	CM	650	10.96	52	601	14.22	26	646	13.43	27
4.5	AM	748	0.79	100	901	2.29	100	908	3.82	100
	CM	614	11.76	38	612	14.83	14	671	13.78	18
5.0	AM	859	1.03	100	978	3.02	100	990	5.19	100
	CM	690	11.38	37	641	14.62	9	674	13.46	14
5.5	AM	830	1.32	100	1001	3.87	100	1039	7.01	100
	CM	629	12.05	25	613	14.57	6	693	14.06	7
6.0	AM	923	1.68	100	1078	5.06	100	1116	8.55	100
	CM	680	13.09	15	597	14.24	4	709	13.35	7
6.5	AM	972	2.04	100	1084	6.41	100	1082	10.69	100
	CM	705	12.11	15	558	14.20	1	633	12.97	7
7.0	AM	1103	2.54	100	1082	7.94	100	1180	13.61	100
	CM	697	13.06	6	523	13.67	2	654	13.30	1
7.5	AM	1031	2.92	100	1114	9.48	100	1171	16.01	100
	CM	676	12.53	9	522	13.44	1	643	12.67	3
8.0	AM	1026	3.44	100	1164	11.45	100	1243	19.37	100
	CM	684	12.32	7	532	13.07	1	648	12.50	3

d — density
alg. — algorithm
AM — proposed algorithm
CM — Clímaco and Martins bicriterion algorithm
(1) — number of nondominated paths determined
(2) — mean time (seconds)
(3) — number of problems that were completely solved

Table I - Computational results using the first objective in the ranking procedure

d	alg.	1000 Nodes (1)	1000 Nodes (2)	1000 Nodes (3)	3000 Nodes (1)	3000 Nodes (2)	3000 Nodes (3)	5000 Nodes (1)	5000 Nodes (2)	5000 Nodes (3)
2.0	AM	425	0.17	100	405	0.24	100	474	0.35	100
	CM	423	1.20	98	403	1.13	98	464	2.74	90
2.5	AM	462	0.21	100	614	0.42	100	567	0.62	100
	CM	451	2.41	93	578	6.27	73	537	5.66	80
3.0	AM	593	0.30	100	652	0.68	100	689	1.13	100
	CM	558	7.10	72	596	9.89	61	589	11.75	48
3.5	AM	650	0.42	100	741	1.09	100	775	1.81	100
	CM	587	10.26	58	641	11.95	45	633	13.23	39
4.0	AM	754	0.59	100	806	1.60	100	842	2.65	100
	CM	633	12.57	42	654	12.27	37	636	14.99	23
4.5	AM	748	0.77	100	901	2.31	100	908	3.74	100
	CM	588	13.90	26	668	13.92	19	613	14.52	16
5.0	AM	859	1.04	100	978	3.11	100	990	5.14	100
	CM	624	14.36	17	682	13.52	20	615	15.20	5
5.5	AM	830	1.32	100	1001	3.88	100	1039	6.91	100
	CM	581	14.21	16	674	13.63	10	552	14.79	3
6.0	AM	923	1.70	100	1078	5.09	100	1116	8.61	100
	CM	581	15.00	3	712	13.21	10	585	13.90	2
6.5	AM	972	2.11	100	1084	6.36	100	1082	10.69	100
	CM	582	14.31	2	687	12.16	7	633	12.97	7
7.0	AM	1103	2.58	100	1082	7.81	100	1180	13.78	100
	CM	640	13.67	4	664	12.96	4	578	13.89	1
7.5	AM	1031	3.01	100	1114	9.27	100	1171	16.32	100
	CM	582	13.35	4	660	12.56	4	531	13.52	0
8.0	AM	1026	3.62	100	1164	11.07	100	1243	19.54	100
	CM	684	13.10	3	677	12.46	1	547	13.31	0

d — density
alg. — algorithm
AM — proposed algorithm
CM — Clímaco and Martins bicriterion algorithm
(1) — number of nondominated paths determined
(2) — mean time (seconds)
(3) — number of problems that were completely solved

Table II - Computational results using the second objective in the ranking procedure

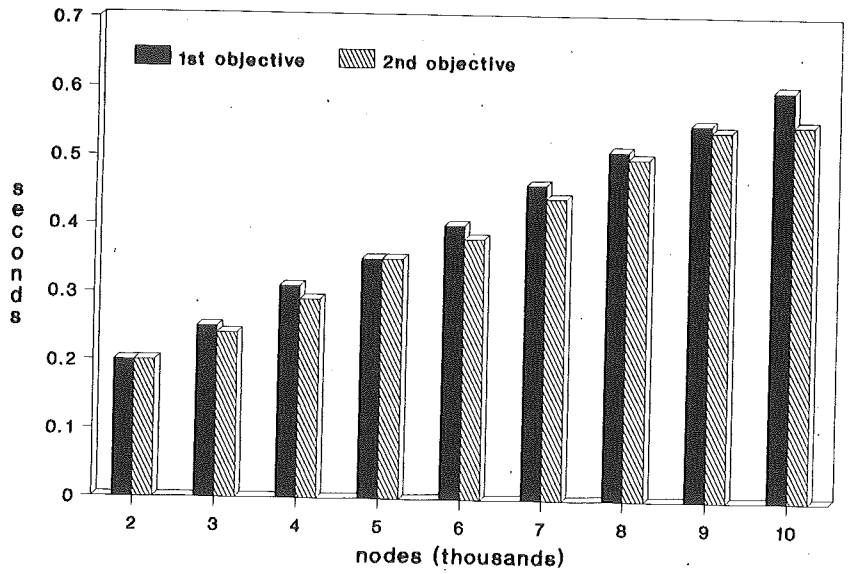


Figure 6: Results for AM algorithm. Network density = 2

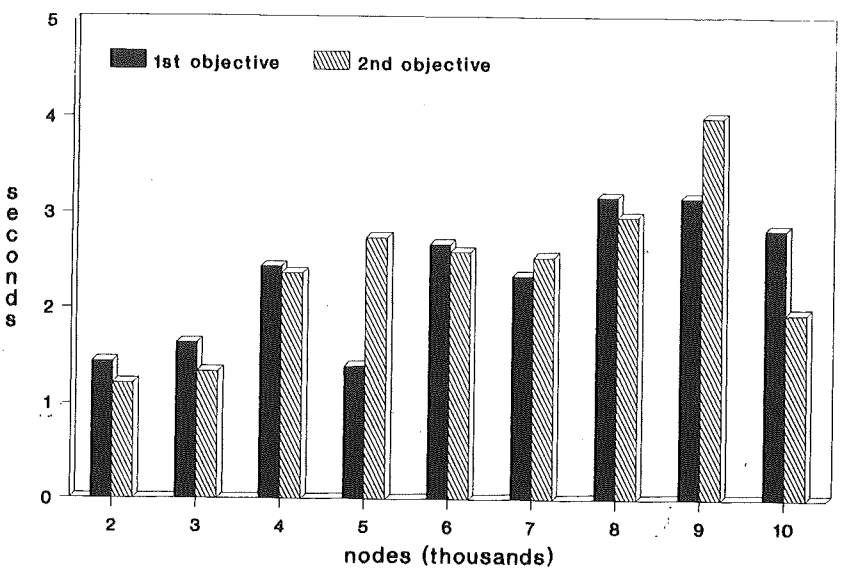


Figure 7: Results for CM algorithm. Network density = 2

Acknowledgment:

This research was partly supported by JNICT Project 87/187: "O Problema do Posicionamento no Contexto da Programação Matemática". We are also grateful to both the referees for their helpful comments and to Joaquim Madeira for his help in writing this paper.

References

[1] Azevedo, J.A. (1988) , "Alguns Algoritmos para o Problema do Caminho Mais Curto com Várias Funções Objectivo", Dissertação de Mestrado em Ciências da Computação, Departamento de Matemática, Universidade de Coimbra.

[2] Azevedo, J.A., Costa, M.E.O.S., Madeira, J.J.E.R.S and Martins, E.Q.V. (1989) , "An algorithm for the ranking of shortest paths", to appear in European Journal of Operational Research.

[3] Azevedo, J.A., Madeira, J.J.E.R.S., Martins, E.Q.V. and Pires, F.M.A. (1990), "A computational improvement for a shortest paths ranking algorithm", submitted.

[4] Azevedo, J.A., Madeira, J.J.E.R.S., Martins, E.Q.V. and Pires, F.M.A. (1990), "A shortest paths ranking algorithm", Proceedings of the Annual Conference AIRO'90, Models and Methods for Decision Support, Operational Research Society of Italy, 1001-1011.

[5] Clímaco, J.C.N., and Martins, E.Q.V. (1982), "A bicriterion shortest path algorithm", European Journal of Operational Research 11, 399-404..

[6] Clímaco, J.C.N., and Martins, E.Q.V. (1980), "On the determination of the nondominated paths in a multiobjective network problem", Proceedings of V Sympösium über Operations Research, Köln, in Methods in Operations Research, 40, (Anton Hain, Königstein, 1981), 255-258.

[7] Dial, R., Glover, G., Karney, D. and Klingman, D. (1977), "A computational analysis of alternative algorithms and labelling techniques for finding shortest path trees", Research Report, Center for Cybernetic Studies, The University of Texas at Austin.

[8] Dijkstra, E. (1959), "A note on two problems in connection with graphs", Numerical Mathematics 1, 395-412.

[9] Hansen, P. (1980), "Bicriterion path problems" in *Multiple Criteria Decision Making: Theory and Applications*, editors: Fandel, G. and Gal, T., Lectures Notes in Economics and Mathematical Systems, 177, 109-127, Springer Heidelberg.

[10] Martins, E.Q.V. (1984), "On a multicriteria shortest path problem", European Journal of Operational Research 16, 236-245.

[11] Martins, E.Q.V. (1984), "Determinação de Caminhos Óptimos em Redes Orientadas", Dissertação de Doutoramento, Departamento de Matemática, Universidade de Coimbra.

INVESTIGAÇÃO OPERACIONAL, Vol. 11, nº1 (Junho de 1991)70-79

UM ALGORITMO HEURÍSTICO DE CONSTRUÇÃO PARALELA PARA O PROBLEMA DO M–CAIXEIRO VIAJANTE

Lourival C. Paraíba
Laboratório de Dinâmica Ambiental
CNPDA, EMBRAPA
Jaguariúna, SP, Brasil

Jurandir F.R. Fernandes
Departamento de Engenharia de Sistemas
FEE, UNICAMP
Campinas, SP, Brasil

António S. Ando
Departamento de Engenharia de Sistemas
FEE, UNICAMP
Campinas, SP, Brasil

Abstract

A solution method is presented to solve the m-traveling salesman problem (m-TSP) based on a parallel tour building heuristic. Computational results concerning the behavior of the proposed algorithm are discussed. The m-TSP set includes networks with different geographical dispersion of the n-costumers, with $30 \leq n \leq 120$ and $1 \leq m \leq 10$.

Resumo

Um algoritmo heurístico para resolver o problema dos múltiplos caixeiros viajantes (m-PCV) é apresentado. Trata-se de um procedimento de construção paralela das m rotas que usa o critério de seleção e inserção mais económica para a determinação de um par cidade-rota. A característica gulosa do algoritmo é minimizada quando usamos procedimentos de trocas de cidades entre rotas.

São apresentados resultados computacionais, obtidos em micro-computadores IBM-PC/286, utilizando-se a linguagem PASCAL, em problemas com até 120 cidades e 10 caixeiros.

Keywords: routing vehicles, parallel insertion, traveling salesman.

1. Introdução

Dentre os problemas tratados pela otimização combinatória, o do caixeiro viajante (PCV) é um dos mais estudados nas três últimas décadas. Consiste em determinar uma rota de custo mínimo para um caixeiro que deve visitar n cidades e retornar ao seu domicílio. Admitindo-se que a rede seja completamente conexa, o caixeiro não deve visitar qualquer dos seus clientes mais do que uma vez. Este problema está associado a uma matriz de custo,

$C = (c_{ij})$, representando as distâncias entre as cidades. Se $c_{ij} = c_{ij}$ para todo i, j, o problema é simétrico. O PCV pode ser representado por um grafo onde as cidades são representadas pelos nós e as ligações entre elas associadas aos arcos. Muitos algoritmos exatos e heurísticos foram e têm sido sugeridos para resolver este problema (veja Bodin et al [1]).

Uma das extensões mais interessantes e aplicáveis do PCV é o problema dos múltiplos caixeiros viajantes (m-PCV): dado um conjunto de n cidades, determinar o conjunto de m rotas para m caixeiros que partem e retornam a um ponto comum, tal que cada cidade seja visitada por um único caixeiro, uma única vez, a um mínimo custo total de deslocamento.

O m-PCV pertence à classe dos problemas NP-completos. Assim, o desenvolvimento de algoritmos eficientes que construam rapidamente um conjunto de rotas de custo o mais próximo possível do mínimo, para um grande número de cidades, tem sido o objectivo de muitos pesquisadores.

Neste trabalho apresenta-se um algoritmo heurístico de construção paralela para o m-PCV que não requer sua transformação num PCV equivalente. Um refinamento da primeira solução fornecida, baseada na troca de cidades entre rotas, é também apresentado e avaliado através de testes computacionais.

2. Algoritmo

O algoritmo proposto é constituído, após a inicialização das m rotas, de três estágios: no primeiro, cada cidade não visitada seleciona uma posição económica para sua provável inserção, em cada uma das m rotas. Em seguida, cada uma destas cidades seleciona sua rota mais económica. Por fim, faz-se a inserção escolhida e redefine-se a rota alterada.

Esgotados estes estágios, propõe-se um refinamento baseado em trocas de cidades entre rotas ou ainda em trocas de sequência de cidades numa mesma rota.

O critério de inserção económica foi escolhido dentre outros, tendo em vista constatações positivas de vários autores, como Bodin et al [1] e Golden e Stewart [2], sobre a eficiência deste procedimento apresentado primeiramente por Rosenkrantz, Sterns e Lewis [3].

Será utilizada a seguinte notação:

n : número de cidades

m : número de caixeiros

c_{ij} : distância entre a cidade i e a cidade j

c_{0i} : distância entre a garagem (cidade 0) e a cidade i

N : conjunto das cidades roteirizadas

U : conjunto das cidades não roteirizadas

$R_i = (i_0, i_1, i_2, \ldots, i_{p-2}, i_{p-1}, i_p)$: i-ésima rota

onde $i_0 = i_p = 0$ e i_j com $1 \leq j \leq p-1$ é a j-ésima cidade da rota i

R : conjunto das m rotas, ou seja, $R = \{R_1, R_2, \ldots, R_m\}$

$p(u, R_i)$: posição mais económica para a inserção da cidade não roteirizada u na rota R_i

$c(u, R_i)$: custo total da rota R_i se u for inserido em R_i na posição $p(u, R_i)$.

PASSO 1: ESCOLHA DAS CIDADES SEMENTES

A escolha das cidades sementes para a construção paralela das m rotas, influe consideravelmente no desempenho das heurísticas de construção - Christofides et al [4]. Assim, adota-se o seguinte procedimento para a seleção eficiente daquelas cidades:

Faça:
começo
$k = 1$
Encontre $i_k \in U$ tal que $c_{oi_k} = \max_{i \in U} \{c_{oi}\}$
Inicialize a rota k, ou seja, faça:
$U = U - \{i_k\}$
$N = N \cup \{i_k\}$
$R_k = (0, i_k, 0)$
fim
Enquanto $1 \leq k < m$ faça:
começo
$k = k + 1$
Encontre $i^* \in U$ tal que
$c_{ii_{k-1}} + c_{ii_{k-2}} + \ldots + c_{ii_2} + c_{ii_1}$
é máximo para $i = i^*$.

Inicialize a k-ésima rota, ou seja, faça:

começo

$i_k = i^*$

$U = U - \{i_k\}$

$N = N \cup \{i_k\}$

$R_k = (0, i_k, 0)$

fim

fim

Este tipo de seleção automática de sementes, evita que todas as m rotas sejam inicializadas por sementes próximas entre si, ou sejam evita que as rotas sejam inicialmente concorrentes em um mesmo agrupamento de cidades.

PASSO 2: SELEÇÃO DE POSIÇÕES

Seja R o conjunto de todas as m rotas inicializadas no passo 1. Para cada $u \in U$ e $R_i \in R$ determine a posição $p(u, R_i)$ de inserção mais económica de u em R_i, ou seja, que minimize a expressão

$$c_{i_j u} + c_{u i_{j+1}} - c_{i_j i_{j+1}}$$

onde i_j e i_{j+1} são respectivamente a j-ésima e a (j+1)-ésima cidade da rota R_i, incluindo a garagem. Por questões de conveniência $p(u, R_i)$ é tomado como o valor j.

Neste passo foram selecionadas m |U| posições

PASSO 3: SELEÇÃO DA MELHOR ROTA PARA CADA u DADO p(u, R)

Para cada $u \in U$ selecione $R_{i^*} \in R$ tal que:

$$c_{i_j u} + c_{u i_{j+1}} - c_{i_j i_{j+1}}$$

é mínima para $R_i = R_{i^*}$ e $j = p(u, R_i)$.

Assim, para cada $u \in U$ existe $R_{i^*} \in R$ tal que a inserção de u em R_{i^*} na posição $p(u, R_{i^*})$ é a mais económica. Neste passo forem selecionados |U| pares cidade-rota.

No próximo passo será selecionado para inserção, um par cidade-rota ótimo entre os |U| pares já selecionados.

PASSO 4: ESCOLHA "PONDERADA" DO MELHOR PAR CIDADE-ROTA

Agora, é necessário que seja selecionado $u^* \in U$ e $R^*_{i^*}$ cuja inserção de u^* em $R^*_{i^*}$ é mais económica dentre aquelas obtidas no passo 3.

Seja c(u, R_{i^*}) o custo da rota R_{i^*} se u for inserido em R_{i^*} na posição $x = p(u, R_{i^*})$, ou seja,

$$c(u, R_{i^*}) = c_{i^*_x u} + c_{u i^*_{x+1}} - c_{i^*_x i^*_{x+1}} + \sum_{j=1}^{p} c_{i^*_{j-1} i^*_j}$$

onde $R_{i^*} = (i^*_0, i^*_1, i^*_2, \ldots, i^*_{p-1}, i^*_p)$ com $i^*_0 = i^*_p = 0$.

Seja $\Delta = c_{i^*_x u} + c_{u i^*_{x+1}} - c_{i^*_x i^*_{x+1}}$.

Selecione $u^* \in U$ e $R^*_{i^*}$ tal que $\Delta/c(u, R_{i^*})$ é mínimo para $u = u^*$ e $R_{i^*} = R^*_{i^*}$

PASSO 5: INSERÇÃO

Faça:

começo

$U = U - \{u^*\}$

$N = N \cup \{u^*\}$

$R^*_{i^*} = (i^*_0, i^*_1, \ldots, i^*_x, u^*, i^*_{x+1}, \ldots, i^*_{p-1}, i^*_p)$

onde $x = p(u^*, R^*_{i^*})$

fim

PASSO 6: FINALIZAÇÃO

Se $U = \emptyset$, então fim. Caso contrário volte ao passo 2.

3. Procedimentos para Melhoria da Solução

O refinamento da solução obtida pelo algoritmo, consiste em remover cidades das posições em que se encontram para outra posição que produza redução do custo total. Os algoritmos de refinamento, em geral, partem de uma solução factível e procuram melhorar sua qualidade através de uma sequência de trocas de arcos ou cidades entre as rotas. Exemplos de procedimentos deste tipo são descritos por Lim & Kernighan [5] e Waters [6].

Neste trabalho foram implementados e testados dois procedimentos de trocas simultâneas de cidades:

PROCEDIMENTO 1: TROCA DE ROTAS ENTRE DUAS CIDADES

Sejam $s_t \in N$ uma cidade localizada na rota R_s entre as cidades s_{t-1} e s_{t+1} e $k_l \in N$ uma outra cidade localizada na rota R_k entre as cidades k_{l-1} e k_{l+1}. Se

$$c_{s_{t-1}k_l} + c_{k_l s_{t+1}} + c_{k_{l-1}s_t} + c_{s_t k_{l+1}} < c_{s_{t-1}s_t} + c_{s_t s_{t+1}} + c_{k_{l-1}k_l} + c_{k_l k_{l+1}}$$

então troque s_t por k_l na rota R_s e k_l por s_t na rota R_k.

Se este procedimento for repetido para todos os pares de cidades então ele reduz a interseção entre os setores de atendimento de cada uma das rotas.

PROCEDIMENTO 2: TROCA SIMPLES MÁXIMA

Dada uma cidade $s_t \in N$ localizada na rota R_s entre as cidades s_{t-1} e s_{t+1} determinar, quando possível, uma cidade $k_j \in N$ localizada na rota R_k entre as cidades k_{j-1} e k_{j+1} tal que a expressão

$$c_{s_{t-1}s_t} + c_{s_ts_{t+1}} + c_{k_jk_{j+1}} - c_{k_js_t} - c_{s_tk_{j+1}} - c_{s_{t-1}s_{t+1}}$$

seja máxima e estritamente positiva, ou seja, dada uma cidade determinar uma posição para troca onde a inserção desta é a que provoca a melhor redução do custo total.

PASSO 7: REFINAMENTO

Seja $C_T = \sum_{i=1}^{m} C(R_i)$ o custo total de roteirização. $C(R_i)$ é o custo da rota i.

PR1: Para todo par i, $j \in N$ com $i \neq j$ aplique o procedimento 1.

PR2: Para todo $i \in N$ aplique o procedimento 2.

Repita os passos PR1 e PR2 até que o custo total não se altere entre uma repetição e outra, isto é, até que nenhuma troca tenha sucesso.

O procedimento PR1 efectua n (n-1)/2 comparações. Já o procedimento PR2 efectua n (n+m-2) comparações.

Eilon et al [7] mostraram que um procedimento r-ótimo necessita da ordem de n! (r–1)! 2^{r-1}/(r! (n–r)!) comparações, ou seja, para r = 2 tem-se n(n–1) comparações. Estes resultados mostram que tanto os procedimentos PR1 e PR2 aqui propostos, bem como o procedimento 2-opt são equivalentes em termos do número de comparações avaliadas.

4. Resultados Computacionais

O algoritmo, programado na linguagem PASCAL, foi testado em um microcomputador IBM–PC/286 com "clock" de 20 MHz. Os problemas--testes foram representados por redes onde todas as ligações entre nós foram consideradas. A distribuição espacial destes nós foi obtida por geração aleatória e as distâncias euclidianas entre eles foram calculadas, constituindo as matrizes de custos dos problemas.

O esforço computacional para se construir soluções aproximadas de problemas m–PCV varia consideravelmente de um problema para outro. Assim, foram considerados 48 grupos de problemas, variando-se o número de cidades (n = 30, 40, 50, 60, 70, 80, 90, 100) e o número de caixeiros (m = 1, 2, 4, 6, 8, 10). Para cada um destes grupos (n, m) a heurística foi aplicada sobre 5 redes diferentes, fornecendo os valores médios de tempos de processamento em segundos apresentados na Tabela 1.

CIDADES	CAIXEIROS					
	1	2	4	6	8	10
30	1.07	0.98	0.71	0.68	0.66	0.64
40	2.33	2.07	1.45	1.30	1.39	1.37
50	4.32	3.80	2.56	2.45	2.44	2.46
60	7.17	6.51	3.96	3.86	3.83	3.91
70	11.09	9.53	5.90	5.64	5.70	5.64
80	16.19	14.72	8.67	7.84	7.72	7.69
90	22.56	21.26	11.67	10.54	10.25	10.26
100	30.47	27.72	15.94	14.61	13.57	13.00

Tabela 1 (tempos em segundos de cpu)

A curva que melhor se ajusta à distribuição destes tempos é:

$$t = \alpha\, n^{2.616}\, m^{-0.327} \qquad \text{com } \alpha = \exp(-8.716)$$

indicando que a heurística de construção paralela de rotas factíveis para m caixeiros viajantes é proporcional ao cubo do número de cidades a serem percorridas. Ressalta-se que estes resultados nada afirmam ou garantem quanto à qualidade das soluções obtidas. A ausência na literatura de soluções, exatas ou não, para diferentes problemas padrões do tipo m-caixeiros torna difícil esta avaliação. Mais adiante esta questão será retomada.

Em recente trabalho de Husbam [8] apresentam-se tempos de processamento para a obtenção de soluções exatas de problemas com até 16 cidades e 4 caixeiros, também utilizando-se microcomputadores IBM-PC compatíveis. A distribuição dos tempos por segundo obtidos pode ser ajustada por:

$$t = \beta \exp(0.872\, n - 0.944\, m) \text{ com } \beta = \exp(-4.310)$$

Fixando-se um tempo de processamento em 2 horas, verificar-se-ia que a heurística forneceria rotas factíveis para um m-PCV de 1100 cidades e 10 caixeiros, ao passo que o método exato de Husbam [8] resolveria um problema com 26 cidades e 10 caixeiros.

Para problemas deste porte, o método aqui sugerido oferece soluções em menos de 1 segundo. Estes resultados confirmam o grande fosso existente entre os métodos exatos e heurísticos no tratamento de problemas reais (Waters e Brodie [9]), via de regra com um número de cidades que compromete a aplicação daqueles métodos.

Waters e Brodie [9] alertam ainda que os métodos exatos publicados têm sido baseados em procedimentos complexos, na maioria das vezes aplicados a problemas especiais com características desejáveis, e fazendo uso de possantes computadores. São condições difíceis de serem preenchidas pela grande maioria dos usuários.

Em geral, quando um algoritmo heurístico se comporta bem em termos de tempo de processamento, compromete-se a minimização da distância total percorrida pelos caixeiros.

Os resultados a seguir avaliam o refinamento proposto, buscando também uma comparação com aqueles apresentados por Christofides et al [4]. A comparação é aproximada uma vez que os problemas de Christofides et al [4] levam em conta restrições temporais e de carga, ou seja, o m-PCV pode ser obtido como relaxação do problema por eles tratado. Não há aqui comparações com os tempos de processamento, uma vez que Christofides et al [4] utilizaram computador de grande porte. Para cada exemplo daquela referência, foram considerados o melhor custo apresentado e o respectivo número de rotas (m), alocados respectivamente nas colunas (A) e (B) da Tabela 2. Ainda nesta tabela os resultados obtidos pela heurística aqui proposta são: (C)-custo sem o refinamento; (D)-custo com o refinamento; (E)-tempo de processamento sem o refinamento e (F)-tempo de processamento do refinamento, em segundos. Na Tabela 3 são apresentadas relações entre os resultados obtidos. Nota-se que após o refinamento a heurística proposta apresenta soluções com custos aderentes aos de Christofides et al [4]. Por outro lado, os tempos de processamento gastos na fase do refinamento não chegam a valores absolutos insuportáveis (pouco mais de 3 minutos para 120

cidades) apesar de, em termos relativos serem bem maiores que os tempos gastos na fase sem refinamento.

n	A	B	C	D	E	F
50	532	5	560	527	2.3	10.9
75	871	11	972	783	6.5	32.8
100	851	8	915	837	11.2	67.6
120	1066	7	1128	1092	20.3	193.3
100*	816	10	1035	821	11.9	123.9

Tabela 2

n	D/A	D/C	F/E
50	0.99	0.94	4.74
75	0.90	0.81	5.05
100	0.98	0.91	6.04
120	1.02	0.97	9.52
100*	1.01	0.79	10.41

Tabela 3

(*) rede com cidades agrupadas

5. Conclusões

O algoritmo apresentado é de natureza heurística e extremamente rápido na construção de uma boa solução factível para o m-PCV. Esta solução pode se tornar a única solução viável quando não se dispõe de recursos computacionais para se resolver um problema de grande porte em curto espaço de tempo. Isto se deve ao fato de que a construção das m rotas se realiza em tempo proporcional ao cubo do número de cidades, e por não precisar transformar o problema original num equivalente caixeiro viajante simples.

A desvantagem deste algoritmo é que resolve os problemas com o número m de caixeiros tomado fixo, ao contrário de outros procedimentos. Veja por exemplo, Laporte e Nobert [10].

O algoritmo apresentado bem como seu refinamento podem ser estendidos facilmente para o uso em diferentes problemas de roteirização de veículos onde estão presentes restrições complicantes como: demanda nos pontos de

visita; jornada de trabalho de uma tripulação; veículos capacitados e janelas de tempo.

A sua aderência às soluções apresentadas em Christofides et al [4], bem como sua versatilidade computacional devido à sua implementação em micros IBM-PC, são atributos marcantes da heurística proposta no tratamento de redes de até duas centenas de cidades, aplicando-se os dois refinamentos descritos. A obtenção de uma solução factível é bastante rápida mesmo para redes maiores.

6. Bibliografia

[1] Bodin, L., Golden, B., Assad, A. and Ball, M. (1983) – Routing and Scheduling of Vehicles and Crews, Comput. Opns. Research 10, 82-96.

[2] Golden, B.L. and Stewart, W.R. (1983) – Empirical Analysis of Heuristic, The Traveling Salesman Problem, Chapter 7, Wiley, New York.

[3] Rosenkrantz, D., Sterns, E. and Lewis, P. (1977) – An Analysis of Several Heuristics for the Traveling Salesman Problem, SIAM J.Computation 6, 563-581.

[4] Christofides, N., Mingozzi, A. and Toth, P. (1979) – The Vehicle Routing Problem, Combinatorial Optimization, Chapter 11, Wiley, New York.

[5] Lin, S. and Kernighan, B. (1973) – An Effective Heuristic Algorithm for the Traveling Salesman Problem, Op.Res. 21, 498-516.

[6] Waters, C.D.J. (1987) – A Solution for the Vehicle Scheduling Problem Based on Iterative Route Improvement, J.Opl.Res.Soc. 38, 833-839.

[7] Eilon, S., Watson-Gandy, C.D.T. and Christofides, N. (1971) – Distribution Management, Griffen, London.

[8] Husban, A. (1989) – An Exact Solution Method for the MTSP, J.Opl.Res.Soc. 40, 461-469.

[9] Waters, C.D.J. and Brodie, G.P. (1987) – *Realistic Sizes for Routing Problems*, J.Opl.Res.Soc. 38, 565-566.

[10] Laporte, G. and Nobert, Y. (1980) – A Cutting Algorithm for the m-Salesman Problem, J.Opl.Res.Soc. 31, 1017-1023.

INVESTIGAÇÃO OPERACIONAL, Vol. 11, nº1 (Junho de 1991) 80-95

DISTRIBUIÇÃO DE EMBARCAÇÕES SALVA-VIDAS POR ESTAÇÕES DE SOCORROS A NAÚFRAGOS

Pedro Pitta Barros
Faculdade de Economia
Universidade Nova de Lisboa

Abstract

Portugal is a country with a wide Atlantic coast. Instituto de Socorros a Náufragos, the national lifeboat institute, has several places (called *estações*) along the portuguese coast, whose needs differ from place to place. Portugal has also a limited number of lifeboats with different characteristics. Those lifeboats are to be assigned among *estações*. A resources allocation problem emerges. One way of solving it is through multicriteria analysis. An application of Saaty's method is presented, and four proposals of lifeboats allocation are introduced. Those proposals contemplate four possible scenarios.

Resumo

Portugal dispõe de uma extensa costa marítima. Para precaver e socorrer possíveis acidentes marítimos, o Instituto de Socorros a Náufragos possui vários locais (estações) ao longo da costa portuguesa. A estas estações deve ser afectado um número limitado de embarcações. As estações diferem nas suas necessidades e as embarcações nas características operacionais. Existe assim um problema de afectação de recursos susceptível de ser resolvido por análise multicritério. Aplicando-se um método compensatório com ponderação proposto por Saaty, foram elaboradas quatro propostas alternativas de afectação, correspondendo a outros tantos cenários base.

Keywords: Multicriteria analysis, Resources allocation.

1. Introdução

É bem sabido que Portugal dispõe de uma extensa costa marítima. Para precaver e socorrer possíveis acidentes marítimos existe, naturalmente, uma entidade específica – o Instituto de Socorros a Naúfragos (ISN). O ISN possui vários locais (estações) ao longo da costa portuguesa que constituem a base da sua faceta operacional de prestação de socorro sempre que lhe seja solicitada a actuação. Possui, por outro lado, um número limitado de embarcações salva-vidas, não sendo porém todas iguais em termos de características relevantes. Existe o problema de assegurar que a distribuição das diferentes embarcações é a que melhor satisfaz as necessidades globais. Trata-se de um problema de natureza essencialmente económica: afectar recursos escassos e susceptíveis de usos alternativos de modo a satisfazer de uma forma óptima um conjunto de necessidades igualmente importantes.

A atribuição de um salva-vidas a uma estação particular do ISN não constitui um problema trivial, pois os diferentes salva-vidas têm diferentes

características e as várias estações de socorros a naúfragos diferentes necessidades e importância. Mais, as diferentes necessidades não são comensuráveis numa mesma escala. A resolução deste problema é no entanto possível de ser obtida através da análise multicritério, que desenvolve técnicas específicas de avaliação relativa das diferentes necessidades (identificadas com critérios de decisão). Esta avaliação vai permitir a sua comparação de modo a alcançar uma distribuição de embarcações salva-vidas por estações ISN que seja óptima, dadas as necessidades e limitações de recursos existentes.

Assim, na segunda secção são apresentados os fundamentos teóricos do método de análise multicritério utilizado. Na terceira secção é feita a aplicação do método à situação em causa. Em seguida, na quarta secção, são apresentados os resultados. Finalmente, na quinta secção, é elaborada a conclusão.

2. Fundamentos Teóricos

O problema existente é fundamentalmente um problema de análise multicritério. Por análise multicritério entende-se uma situação em que se dispõe de um conjunto discreto de alternativas que são avaliadas sob diversos critérios. Estes critérios (focando atributos ou características das alternativas) são, em geral, conflituantes. Por outro lado, diferentes critérios nem sempre são comensuráveis na mesma escala.

Embora existam diferentes métodos de resolução deste tipo de problemas [1] adoptou-se, neste caso concreto, um método compensatório com ponderação. Este tipo de método, permite, como o próprio nome indica, compensações entre critérios: alterações num atributo podem ser compensadas por alterações opostas em quaisquer outros atributos.

Com os métodos compensatórios é usualmente atribuído um único número a cada caracterização multidimensional representando uma alternativa. Nos métodos compensatórios com ponderação levanta-se naturalmente a questão de como calcular os ponderadores, pois é necessário determinar os pesos $w_1,\ldots, w_n$ a associar a cada critério $c_1,\ldots, c_n$.

Na presente aplicação seguiu-se o método dos valores próprios, proposto por Saaty, pois requere apenas informação acerca da importância relativa de cada atributo. Neste método, o decisor é instado a fazer uma comparação

entre pares de critérios, segundo uma escala que representa a intensidade (importância) da comparação relativa dos critérios.

Essa escala é dada, para a comparação entre o critério c_i e o critério c_j, por:

Intensidade/ /Importância	Definição	Explicitação
1	Igual Importância	
3	Dominância Fraca	O 1º critério é ligeiramente mais importante que o 2º
5	Dominância Forte	O 1º critério é mais importante que o 2º
7	Dominância Irrefutável	O 1º critério é significativamente superior ao 2º
9	Dominância Absoluta	O 1º critério domina absolutamente o 2º

Os valores pares (2, 4, 6, 8) são atribuídos aos casos intermédios (por exemplo, quando não se decide entre 1 e 3, escolhe-se 2), e os valores inversos ($\frac{1}{3}$, $\frac{1}{5}$, $\frac{1}{7}$, $\frac{1}{9}$) correspondem à inversão da regra de dominância entre o primeiro e o segundo critérios (por exemplo, quando se escolhe $\frac{1}{9}$, o segundo critério domina absolutamente o primeiro). O mesmo raciocínio é aplicável aos inversos dos números pares.

A justificação desta escala baseia-se no facto de a experiência ter confirmado que é razoável uma escala de nove unidades e que reflecte o grau de descriminação possuído pelos indivíduos quanto à intensidade das relações entre elementos [3, pág. 77].

Torna-se agora necessário concretizar esta escala de comparação de critérios em ponderadores. Para tal, defina-se a matriz B, de elemento genérico b_{ij}, que é o julgamento relativo dos dois critérios i e j. O número de julgamentos é $\frac{n(n-1)}{2}$, para o caso de um número n de critérios.

Esta avaliação relativa deve estar relacionada com os ponderadores w_i e w_j do seguinte modo:

$$b_{ij} = \frac{w_i}{w_j} \qquad (1)$$

Pressupondo que avaliações satisfazem esta relação (1) é possível estabelecer a seguinte identidade:

$$B\,w = n\,w \qquad (2)$$

ou

$$(B - n\,I)\,w = 0 \qquad (2a)$$

onde I é a matriz identidade de ordem apropriada. Deste modo o vector dos ponderadores, w, corresponde ao valor próprio n.

Devido à propriedade de consistência (1), o sistema homogéneo de equações lineares (2a) tem apenas soluções triviais. Em geral, no entanto, os valores precisos de $\frac{w_i}{w_j}$ são desconhecidos e têm que ser estimados, e como tal a relação (1) não será completamente satisfeita. Sabe-se, por outro lado, que pequenas perturbações nos coeficientes implicam pequenas perturbações nos valores próprios.

Definindo B' como a matriz das avaliações estimada, e sendo w' o vector de ponderadores associado a B', então $B'w' = \lambda_{max}\, w'$, onde λ_{max} é maior valor próprio de B'. O vector w' pode ser obtido resolvendo o sistema

$$(B' - \lambda_{max}\, I)\, w' = 0$$

juntamente com a restrição

$$\sum_i w_i = 1$$

Note-se que, por construção, o sistema tem determinante nulo, pois λ_{max} é a maior solução de $|B' - \lambda\, I| = 0$, pelo que as n equações são linearmente dependentes. Deixando cair uma delas e adicionando a restrição de a soma dos ponderadores ter que ser unitária, obtém-se um sistema de n equações a n incógnitas susceptível de ser resolvido [1, pág, 43-44].

Devido à possibilidade de ocorrência de incoerência nas comparações humanas é importante ter uma medida da inconsistência, definida em Saaty (1986) como:

$$CI = \frac{\lambda_{max} - n}{n-1}$$

Por simulação foi determinado o valor do indicador de consistência no caso de as avaliações de critérios serem feitas de um modo puramente aleatório (CI_r^n), que pode servir de termo de comparação. Com essa base determinou-se o seu valor médio e definiu-se

$$RC = \frac{CI}{CI_r^n}$$

em que este quociente RC é considerado por Saaty como um rácio de consistência, propondo este autor a reformulação das comparações feitas quando RC exceder 10%.

O método compensatório com ponderação utilizado foi o método analítico hierárquico, em que os ponderadores foram obtidos do modo acima indicado.

Este método baseia-se no facto de os objectivos não serem todos do mesmo nível. Vão-se compor pesos, por meio do método de Saaty, para diferentes níveis, criando-se assim uma hierarquia. Estando calculados os pesos, comparam-se seguidamente as alternativas com base nos objectivos do nível mais baixo [1, pág. 106-114], [3, pág. 73-74].

3. Aplicação ao Problema em Causa

No caso presente, considera-se o problema de afectação de diversas embarcações salva-vidas por diferentes estações do Instituto de Socorros a Naúfragos (ISN), ao longo da costa portuguesa, continental e ilhas. Cada estação tem as suas especificidades e importância relativa e cada barco as suas características, que serão mais ou menos importantes consoante cada estação de socorros a naúfragos.

Assim, os critérios de primeiro nível serão as próprias estações de socorros a naúfragos, em que se irá obter então a contribuição de cada uma delas para o objectivo global, a melhor distribuição possível das embarcações por estações. Dentro de cada estação são ainda considerados quatro critérios de avaliação de cada embarcação.

As estações de socorros a naúfragos consideradas são, num primeiro caso, vinte seis (ver Quadro 1 para uma apresentação exaustiva das estações), e num segundo caso, trinta (acrescentando às anteriores vinte e seis estações as novas estações possíveis de serem criadas, de Funchal, Velas, Albufeira e Porto Santo). As estações salva-vidas consideradas operacionalmente neste estudo não constituem todas as estações ao dispor do ISN. Existem ainda as estações salva-vidas de Apúlia, S.Martinho do Porto, Vila Nova de Mil Fontes e Alhandra. Estas estações não foram incorporadas na análise devido às suas características particulares.

A estação de Apúlia serve uma zona de diminuta densidade pesqueira e está situada em costa de mar aberto, tendo-lhe sido atribuída uma embarcação alternativa a salva-vidas, que se considera suficiente para suprir as necessidades da estação.

A estação de Alhandra é uma estação fluvial situada no Rio Tejo, não fazendo sentido considerá-la para a atribuição de salva-vidas oceânicos. Esta estação possui o único salva-vidas fluvial possuído pelo ISN.

As estações de S.Martinho do Porto e de Vila Nova de Mil Fontes não foram incorporadas na análise, pois o estado das respectivas barras não

permite a operacionalidade dos salva-vidas oceânicos (o açoreamento que se verifica em ambos os portos limita bastante a capacidade de saída para o mar das embarcações). Adicionalmente, estes pontos perderam importância com a criação dos portos da Nazaré e Sines, respectivamente, para onde se deslocaram as frotas pesqueiras que antes existiam nos portos de S.Martinho do Porto e de Vila Nova de Mil Fontes. Assim, também aqui a impossibilidade de actuação, em termos gerais dos salva-vidas determinou a atribuição de outro tipo de embarcações (bote) substituta para satisfação das necessidades sentidas – fundamentalmente a vigilância das praias.

Estas quatro estações constituem, pois, um problema de afectação de recursos distinto do analisado no presente trabalho.

Os critérios associados às características de cada embarcação em cada uma das estações são a velocidade (avaliado em nós), a capacidade para aguentar mar (avaliada em altura da vaga que a embarcação aguenta), a área que pode cobrir (em milhas marítimas de raio de acção) e a posse de meios de navegação – radar (associando o valor um com a existiencia de radar na embarcação e o valor zero com a sua ausência). A distinção crucial a fazer entre as diferentes classes de embarcações no que toca a meios de navegação respeita à posse, ou não, de radar, uma vez que todas as embarcações têm actualmente agulha magnética e sonda. Por outro lado, o elemento fundamental que permite à embarcação operar mais ao largo é precisamente o radar.

O primeiro passo a ser realizado é a determinação dos ponderadores de cada estação. Estes foram calculados utilizando o método de Saaty exposto e por intermédio do software "Expert Choice". Houve, no entanto, uma dificuldade operacional, o referido software não suporta mais do que comparações entre sete critérios distintos, torneada do seguinte modo:

> classificaram-se as 26 (30) estações segundo sete categorias, e agrupando as estações de igual importância numa mesma categoria. Considerando cada categoria como um critério, foi possível determinar os ponderadores associados a cada categoria. Como dentro de cada categoria todas as estações têm a mesma importância, para obter os ponderadores globais é necessário normalizar, dividindo o ponderador de cada estação pela soma de todos os ponderadores.

O cálculo dos pesos de cada critério em cada estação de socorros a naúfragos foi realizada, sem qualquer probema, através do software "Expert Choice". Os resultados das comparações realizadas, isto é, os pesos normalizados de cada estação e os pesos de cada critério associado a cada estação individual são apresentados no quadro 2.

Com os valores constantes do quadro 1 as características relevantes de cada embarcação, ou, mais tecnicamente, a ordenação segundo cada critério, e com os ponderadores do primeiro e segundo níveis constantes do quadro 2, é possível construir as contribuições marginais (quadro 3) de cada embarcação em cada estação para o objectivo final da melhor afectação de embarcações por localizações.

Quadro 1

Classe da Embarcação	Velocidade (nós)	Capacidade para aguentar (vagas–m)	Área que pode cobrir (milhas)	Posse de meios de navegação (sim-1; não-0)
1.Wilheim-Hubotter	15	8	20	1
2.D.Carlos I	7	5	10	0
3.Vila Chã	6	3	5	0
4.Oakley	9	7	20	1
5.Waveney	15	10	20	1
6.Liverpool	7	7	20	1
7.Atlantic 21-a)	30	1	20	0
-b)	6	4	20	0

NOTA: A embarcação da classe Atlantic 21 tem um comportamento bastante diferente consoante as condições em que se desenrola a sua missão. As características apresentadas sob a denominação a) respeitam as condições óptimas de actuação que serão susceptíveis de ocorre nas seguintes estações, devido ao tipo de missão normalmente nelas envolvido: Vila do Conde, Foz do Douro, Sagres, Ferragudo, Santa Maria-Olhão, Fuzeta, Tavira e Vila Real de Santo António. As restantes estações implicam, na grande maioria dos casos, condições de actuação mais adversas, reflectidas pela avaliação b).

O ISN tem actualmente em operação salva-vidas oceânicos das classes Waveney (2), Wilheim-Hubotter (6), D.Carlos I (6), Oakley (6), Liverpool(1) e Vila Chã (3), estando prestes a ser adquiridos seis salva-vidas da classe Atlantic 21.

Quadro 2

Estações Salva-Vidas	Pesos (26 est.)	Pesos (30 est.)	Crit. 1	Crit. 2	Crit. 3	Crit. 4
1.Viana do Castelo	0.04685	0.0380	0.152	0.390	0.068	0.390
2.Esposende	0.00935	0.0087	0.375	0.375	0.125	0.125
3.Póvoa de Varzim	0.05954	0.0554	0.250	0.250	0.250	0.250
4.Vila do Conde	0.01353	0.0126	0.604	0.201	0.121	0.074
5.Vila Chã	0.00935	0.0087	0.351	0.351	0.189	0.109
6.Angeiras	0.00935	0.0087	0.351	0.351	0.189	0.109
7.Leixões	0.08834	0.0822	0.250	0.250	0.250	0.250
8.Foz do Douro	0.00935	0.0087	0.604	0.201	0.121	0.074
9.Aguda	0.00935	0.0087	0.351	0.351	0.189	0.109
10.Aveiro	0.05954	0.0554	0.250	0.250	0.250	0.250
11.Figueira da Foz	0.05954	0.0554	0.250	0.250	0.250	0.250
12.Nazaré	0.05954	0.0554	0.250	0.250	0.250	0.250
13.Peniche	0.05954	0.0554	0.250	0.250	0.250	0.250
14.Ericeira	0.00689	0.0064	0.200	0.400	0.200	0.200
15.Paço de Arcos	0.08834	0.0822	0.250	0.250	0.250	0.250
16.Sesimbra	0.05954	0.0554	0.250	0.250	0.250	0.250
17.Sines	0.02780	0.0259	0.107	0.282	0.255	0.376
18.Ferragudo	0.02780	0.0259	0.143	0.286	0.286	0.286
19.Sagres	0.02780	0.0259	0.144	0.392	0.144	0.320
20.Sta.Maria–Olhão	0.04085	0.0380	0.223	0.110	0.250	0.418
21.Fuzeta	0.00935	0.0087	0.250	0.250	0.250	0.250
22.Tavira	0.01353	0.0126	0.250	0.250	0.250	0.250
23.V.Real S.António	0.04085	0.0380	0.200	0.200	0.200	0.400
24.Ponta Delgada	0.04085	0.0380	0.250	0.250	0.250	0.250
25.Angra Heroísmo	0.04085	0.0380	0.250	0.250	0.250	0.250
26.Horta	0.08834	0.0822	0.250	0.250	0.250	0.250
27.Funchal		0.0380	0.286	0.143	0.286	0.286
28.Velas		0.0064	0.201	0.604	0.121	0.074
29.Albufeira		0.0126	0.167	0.500	0.167	0.167
30.Porto Santo		0.0126	0.250	0.250	0.250	0.250

Crit. 1 – Velocidade; Crit. 2 – Capacidade para aguentar mar
Crit. 3 – Área que pode cobrir; Crit. 4 – Posse de meios de navegação

Quadro 3.A

Classes Estações (30)	1	2	3	4	5	6	7
1	0.272	0.140	0.092	0.222	0.801	0.211	0.146
2	0.098	0.050	0.035	0.075	0.104	0.069	0.054
3	0.609	0.305	0.194	0.512	0.637	0.485	0.416
4	0.166	0.081	0.061	0.118	0.171	0.102	0.261
5	0.104	0.053	0.036	0.083	0.110	0.077	0.063
6	0.104	0.053	0.036	0.083	0.110	0.077	0.063
7	0.904	0.452	0.288	0.076	0.945	0.719	0.616
8	0.115	0.056	0.042	0.081	0.118	0.071	0180
9	0.104	0.053	0.036	0.083	0.110	0.077	0.060
10	0.609	0.305	0.194	0.512	0.637	0.485	0.416
11	0.609	0.305	0.194	0.512	0.637	0.485	0.416
12	0.609	0.305	0.194	0.512	0.637	0.485	0.416
13	0.609	0.305	0.194	0.512	0.637	0.485	0.416
14	0.067	0.035	0.022	0.056	0.072	0.054	0.044
15	0.904	0.452	0.288	0.760	0.945	0.719	0.616
16	0.609	0.305	0.194	0.512	0.637	0.485	0.416
17	0.241	0.122	0.071	0.217	0.256	0.212	0.178
18	0.270	0.137	0.081	0.240	0.285	0.233	0.266
19	0.220	0.114	0.071	0.187	0.240	0.180	0.196
20	0.366	0.175	0.111	0.311	0.375	0.294	0.448
21	0.096	0.048	0.030	0.080	0.100	0.076	0.111
22	0.139	0.069	0.044	0.116	0.145	0.110	0.161
23	0.342	0.167	0.106	0.289	0.357	0.274	0.388
24	0.418	0.209	0.133	0.352	0.437	0.333	0.285
25	0.418	0.209	0.133	0.352	0.437	0.333	0.285
26	0.904	0.452	0.288	0.760	0.945	0.719	0.616
27	0.435	0.212	0.136	0.364	0.446	0.342	0.304
28	0.066	0.036	0.023	0.055	0.074	0.052	0.039
29	0.126	0.067	0.042	0.107	0.139	0.103	0.080
30	0.126	0.067	0.042	0.107	0.139	0.103	0.080

Quadro 3.B

Classes / Estações (26)	1	2	3	4	5	6	7
1	0.292	0.151	0.099	0.239	0.324	0.226	0.157
2	0.105	0.054	0.037	0.081	0.112	0.074	0.058
3	0.655	0.328	0.208	0.551	0.685	0.521	0.447
4	0.178	0.087	0.065	0.126	0.184	0.110	0.281
5	0.112	0.057	0.038	0.089	0.118	0.082	0.068
6	0.112	0.057	0.038	0.089	0.118	0.082	0.068
7	0.972	0.486	0.309	0.817	1.016	0.773	0.663
8	0.123	0.060	0.045	0.087	0.127	0.076	0.194
9	0.112	0.057	0.038	0.089	0.118	0.082	0.068
10	0.655	0.328	0.208	0.551	0.685	0.521	0.447
11	0.655	0.328	0.208	0.551	0.685	0.521	0.447
12	0.655	0.328	0.208	0.551	0.685	0.521	0.447
13	0.655	0.328	0.208	0.551	0.685	0.521	0.447
14	0.072	0.037	0.023	0.061	0.077	0.058	0.047
15	0.972	0.486	0.309	0.817	1.016	0.773	0.663
16	0.655	0.328	0.208	0.551	0.685	0.521	0.447
17	0.259	0.131	0.077	0.233	0.275	0.227	0.191
18	0.290	0.147	0.087	0.258	0.306	0.250	0.286
19	0.236	0.123	0.077	0.201	0.258	0.193	0.215
20	0.394	0.188	0.119	0.335	0.403	0.317	0.482
21	0.103	0.051	0.033	0.086	0.108	0.082	0.119
22	0.149	0.074	0.047	0.125	0.156	0.118	0.173
23	0.368	0.180	0.114	0.310	0.384	0.294	0.417
24	0.449	0.225	0.143	0.378	0.470	0.357	0.306
25	0.449	0.225	0.143	0.378	0.470	0.357	0.306
26	0.972	0.486	0.309	0.817	1.016	0.773	0.663

A avaliação de cada alternativa, isto é, a avaliação de cada distribuição concebível de embarcações pelas diversas estações implicaria uma enumeração exaustiva, que dadas as dimensões do problema, seria bastante difícil de concretizar. Em alternativa, utilizou-se um procedimento simples. Com os ponderadores calculados é possível resolver este problema de afectação recorrendo a métodos de programação linear inteira [2]. O software

computacional utilizado foi o ZOOM/XMP, implementado no sistema VAX/VMX da Faculdade de Economia, UNL.

A formulação do problema como um problema de programação linear inteira para determinação da solução (soluções) óptima(s) tem a seguinte especificação:

$$\text{Max} \quad \sum_{i=1}^{26} \sum_{j=1}^{7} w_{ij} x_{ij} \qquad \text{onde i indica as estações e j as embarcações}$$

$$\text{s.a.} \quad \sum_{i=1}^{26} w_{ij} \leq d_j \,, \qquad \text{para } j = 1,\ldots, 7$$

$$\sum_{j=1}^{7} w_{ij} \leq b_i \,, \qquad \text{para } i = 1,\ldots, 26$$

$$x_{ij} \in \{0, 1\}$$

sendo d_j a dotação (número) inicial das embarcações do tipo j, w_{ij} a avaliação da embarcação j se atribuída à estação i, b_i o número de embarcações admitido na estação i, e x_{ij} é a variável que toma o valor unitário se a embarcação j for atribuída à estação i, e o valor nulo nos restantes casos. O primeiro conjunto de restrições exige que não seja excedido, na distribuição, o número de embarcações existentes para cada classe. O segundo conjunto de restrições exige que não seja atribuído a cada estação mais do que um número b_i de embarcações. Tipicamente tem-se $b_i = 1$, para qualquer i, isto é, cada estação não pode possuir mais do que uma embarcação.

Existem igualmente restrições adicionais sobre as alternativas possíveis, nomeadamente sobre as estações de Vila Chã e Aguda, pois estão implementadas em barras em mar aberto. As embarcações salva-vidas a serem colocadas nessas estações têm forçosamente que pertencer à categoria Vila Chã, que constituem as embarcações mais leves e susceptíveis de serem lançadas e recolhidas em cada missão que tenham de efectuar. Também Angeiras não pode suportar outro tipo de salva-vidas, já que este teria de ficar fundeado, o que é impossível dadas as características da estação de socorros a naúfragos aí existente.

As embarcações de todas as restantes classes de salva-vidas são demasiadamente pesadas para serem lançadas (recolhidas) ao (do) mar sem a utilização de carro e/ou rampa, instrumentos inexistentes nos três casos supra referidos.

Estas restrições traduzem-se, no problema de programação linear inteira formalizado, pela imposição de valores para determinados x_{ij}. Mais concretamente,

$x_{ij} = 1$, i = Angeiras(6) , Aguda(9) e Vila Chã(5) , j = Vila Chã(3)

$x_{ij} = 0$, i = Angeiras(6) , Aguda(9) e Vila Chã(5) , $j \neq$ Vila Chã(3)

$j = 1,\ldots, 7$

4. Os Resultados

O quadro 4 apresenta as distribuições obtidas, bem como o valor da função objectivo, para quatro possibilidades distintas. Por memória apresenta-se também a actual distribuição. As quatro possibilidades apresentadas correspondem a

a) considerar as vinte e seis estações iniciais e os novos barcos (6) Atlantic 21 adquiridos.

b) idêntico a **a)** excepto que as embarcações atribuídas à estação de Paço de Arcos são fixadas à priori. Esta opção prende-se com o facto de na zona de Lisboa existirem outros meios de socorro a sinistros marítimos que se poderão substituir ao ISN em caso de necessidade. Um exemplo é a Armada Portuguesa, com o navio SAR ("Search and Rescue") que tem disponível para operações de busca e salvamento ao largo da costa.

c) considerar trinta estações de socorros a naúfragos e as novas embarcações Atlantic 21.

d) idêntico a **b)**, mas trinta estações.

Contudo, em qualquer dos casos **a) e c)** não existe uma única solução óptima mas várias. Nos casos **a) e c)**, a solução seria igualmente óptima se o salva-vidas a atribuir a Paço D'Arcos fosse trocado com o atribuído à Horta ou com o destinado a Leixões, e/ou se o salva-vidas a atribuir a Sesimbra fosse trocado com o a atribuir à Póvoa de Varzim (ou Peniche, ou Aveiro, ou ainda Figueira da Foz).

No quadro 4 é apresentada, de dentro das soluções óptimas, a que mais se aproxima da distribuição corrente. Todas as propostas feitas no que segue podem ser alteradas de modo apropriado se se escolher outra solução óptima como referencial.

Quadro 4 – Os Resultados

	Salva-vidas atribuído	Salva-vidas a atribuir			
		Hip.a)	Hip.b)	Hip.c)	Hip.d)
V. Castelo	W.H.	Oakley	Oakley	Oakley	Oakley
Esposende	D.Carlos I	D.Carlos I	D.Carlos I	D.Carlos I	D.Carlos I
P. Varzim	W.H.	W.H.	W.H.	W.H.	W.H.
Vila Conde		Atlantic 21	Atlantic 21	Atlantic 21	Atlantic 21
Vila Chã	Vila Chã	Vila Chã	Vila Chã	Vila Chã	Vila Chã
Angeiras	Vila Chã	Vila Chã	Vila Chã	Vila Chã	Vila Chã
Leixões	Waveney	Waveney	Waveney	Waveney	Waveney
Foz Douro		Atlantic 21	Atlantic 21	Atlantic 21	Atlantic 21
Aguda	Vila Chã	Vila Chã	Vila Chã	Vila Chã	Vila Chã
Aveiro	Oakley	W.H.	W.H.	W.H.	W.H.
Fig. Foz	Oakley	W.H.	W.H.	W.H.	W.H.
Nazaré	W.H.	W.H.	W.H.	W.H.	W.H.
Peniche	Oakley	W.H.	W.H.	W.H.	W.H.
Ericeira		D.Carlos I	D.Carlos I		
P. D'Arcos	W.H.	W.H.	Atlantic 21 D.Carlos I	W.H.	Atlantic 21 D.Carlos I
Sesimbra	Liverpool	Oakley	W.H.	Oakley	W.H.
Sines	Oakley	Liverpool	Oakley	Liverpool	Liverpool
Ferragudo	D.Carlos I	Oakley	Oakley	Atlantic 21	Oakley
Sagres	Oakley	Oakley	Oakley	Oakley	Oakley
S.Mª Olhão	D.Carlos I	Atlantic 21	Atlantic 21	Atlantic 21	Atlantic 21
Fuzeta	D.Carlos I	Atlantic 21	Liverpool	D.Carlos I	D.Carlos I
Tavira	D.Carlos I	Atlantic 21	Atlantic 21	Atlantic 21	Atlantic 21
VRS.António	W.H.	Atlantic 21	Atlantic 21	Atlantic 21	Atlantic 21
P.Delgada	Oakley	Oakley	Oakley	Oakley	Oakley
A.Heroísmo	W.H.	Oakley	Oakley	Oakley	Oakley
Horta	Waveney	Waveney	Waveney	Waveney	Waveney
Funchal				Oakley	Oakley
Velas				D.Carlos I	
Albufeira				D.Carlos I	D.Carlos I
Porto Santo				D.Carlos I	D.Carlos I

W.H. = Wilheim-Hubotter

Não considerando a criação de novas estações do ISN, as alterações sugeridas à distribuição das embarcações pelas estações de socorros a naúfragos são importantes mas não diferem radicalmente da estrutura existente. Também não se verifica uma alteração radical com a restrição imposta por **b)**.

As principais alterações propostas, à luz dos resultados obtidos são:

i) atribuir uma embarcação da classe Oakley a Viana do Castelo, deslocando o salva-vidas atribuído a esta estação (classe Wilheim-Hubotter) para o litoral centro (Aveiro, Figueira da Foz ou Peniche);

ii) atribuição de uma embarcação Atlantic 21 a Vila do Conde, Foz do Douro, Santa Maria-Olhão, Fuzeta, Tavira e Vila Real de Santo António.

iii) as atribuições a realizar em **ii)** permitem libertar embarcações das classes Wilheim-Hubotter (1), que deveria ser distribuído para o litoral centro (Aveiro, Figueira da Foz ou Peniche) e D.Carlos I (3), que deveriam ser retirados de serviço.

iv) o salva-vidas da classe Oakley estacionado em Sines deveria ser encaminhado para Sesimbra, por troca com a embarcação da classe Liverpool atribuída a esta última estação.

v) a substituição no litoral centro dos salva-vidas da classe Oakley por salva-vidas da classe Wilheim-Hubotter permitiria a atribuição referida em **i)** e a substituição em Ferragudo do salva-vidas D.Carlos I por um da classe Oakley.

vi) atribuição à Ericeira de uma embarcação da classe D.Carlos I.

vii) Finalmente, Angra do Heroísmo deveria ceder o salva-vidas da classe Wilheim-Hubotter para o litoral centro (Aveiro, Figueira da Foz ou Peniche) por troca com uma embarcação da classe Oakley pertencente a uma dessas estações.

Atribuindo à priori a Paço D'Arcos duas embarcações (uma da classe Atlantic 21 e outra da classe D.Carlos I), a distribuição anterior modificar-se-ia apenas marginalmente:

i) a embarcação da classe Oakley estacionada em Sesimbra seria substituída por uma da classe Wilheim-Hubotter (libertada de Paço D'Arcos).

ii) a embarcação da classe Oakley dispensada por Sesimbra iria ocupar o lugar em Sines da embarcação da classe Liverpool, atribuída a Fuzeta em substituição do salva-vidas da classe Atlantic 21 destinado a Paço D'Arcos.

Contemplando a criação de mais quatro estações, mantêm-se as principais alterações já enunciadas, diferindo a afectação de embarcações apenas nas estações com menos importância (menor ponderação):

i) a estação de Velas deverá ser aberta, sendo-lhe destinada uma embarcação da classe D.Carlos I.

ii) a Ferragudo seria destinado um salva-vidas Atlantic 21.

iii) à estação de Fuzeta será atribuído um salva-vidas da classe D.Carlos I (como aliás já sucede).

iv) deveriam ser criadas novas estações em Albufeira, Funchal e Porto Santo. Funchal receberá um salva-vidas da classe Oakley, e a Albufeira e Porto Santo serão afectados salva-vidas da classe D.Carlos I.

Considerando a fixação à priori das embarcações atribuídas a Paço D'Arcos, registam-se apenas pequenas modificações, a saber:

i) a estação de Velas deveria ser desactivada, em relação à situação anterior.

ii) o salva-vidas da classe Wilheim-Hubotter libertado da estação de Paço D'Arcos passaria para Sesimbra, deslocando-se a embarcação da classe Oakley atribuída a esta estação na distribuição precedente para Ferragudo, onde ocuparia o lugar do salva-vidas da classe Atlantic 21 desviado para Paço D'Arcos .

5. Conclusões

Neste trabalho foi utilizada uma análise multicritério para determinação da melhor distribuição das embarcações salva-vidas disponíveis pelo Instituto Socorro a Naúfragos pelas suas diversas estações ao longo da costa portuguesa. Com esta abordagem procurou-se tornear a falta de uma base concreta (informação estatística devidamente discriminada e analisada) de avaliação das diferentes estações salva-vidas pertencentes ao ISN, nomeadamente o número de sinistros ocorridos e a actuação efectiva dos salva-vidas (informação não disponível). A valorização feita assentou na avaliação qualitativa dos diferentes aspectos relevantes.

São elaboradas quatro propostas de afectação. Naturalmente, são apenas propostas baseadas na informação recolhida podendo existir outras considerações que justifiquem alterações na distribuição de embarcações. De qualquer modo, considerações adicionais de vária ordem traduzem-se usualmente por restrições sobre as alternativas possíveis. Estas restrições adicionais são facilmente incorporáveis. Assim, o principal objectivo deste trabalho é o de servir de referencial e suporte à tomada de decisão real, e não substituí-la, permitindo uma identificação clara das motivações da decisão tomada.

Agradecimentos

O presente trabalho foi realizado no âmbito da disciplina de Optimização e Análise Económica I, Pós-Graduação em Economia, Faculdade de Economia, Universidade Nova de Lisboa, 1989-90. Expresso os meus agradecimentos ao Prof.Doutor Dias Coelho pelos ensinamentos e esclarecimentos prestados. Agtradeço igualmente toda a colaboração prestada pelo Instituto de Socorros a Naúfragos, na pessoa do Chefe da Divisão Ténica, Capitão de Mar-e-Guerra Virgílio Roma Pitta Barros. Todas as opiniões, erros e omissões são da exclusiva responsabilidade do autor.

Bibliografia

[1] Hwang, C. e Yoon, K (1981) – Multiple Attribute Decision Making: Methods and Applications - A State-of-the-Art Survey, Springer-Verlag, Lecture Notes in Economics and Mathematical Systems, Vol. 186.

[2] Murtagh, B.A. (1981) – Advanced Linear Programming: Computation and Practice, McGraw-Hill.

[3] Saaty, T (1986) – Decision Making for Leaders: the Analysis Hierarchy Process for Decisions in a Complex World, University of Pittsburg.

INVESTIGAÇÃO OPERACIONAL, Vol. 11, nº1 (Junho de 1991) 96-107

RELATIONAL DATABASES IN REGIONAL MANAGEMENT: TWO CASE STUDIES

J.D.Coelho
Faculdade de Economia
Universidade Nova de Lisboa
Tv. Estevão Pinto, Campolide
1000 Lisboa, Portugal

Apolinário, I
Comissão de Coordenação da
Região de Lisboa e Vale do Tejo
Rua Artilharia um, 33
1200 Lisboa, Portugal

Monteiro, R.S.
Faculdade de Economia
Universidade Nova de Lisboa
Tv. Estevão Pinto, Campolide
1000 Lisboa, Portugal

Tábuas, B.G.
Comissão de Coordenação da
Região de Lisboa e Vale do Tejo
Rua Artilharia um, 33
1200 Lisboa, Portugal

Abstract

In this paper our experience in the design and implementation of a relational database with information at local authority level for financial and planning purpose, as well as a database for the evaluation and monitoring of FEDER-financed projects are described.

These databases are implemented using two different database management systems (RDB/VMS and ORACLE). An effort for evaluating both approaches is accomplished.

Resumo

Neste trabalho relata-se a experiência dos autores na concepção e desenvolvimento de uma base de dados com informação ao nível local para planeamento e controlo financeiro e de uma base de dados para acompanhamento e avaliação de projectos financeiros pelo FEDER.

Estas bases de dados foram desenvolvidas usando dois sistemas de gestão de bases de dados (EDB/VMS e ORACLE). São apresentadas algumas considerações sobre a comparação das duas abordagens.

Keywords: Information management, Relational database, ORACLE, RDB/VMS.

1. Introduction

In this paper, we intend to describe our experience in the design and implementation of two databases for regional management. The first one, named project "CCRLVT/FEDER", has been developed to monitor and follow up on projects applying for funding from FEDER in the Tagus Valley and the metropolian area of Lisbon, which is the jurisdiction area of CCRLVT, a regional planning body. The second one, called "DGAA/Database for Local Authorities Information Management", has been designed to provide DGAA, the central administration Department in charge of the links with local authorities, including government financing, with on-line detailed information on these entities and their activities, to support diffusion of local and regional data, and provide tools for analysing and processing it.

A peculiar aspect of this experience is that two different database management systems are used, ORACLE and VAX/ABD, on two distinct computers, UNISYS 5000/50 and VAX 8350, with two operating system environments, UNIX and VMS, respectively.

2. The Project "CCRLVT/FEDER"

2.1. AIMS and general description

The CCRLVT - Comissão de Coordenação da Região de Lisboa e Vale do Tejo is the entity in charge of the process of candidature and follow up of projects for regional development in the Tagus Valley and the metropolian area of Lisbon, applying for financial support from FEDER, the EEC fund for regional development. The CCRLVT is an autonomous institutional body of the Ministry of Planning and Territory Administration entrusted with the coordination, in that particular regional area, of technical, financial and administartive support to local authorities and other administration departments, for the implementation of regional plans and actions aiming at regional development.

From the information management point of view the process of candidature and follow up on projects applying for funding from FEDER requires a large amount of data regarding the scope of the projects (see

Fig.1), the financial streams that they give rise to and the control of their physical and financial execution.

In view of the role entrusted to CCRLVT for the administrative and financial follow up of FEDER financed projects, the need became clear for an information system able to provide safe and dynamic management control over the diversity of documents and processes considered.

The requirements of the information management system include the ability to

- provide well defined decision support and control information
- offer automated execution of administrative procedures;
- receive slices of information that have not been foreseen, imposed by changes of the external environment of the organization or by the evolution of the routines adopted internally.

Thus, as a technological means to ensure the efficiency and effectiveness of the above-mentioned information system, it was decided to build, in a UNIX environment, a relational database supported by a database management system (ORACLE) and some system tools, and application software for users' interface and execution of specific procedures.

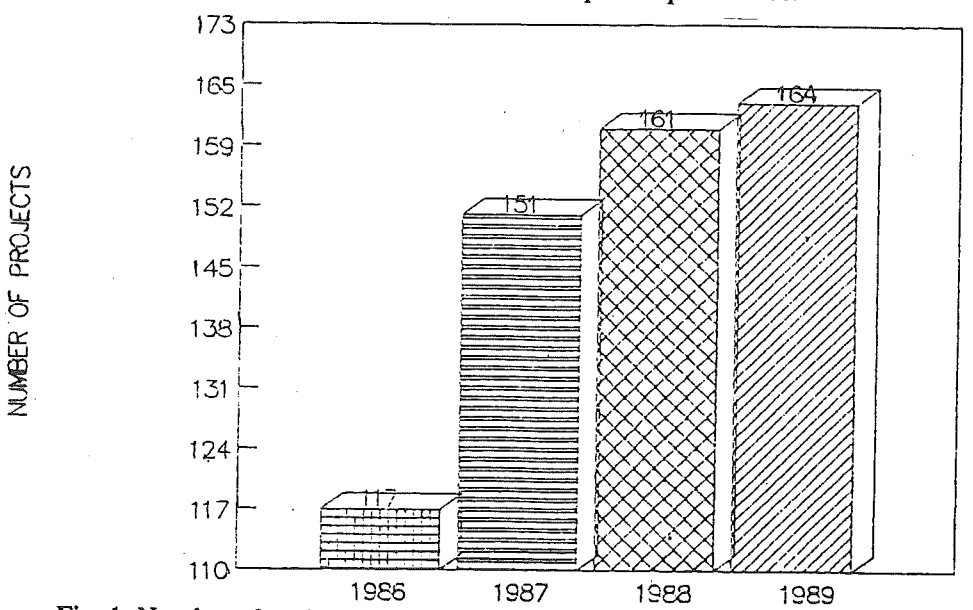


Fig. 1. Number of projects processed by CCRLVT seeking FEDER funding

2.2. The information structure

A schematic representation of the information structure is given here following the entity-relationship model (Chen, 1976) (see Fig. 2).

2.3. Users' Interfaces

2.3.1. General Development Choices

2.3.1.1. Organizational environment and adopted approach

Attending to the organizational conditions, namely those which are likely to provide a basis for a deeply-structured organic analysis and, above all, the functional instability of the object system, regarding either its transactions or their environment, this project has favoured an approach by outcome "targets", which is evident in the sequence of menus building the users' interface.

2.3.1.2. Organizational environment and adopted approach

Software

The following software tools have been used to build the application according to a module structure:

- Interactive screen generator provided by the MBMS as a direct interface to the database;
- Interactive query tools of ORACLE in conjunction with the bourne shell as a map generator;
- Q-menu to generate the tree-structured users' interface menus (Q-menu is the menu generator which is part of the office automation software Q-office);
- Procedures in the programming language C to ensure special features of data integrity, security and specific forms of output;
- Several auxiliary routines in bourne shell.

Hardware

A departmental computer UNISYS 5000/50 (microprocessor MOTOROLA 68020) with terminals SVT 1220.

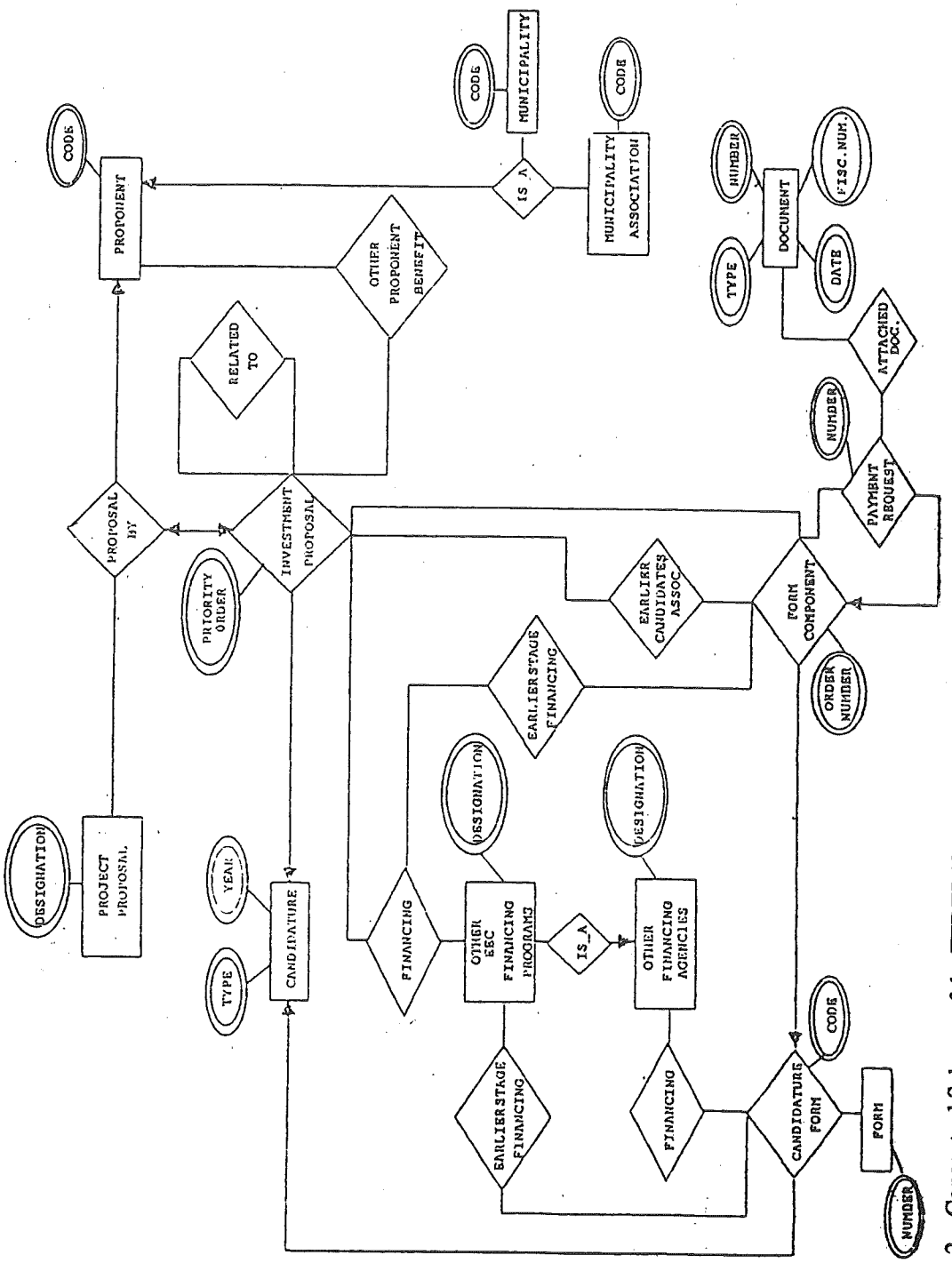


Fig. 2 – Conceptual Scheme of the FEDER/CCRLVT Database According to the Entity-Relationship Model (Chen, 1976)

2.3.2. Implemented solutions

2.3.2.1. Access to CCRLVT/FEDER application and database

. Any access to the CCRLVT/FEDER application requires a user password. Once within the main menu the user will be required to provide the password for every executable option accessing into the database, unless he chooses to fix his username/password through an utility menu "PASS" (options password) for the session. The user may change his own password with the help of the same menu. In all circumstances, the username/password that has been provided regulates the execution of every available program, even the viewing of the corresponding menu, and the operations relevant for each data set (insert, query, update, delete, etc.) according to the protection rules that have been set up by the database administrator.

2.3.2.2. Hierarchic menu structure

The menus have been organized in a hierarchical way and each one is identified by a desingnation and/or by an index number in the hierarchical tree (this is always present in the upper right corner of the screen).

2.3.2.3. Locating options in the menus

By means of an utility denoted "ONDE" (where) it is possible to obtain the list of options available in the set of menus, that match a word or a sequence of words, including wild characters, in the option title. In addition, there is an utility that allows to move directly to any menu by providing its index number.

2.3.2.4. Natural language as a reference paradigm

In the interface design, natural language has come out as a paradigm for the definition of users choices whenever it was reasonable to be inplemented. For those situations in which the input of a sequence of characters is not providing a meaning in correspondence with the user choice, then messages have been provided to translate the consequences of the options, that the user may confirm or correct alternatively.

2.3.2.5. SQL by menu

In addition to the SQL interface provided by the DBMS, a number of interactive tools for automatic definition of SQL statements have been

provided in the menu structure. This approach has been adopted in flexible 'parameterised' programs for output specification and, mainly, for the inclusion of constraints in given relevant domains.

In this last instance, it has been decided either to provide conventional menus for selection of constraint modes (equality, disequality, match with wild characters, etc.) and value assigning or to offer sequences of item oriented menus adding restraints to the selection domain. These options have been taken in function of the domains under consideration.

The constraints are articulated in logical conjunction and disjunction in accordance with the context in which they are set up.

2.3.2.6. Accessing to the operating system

It is implemented a function that provides access to operating system commands and the ability to start autonomous procedures without having to give up the application and repeating the entry commands.

2.3.2.7. Other utilities

Besides the utilities already mentioned, there is available access, at any point in the menu tree, to the following functions:

- querying the data dictionary
- on-line help
- clock
- current monthly calendar
- perpetual calendar, allowing to view any monthly or yearly calendar in Christian era.

2.3.2.8. Iconic representation

In spite of having no mouse, the pre-definition of keys associated with symbols/designations of options in the screen has conceded some form of iconic common options, utilities and value assigning to functional parameters (as, for example, the parameter that allows to select permanently a given candidature) up to the point in which the structure of an output is finished.

3. The Project "DGAA/DATABASE for Local Authorities Information Management"

3.1. AIMS and general description

The DGAA - Direcção Geral de Administração Autárquica is a central administration Department acting as a link between local authorities and central government. In particular, the financing of local authorities is managed through this Department. To that end, a large amount of data regarding geographical and physical characteristics of municipalities, infrastructures, population and housing, local authorities activities and finance must be collected.

The need for collecting and analysing a large amount of data calls for the implementation of a relational database. The availability of a database in a central system widens the scope of work of the technical staff and managers. In addition, it offers the advantage of

- avoiding losses of information collected for general tasks;
- disseminating information that otherwise would be unknown or not available;
- making available information in magnetic media instead of sheets of paper;
- eliminating the redundant input of information;
- allowing for computer networks, including personal computers.

The advantages offered by small systems are maintained by implementing interfaces to the database. Thus, the technical staff may benefit from microcomputer software (Excel, Lotus 123, Symphony or others) with the advantage of incorporating data transferred from the database automatically.

The users may gain access to information accumulated over a much longer period of time, gathered for other studies and by other entities. The access to information is also easier since data is in a single source and user-friendly interface menus are available.

The information management system behind this relational database is VAX/RDB. The information security is enforced by the data administrator who also ensures that regular backups are kept on magnetic tape.

Data protection is another function assigned to the database administrator who provides or denies users' privileges for accessing and manipulating data in each sector or field of information.

In this particular project, the transfer of large bulks of information from other central administration departments (Serviços de Informática do Ministério da Justiça, Serviços Técnicos de Apoio ao Processo Eleitoral, Gabinete de Estudos e Planeamento do Ministério da Educação), public companies (CTT, TLP) and the National Bureau of Statistics (Instituto Nacional de Estatística) has been accomplished.

In its very beginning, the computer facilities of the Faculty of Economics, New University of Lisbon were used to launch the project. DGAA later installed its own computer facilities, where the database is now residing, which act as the central node of a small computer network connecting offices in different locations.

3.2. The information structure

In contrast with the CCRLVT/FEDER project, a more extensive and less schematic description of the information structure is given here.

Since a relational database model has been chosen, we have tried to make use of its advantages. The design of tables has been according to three criteria:

1. Perfomance as trade-off between minimization of disk space and access time;
2. Data coherency;
3. Integrity and security of data.

The integrity and security of data is important in two facets. First, there is the essential need of protecting data against corruption or erasure, either by error or vandalism (it must be kept in mind that the database is aimed at serving a large number of users) and reserving for selected users the access to certain data areas. On the other hand, the objective of decentralizing the input and maintenance of the database requires a large amount of freedom in privileges to write and modify data.

The data coherence has to be ensured first by careful coding, since different sources use distinct codes for the same entities, which may be

caused by various coding approaches or distinct code updates (for instance, when new local authorities are created). Secondly, simple and efficient procedures to control and correct the input data must be implemented, particularly for extensive and repetitive information.

In a first stage, a number of blocks of information have been defined, following a users point of view:

- General data on municipalities
- General data on counties
- Cities
- Towns
- Villages
- Demographic data
- Economic data at the local level
- Electoral data
- Facilities and services
- Municipality finances (budgets and yearly accounts)
- Proxies and ratios of local area management

Later, in parallel with data availability, tables have been defined in the database to allow loading it. The main entities in the database (municipalities, counties, cities, towns, villages, regions and associations of municipalities) have been coded following the National Bureau of Statistics (INE) code, which is based on a hierarchical ranking of three levels of local areas (distrito, concelho and freguesia) ordered alphabetically, with some add-ons originated through local-area creation after the first issue of the code, or by an internal code defined as an extension of INE Code. In addition, the regional EEC code NUT (Nomenclatura de Unidades Territoriais) is used.

3.3. Users' Interfaces

As mentioned above, it is intented that this database be a tool for technical staff enrolled in local-area management and planning, and also to be support diffusion of local and regional information.

Thus, it is required that a number of tools be available, namely those regarding

- Database control

- Loading and maintenance of the database
- Querying and data processing

The control of the database is through RDO (Relational Database Operator), a set of commands that besides allowing data querying and manipulation, provides metadata management (creation, modification and elimination of relations, fields, indexes and views), offers information protection and control of access by user, group of users and type of action (read, write, modify and erase), and gives control on information volumes.

The loading of the database is done primarily from other magnetic media (tapes, diskettes, compact tapes and other computer disks through wide-area networks) by means of specific input programs, or by direct input from surveys and other documentation through procedures that provide for input, validation and error correction. Whenever possible, the transfer from magnetic media is adopted. This has forced the development of expertise on transfer of information from VAXs, MACINTOSHes, UNISYS and IBMs. Reversely, interfaces have been developed to allow transfer to LOTUS 123, EXCEL and WORDSTAR for further information processing.

An important aim is to offer online querying of all information available in the database. This is accomplished through a menu tree that provides nearly-instant access to tables with data in every database information area, including time series whenever applicable. Examples of tables are:

- Census data for each county
- Electoral results by local area
- Municipalities' budgets
- Municipalities' yearly accounts over the last five years

These procedures allow for the copying of the table into a file for printing or for further information processing, either directly from the screen or through a specific menu for that purpose.

Users wishing to process information in statistical-econometric packages, spreadsheets or word processors, may retrieve data from the database through the menu or by using the relational database operator (RDO). This language provides for interactive querying and retrieving of data from database relations, views, and any combination of them, and offers access from users' programs in PASCAL, C, FORTRAN, COBOL and BASIC to

database fields, facilitating information processing prior to using those packages.

4. Concluding Comments

As a first stage to evaluate our experience in the design and implementation of both databases, we have to state that ORACLE and RDB/VMS have allowed us to attain similar results with equivalent efforts. Regarding data security and protection both database management systems offer high level functions ensuring our essential requirements.

The integration of Q-Menu with ORACLE in the UNIX environment has been very convenient to build users' friendly interfaces. We have not had an equivalent tool in RDB/VMS. Therefore, we have opted in this latter case to embody PASCAL routines with the screen generator FMS.

We recognize, however, that fourth generation products VAX/RALLY and VAX/TEAMDATA, that we have not available at the time, might be even a better approach.

No effort has been made to compare access times, since the computer systems were not equivalent and the results would depend on many exogenous parameters.

Aknowledgements

The authors wish to express their gratitude to CCRLVT - Comissão de Coordenação da Região de Lisboa e Vale do Tejo, and to DGAA - Direcção Geral da Administração do Território, for having helped greatly in the development of this work.

References

[1] Chen, P. (1976) – The Entity-Relationship Model: Towards a Unified View of Data, ACM TODS, 1(1).

[2] Digital (1985) – VAX Rdb/VMS - Guide to Database Design and Definition, Digital Equipment Corporation, Massachussetts.

[3] Oracle (1986) – ORACLE Database Manual for the Sperry 5000 Series Operating System, Oracle Corporation, Menlo Park, California.

IFORS 93

XIII World Conference on Operations Research
12–16 July, 1993
LISBOA, PORTUGAL

The International Federation of Operations Research Societies (IFORS) will be 34 years old in 1993. As an association of 39 national OR societies and 5 kindred societies, its purpose is the development of operations research as a unified science and its advancement in all nations of the world. One of IFORS main activities is the organization of a World Conference every three years.

The IFORS XIII Conference will look to the future and highlight new and evolving metodological advances and areas of applications that will take OR into the next century. Invited and contributed sessions will address OR's important role in shaping the future business, industry, government, educational and related scientific fields such as artificial intelligence, computer sciences, decision and information sistems, and statistics. And, recognising that "What's past is prologue", the members of the international OR community are invited and encouraged to present papers describing their current research and applications, and state-of-art reviews.

Call-of-papers and paper submission forms are available from the Conference Secretariat . The abstract should reach the Conference Secretariat by April 1, 1992 for inclusion in the Invitation Programme.

All over the world many colleagues are working under the direction of Prof. Saul Gass, Programme Committee Chairman, to achieve a Conference with high scientific and technical standards. In Portugal, the Organizing Committee is also busy making arrangements to create the right environment for a fruitful meeting. Lisbon is a city full of tradition and character and social programme will provide some pleasant surprises.

Program Committee
Chairman: Prof.Saul I.Gass
Management Science and Statistics
University of Maryland
College Park, Maryland 20742
U.S.A.
Fax–(301) 314–9157

Organizing Committee
Chairman: Prof.J.Dias Coelho
Secretariat: Faculdade de Economia,
Universidade Nova de Lisboa
Travessa Estevão Pinto
1000 Lisboa, Portugal
Fax– 351–1–3871105

INSTRUÇÕES AOS AUTORES

Os autores que desejem submeter um artigo à Investigação Operacional devem enviar três cópias desse trabalho para:

Prof. Joaquim J. Júdice
Departamento de Matemática
Universidade de Coimbra
3000 Coimbra, Portugal

Os artigos devem ser escritos em Português ou Inglês. A primeira página deve conter a seguinte informação:

- Título do artigo
- Autor(es) e instituição(ões) a que pertence(em)
- Abstract (em inglês)
- Resumo
- Keywords (em inglês)
- Título abreviado

As figuras devem aparecer em separado de modo a poderem ser reduzidas e fotocopiadas. As referências devem ser numeradas consecutivamente e aparecer por ordem alfabética de acordo com os seguintes formatos:

Artigos: autor(es), título, título e número da revista (livro com indicação dos editores), ano, páginas.

Livros: autor(es), título, editorial, local de edição, ano.

Fotografia, Montagem
Impressão e Acabamentos
Tip.Nocamil
COIMBRA

ÍNDICE

Associação Portuguesa para o Desenvolvimento da Investigação Operacional

CESUR - Instituto Superior Técnico - Avenida Rovisco Pais
1000 Lisboa - Telef. 80 74 55