Referee report on:
On the solution of large-scale SDP ...
by M. Kocvara and M. Stingl
submitted to: Math Programming

Contents.

The paper deals deals with computationally tractable ways to solve large-scale semidefinite programs (SDP) approximately. The algorithmic starting point is a previous paper by the same authors ([10]), which is based on a generalization of the augmented Lagrangian method. In the present paper, the authors combine their approach [10] with iterative solvers for the linear system to be solved. It is clear that a direct solution of the linear system is too expensive once the number of constraints is larger than a few 1000. The authors investigate various preconditioners and also exploit fast ways for computing Hessian times vector (approximately) by finite differences.

The main part of the paper are the computational tests. First, several preconditioners are compared against each other. Then several classes and sources for SDP are studied in detail. These range from combinatorial applications (Max-Cut, theta-function) to instances from truss design. The computational tests are done quite carefully, with detailed description of stopping conditions and algorithmic features such as time per iteration etc.

There is no clear conclusion by the authors as to which variant of their setttings is the overall champion. All in all, they manage to show that on a wide range of instances, their approach is highly competitive with other large-scale methods.

In summary therefore, there is a clear message to the computationally oriented readership of the journal.

Comments.

From reading the paper, it is not quite clear what exactly is different in [10] from the current paper. This should be stated explicitely somewhere in the introduction.

The paper contains several minor typos and inconsistencies, that should be fixed.

1. page 3, +4: ' ill-conditioned matrices **arise**'
2. page 3, +7: ' does not **necessarily** lead to '
3. page 7, -11: Why don't you need a reasonably exact solution of (11)? It is announced that you return to this point in the next section, but I could not spot this. I would assume that an inaccurate solution of (11) will give a bad search direction. How can you be sure that you get a direction of improvement at all, if you solve only inaccurately?
4. page 8, 2nd line after (12): ' the complexity .... amounts **to** the number ...'
5. page 17ff: the table numbering semms to be messed up. In the text you refer to e.g. Table 5.1, Table 5.2, but there is only a table 1 and a table 2 etc.
6. page 27, 2nd line in 7: 'use iterative solvers **for** the computation ...'
7. page 29: The MON preconditioner should probably also be moved to section 4. It seems that this part was added after the paper was finished.
8. page 29ff: some references do not have a year of appearance, e.g. [12], [22-24].

Recommendation:

The paper is acceptable for publication, but a minor revision should reflect the items given above.