Active Positive-Definite Matrix Completion
Supplementary Material

Charalampos Mavroforakis  Dóra Erdös  Mark Crovella  Evimaria Terzi
{cmav, edori, crovella, evimaria}@cs.bu.edu
Boston University

1 The FindCandidates function of Algorithm 3
In the FindCandidates function (line 4), Select finds all the single edges that can be added to the current mask graph \( G_0 \) while maintaining its chordal structure. To do that, we make use of the clique tree data structure as introduced by Ibarra [1]. Given a graph \( G=(V,E) \), the clique tree is a tree \( C=(V_C,E_C) \), in which each node is a maximal clique of \( G \), i.e., \( V_C \subset 2^V \). In our case the number of nodes \( |V_C| \) is \( O(n) \) because \( G \) is chordal [1]. Two cliques are connected in the clique tree if they share common nodes. According to Ibarra, retrieving the insertable edges \( e=(i,j) \), i.e., those that are attached to node \( i \in V \), requires running a depth-first search on \( C \) (line 4). For every clique \( c \in V_C \) that we visit and for each node \( j \in c \), we check if the edge \((i,j)\) is insertable. As a result, it takes \( O(n^3) \) time to retrieve the set of all insertable edges. The pseudocode of this procedure is summarized in Algorithm 3.

2 Exploring different score functions
In all the experiments that we reported above, we run our algorithms using the edge scoring scheme given by function \( \text{score} \), which we defined in Equation (5.3). In this experiment, we explore the performance of three other scoring schemes and show that indeed using \( \text{score} \) was our best option.

More specifically, we consider the following three scoring alternatives, which in turn define variations of Select.

- **data_score**: We compute the data_score as follows: for \( e(v,v')=x \) assume that \( H_x \) is the maximal leading minor that contains \( x \) and it has the form given in Equation (3.2). Then, we define \( \text{data-score}(e) \) to be the value
  \[
  \frac{1}{\det(A)|\{x \in Z : x \in I_x\}|} \sum_{x\in Z: x \in I_x} \det(H_x),
  \]
  where \( Z \) is the set of observations in \( H_x \).

  That is, for the calculation of the “expected” determinant, instead of using the integral over all values of \( x \in I_x \) (as in Equation (5.3)), we only use the values which appear in the matrix and are observations from the ground truth.

- **norm_score**: In the norm_score, we normalize the “expected” determinant of \( H \) by its maximum value. Using the same notation as above the norm_score is defined as follows:
  \[
  \text{norm-score}(e) = \frac{1}{\det(A)} \frac{1}{\max_{x \in I_x} \det(H_x)} \int_{x \in I_x} \det(H_x).
  \]

- **det_score**: Finally, in the det_score we remove the normalization by \( \det(A) \); in this scoring scheme the edge that maximizes \( \text{det-score} \) is the edge that maximizes the expected determinant of the corresponding maximal leading minor. That is, using the same notation as above \( \text{det-score} \) is defined as follows:
  \[
  \text{det-score}(e) = \int_{x \in I_x} \det(H_x).
  \]

We run Select with these different score variants as well as the original score function, defined as score in Equation (5.3). We also run for all the data the RandComplete algorithm and we report the LogDetRatio of the different versions of Select. For this experiment we use the synthetic partial matrices generated by random hiding. As in the budget experiment, we only reveal 25% of the original matrix as our observed entries.

Figure 1 summarizes our findings: the x-axis corresponds to the query budget, while the y-axis reports the LogDetRatio for score, data_score, norm_score and det_score. The dashed line corresponds to the LogDetRatio of an algorithm that has the same performance as RandComplete, i.e., our baseline. Anything above this line corresponds to an algorithm that performs worse than the baseline and anything below the dashed line corresponds to an algorithm that performs better than RandComplete. Clearly, using the score scoring with Select performs consistently the best. Almost as good is the performance of data_score. This is somehow expected as the score and data_score are defined using the same rationale in mind: they try to see
the impact of the value of $x$ in $\det(H_x)$ by subtracting the value of $\det(A)$, which consists of entries independent of $x$. The only difference between these two scoring schemes is that $\text{score}$ computes the expected value of $\det(H_x)$ using all possible values in $I_x$, while $\text{data\_score}$ computes the same value using the entries that appear in $H_x$ and are observations from the ground truth.

The results also showcase that $\text{norm\_score}$ is not as good as $\text{score}$ or $\text{data\_score}$, as it performs almost as good as the random baseline. Finally, $\det\_score$ is significantly worse than all other scoring schemes, including the random baseline. We conjecture that the reason for this is that $\det\_score$ does not isolate the impact of $x$ in the value of $\det(H_x)$, as it does not subtract the impact of $A$. As a result, the values appearing in $A$ may dominate the $\det\_score$, preventing this version of the algorithm from really evaluating $x$ itself.

References