# Referee's Report

M. V. Nayakkankuppam

## Solving Large-Scale Semidefinite Programs in Parallel

The author describes, and reports numerical results of, a parallel implementation of the spectral bundle method with special emphasis on block structured problems. In my impression, a good step forward is the "Block Structured Lanczos Method with Active Block Strategy" for speeding up function evaluations in the bundle method, which coordinates simultaneous computation of the maximal eigenvalues of the blocks. Up to some hopefully minor mistakes in the presentation of the general method, the paper is well written, the numerical examples are well chosen, and the discussion of the advantages and disadvantages of the approach is solid and unbiased.

In Section 2.1, equation (4) is wrong and should read

$$\hat{f}^k(y) = \max_{j \in J^k} f(y^j) + \left\langle g^j, y - y^j \right\rangle$$

Likewise, (5) and (7) need corrections. Theorem 1 needs $\delta = 0$. In Algorithm 2.1, 14: should be more restrictive in the update requirements on $\rho^k$. To my knowledge, it has only bee proved, [16], that $\rho^k$ may be increased after null steps and may be chosen arbitrarily above $\rho_{\min}$ after serious steps. Equation (10) lacks an explanation of $P$ and is correct only if $PP^T = P^TP = I$, so if $P$ spans the whole space. It would be simpler to write

$$\partial_\varepsilon \lambda_{\max}(Z) = \{W \succeq 0 : \operatorname{tr}(W) = 1, \langle Z, W \rangle \geq \lambda_{\max}(Z) - \varepsilon\}.$$

In (11), I guess, $f = \lambda_{\max}(A^Ty - C) - \langle b, y \rangle$, but is this really the same $\varepsilon$ appearing twice? Also, according to general experience it is not true, that the leading few eigenvalues can be computed at only slightly larger computational expense by the Lanczos method. This might only be the case, if the eigenvalues are well separated. Here and in several places in the rest of the paper the difficulty arises that Ritz vectors are used instead of true eigenvectors. Since the author uses the name Ritz vectors in his treatment of the Lanczos method, he might as well use it in the rest of the paper.

I am not quite sure whether it is really worth to go into the details of the Lanczos method and implicit restarting in sections 5.1, 5.2 with corresponding algorithms 5.1 and 5.2 since the paper describes no relevant modifications to these codes. In my view, the sections could be dropped almost completely, replacing them mostly by references to the papers/books. I have not checked these sections in detail (but sometimes $\nu^0$ is used instead of $\nu^1$).

Section 5.3, however, is definitely worth reporting and, together with this, Algorithm 5.3. This really seems to be a clever approach for speeding up the computation. Yet, there are some details that must be clarified. E.g., by what means is it ensured that there is no false convergence? As listed, Algorithm 5.3 would be fooled by giving e.g., for $p = 3$, three matrices that exhibit converged maximum eigenvalues and eigenvectors after the first spectral factorization (or just one matrix that has 3 large converged eigenvalues), while the largest eigenvalue is part of a fourth block and has not yet emerged. What mechanism makes sure, that all other blocks cannot have a

larger eigenvalue? Why is there a need for $p$ converged eigenvalues and eigenvectors, why is not just one enough (as long as that one is indeed the maximum eigenvalue)? What about the claim, that the code returns the maximal $p$ eigenvalue/eigenvector pairs of $Z$ if the maximum eigenvalue is clustered in one block? The described Lanczos method would exhibit at most one out of this block. Mind, I definitely see no need or advantage in using a block Lanczos (in fact, this makes it even more difficult to avoid false convergence), but making sure that one maximum eigenvalue/eigenvector pair is found is enough.

Section 5.6 does not really convince me. I would guess, that constructing the matrix $A^T y - C$ first in sparse representation (maybe by first forming the sparse representation on each set of $A_i$ of each computer) and then distributing the work according to a partition of the rows is much more efficient than forming the sum again for every matrix vector multiplication (unless the $A_i$ have some nonsparse structure, of course). But quite likely this would have been a lot more work to implement.

Also, it seems dangerous to me, that the vector of all ones is used as a starting vector for the Lanczos process, because for such a special vector probability seems high that it does not have a component in the direction of the maximum eigenvector. In my impression, it is generally agreed, that one should take some randomized vector as a starting vector to avoid this effect.

Section 6 looks convincing to me. Indeed, I would expect, that here the approach taken is very well suited for exploiting parallelism.

In Section 8 the importance of choosing the parameter $\rho$ was obviously already observed in the initial work of Lemarechal and also gave rise to [16].

In the computational results, an accuracy of 0.01 indeed seems very low. In my eyes, already three digits is almost nothing, but maybe we cannot hope for more.

In Section 9.4 the meaning of columns Total and Sub is not explained. I guess, it is total wall clock time in seconds and the wall clock time spent in solving the subproblem in seconds.

In Table 7, the entries BeO/64 and LiF/64 catch the eye, because computing time suddenly doubles in comparison to 32. Is this indeed only due to communication losses? Maybe it is good to mention the cause for these anomalies explicitly in the text.

Section 9.6 suddenly uses fewer exact eigenvalues than before. Is it indeed true, that the use of more exact eigenvalues improves performance in the previous examples?