An Extension of a Variant of a Predictor-Corrector Primal-Dual Method from Linear Programming to Semidefinite Programming

F. Bastos [*] [‡] A. Teixeira [*] [†]

[‡] Departamento de Estatística e Investigação Operacional
Universidade de Lisboa
Lisboa - Portugal
fbastos@fc.ul.pt

[†] Departamento de Matemática
Universidade de Trás-os-Montes e Alto Douro
Vila Real - Portugal
ateixeir@utad.pt

[*] CIO - Centro de Investigação Operacional

Abstract

We extend a variant of a predictor-corrector primal-dual method for Linear Programming to Semidefinite Programming. Two versions are proposed. One of the versions uses the HKM direction and the other the NT direction. We present the algorithms associated with these versions and the computational experience using the SDPLIB 1.2 collection of Semidefinite Programming test problems. We show that, in general, the algorithm using the HKM direction is the best and is also better than the one relative to the classical method.

Keywords: Semidefinite Programming, predictor-corrector interior point variant, HKM direction, NT direction.

1 Introduction

Semidefinite Programming, SDP, is a recent, but important, field of Mathematical Programming and, although its roots are older [3, 12], most of its remarkable advances were made in the 90's [1, 14]. With this work, we intend to obtain a predictor-corrector primal-dual interior point algorithm with better performance and more precise than the other algorithms of the same type already known.

Many interior-point methods for SDP are extensions of the methods that already exist for Linear Programming, LP [1, 14]. In [2] a variant of Mehrotra's predictor-corrector method for LP was presented and it was shown to be more efficient than the original for the class of problems studied in that work. We propose a new variant for SDP that is an extension of that one for LP.

We present two versions of the new variant. One of them uses a direction, that was independently proposed by Helmberg, Rendl, Vanderbei and Wolkowicz in [8] and Kojima, Shindoh and Hara [11] and later rediscovered by Monteiro [13], which is usually refered as *HKM direction.* The other version uses the *NT direction* introduced by Nesterov and Todd in [15]. We also compare the results of these versions and the ones obtained with the classical method.

This paper is organized as follows. In Section 2, we present some useful notions and results about Semidefinite Programming and the solution of the the *Lyapunov equation*

$$AX + XA^T = H, \tag{1}$$

with $A \in \mathcal{S}_n^{++}$ and $H \in \mathcal{M}_n$. In Section 3, the primal-dual and the predictor-corrector algorithms, bases of the new variant, are described. In Section 4, the version of the variant with the HKM direction is presented. In Section 5, the NT direction for the new variant is computed and the corresponding version of the variant is presented. In Section 6, the computational experience is described. A brief description of the used tools is given and the obtained results with the new versions and the classical method are compared.

2 Preliminaries

The set of real $m \times n$ matrices, $\mathcal{M}_{m,n}$, is a real vector space isomorphic to $\mathbb{R}^{mn}$. In this vector space we define the inner product between $A, B \in \mathcal{M}_{m,n}$ by

$$\langle A, B\rangle = \mathrm{Tr}\,(B^T A) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij},$$

where $\mathrm{Tr}\,(\cdot)$ represents the trace, which is the sum of the main diagonal entries of a matrix. The associated norm, $\langle A, A\rangle^{1/2}$, is called *Frobenius norm* and is represented by $\|A\|_F$. The particular case of the square matrices of order n will be represented by $\mathcal{M}_n$. The set of real symmetric matrices of order n, $\mathcal{S}_n = \{A \in \mathcal{M}_n : A^T = A\}$, is a subspace of $\mathcal{M}_n$ isomorphic to $\mathbb{R}^{n(n+1)/2}$. For $A, B \in \mathcal{S}_n$ $\langle A, B\rangle = \mathrm{Tr}\,(AB)$. The set of positive semidefinite matrices, $\mathcal{S}_n^+$, is a full dimensional, non polyhedral convex cone in $\mathcal{S}_n$ and induces a partial order on the set of symmetric matrices. We write $A \succeq B$ if $A - B \in \mathcal{S}_n^+$. The interior of the cone $\mathcal{S}_n^+$ is the set of positive definite matrices and is denoted by $\mathcal{S}_n^{++}$. Similarly, $A \succ B$ means that $A - B \in \mathcal{S}_n^{++}$.

Consider the matrices $A_i \in \mathcal{S}_n$, $i = 1, ..., m$ and the linear operator $\mathcal{A} : \mathcal{S}_n \longrightarrow \mathbb{R}^m$ defined by

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}.$$

The adjoint operator of $\mathcal{A}$, the operator $\mathcal{A}^T : \mathbb{R}^m \to \mathcal{S}_n$ defined by $\langle \mathcal{A}(X), y \rangle = \langle X, \mathcal{A}^T(y) \rangle$, for all $X \in \mathcal{S}_n$ and $y \in \mathbb{R}^m$, is $\mathcal{A}^T(y) = \sum_{i=1}^m y_i A_i$.

The *standard* primal-dual pair of Semidefinite Programming problems is

$$\begin{array}{llll} \text{(P)} & \min\limits_{X} & \langle C, X \rangle & \qquad\qquad \text{(D)} & \max\limits_{y,Z} & b^T y \\ & \text{s.t} & \mathcal{A}(X) = b & & \text{s.t} & \mathcal{A}^T(y) + Z = C \\ & & X \succeq 0 & & & Z \succeq 0, \end{array}$$

where $C \in \mathcal{S}_n$ and $b \in \mathbb{R}^m$.

The primal problem is (P) while (D) is the Lagrangian dual of (P). Any X ((y, Z)) that satisfy the constraints of (P) ((D)) is called a *feasible solution* of (P) ((D)). If X ((y, Z)) is a feasible solution of (P) ((D)) and $X \succ 0$ ($Z \succ 0$) then X ((y, Z)) is called a *strictly feasible solution* of (P) $((D))$.

Any primal-dual pair of feasible solutions verifies the relation $\langle C, X \rangle - b^T y = \langle X, Z \rangle \geqslant 0$, called *weak duality* property. If $\langle X, Z \rangle = 0$, then the primal-dual pair (X, Z) is optimal but the reciprocal is not true, that is, the *strong duality* property is not always true. A sufficient condition for the optimal values of (P) and (D) be the same is the existence of strictly feasible solutions for both the primal and the dual problems.

We end this section with a result that gives the solution of the *Lyapunov equation*

$$AX + XA^T = H, \tag{2}$$

with $A \in \mathcal{S}_n^{++}$ and $H \in \mathcal{M}_n$ known [10].

Let us first introduce the definition of Hadamard product [9], necessary to the next theorem.

Definition 2.1. *The Hadamard product of* $A = [a_{ij}], B = [b_{ij}] \in \mathcal{M}_{m,n}$ *is* $A \circ B = [a_{ij} b_{ij}]$.

Theorem 2.1. *Let* $A \in \mathcal{S}_n^{++}$ *be a matrix satisfying* $A = S \Lambda S^T$, $S, \Lambda \in \mathcal{M}_n$, *$S$ orthogonal and* $\Lambda = \mathrm{Diag}(\lambda_1(A), \ldots, \lambda_n(A))$, *where* $\lambda_i(A)$, $i = 1, \ldots, n$ *are the eigenvalues of A. Consider that* $H \in \mathcal{M}_n$, *then the solution of (2) is*

$$X = S[L \circ (S^T H S)] S^T,$$

with $L = [l_{ij}]$, *where* $l_{ij} = (\lambda_i(A) + \lambda_j(A))^{-1}$.

3 A primal-dual method and a classical predictor-corrector variant

The SDP algorithms presented in this paper correspond to different versions of a new predictor-corrector variant of the primal-dual method developed by Christopher Helmberg, Franz Rendl, Robert J. Vanderbei and Henry Wolkowicz [8]. In this section we will briefly describe the infeasible versions of this primal-dual method and of the classical predictor- -corrector variant.

Although it is not needed a feasible solution, the method requires the existence of an initial primal-dual pair (X^0, Z^0) satisfying $X^0, Z^0 \in \mathcal{S}_n^{++}$.

The primal barrier problem is

$$\begin{array}{lll} \mathrm{PB}(\mu): & \min\limits_{X} & \langle C, X\rangle - \mu \ln(\det(X)) \\ & \text{s.a} & \mathcal{A}(X) = b, \end{array}$$

where $\mu > 0$ is the *barrier parameter.*

For each μ, the corresponding Lagrangean is

$$\mathcal{L}_\mu(X, y) = \langle C, X\rangle - \mu\ \ln(\det(X)) + \langle y, b - \mathcal{A}(X)\rangle.$$

The first order optimality conditions for $\mathrm{PB}(\mu)$, necessary and sufficient in this case, are

$$\begin{array}{rcl} \nabla_X \mathcal{L}_\mu & = & C - \mu X^{-1} - \mathcal{A}^T(y) = 0 \\ \nabla_y \mathcal{L}_\mu & = & b - \mathcal{A}(X) = 0. \end{array}$$

Taking $Z = \mu X^{-1}$ we can rewrite these conditions in the following form

$$\mathrm{CPS}(\mu) \quad \left\{ \begin{array}{c} \mathcal{A}(X) = b \\ \mathcal{A}^T(y) + Z = C \\ XZ = \mu I \\ X \succ 0 \\ Z \succ 0. \end{array} \right. \tag{3}$$

The set of solutions of $\mathrm{CPS}(\mu)$ for $\mu > 0$,

$$\mathcal{C} = \left\{ (X_\mu, y_\mu, Z_\mu) : \mu > 0 \right\},$$

is called *central path.* Note that, for a point (X, y, Z) in the central path we obtain $\mu = \langle X, Z\rangle / n$.

Given $(X, y, Z) \in \mathcal{C}$ we want to find a search direction $(\Delta X, \Delta y, \Delta Z)$ so that $(X + \Delta X, y + \Delta y, Z + \Delta Z)$ is still in the central path. Let

$$F_\mu(p) = \left[\begin{array}{c} \mathcal{A}(X) - b \\ \mathcal{A}^T(y) + Z - C \\ XZ - \mu I \end{array} \right], \tag{4}$$

with $p = (X, y, Z)$. Using Newton's method we can obtain a search direction $\Delta p = (\Delta X, \Delta y, \Delta Z)$ by solving

$$F_\mu(p) + \nabla F_\mu(p) \Delta p^T = 0.$$

This system can be written as

$$\text{NFS}(\mu)\begin{cases} \mathcal{A}(\Delta X) & = & F_p \\ \mathcal{A}^T(\Delta y) + \Delta Z & = & F_d \\ \Delta X Z + X \Delta Z & = & \mu I - XZ, \end{cases} \tag{5}$$

where $F_p = b - \mathcal{A}(X)$ and $F_d = C - Z - \mathcal{A}^T(y)$.

Rewriting NFS(μ) we obtain

$$\begin{cases} \mathcal{A}(X\mathcal{A}^T(\triangle y)Z^{-1}) & = & -\mu\mathcal{A}(Z^{-1}) + b + \mathcal{A}(XF_dZ^{-1}) \\ \Delta Z & = & F_d - \mathcal{A}^T(\Delta y) \\ \Delta X & = & \mu Z^{-1} - X - XF_dZ^{-1} + X\mathcal{A}^T(\Delta y)Z^{-1}. \end{cases} \tag{6}$$

From the first equation we get Δy and, replacing it in the remaining equations, we get ΔX and ΔZ. The obtained ΔZ is symmetric, consequence of the second equation, but, as X and Z do not commute, this is not the case for ΔX. As the first and the third components of the next iterate must belong to $\mathcal{S}_n^{++}$ we have to obtain a symmetric ΔX. There are several ways of doing this, one of them is just using the symmetric part of ΔX, that is, supposing that $(\widehat{\Delta X}, \Delta y, \Delta Z)$ is the solution of (6) then, our search direction will be $(\Delta X, \Delta y, \Delta Z)$ with $\Delta X = (\widehat{\Delta X} + (\widehat{\Delta X})^T)/2$. This search direction is known as *HRVW/KSH/M direction*.

In each iteration, two step lengths are computed, one associated with the primal direction, α_p, and the other with the dual, α_d. Both values are obtained using the backtracking method. After computing α_p and α_d the algorithm moves from the current iterate (X, y, Z) to the next iterate

$$(X + \alpha_p \Delta X, y + \alpha_d \Delta y, Z + \alpha_d \Delta Z).$$

Brian Borchers [4] developed an infeasible predictor-corrector variant of the primal-dual method presented in [8]. We will call it *classical predictor-corrector*, PCC. In this variant, to obtain the search direction we assume that is possible to take a step of length one in the direction $(\Delta X, \Delta y, \Delta Z)$ and replace in the first three equations of CPS(μ), in (3), the current iterate (X, y, Z) by the new iterate $(X + \Delta X, y + \Delta y, Z + \Delta Z)$. The result is the system

$$\begin{cases} \mathcal{A}(\Delta X) & = & F_p \\ \mathcal{A}^T(\Delta y) + \Delta Z & = & F_d \\ X\Delta Z + \Delta X Z & = & \mu I - XZ - \Delta X \Delta Z. \end{cases} \tag{7}$$

To obtain the predictor direction, in (7) we take $\mu = 0$ and neglect the non-linear term $\Delta X \Delta Z$. Solving this system we get the triple $(\widehat{\Delta X^\circ}, \Delta y^\circ, \Delta Z^\circ)$ satisfying

$$\begin{cases} \mathcal{A}(X\mathcal{A}^T(\Delta y^\circ)Z^{-1}) & = & b + \mathcal{A}(XF_dZ^{-1}) \\ \Delta Z^\circ & = & F_d - \mathcal{A}^T(\Delta y^\circ) \\ \widehat{\Delta X^\circ} & = & -X - XF_dZ^{-1} + X\mathcal{A}^T(\Delta y^\circ)Z^{-1}, \end{cases}$$

with $F_d = C - Z - \mathcal{A}^T(y)$. As in the primal-dual case, the predictor direction $(\Delta X^\circ, \Delta y^\circ, \Delta Z^\circ)$ is obtained symmetrizing $\widehat{\Delta X^\circ}$.

The corrector direction is also obtained from (7). We use $(\Delta X^\circ, \Delta y^\circ, \Delta Z^\circ)$ to compute μ and to approximate the non-linear term $\Delta X \Delta Z$ by $\Delta X^\circ \Delta Z^\circ$. The final direction is the sum of the predictor and the corrector directions, that is, we take $(\Delta X, \Delta y, \Delta Z) = (\widehat{\Delta X^\circ} + \widehat{\Delta X^c}, \Delta y^\circ + \Delta y^c, \Delta Z^\circ + \Delta Z^c)$. Now, considering the conditions satisfied by the predictor direction we obtain the following system

$$\left\{ \begin{array}{rcl} \mathcal{A}(\widehat{\Delta X^c}) & = & 0 \\ \mathcal{A}^T(\Delta y^c) + \Delta Z^c & = & 0 \\ X\Delta Z^c + \widehat{\Delta X^c} Z & = & \mu I - \Delta X^\circ \Delta Z^\circ. \end{array} \right. \tag{8}$$

From (8) we get the triple $(\widehat{\Delta X^c}, \Delta y^c, \Delta Z^c)$ satisfying

$$\left\{ \begin{array}{rcl} \mathcal{A}(X\mathcal{A}^T(\Delta y^c)Z^{-1}) & = & -\mathcal{A}(\mu Z^{-1} - \Delta X^\circ \Delta Z^\circ Z^{-1}) \\ \Delta Z^c & = & -\mathcal{A}^T(\Delta y^c) \\ \widehat{\Delta X^c} & = & \mu Z^{-1} - \Delta X^\circ \Delta Z^\circ Z^{-1} + X\mathcal{A}^T(\Delta y^c)Z^{-1}. \end{array} \right.$$

We obtain the corrector direction $(\Delta X^c, \Delta y^c, \Delta Z^c)$ after symmetrizing $\widehat{\Delta X^c}$. The final direction is

$$(\Delta X, \Delta y, \Delta Z) = (\Delta X^\circ + \Delta X^c, \Delta y^\circ + \Delta y^c, \Delta Z^\circ + \Delta Z^c).$$

In the classical predictor-corrector algorithm, like in the primal-dual, different choices for the primal and dual step lengths are made, using the backtracking method. In each iteration,

$$\begin{array}{rcl} \alpha_p^k & = & 0.99 \sup\limits_\alpha \{\alpha \in [0,1] : \ X^k + \alpha \Delta X^k \succ 0\} \\ \alpha_d^k & = & 0.99 \sup\limits_\alpha \{\alpha \in [0,1] : \ Z^k + \alpha \Delta Z^k \succ 0\}. \end{array} \tag{9}$$

To obtain the supreme, we begin with $\alpha = 1$ and α is actualized to be 90% of the previous α, this is, $\alpha_{\text{new}} = 0.9\alpha_{\text{old}}$.

The computation of the duality measure, μ, is not only based on the quantities $\langle X, Z\rangle/n$ and $\langle X, Z\rangle/2n$ but it also uses two line searches associated with the predictor direction.

The initial solution used is

$$X^0 = 100\alpha I, \qquad y = 0, \qquad Z^0 = 100\beta I, \tag{10}$$

with

$$\begin{array}{rcl} \alpha & = & n \max\limits_{1 \leqslant i \leqslant m} \dfrac{1 + |b_i|}{1 + \|A_i\|_F}, \\ \beta & = & \dfrac{1 + \max\left(\max\limits_{1 \leqslant i \leqslant m} \|A_i\|_F, \|C\|_F\right)}{\sqrt{n}}. \end{array} \tag{11}$$

Finally, the termination criteria is

$$\begin{array}{ll} \dfrac{\|b - \mathcal{A}(X^k)\|_2}{1 + \|b\|_2} \leqslant 1.0 \times 10^{-06}; & \dfrac{\|C - \mathcal{A}^T(y^k) - Z^k\|_F}{1 + \|C\|_F} \leqslant 1.0 \times 10^{-06} \\ \dfrac{|\langle C, X^k\rangle - b^T y^k|}{1 + |b^T y^k|} \leqslant 1.0 \times 10^{-07}; & X^k \succ 0; \quad Z^k \succ 0. \end{array} \tag{12}$$

4 A new predictor-corrector variant

From the previous section, we know that the classical predictor-corrector algorithm performs four line searches by iteration, two of them after obtaining the predictor direction to estimate the duality measure and the other two after computing the final direction. The algorithm we extend from LP [2] just makes two line searches per iteration and so it has the advantage of decreasing the running time of each iteration.

The major differences between this new variant and the classical predictor-corrector for LP are:

- the predictor direction is computed as in the primal-dual method;
- it uses the same μ for both the predictor and the corrector directions;
- to obtain μ no line search is needed.

We first tried to extend this variant to SDP literally. In particular, we used $\mu = \langle X, Z\rangle/2n$, indicated in [8] as a good heuristic for the primal-dual algorithm, in the predictor and corrector steps. We verified that this option did not work in this context. We also tried some other heuristics mentioned in [2] such as

$$\mu = \frac{\langle X, Z\rangle}{\theta(n)} \qquad \text{with} \qquad \theta(n) = \begin{cases} n^2 & \text{if} \quad n \leqslant 5000 \\ n\sqrt{n} & \text{if} \quad n > 5000, \end{cases}$$

always using the same μ in both steps.

After some computational experience, we verified that we could get substantial improvements if we consider different values of μ in the predictor and corrector steps. From that computational experience we concluded that the best heuristics for μ are:

$$\begin{aligned} \mu = \mu_p &= \frac{\langle X, Z\rangle}{2.2n^2}, && \text{in the predictor step} \\ \mu = \mu_c &= \frac{\langle X, Z\rangle}{2n^{5/4}}, && \text{in the corrector step.} \end{aligned} \tag{13}$$

These expressions for the duality measure, although basically empirical, have some theoretical foundation. Let us assume this is a feasible method, that is, $F_p = 0$ and $F_d = 0$ in NFS(μ), (5). Then, at k-th iteration

$$\begin{aligned} \langle C, X^k\rangle - \langle b, y^k\rangle &= \langle \mathcal{A}^T(y^k) + Z^k, X^k\rangle - \langle \mathcal{A}(X^k), y^k\rangle \\ &= \langle \mathcal{A}^T(y^k) + Z^k, X^k\rangle - \langle \mathcal{A}^T(y^k), X^k\rangle \\ &= \langle Z^k, X^k\rangle. \end{aligned}$$

Suppose now that, instead of considering α_p^k and α_d^k, we would consider $\alpha^k = \min\{\alpha_p^k, \alpha_d^k\}$. Then, we would get

$$\begin{aligned} \langle C, X^{k+1}\rangle - \langle b, y^{k+1}\rangle &= \langle Z^{k+1}, X^{k+1}\rangle = \langle Z^k + \alpha^k \Delta Z^k, X^k + \alpha^k \Delta X^k\rangle \\ &= \langle C, X^k\rangle - \langle b, y^k\rangle + \alpha^k \left(\langle Z^k, \Delta X^k\rangle + \langle \Delta Z^k, X^k\rangle\right) + \\ &\quad + (\alpha^k)^2 \langle \Delta Z^k, \Delta X^k\rangle. \end{aligned}$$

On the other side, from the third equation of NFS(μ), (5), we have $\mathrm{Tr}\,(\Delta X^k Z^k + X^k \Delta Z^k) = \mathrm{Tr}\,(\mu^k I - X^k Z^k)$ that is equivalent to

$$\langle Z^k, \Delta X^k\rangle + \langle \Delta Z^k, X^k\rangle = n\mu^k - \langle Z^k, X^k\rangle.$$

Furthermore, using the definition of adjoint operator and the assumption of the method being feasible, we obtain

$$\begin{aligned}\langle \Delta Z^k, \Delta X^k\rangle &= \langle -\mathcal{A}^T(\Delta y^k), \Delta X^k\rangle = -\langle \mathcal{A}(\Delta X^k), \Delta y^k\rangle \\ &= \langle 0, \Delta y^k\rangle = 0.\end{aligned}$$

Then, $\langle C, X^{k+1}\rangle - \langle b, y^{k+1}\rangle = \langle C, X^k\rangle - \langle b, y^k\rangle + \alpha^k \left(n\mu^k - \langle Z^k, X^k\rangle\right)$. This allow us to conclude that

$$\langle C, X^{k+1}\rangle - \langle b, y^{k+1}\rangle < \langle C, X^k\rangle - \langle b, y^k\rangle$$

if

$$\mu^k < \frac{\langle Z^k, X^k\rangle}{n}. \tag{14}$$

Both options for μ^k in (13) satisfy (14). But, as the method is not feasible and we do not use $\alpha^k = \min\{\alpha_p^k, \alpha_d^k\}$, we can not guarantee a decrease of $\langle C, X\rangle - \langle b, y\rangle$. However, in practice, we verified that these options are much better than any other already proposed.

To implement this new variant we first used the HKM direction and the backtracking method to compute the step length, as in the classical predictor-corrector. For the backtracking method the proceeding is similar to the one described in (9), but the new α is chosen as 85% of the previous one and not 90% as it happened before. This version of the new variant can be formalized in the following algorithm:

Algorithm 1 - BHKM

```
Require: $(X^0, y^0, Z^0)$ satisfying (10) and (11);
  Take $k = 0$;
  while the termination criteria (12) is not satisfied do
    (a) Take $\mu_p^k = \sigma^k \frac{\langle X^k, Z^k\rangle}{2.2n^2}$, with $\sigma^k \in [0,1]$;
    (b) Solve system (6) for $(\widehat{\Delta X^{\circ^k}}, \Delta y^{\circ^k}, \Delta Z^{\circ^k})$, considering $\mu = \mu_p^k$;
    Symmetrize $\widehat{\Delta X^{\circ^k}}$ to get $(\Delta X^{\circ^k}, \Delta y^{\circ^k}, \Delta Z^{\circ^k})$;
    (c) Take $\mu_c^k = \sigma^k \frac{\langle X^k, Z^k\rangle}{2n^{5/4}}$, with $\sigma^k \in [0,1]$;
    (d) Solve system (8) for $(\widehat{\Delta X^{c^k}}, \Delta y^{c^k}, \Delta Z^{c^k})$, considering $\mu = \mu_c^k$;
    Symmetrize $\widehat{\Delta X^{c^k}}$ to get $(\Delta X^{c^k}, \Delta y^{c^k}, \Delta Z^{c^k})$;
    (e) Take $(\Delta X^k, \Delta y^k, \Delta Z^k) = (\Delta X^{\circ^k}, \Delta y^{\circ^k}, \Delta Z^{\circ^k}) + (\Delta X^{c^k}, \Delta y^{c^k}, \Delta Z^{c^k})$;
    (f) Compute $\alpha_p^k$ and $\alpha_d^k$ using (9) with $\alpha_{\text{new}} = 0.85\alpha_{\text{old}}$;
    (g) Take $X^{k+1} = X^k + \alpha_p^k \Delta X^k$ and $(y^{k+1}, Z^{k+1}) = (y^k, Z^k) + \alpha_d^k(\Delta y^k, \Delta Z^k)$;
    (h) Take $k = k + 1$;
  end while
```

5 The new predictor-corrector variant with NT direction

Another direction that is often used in practice and also theoretically studied is due to Nesterov and Todd and is called *NT direction* [15, 16, 17]. Thus, we decided to implement this direction in our predictor-corrector variant.

As this direction is not directly obtained, as it happens with HKM, we will present its computation. Once more, we will assume that is possible to take a step of length one in the direction $(\Delta X, \Delta y, \Delta Z)$. This way we get the system

$$\left\{\begin{array}{ccc} \mathcal{A}(\Delta X) & = & F_p \\ \mathcal{A}(\Delta y) + \Delta Z & = & F_d \\ (X+\Delta X)(Z+\Delta Z) & = & \mu I, \end{array}\right. \tag{15}$$

with $F_p = b - \mathcal{A}(X)$ and $F_d = C - Z - \mathcal{A}^T(y)$.

The NT direction is obtained using a scaling matrix to reformulate the third equation of (15), so that the solution of the new system is symmetric. Let us consider the Cholesky factorizations of X and Z, $X = LL^T$ and $Z = RR^T$, the singular value decomposition of R^TL, $UDV^T = R^TL$, and the scaling matrix $W = X^{1/2}(X^{1/2}ZX^{1/2})^{-1/2}X^{1/2}$. It can be shown that W satisfies the condition $W^{-1}X = ZW$ and that it can be written as $W = GG^T$, with $G = LVD^{-1/2}$ [17]. Then, $G^{-1}XG^{-T} = G^TZG$.

Consider now $D = G^{-1}XG^{-T} = G^TZG$, $D_X = G^{-1}\Delta XG^{-T}$ and $D_Z = G^T\Delta ZG$. Thus, the third equation of (15) is equivalent to $(D+D_X)(D+D_Z) = \mu I$. Weakening this equation, by replacing the left member by its symmetric part, we get

$$\frac{1}{2}\left[(D_X+D_Z)D + D(D_X+D_Z)\right] = \mu I - D^2 - \frac{1}{2}\big(D_XD_Z + D_ZD_X\big). \tag{16}$$

To obtain the predictor direction we take $\mu = \mu_p$ and neglect the non-linear terms in (16). This way we get a Lyapunov equation in $D_X^\circ + D_Z^\circ$ which has the symmetric solution $\mu_pD^{-1} - D$. Multiplying both members of

$$D_X^\circ + D_Z^\circ = \mu_pD^{-1} - D$$

by G on the left and G^T on the right we get

$$\Delta X^\circ = \mu_pZ^{-1} - X - W\Delta Z^\circ W.$$

This equation, jointly with (15), allow us to obtain

$$\left\{\begin{array}{ccl} \mathcal{A}(\Delta X^\circ) & = & F_p \\ \mathcal{A}^T(\Delta y^\circ) + \Delta Z^\circ & = & F_d \\ \Delta X^\circ & = & \mu_pZ^{-1} - X - W\Delta Z^\circ W. \end{array}\right. \tag{17}$$

Solving (17) we get the triple $(\Delta X^\circ, \Delta y^\circ, \Delta Z^\circ)$ satisfying

$$\left\{\begin{array}{ccl} \mathcal{A}(W\mathcal{A}^T(\Delta y^\circ)W) & = & \mathcal{A}(WF_dW + X - \mu_pZ^{-1}) + F_p \\ \Delta Z^\circ & = & F_d - \mathcal{A}^T(\Delta y^\circ) \\ \Delta X^\circ & = & \mu_pZ^{-1} - X - WF_dW + W\mathcal{A}^T(\Delta y^\circ)W. \end{array}\right. \tag{18}$$

The corrector direction is obtained taking in (16) $D_X = D_X^\circ + D_X^c$, $D_Z = D_Z^\circ + D_Z^c$ and $\mu = \mu_p + \mu_c$. The non-linear terms D_XD_Z and D_ZD_X are also approximated by $D_X^\circ D_Z^\circ$ and

$D_Z^\circ D_X^\circ$, respectively. Then, considering the conditions satisfied by the predictor direction, we can write

$$(D_X^c + D_Z^c)D + D(D_X^c + D_Z^c) = H_c,$$

with $H_c = 2\mu_c I - (D_X^\circ D_Z^\circ + D_Z^\circ D_X^\circ)$. By Theorem 2.1, the solution of this Lyapunov equation is $L \circ H_c$, with $L = [l_{ij}]$ where $l_{ij} = (d_i + d_j)^{-1}$ and d_i and d_j represent the i-th and the j-th entries of the diagonal matrix D, respectively. Multiplying both members of

$$D_X^c + D_Z^c = L \circ H_c$$

by G on the left and by G^T on the right we get

$$\Delta X^c = G(L \circ H_c)G^T - W\Delta Z^c W. \tag{19}$$

Now, consider $D_X^\circ = G^{-1}\Delta X^\circ G^{-T}$, $D_X^c = G^{-1}\Delta X^c G^{-T}$, $D_Z^\circ = G^T \Delta Z^\circ G$ and $D_Z^c = G^T \Delta Z^c G$. Then, it is possible to show that the equalities $D_X = D_X^\circ + D_X^c$ and $D_Z = D_Z^\circ + D_Z^c$ are equivalent to $\Delta X = \Delta X^\circ + \Delta X^c$ and $\Delta Z = \Delta Z^\circ + \Delta Z^c$, respectively. Furthermore, $D_X^\circ D_Z^\circ + D_Z^\circ D_X^\circ = G^{-1}\Delta X^\circ \Delta Z^\circ G + (G^{-1}\Delta X^\circ \Delta Z^\circ G)^T$. Thus, taking $(\Delta X, \Delta y, \Delta Z) = (\Delta X^\circ + \Delta X^c, \Delta y^\circ + \Delta y^c, \Delta Z^\circ + \Delta Z^c)$ and considering (19) and the conditions satisfied by the predictor direction we get the system

$$\left\{\begin{array}{ccl} \mathcal{A}(\Delta X^c) & = & 0 \\ \Delta Z^c + \mathcal{A}^T(\Delta y^c) & = & 0 \\ \Delta X^c & = & G(L \circ H_c)G^T - W\Delta Z^c W, \end{array}\right. \tag{20}$$

with $H_c = 2\mu_c I - [(G^{-1}\Delta X^\circ \Delta Z^\circ G) + (G^{-1}\Delta X^\circ \Delta Z^\circ G)^T]$. The solution of (20) is the corrector direction $(\Delta X^c, \Delta y^c, \Delta Z^c)$ that satisfies

$$\left\{\begin{array}{ccl} \mathcal{A}(W\mathcal{A}^T(\Delta y^c)W) & = & -\mathcal{A}(G(L \circ H_c)G^T) \\ \Delta Z^c & = & -\mathcal{A}^T(\Delta y^c) \\ \Delta X^c & = & G(L \circ H_c)G^T + W\mathcal{A}^T(\Delta y^c)W. \end{array}\right.$$

The final direction, $(\Delta X, \Delta y, \Delta Z)$, is obtained adding the predictor and the corrector directions.

From the computational experience with this version we verified that we got better results when we used a different heuristic for the duality measure in the predictor direction,

$$\mu_p = \frac{\langle X, Z\rangle}{2.3n^2}.$$

The algorithm corresponding to the version of our predictor-corrector variant that uses NT direction will be called *BNT algorithm* and differs from the BHKM algorithm in the following points:

(a) Take $\mu_p^k = \sigma^k \dfrac{\langle X^k, Z^k\rangle}{2.3n^2}$, with $\sigma^k \in [0,1]$;

(b) Solve system (17) for $(\Delta X^{\circ^k}, \Delta y^{\circ^k}, \Delta Z^{\circ^k})$, considering $\mu_p = \mu_p^k$;

(d) Solve system (20) for $(\Delta X^{c^k}, \Delta y^{c^k}, \Delta Z^{c^k})$, considering $\mu_c = \mu_c^k$;

6 Computational experience

Now, we will describe the computational experience that we have done to compare the two versions of our predictor-corrector variant and the classical predictor-corrector method, described in the previous sections.

6.1 Brief description of the used tools

The computational tests were performed in a Pentium III, 1100 Mhz, 245232 KB of RAM, with the 7.3 version of the Red Hat Linux operating system and the 2.96 version of the gcc compiler. To implement the new predictor-corrector variant we used the 3.2 version of the source code of the package CSDP by Brian Borchers [4, 6, 7]. The code was modified to achieve two main purposes: it was adapted to be possible to implement the different versions of the predictor-corrector variant and it was optimized to become faster and more robust.

To compare the performance of the algorithms we used the SDPLIB 1.2 collection of SDP test problems [5]. It has 92 problems in a wide range of sizes and drawn from many different applications, including truss topology design, control systems engineering and combinatorial optimization. It also includes primal and dual infeasible instances.

6.2 Results

We will present the results corresponding to both versions of the predictor-corrector variant described earlier and compare those results with the ones obtained with the classical predictor-corrector algorithm (PCC algorithm). We will use tables with information about the number of iterations, It, the total CPU time (in seconds), T, and the relative duality gap, RDgap. We will also include information about the relative admissibility, that specifies if the algorithm stops with a solution that is far from satisfying any of the admissibility criteria or not. All the solutions obtained satisfy the relative dual admissibility termination criteria but the same does not happen with the primal one. Thus, in RDgap column will appear * if the relative primal admissibility belongs to $]1.0 \times 10^{-06}, 1.0 \times 10^{-03}[$ and ** if it is non inferior to 1.0×10^{-03}. In all the experiences we used (12) as termination criteria and (10), satisfying (11), as initial solution.

Due to memory problems it was not possible to obtain any results for the problems *maxG55* and *maxG60*, so they were not included in the tables. For the same reason, it was also not possible to obtain some results with the BNT algorithm. In this case, the fields of the table will contain $\underline{\text{m}}$.

In the next section we will analyse the results presented.

Name	It	T	RDgap
arch0	30	31	3.51e-08
arch2	30	34	6.28e-08
arch4	29	33	9.99e-09
arch8	30	34	8.66e-08
control1	26	< 1	4.07e-08
control2	30	1	1.33e-08
control3	37	9	3.49e-07
control4	31	29	7.52e-08
control5	38	134	6.85e-07
control6	39	349	6.68e-08 *
control7	39	792	5.45e-08
control8	44	1680	1.69e-06
control9	39	2629	5.84e-07
control10	39	4476	2.87e-06
control11	41	7466	3.35e-06 *
equalG11	21	3378	1.06e-08
equalG51	24	7727	1.35e-08
gpp100	20	7	3.11e-08
gpp124-1	29	18	7.38e-07 *
gpp124-2	26	14	3.70e-08
gpp124-3	21	12	9.55e-08
gpp124-4	24	13	8.34e-08
gpp250-1	32	157	2.53e-05
gpp250-2	28	125	4.77e-07
gpp250-3	21	83	5.86e-08
gpp250-4	27	113	4.04e-08
gpp500-1	26	1070	7.56e-08
gpp500-2	34	1715	4.83e-07 *
gpp500-3	39	2099	5.49e-07
gpp500-4	32	1618	3.56e-08
hinf1	40	< 1	7.46e-06
hinf2	40	< 1	9.89e-09
hinf3	40	< 1	6.25e-06
hinf4	19	< 1	3.04e-08
hinf5	40	1	3.23e-05
hinf6	40	< 1	1.55e-05
hinf7	33	< 1	1.25e-08
hinf8	40	< 1	2.13e-06
hinf9	23	< 1	1.89e-08
hinf10	40	< 1	1.36e-04
hinf11	40	< 1	7.13e-05
hinf12	49	< 1	5.62e-04
hinf13	40	< 1	1.92e-05 *
hinf14	40	1	1.96e+00 **
hinf15	45	1	3.89e-06 *

Name	It	T	RDgap
infd1	14	< 1	2.69e+00
infd2	15	< 1	3.03e-01
infp1	40	2	2.09e+00
infp2	40	3	1.93e+00
maxG11	19	1691	5.22e-08
maxG32	20	51059	7.84e-08
maxG51	21	3962	1.00e-08
mcp100	16	4	1.33e-08
mcp124-1	16	5	9.59e-08
mcp124-2	17	5	1.00e-08
mcp124-3	18	6	1.00e-08
mcp124-4	17	7	1.48e-08
mcp250-1	18	35	1.00e-08
mcp250-2	17	34	8.03e-08
mcp250-3	18	40	1.00e-08
mcp250-4	17	42	6.99e-08
mcp500-1	20	430	9.50e-09
mcp500-2	20	459	2.62e-08
mcp500-3	20	483	1.00e-08
mcp500-4	20	526	1.00e-08
qap5	19	1	4.95e-06
qap6	18	2	6.07e-08
qap7	41	21	1.04e-07
qap8	26	49	3.92e-07
qap9	19	83	3.51e-08
qap10	22	234	4.42e-08
qpG11	20	15225	9.50e-09
qpG51	22	44005	1.00e-08
ss30	30	466	7.28e-08
theta1	18	< 1	9.50e-09
theta2	19	26	6.19e-08
theta3	19	247	4.14e-08
theta4	21	1387	3.14e-08
theta5	20	4727	6.84e-08
theta6	20	13951	6.73e-08
thetaG11	27	6221	2.19e-08
thetaG51	40	193855	2.19e-07
truss1	16	< 1	1.00e-09
truss2	18	< 1	2.42e-08
truss3	18	< 1	3.13e-08
truss4	16	< 1	1.00e-08
truss5	22	2	6.05e-08
truss6	24	1	5.11e-08
truss7	21	1	8.29e-08
truss8	27	41	4.71e-08

Table 1: PCC algorithm

Name	It	T	RDgap
arch0	31	28	2.18e-08
arch2	31	31	3.49e-08
arch4	30	31	1.42e-08
arch8	31	32	3.10e-08
control1	35	< 1	6.93e-08
control2	34	1	7.34e-08
control3	39	9	3.20e-07
control4	32	30	7.10e-08
control5	39	136	3.38e-07
control6	37	329	9.50e-07 *
control7	40	766	1.66e-06
control8	42	1515	1.61e-06
control9	38	2538	7.18e-07
control10	39	4457	1.12e-05 *
control11	41	7441	4.72e-06 *
equalG11	22	2888	6.60e-08
equalG51	24	6220	9.09e-08
gpp100	25	6	7.17e-08
gpp124-1	25	11	1.54e-08
gpp124-2	24	10	5.48e-08
gpp124-3	27	12	1.59e-08
gpp124-4	24	11	4.83e-08
gpp250-1	25	86	5.62e-08
gpp250-2	25	86	5.71e-08
gpp250-3	25	86	7.37e-08
gpp250-4	23	79	8.90e-09
gpp500-1	26	895	1.33e-08
gpp500-2	25	865	4.49e-08
gpp500-3	24	831	4.03e-08
gpp500-4	24	832	2.56e-08
hinf1	24	< 1	8.23e-08
hinf2	34	< 1	1.00e-07
hinf3	23	< 1	8.38e-08
hinf4	25	< 1	2.45e-08
hinf5	40	< 1	5.70e-06 *
hinf6	31	< 1	2.22e-06
hinf7	43	< 1	5.93e-08
hinf8	34	< 1	3.99e-07
hinf9	29	< 1	8.73e-09
hinf10	30	< 1	1.94e-05
hinf11	30	< 1	1.24e-05
hinf12	40	< 1	7.89e-02
hinf13	32	< 1	1.08e-05 *
hinf14	37	< 1	5.51e-06 *
hinf15	35	2	2.83e-05 *
infd1	14	< 1	2.58e+00
infd2	13	< 1	8.76e-01
infp1	30	1	1.87e+00
infp2	30	1	1.93e+00
maxG11	20	1364	3.49e-08
maxG32	22	52270	6.08e-08
maxG51	21	3047	6.61e-08
mcp100	17	2	5.37e-08
mcp124-1	18	4	6.91e-08
mcp124-2	19	5	3.67e-08
mcp124-3	18	5	3.76e-08
mcp124-4	19	5	3.02e-08
mcp250-1	20	31	1.55e-08
mcp250-2	20	32	3.44e-08
mcp250-3	19	34	3.51e-08
mcp250-4	19	39	5.91e-08
mcp500-1	20	322	2.04e-08
mcp500-2	20	351	2.11e-08
mcp500-3	19	353	4.92e-08
mcp500-4	20	403	1.83e-08
qap5	28	< 1	2.06e-08
qap6	26	3	6.25e-08
qap7	37	18	1.09e-07
qap8	25	39	8.69e-08
qap9	24	101	5.81e-08
qap10	27	280	9.31e-08
qpG11	21	12223	7.80e-08
qpG51	25	43256	8.52e-08
ss30	27	369	8.14e-08
theta1	22	< 1	3.20e-08
theta2	22	28	6.82e-08
theta3	22	276	3.65e-08
theta4	22	1426	3.57e-08
theta5	22	5115	5.05e-08
theta6	22	15187	1.86e-08
thetaG11	28	5233	2.85e-08
thetaG51	38	142052	1.68e-07
truss1	24	< 1	5.04e-08
truss2	20	< 1	2.19e-08
truss3	22	< 1	3.79e-08
truss4	23	< 1	3.88e-08
truss5	27	3	2.25e-08
truss6	24	1	1.47e-08
truss7	26	< 1	1.76e-08
truss8	27	38	2.82e-08

Table 2: BHKM algorithm

Name	It	T	RDgap
arch0	32	65	3.13e-08
arch2	30	65	8.68e-08
arch4	31	67	2.53e-08
arch8	31	67	8.54e-08
control1	34	1	4.98e-08
control2	33	1	4.75e-08
control3	39	10	2.43e-07
control4	32	37	3.99e-08
control5	44	160	1.58e-06 *
control6	38	354	2.61e-07 *
control7	39	786	5.64e-07
control8	38	1463	1.41e-06
control9	38	2708	7.68e-07
control10	41	4932	9.54e-06 *
control11	39	7211	3.33e-06 *
equalG11	21	6474	4.39e-08
equalG51	23	15046	3.19e-08
gpp100	25	12	9.99e-08
gpp124-1	26	23	1.33e-08
gpp124-2	29	26	2.36e-08
gpp124-3	23	20	9.13e-08
gpp124-4	22	19	4.62e-08
gpp250-1	25	207	1.55e-08
gpp250-2	25	210	4.53e-08
gpp250-3	23	193	7.33e-09
gpp250-4	23	194	4.96e-08
gpp500-1	27	2097	2.28e-08
gpp500-2	26	1995	6.01e-08
gpp500-3	24	1847	2.04e-08
gpp500-4	22	1678	3.84e-08
hinf1	33	< 1	1.91e-07
hinf2	33	< 1	1.11e-07
hinf3	32	< 1	1.45e-06
hinf4	25	< 1	8.64e-08
hinf5	30	< 1	2.02e-05 *
hinf6	20	< 1	7.88e-08
hinf7	39	< 1	1.83e-07 **
hinf8	32	< 1	1.18e-07
hinf9	28	< 1	8.52e-08
hinf10	30	< 1	4.06e-05
hinf11	30	< 1	4.41e-05
hinf12	40	< 1	7.49e-02
hinf13	30	2	7.23e-05 *
hinf14	38	< 1	9.57e-08 *
hinf15	33	1	2.15e-04 *
infd1	30	1	2.92e+00
infd2	30	2	4.36e-01
infp1	9	1	1.95e+00
infp2	10	< 1	1.98e+00
maxG11	19	5325	2.79e-08
maxG32	m	m	m
maxG51	22	12100	1.96e-08
mcp100	18	7	8.00e-08
mcp124-1	18	13	2.83e-08
mcp124-2	17	13	1.87e-08
mcp124-3	17	13	2.36e-08
mcp124-4	17	14	1.86e-08
mcp250-1	19	143	5.31e-08
mcp250-2	18	138	4.09e-08
mcp250-3	18	138	6.58e-08
mcp250-4	18	140	5.59e-08
mcp500-1	21	1426	2.83e-08
mcp500-2	20	1410	2.71e-08
mcp500-3	19	1347	4.02e-08
mcp500-4	19	1333	1.25e-08
qap5	19	< 1	1.60e-06
qap6	26	3	5.62e-08
qap7	28	14	7.95e-08
qap8	27	48	9.04e-08
qap9	28	128	7.89e-09
qap10	22	246	4.61e-08
qpG11	m	m	m
qpG51	m	m	m
ss30	27	655	9.30e-08
theta1	22	2	8.78e-08
theta2	22	33	2.01e-08
theta3	20	279	6.08e-08
theta4	22	1523	3.10e-08
theta5	21	4975	2.22e-08
theta6	24	16751	2.20e-8
thetaG11	28	11021	4.89e-08
thetaG51	m	m	m
truss1	24	< 1	4.07e-08
truss2	19	1	7.57e-08
truss3	23	< 1	3.83e-08
truss4	23	< 1	8.57e-08
truss5	22	3	3.38e-08
truss6	25	3	6.44e-08
truss7	24	< 1	3.72e-09
truss8	25	39	1.16e-08

Table 3: BNT algorithm

6.3 Analysis of the results

To help the analysis of the previous tables, for each class of problems, we will construct graphics with information about the time and the number of iterations. As the running times of the problems in *hinf*, *inf* and *truss* classes are very small, we do not present the corresponding graphics with the time information. Furthermore, when, for a certain class, the amplitude of the interval of time is too big we will present two graphics to facilitate the reading.

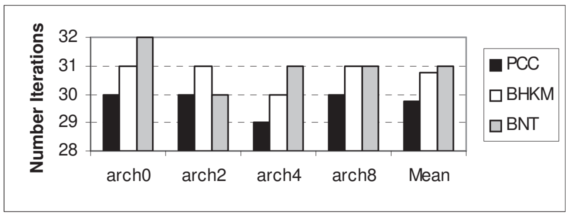

Figure 1: Number of iterations for the *arch* class.

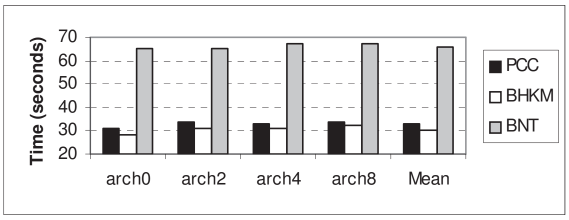

Figure 2: Time, in seconds, for the *arch* class.

As we can observe in Figure 1, for the *arch* class the algorithms that performed less and more iterations were PCC and BNT, respectively. From Figure 2 we can conclude that BHKM was the fastest while BNT was the most time consuming.

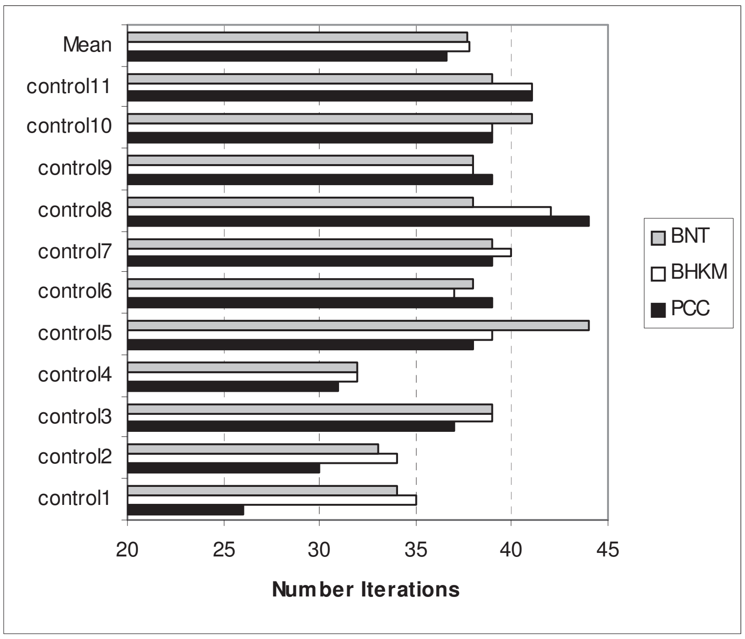

Figure 3: Number of iterations for the *control* class.

Consider now the *control* class. In Figure 3 is possible to observe that PCC was the algorithm with less iterations while the other two presented similar means for the number of iterations. For the time, Figures 4 and 5, all the algorithms presented identical means.

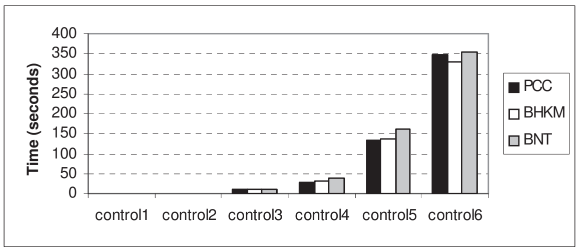

Figure 4: Time, in seconds, for the problems *control1* to *control6*.

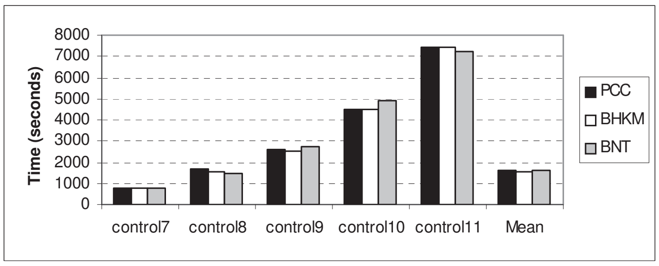

Figure 5: Time, in seconds, for the problems *control7* to *control11*.

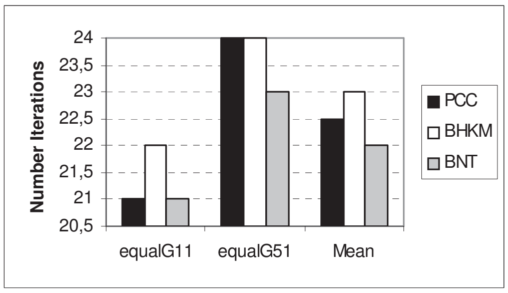

Figure 6: Number of iterations for the *equalG* class.

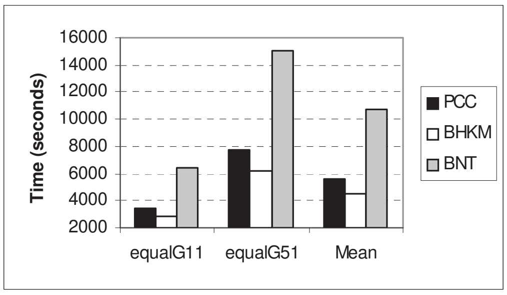

Figure 7: Time, in seconds, for the *equalG* class.

For the *equalG* class, the algorithm with less iterations was BNT while the one with more was BHKM, Figure 6. These positions were inverted for the time, Figure 7.

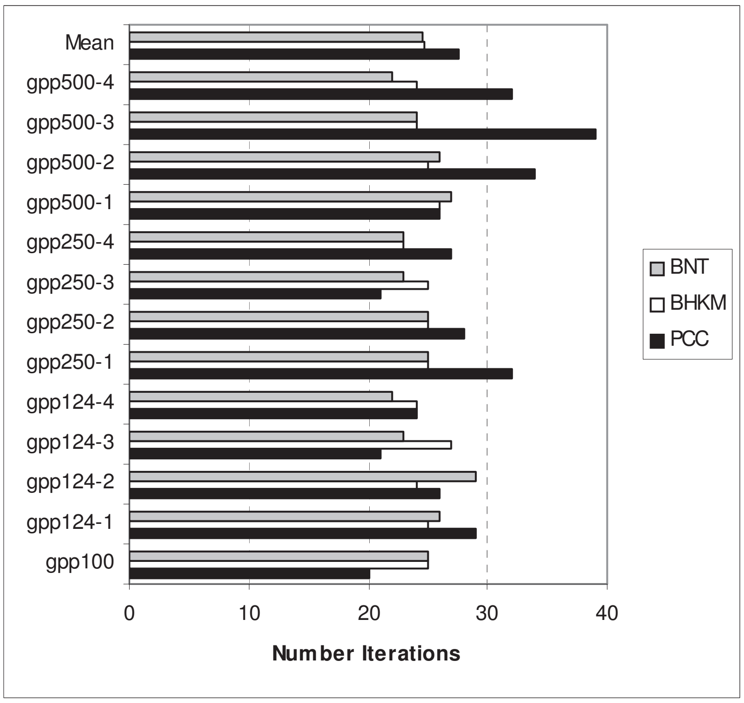

Figure 8: Number of iterations for the *gpp* class.

Relatively to the *gpp* class, we can observe in Figure 8 that the algorithms with less iterations were BHKM and BNT, being the first one the fastest and the second the slowest, Figures 9 and 10.

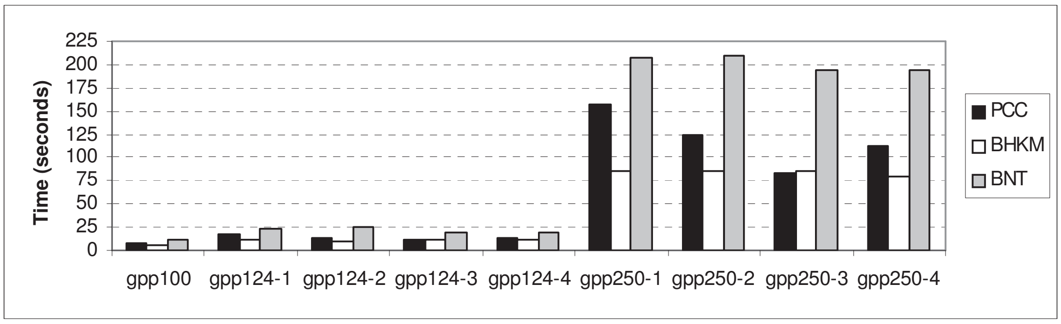

Figure 9: Time, in seconds, for the problems *gpp100* to *gpp250-4*.

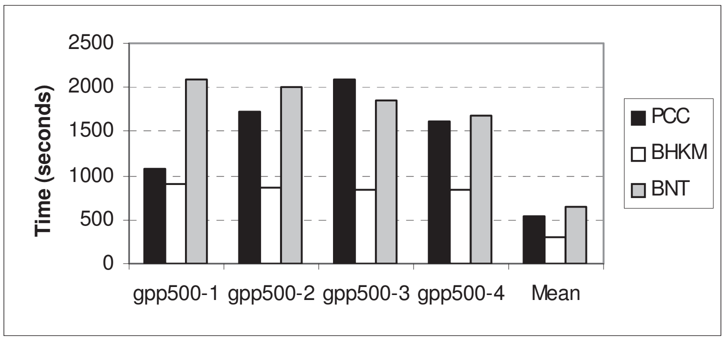

Figure 10: Time, in seconds, for the problems *gpp500*.

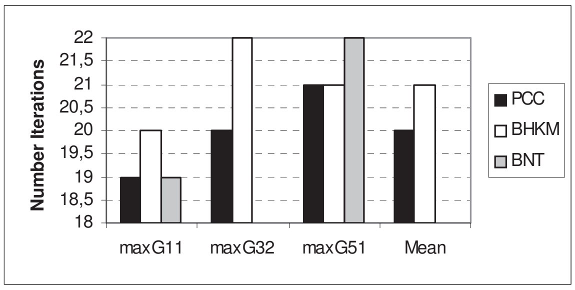

Figure 11: Number of iterations for the *maxG* class.

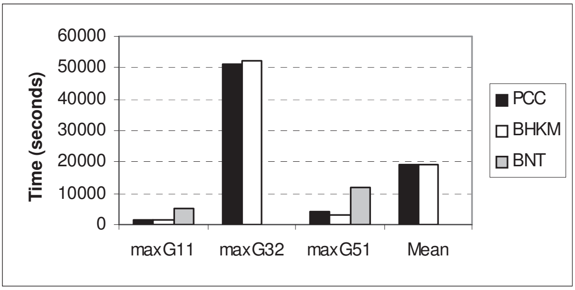

Figure 12: Time, in seconds, for the *maxG* class.

Consider now the *maxG* class. Due to memory problems, it was not possible to obtain any results for *maxG32* with BNT. But, in general, this algorithm was the worst one for the remaining problems. PCC was the algorithm that presented less iterations, Figure 11. PCC and BHKM spent similar amounts of time, Figure 12.

For the *mcp* class, in Figure 13 is possible to see that PCC was the algorithm with less iterations while BHKM was the one with more. But, this last one was the fastest, being BNT the most expensive, Figures 14 and 15.

For the *qap* class, the best and the worst algorithms were PCC and BHKM, respectively, Figures 16, 17 and 18.

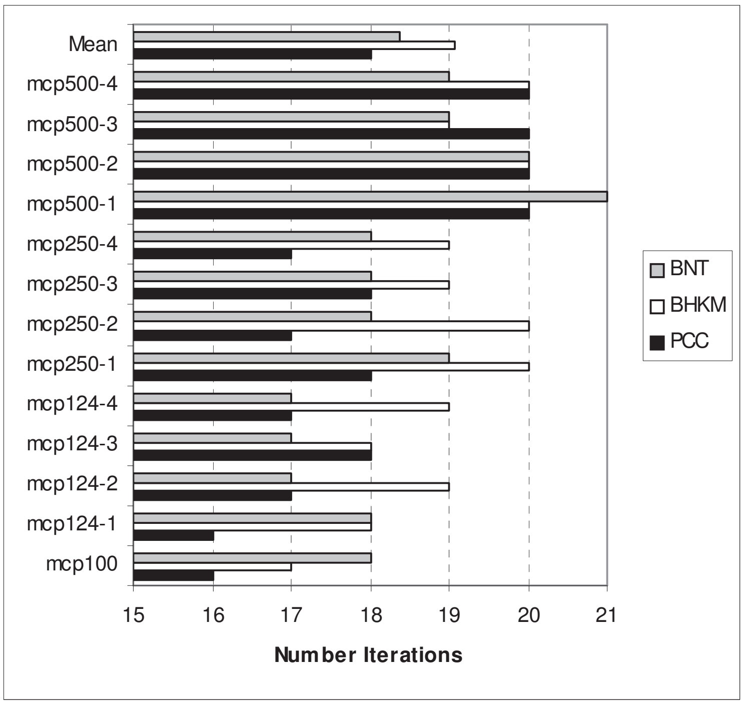

Figure 13: Number of iterations for the *mcp* class.

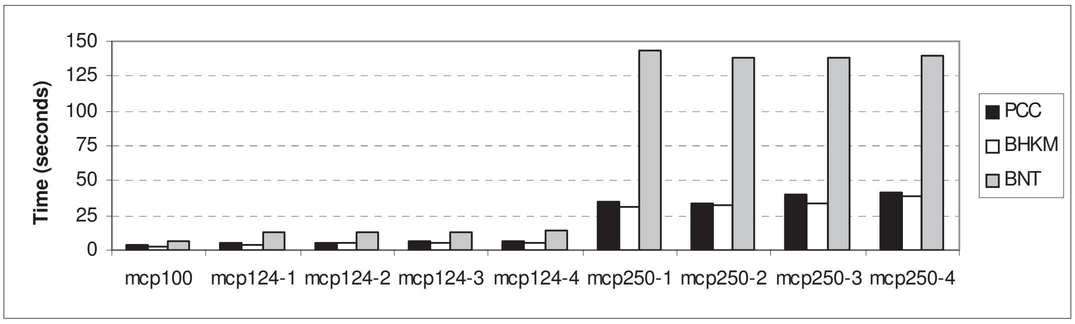

Figure 14: Time, in seconds, for the problems *mcp100* to *mcp250-4*.

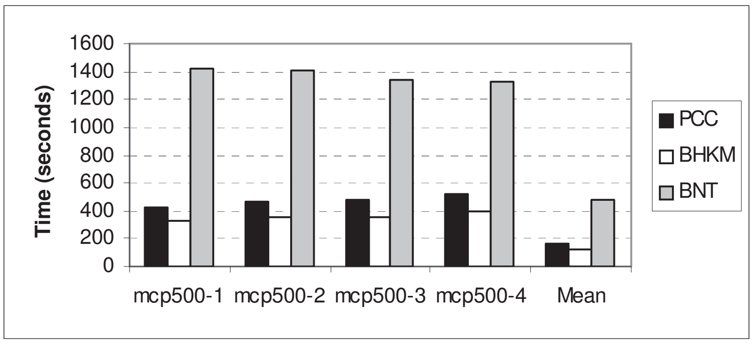

Figure 15: Time, in seconds, for the problems *mcp500*.

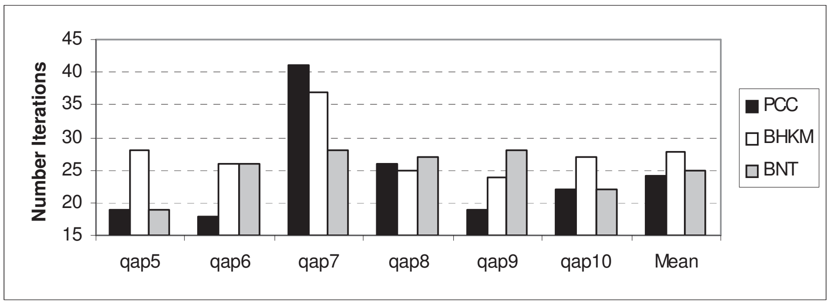

Figure 16: Number of iterations for the *qap* class.

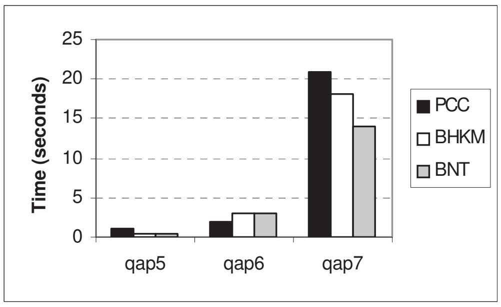

Figure 17: Time, in seconds, for the problems *qap5* to *qap7*.

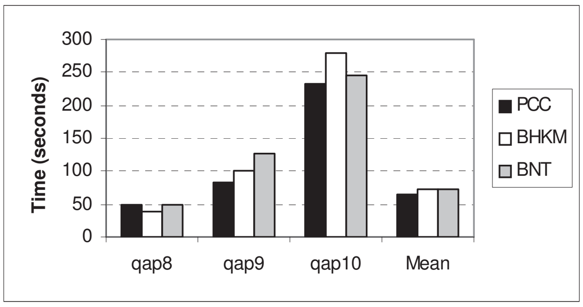

Figure 18: Time, in seconds, for the problems *qap8* to *qap10*.

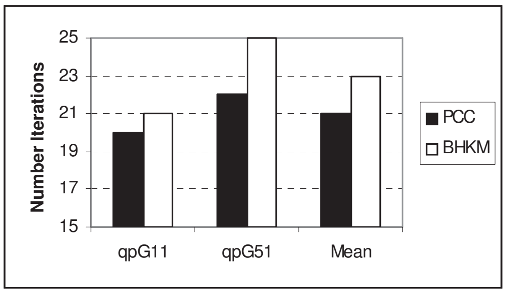

Figure 19: Number of iterations for the *qpG* class.

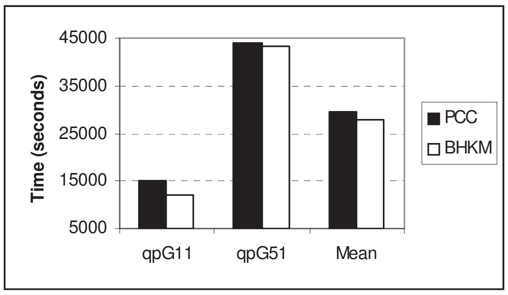

Figure 20: Time, in seconds, for the *qpG* class.

Consider, now, the *qpG* class. Due to memory problems, it was not possible to obtain any results with BNT. Once more, BHKM was the algorithm with more iterations, Figure 19, but also the fastest, Figure 20.

From Figures 21 and 22 is possible to conclude that, for the *ss30* problem, BHKM and BNT were the algorithms that presented less iterations being the first one the fastest and the second one the slowest.

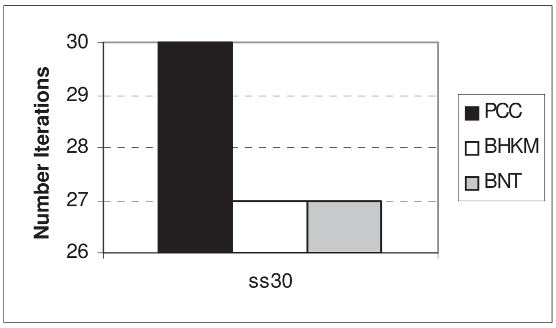

Figure 21: Number of iterations for the *ss30* problem.

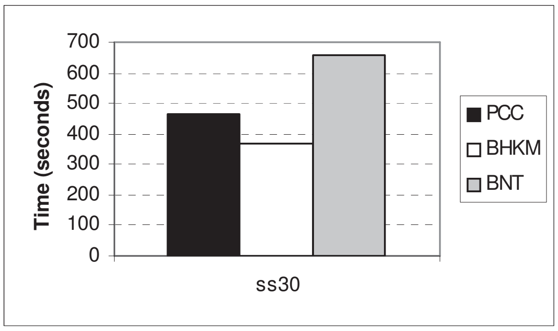

Figure 22: Time, in seconds, for the *ss30* problem.

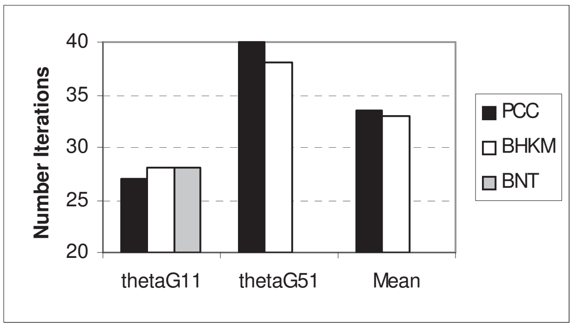

Figure 23: Number of iterations for the *thetaG* class.

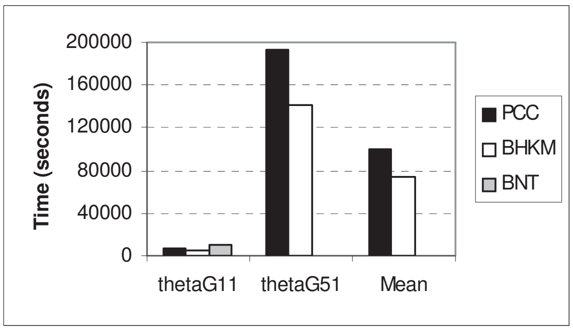

Figure 24: Time, in seconds, for the *thetaG* class.

As it happened before, due to memory problems, it was not possible to obtain any results for *thetaG51* with BNT. For *thetaG11* this algorithm was the most expensive. As we can see in Figures 23 and 24, the BHKM algorithm was the best one for the *thetaG* class.

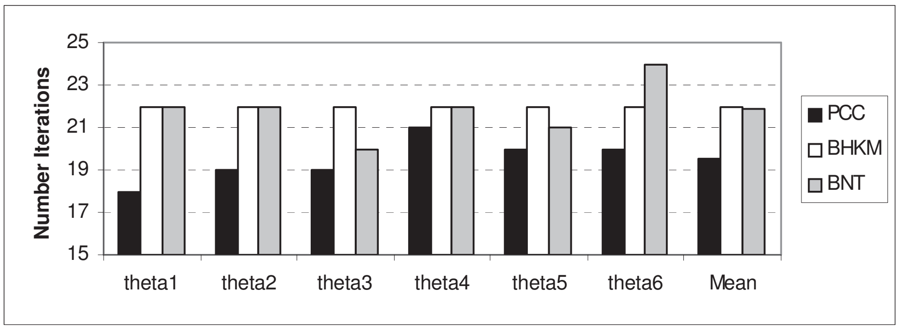

Figure 25: Number of iterations for the *theta* class.

From Figures 25, 26 and 27 we can conclude that PCC was the best algorithm for the *theta* class, while BHKM was the one with more iterations and BNT was the slowest.

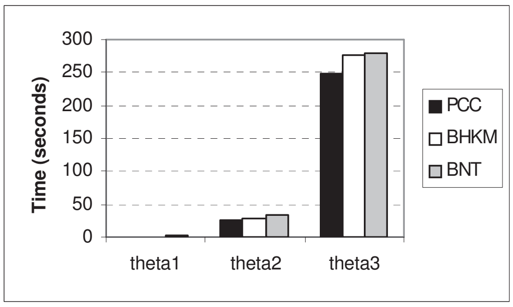

Figure 26: Time, in seconds, for the problems *theta1* to *theta3*.

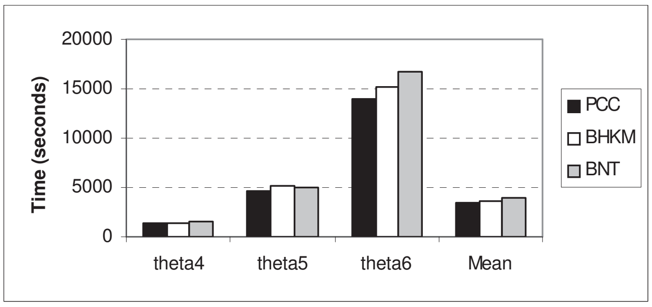

Figure 27: Time, in seconds, for the problems *theta4* to *theta6*.

It remains to analyse the results relating to *truss*, *hinf* and *inf* classes. As these problems are of quick resolution we decided to study only the results about the number of iterations.

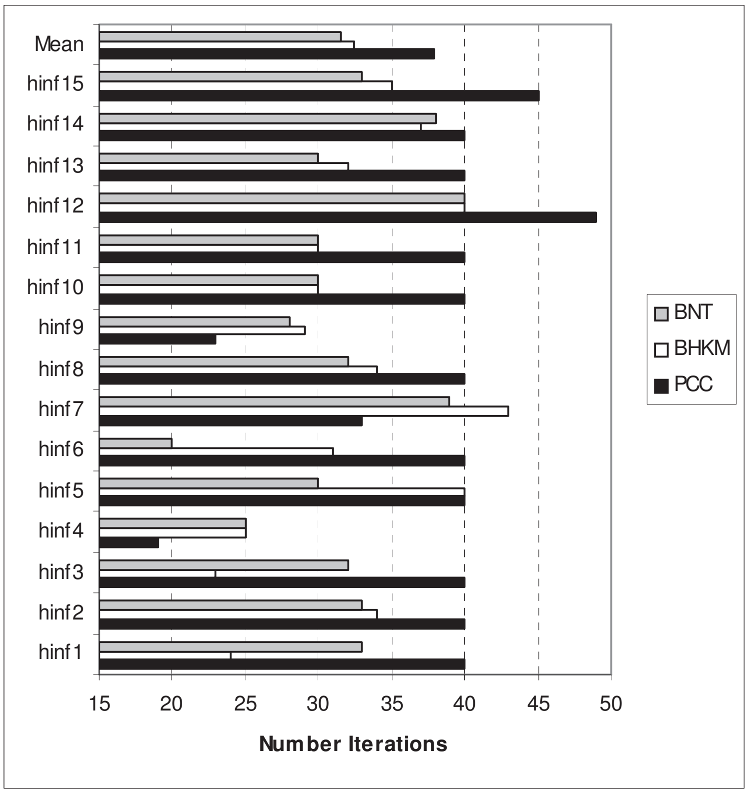

Figure 28: Number of iterations for the *hinf* class.

As we can see in Figure 28, for the *hinf* class the algorithm that presented less iterations was BNT while the one with more was PCC.

For the *truss* class, Figure 29, the algorithm with less iterations was PCC and the one with more was BHKM.

For the *infd* problems, Figure 30, BHKM was the algorithm that took less iterations while BNT was the one with more. Finally, for the *infp* problems, Figure 31, BNT was the best

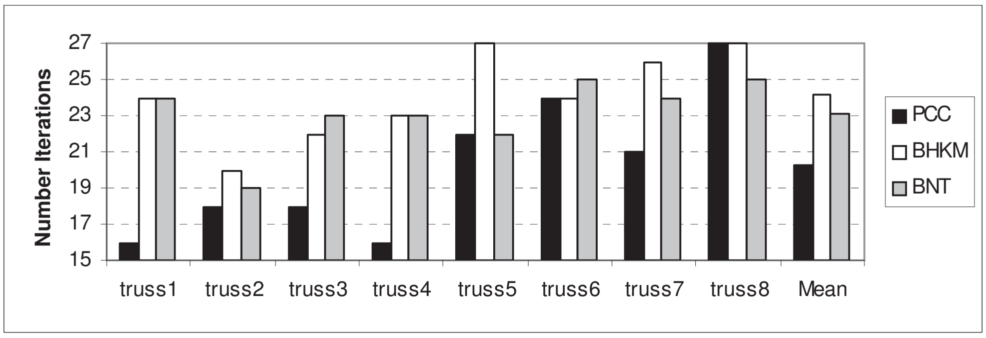

Figure 29: Number of iterations for the *truss* class.

algorithm while PCC was the worst.

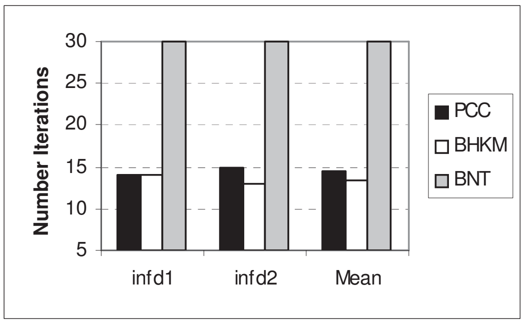

Figure 30: Number of iterations for the *infd* problems.

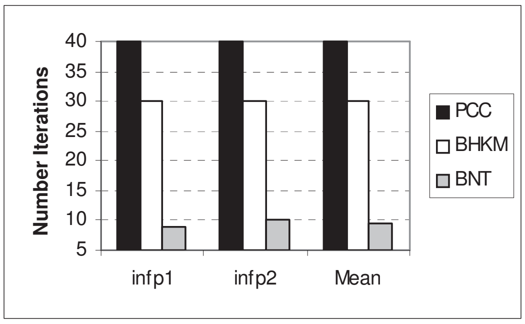

Figure 31: Number of iterations for the *infp* problems.

To summarize we can say that:

- although BHKM was the algorithm that, in general, presented more iterations it was the fastest one;
- PCC was the algorithm that presented, for most of the cases, less iterations;
- BNT was the algorithm that spent more time.

For the classes of problems with results that satisfy all the termination criteria, the efficiency of the algorithms can be measured just in terms of the two previously analysed characteristics. For the other classes we considered that the efficiency should be measured in terms of how far the solution is from satisfying all those criteria. The *control* is one of these classes. For this class, PCC was the algorithm that performed better while BHKM and BNT presented similar behaviour. For the *gpp* class the only algorithm that stopped without satisfying all the termination criteria was PCC. The BHKM was the best algorithm for *hinf* and *qap* classes while PCC was the worst. As *infd* and *infp* are infeasible problems and problems with an infeasible dual, respectively, the values in RDgap do not mean, just by themselves, that the results were not satisfactory. All the algorithms solved the *infd* problems but only BNT solved the *infp* ones. Finally, for *thetaG51* both PCC and BHKM presented similar results while, as already mentioned, due to memory problems it was not possible to obtain any results with BNT.

7 Final conclusions

From the obtained results we can conclude that, in general, the algorithm that uses the NT direction was the slowest and, due to memory problems, did not allow to obtain results for the problems with bigger dimensions. It was only better than the remaining algorithms for the *infp* problems. Although, for most of the problems, the BHKM algorithm needed more iterations than the others it was, in general, the fastest. Relatively to the classes of problems for which at least one of the algorithms ends without satisfying all the termination criteria, although PCC was the best for *control* class the BHKM was superior for *gpp*, *hinf* and *qap* classes, having both presented similar behaviour for the *thetaG51* problem. Finally, we can conclude that the version of our predictor-corrector variant with HKM direction is better than the classical predictor-corrector for the generality of the problems.

References

[1] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal Optimization, 5(1) (1995), 13-51.

[2] F. J. Bastos, *Problemas de transporte e métodos de ponto interior*, Dissertação de Doutoramento, Universidade Nova de Lisboa, Lisboa, 1994.

[3] R. Bellman, K. Fan, *On systems of linear inequalities in Hermitian matrix variables*, V. L. Klee (editor), Convexity, Proceedings of Symposia in Pure Mathematics, American Mathematical Society, 7 (1963), 1-11.

[4] B. Borchers, *CSDP, a C library for semidefinite programming*, Optimization Methods and Software, 11(1) (1999), 613-623.

[5] B. Borchers, *SDPLIB 1.2, a library for semidefinite programming test problems*, Optimization Methods and Software, 11(1) (1999), 683-690.

[6] B. Borchers, *CSDP 3.2, User's Guide*, Technical Report, Department of Mathematics, New Mexico Tech, USA, 2000.

[7] B. Borchers, *Implementation issues in CSDP*, Technical Report, Department of Mathematics, New Mexico Tech, USA, 2001.

[8] C. Helmberg, F. Rendl, R. J. Vanderbei, H. Wolkowicz, *An interior-point method for semidefinite programming*, SIAM Journal Optimization, 6(2) (1996), 342-361.

[9] R. Horn, C. Johnson, *Matrix Analysis* , Cambridge University Press, Cambridge, 1991.

[10] R. Horn, C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[11] M. Kojima, S. Shindoh, S. Hara, *Interior-point methods for the monotone semidefinite linear complementary problem in symmetric matrices*, SIAM Journal Optimization, 7(1) (1997), 86-125.

[12] L. Lovász, *On the Shannon Capacity of a Graph*, IEEE Transactions of Information Theory, IT-25(1) (1979), 1-7.

[13] R. D. C. Monteiro, *Primal-dual path-following algorithms for semidefinite programming*, SIAM Journal Optimization, 7(3)(1997), 663-678.

[14] A. Nemirovskii, Y. Nesterov, *Interior-Point Polynomial Algorithms in Convex Programming*, Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[15] Y. E. Nesterov, M. J. Todd, *Self-scaled barriers and interior-point methods for convex programming*, Mathematics of Operations Research, 22(1) (1997), 1-42.

[16] M. J. Todd, *A study of search directions in primal-dual interior-point methods for semidefinite programming*, Optimization Methods and Software, 11 (1999), 1-46.

[17] M. J. Todd, K. C. Toh, R. H. Tutuncu, *On the Nesterov-Todd direction in semidefinite programming*, SIAM Journal Optimization, 8(3) (1998), 769-796.