Semidefinite Programming for Graph Partitioning with Preferences in Data Distribution

Suely Oliveira[1], David Stewart[2], and Takako Soma[1]

[1] The Department of Computer Science, The University of Iowa
Iowa City, IA 52242, USA
{oliveira,tsoma}@cs.uiowa.edu

[2] The Department of Mathematics, The University of Iowa
Iowa City, IA 52242, USA
dstewart@math.uiowa.edu

Abstract. Graph partitioning with preferences is one of the data distribution models for parallel computer, where partitioning and mapping are generated together. It improves the overall throughput of message traffic by having communication restricted to processors which are near each other, whenever possible. This model is obtained by associating to each vertex a value which reflects its net preference for being in one partition or another of the recursive bisection process. We have formulated a semidefinite programming relaxation for graph partitioning with preferences and implemented efficient subspace algorithm for this model. We numerically compared our new algorithm with a standard semidefinite programming algorithm and show that our subspace algorithm performs better.

1 The Graph Partitioning Problem and Parallel Data Distribution

Graph partitioning is universally employed in the parallelization of calculations on unstructured grids, such as finite element and finite difference calculations, whether using explicit or implicit methods. Once a graph model of a computation is constructed, graph partitioning can be used to determine how to divide the work and data for efficient parallel computation. The goal of the graph partitioning problem is to divide a graph into disjoint subgraphs subject to the constraint that each subgraph has roughly equal number of vertices, and with the objective of minimizing the number of edges that are cut by the partitionings. In many calculations the underlying computational structure can be conveniently modeled as a graph in which vertices correspond to computational tasks and edges reflect data dependencies. The objectives here are to evenly distribute the computations among the processors while minimizing interprocessor communication, so that the corresponding assignment of tasks to processors leads to efficient execution. Therefore, we wish to divide the graph into subgraphs with roughly equal numbers of nodes with the minimum number of edges crossing between the subgraphs. Graph partitioning is an NP-hard problem [7]. Therefore, heuristics need to be used to get approximate solutions for these problems.

The graph partitioning problem for high performance scientific computing has been studied extensively over the past decades. The standard approach is to use Recursive Bisection [9, 21]. In Recursive Bisection, the graph is broken in half, the halves are halved independently, and so on, until there are as many pieces as desired.

The justification for traditional partitioning is that the number of edges cut in a partition typically corresponds to the volume of communication in the parallel application. Since communication is an expensive operation, minimizing this volume is extremely important in achieving high performance. In traditional recursive partitioning, after each step in a recursive decomposition the subgraphs are decoupled and interact no further. An edge crossing between two sets does not affect the later partitioning of either set. Consequently, there is nothing preventing the two adjacent vertices from being assigned to processors that are quite far from each other. A message between distant processors must traverse many wires, which are therefore rendered unavailable to transmit other messages. Conversely, if each message consumes only a small number of wires, more messages can be sent at once. In parallel computing, messages traveling between architecturally distant processors should be minimized by improving the data locality, since they tie up many communication links. Therefore, a good mapping is one that reduces message congestion and thereby preserves communication bandwidth. Many scientific computing applications of interest, for example those employing an iterative sparse solver kernel, have a structure in which many messages simultaneously compete for limited communication bandwidth. Good mappings are especially important in these cases.

Recently, Hendrickson et al. [8, 9, 11] pointed out problems with traditional models. Assume we have already partitioned the graph into left and right halves, and that we have similarly divided the left-half graph into top and bottom quadrants (see Figure 1). When partitioning the right-half graph between processors 3 and 4, we want the messages to travel short distances. The mapping shown in the left-hand of Figure 1 is better since the total message distance is less than that for the right-hand figure.

These models are obtained by associating to each vertex a value which reflects its net preference for being in one subgraph or another. Note that this preference is a function only of edges that connect the vertex to vertices which are not in the current subgraph. These preferences should be propagated through the recursive partitioning process. If the graph partitioning problem with preferences is relaxed as is done to obtain spectral graph partitioning, we obtain an extended eigenproblem: Find the minimum μ for which there is a $y \neq 0$ satisfying $Ay = \mu y + g$ with a specified norm [11].

In [18] we developed subspace methods to solve extended eigenproblems. In [19] we have developed a subspace algorithm for a SDP of the original graph partitioning. In this paper we will develop a semidefinite program for graph

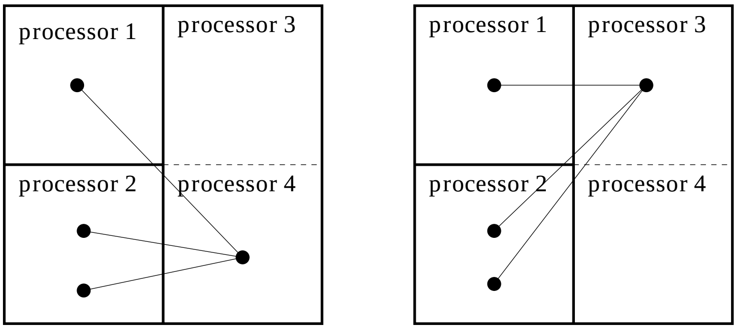

Fig. 1. Partition between the 4 processors

partitioning *with preferences* and present an efficient subspace algorithm for this model.

2 Spectral Graph Patitioning and Semidefinite Programming

Like combinatorial methods, spectral methods have proven to be effective for large graphs arising from FEM discretizations [21]. Software packages that use spectral methods combined with combinatorial algorithms include METIS [15], Chaco [10], JOSTLE [31], PARTY [22], and SCOTCH [20]. The spectral algorithms which are used in these packages are based on models that partition a graph by finding an eigenvector associated with the second smallest eigenvalue of its graph Laplacian matrix using an iterative method. This eigenvector model is used in [12, 13] for example.

Another approach to get better approximation of the graph partitioning problem is semidefinite programming. It gives tighter relaxations to the original graph partitioning problem than does spectral graph partitioning, leading to better partitionings.

The semidefinite programming problem [1, 24, 29] is the problem of minimizing a linear function over symmetric matrices with linear constraints and the constraint that the matrix is symmetric positive semi-definite. Semidefinite programming reduces to linear programming when all the matrices are diagonal. Currently there are various software packages for solving semidefinite programs available using interior-point algorithms, such as CSDP [4], SDPA [6], SDPpack [2], SDPT3 [28], SP [30], and a Matlab toolbox by Rendl [23]. Semidefinite programming relaxation technique for equal-partitioning problem has been developed in [14]. In the next section we develop an SDP relaxation for the GP problem with preferences. In Section 4 we present an efficient algorithm for the new model of Section 3

3 A Semidefinite Programming Model of the Graph Partitioning Problems with Preferences

The graph partitioning with preferences (GP/P) problem is as follows: Given an undirected graph $G = (V, E)$ and a preference vector $d = [d_i | i \in V]$, and if a partition is represented by a vector x where $x_i \in \{-1, 1\}$ depending on whether x_i belongs to set P_1 or P_2,

$$\begin{aligned} \min_x \frac{1}{4} \sum_{\{i,j\} \in E} (x_i - x_j)^2 - \frac{1}{2} \sum_{i \in V} d_i x_i &= \min_x \frac{1}{4} x^T L x - \frac{1}{2} d^T x \\ \text{Subject to (a) } & x_i = \pm 1, \ \forall i \\ \text{(b) } & \sum_{i \in V} x_i \approx 0, \end{aligned}$$

where L is the graph Laplacian matrix. A straight forward way of approximating this problem is by using the following relaxation:

$$\begin{aligned} \min_x \frac{1}{4} x^T L x - \frac{1}{2} d^T x \\ \text{Subject to (a) } x^T x = n \\ \text{(b) } e^T x = 0, \end{aligned}$$

where e is the vector of all ones.

The above relaxation replaces constraint (a) $x_i = \pm 1, \ \forall i$ with $\sum_i x_i^2 = x^T x = n$, where $n = |V|$. A subspace algorithms for solving this extended eigenvalue relaxation was developed in [18]. In this paper we will develop a new semidefinite programming relaxation for the GP/P problem. This development is based on the following theorem which shows that the GP/P problem above can be rewritten as a semi-definite programming with a rank one constraint. An SDP relaxation can then be obtained by dropping the rank one constraint. In order to efficiently solve the SDP, we use a subspace approach.

Theorem 1. *The GP/P*

$$\begin{aligned} \min_x \frac{1}{4} x^T L x - \frac{1}{2} d^T x \\ \textit{Subject to (a) } x_i = \pm 1, \ \forall i \\ \textit{(b) } \sum_{i \in V} x_i = 0 \end{aligned} \tag{1}$$

is equivalent to

$$\begin{aligned} \min_{\tilde{X}} \tilde{X} \bullet \frac{1}{4} \begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix} \\ \textit{Subject to (a) } & \tilde{X} \succeq 0 \\ \textit{(b) } & \xi = 1 \\ \textit{(c) } & \operatorname{diag}(X) = e \\ \textit{(d) } & X \bullet (ee^T) = 0 \\ \textit{(e) } & \operatorname{rank}(X) = 1, \end{aligned} \tag{2}$$

where $\tilde{X} = \begin{pmatrix} \xi & u^T \\ u & X \end{pmatrix}$*, and the solution to (2) is* $\tilde{X} = \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix}$*, where* x *is the solution of (1).*

Proof:

(i) If x solves (1), then for $\tilde{X} = \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix}$, the constraints are: $\tilde{X} \succeq 0$, $\operatorname{diag}(xx^T) = [x_i^2 | i \in V] = e$, and $X \bullet (ee^T) = \operatorname{tr}(Xee^T) = \operatorname{tr}(xx^Tee^T) = \operatorname{tr}(e^Txx^Te) = (e^Tx)^2 = 0$. Since

$$\begin{aligned}
\tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix} &= \begin{pmatrix} 1 & x^T \\ x & xx^T \end{pmatrix} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \\
&= \frac{1}{4}\operatorname{tr}\begin{pmatrix} -x^Td & -d^T + x^TL \\ -xx^Td & xx^TL - xd^T \end{pmatrix} \\
&= \frac{1}{4}(\operatorname{tr}(xx^TL) - 2d^Tx) \\
&= \frac{1}{4}L \bullet xx^T - \frac{1}{2}d^Tx \\
&= \frac{1}{4}x^TLx - \frac{1}{2}d^Tx,
\end{aligned}$$

we have $\tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix} = \frac{1}{4}x^TLx - \frac{1}{2}d^Tx$, so the value of (2) is less than or equal to the value of (1).

(ii) On the other hand, suppose $\tilde{X} = \begin{pmatrix} \xi & u^T \\ u & X \end{pmatrix}$ solves (2). Since $\operatorname{rank}(X) = 1$, and $X \succeq 0$, $X = zz^T$ for some z. Also $\xi = 1$, so

$$\tilde{X} = \begin{pmatrix} 1 & u^T \\ u & zz^T \end{pmatrix} \succeq 0.$$

By taking one step of Cholesky factorization, we see that $\tilde{X}$ is positive semidefinite if and only if

$$zz^T - uu^T \succeq 0.$$

If we write $u = \beta z + w$, where $w \perp z$, then,

$$\begin{aligned}
w^T(zz^T - uu^T)w &= w^T(zz^T - (\beta z + w)(\beta z + w)^T)w \\
&= w^T(zz^T - (\beta^2 zz^T + \beta zw^T + \beta wz^T + ww^T))w \\
&= w^T(zz^T - \beta^2 zz^T - \beta zw^T - \beta wz^T - ww^T)w \\
&= w^T(-ww^T)w \\
&= -(w^Tw)^2 \geq 0.
\end{aligned}$$

So, $w = 0$. Therefore, $u = \beta z$. Then,

$$\begin{aligned}
zz^T - uu^T &= zz^T - (\beta z)(\beta z)^T \\
&= zz^T - \beta^2 zz^T \\
&= (1 - \beta^2)zz^T \succeq 0
\end{aligned}$$

is equivalent to $|\beta| \leq 1$, provided $z \neq 0$. If $z = 0$, then $zz^T - uu^T \succeq 0$, implies $u = 0$. In either case, $u = \beta z$, where $|\beta| \leq 1$.

Since $\tilde{X}$ minimizes $\tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix}$ over the constraints in (2), and the only constraint in (2) involving u is $\tilde{X} \succeq 0$ (which we just showed is equivalent to $u = \beta z$, where $|\beta| \leq 1$). We can then rewrite the problem in terms of variable z in the following way

$$\begin{aligned}
\min_u \tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} &= \min_u \begin{pmatrix} 1 & u^T \\ u & zz^T \end{pmatrix} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \\
&= \min_u \frac{1}{4}\mathrm{tr}\begin{pmatrix} -u^T d & -d^T + u^T L \\ -zz^T d & -ud^T + zz^T L \end{pmatrix} \\
&= \min_u \frac{1}{4}(\mathrm{tr}(zz^T L) - 2d^T u) \\
&= \min_u \frac{1}{4} L \bullet zz^T - \frac{1}{2} d^T u \\
&= \min_u \frac{1}{4} z^T L z - \frac{1}{2} d^T u
\end{aligned}$$

$$\begin{aligned}
\text{Subject to (a) } & u = \beta z \\
\text{(b) } & |\beta| \leq 1.
\end{aligned}$$

This minimum value is

$$\min_{-1 \leq \beta \leq +1} \frac{1}{4} z^T L z - \frac{1}{2}\beta d^T z = \frac{1}{4} z^T L z - \frac{1}{2}|d^T z|.$$

We can assume without loss of generality that $d^T z \geq 0$, since if we change the sign of z then $X = zz^T$ is unchanged. So, this minimum value over u is

$$\frac{1}{4} z^T L z - \frac{1}{2} d^T z.$$

Furthermore, the optimal choice of u is $u = z$. So the value of (2) is

$$\begin{aligned}
&\min_z \frac{1}{4} z^T L z - \frac{1}{2} d^T z \\
&\text{Subject to (a) } \mathrm{diag}(zz^T) = e \\
&\qquad\qquad \text{(b) } (zz^T) \bullet (ee^T) = 0,
\end{aligned}$$

which is,

$$\begin{aligned}
&\min_z \frac{1}{4} z^T L z - \frac{1}{2} d^T z \\
&\text{Subject to (a) } z_i^2 = 1, \ \forall i \\
&\qquad\qquad \text{(b) } e^T z = 0.
\end{aligned}$$

Therefore, the value of (2) is

$$\begin{aligned}
&\min_z \frac{1}{4} z^T L z - \frac{1}{2} d^T z \\
&\text{Subject to (a) } z_i = \pm 1, \ \forall i \\
&\qquad\qquad \text{(b) } \sum_{i \in V} z_i = 0.
\end{aligned}$$

So, the minimum value of (2) is the minimum value of (1), and the solutions are related by $X = xx^T$ and $u = x$. □

The semidefinite programming relaxation removes the constraint $\mathrm{rank}(X) = 1$:

$$\begin{array}{l} \min\limits_{\tilde{X}} \tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix} \\ \text{Subject to (a) } \tilde{X} \succeq 0 \\ \qquad\qquad\quad \text{(b) } \xi = 1 \\ \qquad\qquad\quad \text{(c) } \mathrm{diag}(X) = e \\ \qquad\qquad\quad \text{(d) } X \bullet (ee^T) = 0, \end{array}$$

where $\tilde{X} = \begin{pmatrix} \xi & u^T \\ u & X \end{pmatrix}$. To obtain the resulting partition, use the u vector from $\tilde{X}$ just as the Fiedler vector is used in spectral graph partitioning.

Constraints (b) and (c) can be combined together as $\mathrm{diag}(\tilde{X}) = e$, where e is now a vector of length $n + 1$. Therefore,

$$\begin{array}{l} \min\limits_{\tilde{X}} \tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d^T & L \end{pmatrix} \\ \text{Subject to (a) } \tilde{X} \succeq 0 \\ \qquad\qquad\quad \text{(b) } \mathrm{diag}(\tilde{X}) = 1 \\ \qquad\qquad\quad \text{(c) } X \bullet (ee^T) = 0, \end{array}$$

It is easily shown that for the particular case when the graph partitioning problem is assumed without preferences (i.e. $d = 0$), the above SDP relaxation corresponds to the SDP relaxation of graph partitioning in [19].

4 Solving the New SDP Relaxation Efficiently

While solving semidefinite programs is believed to be a polynomial time problem, current algorithms typically take $\Omega(n^3)$ time and $\Omega(n^2)$ space, making them impractical for realistic data distribution problems. This makes the development of efficient algorithms important.

Subspace methods build sequences of subspaces that are used to compute approximate solutions of the original problem. They have been previously used to solve systems of linear equations [16] and to compute eigenvalues and eigenvectors [3]. Of the methods for eigenvalues, the Lanczos and Arnoldi methods use Krylov subspaces (generated using a single matrix and a starting vector), while Davidson-type methods (Generalized Davidson [25], Jacobi–Davidson [26]) use Rayleigh matrices. Theoretical studies of the Krylov subspace methods are well-known in the numerical analysis community, but they have only recently been developed for Davidson-type methods [5, 17]. Davidson-type subspace methods have been applied to spectral graph partitioning [12, 13].

A subspace algorithm for the SDP relaxation of the spectral partitioning model was developed in [17]. Below we describe a subspace algorithm for the semidefinite programming relaxation described above, in order to efficiently find high-quality solutions to the graph partitioning problem with preferences.

$$\begin{aligned}
&\min_{\tilde{X}} \tilde{X} \bullet \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \\
&\text{Subject to (a) } \tilde{X} \succeq 0 \\
&\qquad\qquad\text{(b) } \xi = 1 \\
&\qquad\qquad\text{(c) } \operatorname{diag}(X) = e \\
&\qquad\qquad\text{(d) } X \bullet (ee^T) = 0,
\end{aligned}$$

where $\tilde{X} = \begin{pmatrix} \xi & u^T \\ u & X \end{pmatrix}$. We define the operator $\mathcal{A}$ by $\mathcal{A}(\tilde{X})_i = \mathcal{A}_i \bullet \tilde{X} = \operatorname{trace}(\mathcal{A}_i\tilde{X})$ for $i = 1, 2, \ldots, m$ where each $\mathcal{A}_i$ is an $(n+1) \times (n+1)$ matrix: $\mathcal{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & ee^T \end{pmatrix}$, $\mathcal{A}_2 = \operatorname{diag}(1, 0, 0, \ldots, 0)$, $\mathcal{A}_3 = \operatorname{diag}(0, 1, 0, \ldots, 0)$, $\ldots$, $\mathcal{A}_m = \operatorname{diag}(0, 0, 0, \ldots, 1)$, where $m = n + 2$. Now consider the vector $a = (a_1, a_2, a_3, \ldots, a_{m+1})^T = (0, 1, 1, \ldots, 1)^T$ Note that $\mathcal{A}_1 \bullet \tilde{X} = a_1$ ensures that (d) $X \bullet (ee^T) = 0$; $\mathcal{A}_2 \bullet \tilde{X} = a_2$ ensures that (b) $\xi = 1$; and $\mathcal{A}_3 \bullet \tilde{X} = a_3$, $\ldots$, $\mathcal{A}_m \bullet \tilde{X} = a_m$ ensures that (c) $\operatorname{diag}(X) = e$.

Notice that the subspace framework here is similar to that of [17], the input data for the subspace SDP being the main difference. Below we present the main steps of our subspace semidefinite programming for extended eigenproblems.

Algorithm 1 – Subspace Semidefinite Programming for SDP Relaxation of Graph Partitioning with Preferences

1. Define $(n+1) \times (n+1)$ matrix $C = \frac{1}{4}\begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix}$, $m \times 1$ vector a, and an operator $\mathcal{A}$ which is a set of m number of $(n+1) \times (n+1)$ matrices.
2. Compute an interior feasible starting point y for the original problem.
3. Define $(n+1) \times 3$ vector V_1 and $m \times 3$ vector W_1. The first column of V_1 should be the e_1, the first standard basis vector, and the the first two columns of W_1 should be e_1 and e_2 respectively.
4. Compute the semidefinite programming solution $(\hat{X}, \hat{y})$ on the subspace. The projected matrix, operator and vector on the subspace are $\hat{C} = V_j^T C V_j$, $\hat{\hat{\mathcal{A}}}(\hat{X}) = W_j^T \hat{\mathcal{A}}(\hat{X}) = W_j^T \begin{pmatrix} V_j^T A_1 V_j \bullet \hat{X} \\ \vdots \\ V_j^T A_m V_j \bullet \hat{X} \end{pmatrix}$ and $\hat{a} = W_j^T a$, where V_j and W_j are the current orthogonal bases.
5. Estimate the "residual" matrix $\tilde{R} = C - \mathcal{A}^T(y)$ and compute r which is an eigenvector corresponding to the minimum eigenvalue of $\tilde{R}$.
6. Orthonormalize r against the current orthogonal basis V_j. Append the orthonormalized vector to V_j to give V_{j+1}.
7. Estimate residual $p = \mathcal{A}(X) - a = \mathcal{A}(V_j \hat{X} V_j^T) - a$.

8. Orthonormalize p against the current orthogonal basis W_j. Append the orthonormalized vector to W_j to give W_{j+1}.
9. Compute $\tilde{X} = V_j \hat{X} V_j^T$ of original size, where $\tilde{X} = \begin{pmatrix} \xi & x^T \\ x & X \end{pmatrix}$.

5 Numerical Results

Here we describe how to reduce the number of constraints in original problem. The semidefinite programming formulation of extended eigenproblem is written as follows:

$$\begin{aligned} \min_{\tilde{X}} \frac{1}{4} \begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \bullet \tilde{X} \\ \text{Subject to (a) } & \tilde{X} \succeq 0 \\ \text{(b) } & \xi = 1 \\ \text{(c) } & \mathrm{diag}(X) = e \\ \text{(d) } & X \bullet (ee^T) = 0, \end{aligned}$$

where $\tilde{X} = \begin{pmatrix} \xi & x^T \\ x & X \end{pmatrix}$ and $X = xx^T$. The constraints (b) and (c) can be combined together.

$$\begin{aligned} \min_{\tilde{X}} \frac{1}{4} \begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \bullet \tilde{X} \\ \text{Subject to (a) } & \tilde{X} \succeq 0 \\ \text{(b) } & \mathcal{A}(\tilde{X}) = a \\ \text{(c) } & X \bullet (ee^T) = 0. \end{aligned}$$

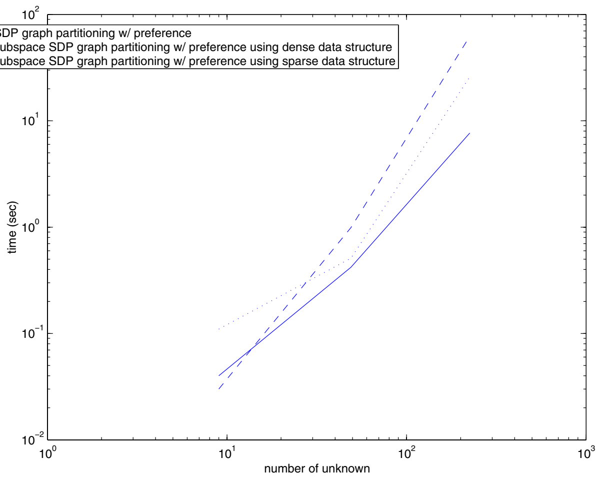

Fig. 2. Timings comparing subspace SDP extended eigenproblem against SDP extended eigenproblem

There are n number of constrains from (c) in (3). We rewrite the problem so that we can reduce the number of constraints in the original problem. The operator $\mathcal{A}$ is defined in terms of m diagonal matrices A_i where $(A_i)_{jj}$ is one if $j \equiv i$ (mod m), and zero otherwise, $i = 1, \cdots, m$ and $m \leq n$:

$$\begin{aligned} &\min_{\tilde{X}} \frac{1}{4} \begin{pmatrix} 0 & -d^T \\ -d & L \end{pmatrix} \bullet \tilde{X} \\ &\text{Subject to (a) } \tilde{X} \succeq 0 \\ &\qquad\qquad\text{(b) } \mathcal{A}(\tilde{X}) = (n/4m)/e \\ &\qquad\qquad\text{(c) } X \bullet (ee^T) = 0, \end{aligned}$$

where e is the vector of ones of the appropriate size.

The new algorithms have been implemented within the existing semidefinite software, CSDP [4] and matrix computation was utilized by Meschach [27]. The codes were run on a HP VISUALIZE Model C240 workstation, with a 326MHz PA-8200 processor and 512MB RAM.

The above subspace algorithm and CSDP were run with square matrices of various sizes. The subspace algorithm was implemented using both dense data structures and sparse data structures. Figure 2 compares the observed running timings for using CSDP directly, and the two implementations of the subspace algorithm. The vertical-axis shows the time in seconds. The horizontal-axis shows the number of unknowns: 9, 49, and 225. The number of constraints of the original problem for both methods was reduced to 5 for the matrix with 9 unknowns, 9 for the matrix with 49 unknowns, and 17 for the matrix with 225 unknowns, as described above. From this graph, we can see that the subspace algorithm takes less than the original algorithm as the number of unknowns increases. And implementation using sparse data structure perform much better. The timing for original method and subspace method with both dense and sparse data structures are summarized in Table 1, Table 2, and Table 3.

Table 1. Timings for SDP extended eigenproblem with reduced number of constraints

size of unknowns	9	49	225
number of constraints	5	9	17
running time (sec)	0.030	0.990	62.440

Table 2. Timings for subspace SDP extended eigenproblem using dense data structure with reduced number of constraints

size of unknowns	9	9	49	225
number of constraints	5	5	9	17
number of iterations	3	5	7	17
$\|p\|_2$	1.19×10^{-6}	5.25×10^{-7}	1.08×10^{-5}	9.54×10^{-5}
running time (sec)	0.110	0.120	0.510	30.100

Table 3. Timings for subspace SDP extended eigenproblem using sparse data structure with reduced number of constraints

size of unknowns	9	9	49	225
number of constraints	5	5	9	17
number of iterations	3	5	7	15
number of Arnoldi iterations	9	9	30	30
$\|p\|_2$	1.86×10^{-6}	6.24×10^{-6}	3.083×10^{-5}	4.10×10^{-4}
running time (sec)	0.040	0.130	0.420	7.690

The Table 4 shows the behavior of $\|p\|_2$ as the subspace was expanded in subspace SDP extended eigenproblem using dense data structure. Data was taken for the matrix with 9 unknowns and number of constraints is 5, the matrix with 49 unknowns and number of constraints is 9, and the matrix with 225 unknowns and number of constraints is 17. As you can see, all the cases converged at the end. The Table 5 is the result for subspace SDP extended eigenproblem using sparse data structure. The behavior of the $\|p\|_2$ is same as that for dense data structure.

Table 4. $\|p\|_2$ of subspace SDP extended eigenproblem using dense data structure with reduced number of constraints for matrices with number of unknown = 9, 49, and 225

size of unknowns	9	49	225
number of constraints	5	9	17
number of iteration			
1	68.7507477	12.4849939	15.341258
2	7.34615088	3.8352344	14.1316462
3	1.19470394e-06	4.51103926	1218.76562
4	5.49540289e-07	3.52121067	2978.45337
5	5.24935558e-07	4.89502668	16.0653
6		0.531660855	3359.63013
7		1.07650176e-05	17.5198021
8		2.97533984e-06	5.35400295
9		36342.1484	17.7656918
10			7.57762194
11			696.369324
12			5.29387665
13			737332.562
14			484.507904
15			16.7516613
16			0.000186197911
17			9.53856725e-05

Table 5. $||p||_2$ of subspace SDP extended eigenproblem using sparse data structure with reduced number of constraints for matrices with number of unknown = 9, 49, and 225

size of unknowns	9	49	225
number of constraints	5	9	17
number of iteration			
1	68.7507477	12.4849596	16.1067295
2	7.34615374	4.14506721	15.4141665
3	1.86033265e-06	5.64862299	16.7704239
4	7.83976702e-07	3.44969082	799.051331
5	6.24131007e-06	4.71650219	29.9862556
6		1.86342728	7.08970928
7		3.08254312e-05	11.917738
8		33748.1367	844.419495
9		36664.4961	14.5084686
10			447.563751
11			4.25445318
12			56.6501083
13			10.7289038
14			0.814486623
15			0.000409778644
16			11.1356792
17			201.696182

References

[1] F. Alizadeh. Interior-point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization,* 5(1):13-51, 1995. 705

[2] F. Alizadeh, J. P. A. Haeberly, M. V. Nayakkankuppam, M. L. Overton, and S. Schmieta. SDPpack user's guide - version 0.9 beta for Matlab 5.0. Technical Report TR1997-737, Computer Science Department, New York University, New York, NY, June 1997. 705

[3] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenproblem. *Quarterly Applied Mathematics,* 9:17-29, 1951. 709

[4] B. Borchers. CSDP, 2.3 User's Guide. *Optimization Methods and Software,* 11(1):597-611, 1999. 705, 712

[5] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.,* 15(1):62-76, 1994. 709

[6] K. Fujisawa, M. Kojima, and K. Nakata. SDPA User's Manual - Version 4.50. Technical Report B, Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan, July 1999. 705

[7] M. R. Carey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. *Theoretical Computer Science,* 1:237-267, 1976. 704

[8] B. Hendrickson. Graph partitioning and parallel solvers: Has the emperor no clothes? (extended abstract). In *Lecture Notes in Computer Science,* volume 1457, 1998. 704

[9] B. Hendrickson and T. G. Kolda. Graph partitioning models for parallel computing. *Parallel Comput.,* 26(12):1519-1534, 2000. 704

[10] B. Hendrickson and R. Leland. The Chaco user's guide, version 2.0. Technical Report SAND-95-2344, Sandia National Laboratories, Albuquerque, NM, July 1995. 705

[11] B. Hendrickson, R. Leland, and R. Van Driessche. Enhancing data locality by using terminal propagation. In *Proc. 29th Hawaii Intl. Conf. System Science,* volume 16, 1996. 704

[12] M. Holzrichter and S. Oliveira. A graph based Davidson algorithm for the graph partitioning problem. *International Journal of Foundations of Computer Science,* 10:225-246, 1999. 705, 709

[13] M. Holzrichter and S. Oliveira. A graph based method for generating the Fiedler vector of irregular problems. In *Lecture Notes in Computer Science,* volume 1586, pages 978-985. Springer, 1999. Proceedings of the 11th IPPS/SPDP'99 workshops. 705, 709

[14] S. E. Karisch and F. Rendl. Semidefinite programming and graph equipartition. In P. M. Pardalos and H. Wolkowicz, editors, *Topics in Semidefinite and InteriorPoint Methods,* volume 18, pages 77-95. AMS, 1998. 705

[15] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system Version 2.0. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, August 1995. 705

[16] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Research Nat'l Bureau of Standards,* 49:33-53, 1952. 709

[17] S. Oliveira. On the convergence rate of a preconditioned subspace eigensolver. *Computing,* 63(2):219-231, December 1999. 709, 710

[18] S. Oliveira and T. Soma. A multilevel algorithm for spectral partitioning with extended eigen-models. In *Lecture Notes in Computer Science,* volume 1800, pages 477-484. Springer, 2000. Proceedings of the 15th IPDPS 2000 workshops. 704, 706

[19] S. Oliveira, D. Stewart, and T. Soma. A subspace semidefinite programming for spectral graph partit ioning. In P.M.A Sloot, C.K.K. Tan, J.J. Dongarra, and A. G. Hoekstra, editors, *Lecture Notes in Computer Science,* volume 2329, pages 10581067. Springer, 2002. Proceedings of International Conference on Computational Science-ICCS 2002, Part 1, Amsterdam, The Netherlands. 704, 709

[20] F. Pellegrini. SCOTCH 3.1 user's guide. Technical Report 1137-96, Laboratoire Bordelais de Recherche en Informatique, Universite Bordeaux, France, 1996. 705

[21] A. Pothen, H. D. Simon, and Kang-Pu K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.,* 11(3):430-452, 1990. 704, 705

[22] R. Preis and R. Diekmann. The PARTY Partitioning-Library, User Guide - Version 1.1. Technical Report tr-rsfb-96-024, University of Paderborn, Germany, 1996. 705

[23] F. Rendl. A Matlab toolbox for semidefinite programming. Technical report, Technische Universitdt Graz, Institut fiir Mathematik, Kopernikusgasse 24, A-8010 Graz, Austria, 1994. 705

[24] F. Rendl, R. J. Vanderbei, and H. Wolkowicz. primal-dual interior point algorithms, and trust region subproblems. *Optimization Methods and Software,* 5:1-16, 1995. 705

[25] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Oxford Road, Manchester M13 9PL, UK, 1992. 709

[26] G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.,* 17(2):401-425, 1996. Max-min eigenvalue problems, 709

[27] D. E. Stewart and Z. Leyk. *Meschach: Matrix Computations in C.* Australian National University, Canberra, 1994. Proceedings of the CMA, # 32. 712

[28] K. C. Toh, M. J. Todd, and P. H. Tiitüncii. SDPT3 - a Matlab software package for semidefinite programming, version 2.1. Technical report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, September 1999. 705

[29] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review,* 38:49-95, 1996. 705

[30] L. Vandenberghe and S. Boyd. SP Software for semidefinite programming User's guide, version 1.0. Technical report, Information System Laboratory, Stanford University, Stanford, CA, November 1998. 705

[31] C. Walshaw, M. Cross, and M. Everett. Mesh partitioning and load-balancing for distributed memory parallel systems. In B. Topping, editor, Proc. *Parallel & Distributed Computing for Computational Mechanics, Lochinver, Scotland,* 1998. 705