Referee 2

Comments on "Large-scale semidefinite programming via saddle point mirror-prox algorithm" by Z. Lu, A. Nemirovski, R.D.C. Monteiro

Let an SDP where the feasible region of the variable vector is bounded and we know some bounds related to the problem a priori. Suppose also that the positive semidefinite affine combination of the data matrices is well-structured (or favorably-structured). Based on the saddle-point mirror-prox algorithm [6], this paper proposes a reformulation of an SDP as a saddle-point problem and an algorithm which converges basically in $\mathcal{O}(\epsilon^{-1})$ steps where each step is dominated by eigenvalue decompositions of smaller "block diagonal matrices" of the problem.

The decomposition of a large well-structured sparse matrix into smaller matrices where the positive (semi)definitess is preserved, makes possible to overcome one of the weakness of the mirror-prox method.

Two particular problems: the Lovász capacity and the SDP relaxation of the max-cut problem on the so-called stair-case structure are reformulated as saddle-point problems, and numerical experiments on large-scale instances are given. Certainly, these problems can not be solved with interior-point methods which have expensive iteration but converges in $\mathcal{O}(\ln \epsilon^{-1})$ steps compared to the proposed method which requires much cheaper iterations but $\mathcal{O}(\epsilon^{-1})$ steps.

Main comments

1. page 3, line 11 of sec. 2: "In other words, when ... Cholesky factor." A more careful discussion about the ordering of rows/columns and fill-in is necessary. For instance, consider a matrix whose nonzeros are specified by the positions $(1,i),(i,1),(i,i), \quad i=1,\ldots,n$. If we apply the Cholesky factorization with lower triangular factors to it, we will get a complete fill-in. Instead, we should swap the indices 1 and n, apply the symbolic Cholesky factorization which will not produce any fill-in, and then swap the indices 1 and n again to make use of the procedure discussed here.

2. page 10, Proposition 3.4 and Remark 3.1. Although the authors claim that Prop. 3.4 can be proved from [4] with "moderate effort", it seems rather straightforward. Also, you would prefer to state Prop. 3.4 as a particular case of the result in [4]. In fact, [4] says that a partially defined symmetric matrix admits a positive definite matrix completion if and only if the associated graph is chordal. Prop. 3.4 will follow if we show that the graph corresponding to the structure defined in section 2 is chordal. Suppose that we have a cycle of length $t > 3$ in this graph, and let reorder the indices of the vertices of this cycle according to the order which they appears as matrix indices $i_1 < i_2 < \cdots < i_t$. Let i_s the largest among these indices where (i_1, i_s) is in the original graph. If (i_1, i_s) is not one of the arcs in the cycle, our cycle in fact is not a cycle. Otherwise, because of the structure defined in section 2, (i_1, i_r) is an arc for any vertex i_r in the cycle with $i_1 < i_r < i_s$. Therefore we can only have cycles of length at most 3, and the graph is chordal.

3. The notation given in section 2 is quite complicate. Though I am not sure how it could be improved, and it will depend if you will follow or not the next suggestion. However, it can be partially improved if you avoid calling same values, sets or structures with different names or equivalent expressions. Adding explanations with few words can help the interpretation of those definitions. Some suggestions are given below.

4. page 5, Prop. 3.1 and the subsequence results. From the fact 1. above, I think you can easily extend the results here. That is, Prop. 3.1 is valid even when the structure of nonzeros corresponds to a chordal graph which is more general than the simple sparsity structure. For instance, suppose that the maximal cliques $C_1, \ldots, C_t$ of the corresponding chordal graph are ordered to satisfy the running intersection property as state in [3] and references therein. Let $S_1 = C_1 \backslash (C_2 \cup \cdots \cup C_t) \neq \emptyset$ and $U_1 = C_2 \cup \cdots \cup C_t$. Then, Prop. 3.1 can be stated if we consider a block matrix P with indices in U_1, R with indices in S_1, and using the induction hypothesis. Although the results will be more neat with this suggestion, it will consequently require a complete change in the notation throughout the paper.

Minor comments

1. several places: "Lovász"
2. there are many equation with number which are not used later.
3. Abstract, line 1: preference for alternative expression for ""economical" representation".
4. page 1, (1): $x \in \mathcal{N} \cap \boldsymbol{S}_+$
5. page 1, 4 lines below (1): add "partially overlapped" before "diagonal blocks".
6. page 2, line 6: 10^5 is more realistic.
7. page 2, line 12: add "present" before "method is in".
8. page 2, line 15: μ is already used in section 4.
9. page 2, line -15: remove extra close parenthesis.
10. page 2, line -12, page 3, line 5: "first-order".
11. page 3, section 2, motivation: explanation is too long and confusing. It is better to refer only to Cholesky factorization with upper triangular factor here. There is no clear relation between n and A^ℓ.
12. page 4, line 1: "a simple sparsity structure" is defined here or at the previous page?
13. page 4: it might be useful if the meaning of some sets are spelled out in words. For instance, J_k is the set of indices which will correspond to smaller blocks, J'_k is the set of common indices between J_k and J_{k-1}, etc.
14. page 4, item 3: You can also derived that $J'_k = J_{k-1} \cap J_k$.
15. page 4, item 4, line -2: J_k.
16. page 5, Proposition 3.1: W not defined here.
17. page 6, proof of proposition 3.1: there is a confusion in using the indices m and s here. State that $A = [P \quad Q; Q^T \quad R]$.
18. page 6, line 5 of the proof: extra "$\succeq 0$".

19. page 6, line 8: no need to say again that "$B^m \succeq 0$".

20. page 7, Lemma 3.1: add "a" after "if and only if there exists".

21. page 8, line 1: $[\Delta^k]^T = [\Delta^k_{ij}]_{i,j\in J'_{k+1}}$

22. page 8, line 6: $i = 1, 2, \ldots, i_k$

23. page 8, before (11): use $k \in D_{ij}$ instead. B^k instead of $\mathcal{B}^k$.

24. page 9, after Proposition 3.2: It may be better to explicitly refer to some system as $\mathcal{S}$. In the next line, A is not a variable but given.

25. page 9, example: using symbol μ may be confusing here.

26. page 10, line 3 of section 3.2: "which belongs to the dual" instead of "which is dual"

27. page 10, proof of Prop. 3.4: It might be not necessary to state that $\Sigma Tr([Z^k_{ij}][B^k_{ij}]) \equiv \ldots Z_{ij} = Z^k_{ij} \quad i, j \in J_k, k = 1, \ldots, m$. What is B^k_{ij} here?

28. page 11, line 1 of section 4.1: "Consider THE semidefinite".

29. page 12, before (20): $(x_\epsilon \in X, \Delta_\epsilon \in \Delta_\rho)$.

30. page 13, Theorem 4.1 (c): delete ", or the simplex $\{\ldots\}$".

31. page 13, line -7: $\Delta^\ell = [\Delta^\ell_{ij}]_{i,j\in J'_{\ell+1}}, \quad \ell = 1, 2, \ldots, m-1.$

32. page 14, line 4 and 12: $\hat{S}$ bold.

33. page 14, 16: $L_{\Delta Y}$.

34. page 14, (24): remove since it is not used later on.

35. page 15, definition of $\mathcal{P}(A)$. Change the notation for $\hat{A}$. Then you can avoid introducing the symbol X in the next page to refer to $\hat{\tilde{B}}$.

36. page 16, (30): why the term $\ln(n)$ do not appear here at this point?

37. page 16, after (32): avoid calling $\mathcal{P}(\mathcal{S}(\tilde{B}))$ as C.

38. page 17, Proposition 4.1: (28) instead of (29) and delete reference to (26).

39. page 17, Corollary 4.2: add "Proof" before the proof.

40. page 17, Example: it is very confusing to use m here, since it refers to the number of small block matrices along the paper.

41. page 18, item 1: is $\ln(pm)$?

42. page 19, line -9: I am not sure if the cost is right here. Is it $\mathcal{O}(1)\frac{Opt}{\epsilon} pm^{1/2}\sqrt{\ln(pm)}$?

43. page 19, next line: what do you exactly mean by ""most economical" dual formulation of (37)"?

44. page 20, line 7: Please check the number of arcs.

45. The numerical implementation shows that it actually can solve large-scale SDPs within 1% of gap. I was wondering if the actual implementation suffers from finite precision computation when more tight precisions is required.