Fast and Efficient Eigensolvers for Sparse Similarity and Graph Connectivity Matrices

1 Overview and Objectives

This research will propose fast and efficient eigensolvers for large-scale sparse Hermitian matrices with applications in graph connectivity analysis arising in graph feature extraction, anomaly detection, data clustering problems [11, 6, 2] of the interest of Oracle Lab. The matrix size considered in this proposal is of order more than millions, and, hence, no efficient matrix factorization or simple preconditioner is affordable to speed up traditional iterative eigensolvers, not to mention direct eigensolvers that require huge computer memory. Given such a large-scale sparse Hermitian matrix $A \in \mathbb{C}^{N\times N}$ stored in a sparse matrix format (e.g., the coordinate list format (COO) that only stores nonzero entries and their indices), it is assumed that only matrix-vector multiplication (matvec) is affordable and the goal is to compute the largest (or the smallest) k eigenpairs of A. This research will explore the Chebyshev-Davidson method [12] and propose a novel coordinate-wise descent method for leading eigenpairs in both sequential computing in year one and parallel computing in years two and three. Accordingly, two specific research aims are as follows.

Aim 1: Develop the Chebyshev-Davidson method for sparse similarity matrices of graphs. Davidson-type algorithms have been an efficient eigensolver for many applications due to their fast convergence and robustness [1, 9], but they require a good preconditioner for solving the correction-equation in each iteration of the algorithm, which is not affordable for large-scale graph similarity matrices. The Chebyshev-Davidson method [12] replaces this expensive step by a Chebyshev polynomial filtering step that only requires $O(1)$ matvec of A in each iteration. Polynomial filters amplify components of the desired eigenvectors in the search subspace, which can reduce both the number of steps required for convergence and the cost in orthogonalization and restart in the original Davidson-type methods. Therefore, in the case when the matvec of A and the orthogonalization of the current search are affordable, the Chebyshev-Davidson method can be a fast and efficient eigensolver for sparse similarity and graph connectivity matrices. We will explore the sparsity of these matrices, the sparsity of eigenvectors, and parallel computing to further accelerate the computation in this proposal.

Aim 2: Design the coordinate-wise descent method for identifying the leading eigen-subspace of sparse similarity matrices of graphs. By reformulating the leading eigenvalue problem as a non-convex optimization problem, we will propose novel coordinate-wise descent methods to solve the optimization efficiently with a theoretical guarantee for global minimizers to identify the leading eigen-subspace. After this, an eigenvalue problem of size only k by k is required to identify the leading eigenpairs of A. In each iteration, computation with only $O(1)$ memory and flops is needed, making the coordinate-wise descent method applicable to large-scale graph problems. With adaptive selection strategies for choosing the coordinates, the convergence of the proposed method will be accelerated to make it highly attractive. Parallel computing and the sparsity of A and its eigenvectors will be further explored to accelerate the computation. Compared with the Chebyshev-Davidson method, no orthogonalization is required in each iteration of the coordinate-wise descent method, making it attractive for distributed memory parallel computing because distributed memory parallel orthogonalization is expensive.

2 Expected significance

In this project, we expect to build a mathematical software package for solving large-scale eigenvalue problems for sparse Hermitian matrices based on Matlab in the first year and C++ or Fortran for distributed memory parallel computing in the second and third year. It is expected that this package can handle matrices of size 10^6 to 10^8 on a single workstation and the matrix size can be much larger with distributed memory parallel computing. The numerical performance of this software will be robust and efficient with a theoretical guarantee.

3 Research Plans

In this section, the Chebyshev-Davidson method and the coordinate-wise descent method will be introduced in detail with preliminary results to support this proposal.

3.1 Chebyshev-Davidson Method

3.1.1 Problem Setup

Without loss of generality, it is assumed that a positive semi-definite Hermitian sparse matrix $A \in \mathbb{C}^{N\times N}$ is given. If A is not positive semi-definite $A + \mu I$ with a sufficiently large $\mu > 0$ will be positive definite. It is further assumed that each row of A has n nonzero entries with $n = O(1)$ independent of N due to the sparse connectivity of graphs considered in Oracle lab. Since the largest eigenpairs of A are the smallest eigenpairs of $-A$, only the algorithm for identifying the smallest k eigenpairs of A will be presented. It is straightforward to extend it to find the largest k eigenpairs. In sum, suppose (λ_i, v_i) for $i = 1, \dots, N$ are the N eigenpairs of A with eigenvalues $0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_N$. Our goal is to identify (λ_i, v_i) for $i = 1, \dots, k$. It is assumed that $\lambda_k < \lambda_{k+1}$ to make the target eigenvectors belong to a unique eigen-subspace.

3.1.2 Algorithm Description

Overview. The Chebyshev-Davidson method [12] applies a Chebyshev polynomial filter to accelerate the convergence of the Davidson method [1]. The Davidson method can augment the searching subspace for eigenvectors by a potentially much better new vector than the one based on a strict Krylov subspace structure, resulting in faster convergence. However, the original Davidson method [1] was designed for diagonally dominant matrices arising from quantum chemistry instead of large-scale sparse matrices in graph analysis. Besides, the augmentation vector added to the subspace at each step requires solving a correction-equation that is not affordable for big data, even though the Jacobi-Davidson method [9] has been designed so as to favor the efficient use of modern iterative techniques for the correction-equation, based on preconditioning and Krylov subspace acceleration. Compared to all other kinds of Davidson-type methods, there is no need to form or solve any correction-equations in the Chebyshev-Davidson method, instead, interval-wise filtering based on Chebyshev polynomials are utilized, making it very suitable for large-scale graph similarity matrices of size at least $O(10^6)$. The Chebyshev filter can enhance the eigensubspace of interest and dampen the eigensubspace undesired, making the Chebyshev-Davidson method efficiently applicable to general matrices including the sparse Hertimian matrices in graph analysis.

```
1 Goal: Given $a$, $b$, $a_0$, and a vector $v$, compute $u = \phi(A)v$.
2 Denote this algorithm as $u = \mathrm{Cheb}(v, m, a, b, a_0)$.
3 Let $e = (b-1)/2$, $c = (b+a)/2$, $\sigma = e/(a_0 - c)$, and $\sigma_1 = \sigma$.
4 Compute $y = (Av - cv)\sigma_1/e$.
5 for $i = 2$ to $m$ do
6     Compute $\sigma_{\mathrm{new}} = \frac{1}{2/\sigma_1 - \sigma}$.
7     Compute $y_{\mathrm{new}} = 2(Ay - cy)\sigma_{\mathrm{new}}/e - \sigma\sigma_{\mathrm{new}} v$.
8     Let $v = y$, $y = y_{\mathrm{new}}$, and $\sigma = \sigma_{\mathrm{new}}$.
```

Algorithm 1: The application of the Chebyshev polynomial filter $\phi(A)$ of degree m that dampens the eigenvalue of A in $[a, b]$ while magnifying the eigenvalues in the interval to the left of $[a, b]$. a_0 is a crude approximation of the smallest eigenvalue of A.

Chebyshev Polynomial Filter. Let $V \in \mathbb{C}^{N\times N}$ be the unitary matrix consisting of the N eigenvectors $\{v_i\}_{i=1}^N$ of the target Hermitian matrix A and let $\Lambda \in \mathbb{R}^{N\times N}$ be a diagonal matrix with the i-th diagonal entry as λ_i, the i-th eigenvalue of A. For any polynomial $\phi(x) : \mathbb{R} \to \mathbb{R}$, it holds

$$\phi(A) = V\phi(\Lambda)V^*, \tag{1}$$

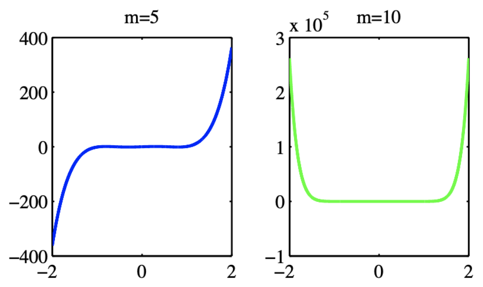

Figure 1: The configuration of Chebyshev polynomials of degree $m = 5$ and 10. The function magnitude quickly grows to infinity outside $[-1, 1]$.

where * denotes the conjugate transpose of a matrix. Here, we consider Chebyshev polynomials because of two advantages. First of all, the Chebyshev polynomial $C_m(x)$ of degree m defined by

$$C_m(x) = \begin{cases} \cos(m \cos^{-1}(x))), & -1 \leq x \leq 1, \\ \cosh(m \cosh^{-1}(x)), & |x| > 1, \end{cases}$$

rapidly grows outside the interval $[-1, 1]$ as illustrated in Figure 1. By (1), if $\phi(x) = C_m(ax + b)$ with appropriate a and b, then the smallest k eigenvalues of A becomes the largest k eigenvalues of $\phi(A)$ and they are significantly much larger than other eigenvalues, making they well separated from others, which means that it is relatively much easier to identify the leading k eigenpairs of $\phi(A)$ using the Davidson method without correction-equation. Note that the eigenvectors of A and $\phi(A)$ remain unchanged by (1). Therefore, it is sufficient to find the k largest eigenpairs of $\phi(A)$ essentially using the Davidson method without correction-equation. Secondly, Chebyshev polynomials admit a three-term recurrence relation such that $\phi(A)$ can be efficiently applied to an arbitrary vector v as long as the fast matvec of A is available. For example, $\phi(A) = C_m(aA + b)$ can be applied to an arbitrary vector v efficiently following Algorithm 1 via the following recursive computation:

$$C_{k+1}(aA + b)v = 2(aA + b)C_k(aA + b)v - C_{k-1}(aA + b)v,$$

for $k = 1, \ldots, m - 1$, $C_0(aA + b) = I_N$, and $C_1(aA + b) = aA + b$, where I_N is the identify matrix of size N. The computation above only requires a fast algorithm for the matvec of A, which is available since A is sparse.

Chebyshev-Davidson method and Features. Incorporating the Chebyshev polynomial filter into the traditional Davidson method gives the Chebyshev-Davidson method in Algorithm 2. The Chebyshev-Davidson method in Algorithm 2 admits several favorable features for large-scale sparse Hermitian matrices arising in graph analysis. 1) **Matrix-free**, i.e. does not require storing the coefficient matrix explicitly, but can access the matrix by evaluating matrix-vector products. 2) **Factorization-free**, i.e. does not require any matrix decomposition for solving linear systems, though a small eigenvalue problem of size at most k by k is required. 3) **The costs per iteration and the RAM memory** are $O(Nn(m + k) + kN + k^3)$ and $O(N(n + k))$, respectively, where n is the number of nonzero entries of A per row and m is the degree of the Chebyshev polynomial. 4) **Chebyshev polynomial acceleration** works for general matrices and is faster than typical eigensolvers, e.g., LOBPCG [4], the Jacob-Davidson method [9]. 5) **Blocking** allows utilizing highly efficient matrix-matrix operations, e.g., BLAS3, to compute multiple eigenpairs simultaneously.

The Chebyshev-Davidson method originally proposed in [12] focuses on quantum chemistry problems and does not take full advantage of the sparsity of eigenvectors and parallel computing. In this proposal, we will explore the application to graph analysis, the sparsity structure of similarity matrices and their eigenvectors, and parallel computing for big data.

It is worth mentioning that the Chebyshev-Davidson method as well as the LOBPCG and the Jacob-Davidson method require orthogonalization that is expensive in distributed memory parallel computing. All these methods also require the access of all nonzero entries of A and all entries of the desired eigenvectors, resulting in at least $O(N)$ computational and memory cost. These two requirements may be too expensive for extra-scale computing, which is the main motivation to propose a novel coordinate-wise descent method later in this proposal.

```
1  Input: $x$-initial vector; $m$-polynomial degree; $k_{keep}$- the number of vectors to keep during restart;
   $dim_{\max}$-maximum subspace dimension; $\tau$-convergence tolerance.
2  Output: converged eigenvalues $eval(1:k_c)$ (in non-increasing order) and their corresponding
   eigenvectors $V(:,1:k_c)$, where $k_c$ denotes the number of converged eigenpairs.
3  Start with the unit vector $x$, $V = x$.
4  Compute $W = Ax$, $H = \mu$, where $\mu = x^*W$.
5  Compute the residual vector: $r = W(:,1) - \mu x$.
6  if $\|r\| \leq \tau$ then
7      Set $eval(1) = \mu$, $k_c = 1$, $H = []$.
8  else
9      Set $k_c = 0$.
10 Estimate the upper bound (denoted as $b$) of eigenvalues.
11 Set the lower bound of eigenvalues $a = (b + \mu)/2$, $a_0 = b$.
12 Set $k_{sub} = 1$ ($k_{sub}$ stores the current subspace dimension).
13 while $iter \leq iter_{\max}$ do
14     Call the Chebyshev polynomial filter: $t = \mathrm{Cheb}(x, m, a, b, a_0)$.
15     Orthonormalize $t$ against $V(:,1:k_{sub})$ to get a unit vector $V(:,k_{sub}+1)$; set $k_{sub} \leftarrow k_{sub}+1$; set
        $k_{old} \leftarrow k_{sub}$.
16     Compute $W(:,k_{sub}) = AV(:,k_{sub})$.
17     Compute the last column of the symmetric Rayleigh-Quotient matrix $H$:
        $H(1:k_{sub}-k_c, k_{sub}-k_c) = V(:,k_c+1:k_{sub})^*W(:,k_{sub})$.
18     Compute the eigen-decomposition of $H$: $HY = YD$, where $diag(D)$ is in non-increasing order.
        Set $\mu = D(1,1)$.
19     if $k_{sub} \geq dim_{\max}$ then
20         Restart: set $k_{sub} = k_c + k_{keep}$.
21     Update basis: $V(:,k_c+1:k_{sub}) \leftarrow V(:,k_c+1:k_{old})Y(:,1:k_{sub}-k_c)$; update $W$:
        $W(:,k_c+1:k_{sub}) \leftarrow W(:,k_c+1:k_{old})Y(:,1:k_{sub}-k_c)$.
22     Compute the residual vector: $r = W(:,k_c+1) - \mu V(:,k_c+1)$.
23     Set $noswap = 0$, $iter \leftarrow iter + 1$.
24     if $\|r\| \leq \tau \max(diag(D))$ then
25         Set $k_c = k_c + 1$, set $eval(k_c) = \mu$.
26         Swap eigenpairs if necessary to make converged eigenvalues are in non-increasing order.
27         Set $noswap = 1$ if any swap happens.
28     if $k_c \geq k$ and $noswap = 0$ then
29         Return $eval(1:k_c)$ and $V(:,1:k_c)$ as the converged desired eigenpairs.
30     Update lower bounds: let $a$ be the median of the diagonal entries of $D$.
31     if $a_0 > \min(diag(D))$ then
32         Set $a_0 \leftarrow \min(diag(D))$.
33     Set the next Ritz vector for filtering: $x = V(:,k_c+1)$.
34     Update $H$: $H = D(k_c+1:k_{sub}, k_c+1:k_{sub})$.
```

Algorithm 2: The Chebyshev-Davidson method for computing the k smallest eigenpairs of A [12].

3.1.3 Numerical Results

Here, we provide several examples arising from quantum chemistry to demonstrate the efficiency of the Chebyshev-Davidson method against LOBPCG and the Jacob-Davidson method used in [12]. In these tests, ChebyD means the Chebyshev-Davidson method; JDminres means the Jacob-Davidson method with the MINRES [8] algorithm for the correction equation; JDQR is the Jacob-Davidson method considered in [3]; JDCG is the Jacob-Davidson method proposed in [7]; LOBPCG is the algorithm in [4]. k eigenpairs of

Method	CPU Time (sec.)	Iteration Number	Error
ChebyD	1204	706	4.16E-11
JDminres	1968	536	6.44E-11
JDQR	3734	2183	2.73E-13
JDCG	3597	927	2.90E-12
LOBPCG	23190	5289	2.98E-10

Table 1: Silicon quantum dot model Si$_{34}$H$_{36}$ from the SuiteSparse Matrix Collection. Matrix size $N = 95769$. Compute $k = 100$ eigenpairs with $m = 20$ degrees in the Chebyshev polynomial.

Method	CPU Time (sec.)	Iteration Number	Error
ChebyD	418	587	2.38E-11
JDminres	850	577	3.55E-11
JDQR	1186	1441	5.80E-14
JDCG	1250	695	8.86E-13
LOBPCG	1974	686	9.96E-11

Table 2: bcsstk32 from the NIST Matrix Market. Matrix size $N = 44609$. Compute $k = 100$ eigenpairs with $m = 20$ degrees in the Chebyshev polynomial.

A are computed and the numerical error is defined as $\|AV - VD\|/\|A\|$, where V consists of all computed eigenvectors and D is a diagonal matrix representing the corresponding estimated eigenvalues.

The first example comes from the silicon quantum dot model available from the University of SuiteSparse Matrix Collection[1]. The second example is from the NIST Matrix Market[2]. They are all large-scale sparse matrices. From Tables 1 and 2, it is clear that the Chebyshev-Davidson method is the best among all methods compared.

3.1.4 Project Management

In the first year of this proposal, we will apply the Chebyshev-Davidson method to solve leading eigenvalue problems for graph analysis using sequential computing in Matlab, leveraging the efficient sparse matrix computation in Matlab. In the Chebyshev-Davidson method, it is crucial to estimate the range of desired leading eigenvalues and the range of all eigenvalues. Though an algorithm has been proposed in the original Chebyshev-Davidson method in [12], it is also important to design refined estimation using more advanced techniques to make the estimation more accurate, especially when eigenvalues are clustering together. Better estimation can lead to the more robust performance of the Chebyshev-Davidson method. In the second and third years of the proposal, if this proposal is renewable, we will explore the sparsity of matrices and eigenvectors, and distributed memory parallel computing to make the Chebyshev-Davidson method applicable to larger matrices.

3.2 Coordinate-Wise Descent Method

3.2.1 Problem Setup

The goal of the coordinate-wise descent method is the same as the goal of the Chebyshev-Davidson method. The problem setup is also almost the same except that the coordinate-wise descent method aims at much larger matrices when it is too expensive to store the whole sparse matrix and all eigenvectors at the same time in computer RAM in sequential computing, and it is too expensive to update k eigenvectors at the same time even if distributed memory parallel computing is applied. To tackle this difficulty, we will propose a novel coordinate-wise descent method to identify k eigenvectors of a given Hermitian sparse

[1]`https://sparse.tamu.edu/`

[2]`https://math.nist.gov/MatrixMarket/`

matrix A with $O(1)$ computational and memory cost per iteration without the need for orthogonalization in every iteration.

3.2.2 Algorithm Description

Overview. Without loss of generality, let us focus on a positive semi-definite sparse Hermitian matrix A and identify its smallest k eigenpairs. From the viewpoint of manifold constrained optimization, the eigenvalue problem can be transformed into the following constrained optimization problem

$$\min_{V\in\mathbb{C}^{N\times k}, V^*V=I_k} f(V) := \|A+VV^*\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix and I_k is the identity matrix of size k by k. However, the constraint $V^*V = I_k$ typically requires the orthonormalization of V in each iteration of the optimization, which involves the computation of the whole V and is also expensive especially in parallel computing. It has been the main challenge for decades to design an efficient optimization algorithm to solve the above problem for leading eigenpairs. In this proposal, we will theoretically show that, instead of solving the above challenging constrained optimization, it is sufficient to solve the following unconstrained optimization to identify a matrix $V \in \mathbb{C}^{N\times k}$ with a column space as the desired eigensubspace:

$$\bar{V} = \arg\min_{V\in\mathbb{C}^{N\times k}} f(V) = \|A+VV^*\|_F^2. \tag{2}$$

Then applying the Rayleigh-Ritz method with A and $\bar{V}$ can identify the k desired eigenpairs with an eigen-decomposition of size k by k and a matrix-matrix multiplication of cost $O(Nk^2)$.

Algorithm Design. Typically, second-order and quasi-second-order optimization methods may converge faster than first-order optimization methods for solving (2). However, second-order methods are too expensive because they require the computation of the Hessian of $f(V)$, and quasi-second-order methods are also not applicable since we aim at $O(1)$ computational and memory cost per iteration. Therefore, the coordinate-wise descent method based on gradient descent is proposed to solve (2). The proposed method is similar to the one in [5], but the method in [5] can only identify the smallest eigenpair instead of k pairs. The gradient of $f(V)$ is

$$\nabla f(V) = 4AV + 4V(V^*V),$$

and, hence, the gradient descent updating rule in the i-th iteration is

$$V_{i+1} = V_i - \alpha\left(4AV_i + 4V_i(V_i^*V_i)\right), \tag{3}$$

where α is a step size. The most important observation of the proposed method is the fact that (3) can be applied coordinate-wise with $O(1)$ computational cost and $O(1)$ RAM memory. In the coordinate-wise updating rule, suppose in the i-th iteration, we would like to only update the j-th row of V_i to obtain V_{i+1}, and we update the $(j-1)$-th row of V_{i-1} to get V_i. Let c_j denote the index set of the nonzero entries of $A(j,:)$. Then it is sufficient to evaluate

$$V_{i+1}(j,:) = V_i(j,:) - \alpha\left(4A(j,c_j)V_i(c_j,:) + 4V_i(j,:)K_i\right),$$

where $K_i \in \mathbb{C}^{k\times k}$ represents $V_i^*V_i$ and can be computed efficiently via a recursive algorithm

$$K_i = K_{i-1} + V_{i-1}^*(j-1,:)U + U^*V_{i-1}(j-1,:) + U^*U,$$

where

$$U = -\alpha\left(4A(j-1,c_{j-1})V_{i-1}(c_{j-1},:) + 4V_{i-1}(j-1,:)K_{i-1}\right).$$

Since $K_{i=1}$ has been computed in the previous step, computing K_i to obtain V_{i+1} takes only $O(k^2+kn)$ operations and RAM memory, where n is the number of nonzero entries of each row of A. The above discussion can be summarized as the following algorithm to identify the k smallest eigenpairs of A using $O(k^2+kn)$ operations and RAM memory per iteration and one step of Rayleigh-Ritz. The Rayleigh-Ritz step only requires the orthonormalization of V once and an eigendecomposition of size $k \times k$ once, which has a total computational cost of $O(Nk^2 + k^3)$ in operations and $O(Nk)$ in RAM memory.

```
1  **Input:** $iter_{\max}$-the maximum iteration number; $\alpha$-step size; $k$-the desired number of eigenpairs; $A$-a given matrix stored in a COO format.
2  **Output:** eigenvalues (in non-increasing order) stored in a diagonal matrix $\Sigma$ and their corresponding eigenvectors stored in $V \in \mathbb{C}^{N\times k}$.
3  **Initialization:** generate a random matrix $V \in \mathbb{C}^{N\times k}$ using i.i.d. Gaussian random variables and let $i = 1$.
4  Compute $K = V^*V$.
5  Let $r$ be a zero vector of size $k$, $K_0 = K$, and $V_0 = V$.
6  while $i \leq iter_{\max}$ do
7      Let $j = \mod (i-1, N) + 1$.
8      Compute $U = -\alpha\,(4A(j, c_j)V_0(c_j, :) + 4V_0(j, :)K_0)$.
9      Compute $G = V_0^*(j, :)U + U^*V_0(j, :) + U^*U$.
10     Update $V(j, :) \leftarrow V(j, :) + U$.
11     Update $r \leftarrow r + V(j, :) \bullet V(j, :)$.
12     Update $K \leftarrow K + G$.
13     Update $i \leftarrow i + 1$.
14     if $j = N$ then
15         Define $D = diag(r)$.
16         Compute $K \leftarrow D^{-1}KD^{-1}$ and let $K_0 = K$.
17         Compute $V \leftarrow VD^{-1}$ and let $V_0 = V$.
18         Let $r$ be a zero vector of size $k$.
19 Orthonormalize vectors in $V$.
20 Compute $H = V^*AV \in \mathbb{C}^{k\times k}$.
21 Compute the eigendecomposition $HQ = Q\Sigma$.
22 Update $V = VQ$.
```

Algorithm 3: The coordinate-wise descent method for computing the k smallest eigenpairs of A. To identify the k largest eigenpairs, apply this algorithm to $-A$ and change the signs of identified eigenvalues. In this algorithm, "$\bullet$" means the dot product of two vectors. Line 17 takes $O(N)$ cost but it can be simply implemented via one more cycle with $O(1)$ cost per iteration.

Generally, coordinate-wise descent methods have a cost per iteration much smaller than traditional iterative methods. Such a gap of the computational cost per iteration enables coordinate-wise descent methods focusing on the update of more important coordinates throughout iterations. The increase of the number of iterations can be leveraged by the choice of stepsize in the updating. The upper bound for the stepsize with guaranteed convergence in the coordinate-wise descent methods could be much larger than that in traditional iterative methods. Besides, an advanced and adaptive coordinate selection strategy can also reduce the number of iterations. Therefore, although coordinate-wise descent methods require a little larger number of iterations to achieve the convergence criteria, the operations counts and the runtime could be cheaper than that of traditional iterative methods. Most importantly, the low cost per iteration makes it applicable to large-scale problems that traditional iterative methods might not be capable of.

Main Features. The coordinate descent Algorithm 2 admits several favorable features for large-scale sparse Hermitian matrices arising in graph analysis. 1) **Matrix-free**, i.e. does not require storing the coefficient matrix explicitly, but can access the matrix by evaluating matrix-vector products. 2) **Factorization-free**, i.e. does not require any matrix decomposition for solving linear systems, though a small eigenvalue problem of size at most k by k is required. 3) **The costs per iteration and the RAM memory** are both $O(k^2 + kn)$, where n is the number of nonzero entries of A per row. 4) **Adaptive step size and coordinate selection** can facilitate the convergence. 5) **Blocking** allows utilizing highly efficient matrix-matrix operations, e.g., BLAS3, to compute multiple eigenpairs simultaneously.

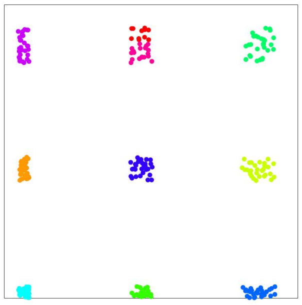

Figure 2: An artificial example of a point distribution for spectral clustering. There are 9 clusters, each of which has 300 points in a two-dimensional domain. The spectral clustering is applied with 9 largest eigenpairs of the corresponding similarity matrix identified by the coordinate-wise descent method..

3.2.3 Numerical Results

We will provide two sets of examples to illustrate the performance of the coordinate-wise descent method. The first set of examples comes from the full configuration interaction in quantum many-body problems, where it is interesting to identify the smallest eigenpair of large-scale sparse Hermitian matrices. We will take the example of H_2O and N_2 in [10] using the same error criteria as in the Chebyshev-Davidson method. The numerical results are presented in Table 3 and show that the coordinate-wise descent method can handle large-scale matrices.

Item	Matrix Size	Error	CPU Time (sec.)	Item	Matrix Size	Error	CPU Time (sec.)
H_2O	4.53E08	1.0E-02	3.7	N_2	1.75E11	1.0E-02	33.4
		1.0E-03	96.2			1.0E-03	752.6
		1.0E-04	592.5			1.0E-04	7892.6
		1.0E-05	2780.0			1.0E-05	49862.6

Table 3: The performance of the coordinate-wise descent method in Algorithm 3 for computing the smallest eigenpair of a large sparse Hermitian matrix from the full configuration interaction model for H_2O and N_2.

Next, we present an artificial example for spectral clustering. 9 clusters are constructed, each of which has 300 points in a two-dimensional domain. We apply the proposed coordinate-wise descent method with $\alpha = 0.001$ to identify the 9 largest eigenpairs of the corresponding similarity matrix and the spectral clustering method to cluster these 2700 points visualized in Figure 2. The performance of the coordinate-wise descent method is presented in Table 4. As we can see, the proposed method converges with good accuracy.

Number of Coordinate Update	Number of Gradient Cycles	Error
2,700,000	1000	5.0E-01
5,400,000	2000	8.0E-02
8,100,000	3000	1.6E-03
10,800,000	4000	4.5E-05
13,500,000	5000	1.9E-06

Table 4: The performance of the coordinate-wise descent method in Algorithm 3 for computing the largest 9 eigenpair of a similarity matrix from the clustering problem in Figure 2.

3.2.4 Project Management

Year 1: The coordinate-wise descent method in Algorithm 3 applies a cyclic strategy to select coordinates and a fixed step size. In this proposal, we will explore different adaptive methods to choose the step size and coordinates to reduce the total number of iteration to make the proposed method more attractive. The implementation will be in Matlab because of its efficient sparse matrix computation. It is expected that a single workstation can compute the leading eigenpairs of a sparse matrix of size $O(10^6)$ to $O(10^8)$ using the proposed method.
Year 2 and 3: If this project is renewable, we will explore the sparsity of eigenvectors and distributed memory parallel computing to further increase the computational limit of the problem size. If eigenvectors (or eigensubspace) admit sparse structures, a single workstation could handle matrices of size $O(10^{11})$. Armed with distributed memory parallel computing, the proposed method can handle matrices even larger. A convergence theory will be established for the proposed coordinate-wise descent method as well.

References

[1] M. Crouzeix, B. Philippe, and M. Sadkane. The davidson method. *SIAM Journal on Scientific Computing*, 15(1):62–76, 1994.

[2] H. E. Egilmez and A. Ortega. Spectral anomaly detection using graph-based filtering for wireless sensor networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1085–1089, 2014.

[3] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst. Jacobi–davidson style qr and qz algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20(1):94–125, 1998.

[4] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing*, 23(2):517–541, 2001.

[5] Y. Li, J. Lu, and Z. Wang. Coordinatewise descent methods for leading eigenvalue problem. *SIAM Journal on Scientific Computing*, 41(4):A2681–A2716, 2019.

[6] J. Liu, C. Wang, M. Danilevsky, and J. Han. Large-scale spectral clustering on graphs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 14861492. AAAI Press, 2013.

[7] Y. Notay. Combination of jacobidavidson and conjugate gradients for the partial symmetric eigenproblem. *Numerical Linear Algebra with Applications*, 9(1):21–44, 2002.

[8] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[9] G. L. G. Sleijpen and H. A. Van der Vorst. A jacobi–davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2):267–293, 2000.

[10] Z. Wang, Y. Li, and J. Lu. Coordinate descent full configuration interaction. *Journal of Chemical Theory and Computation*, 15(6):3558–3569, 06 2019.

[11] L. Wu, I. E.-H. Yen, Z. Zhang, K. Xu, L. Zhao, X. Peng, Y. Xia, and C. Aggarwal. Scalable global alignment graph kernel using random features: From node embedding to graph embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 14181428, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Y. Zhou and Y. Saad. A chebyshevdavidson algorithm for large symmetric eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):954–971, 2007.