Nonlinear spectral clustering with C++ GraphBLAS

Dimosthenis Pasadakis & Olaf Schenk & Verner Vlacic & Albert-Jan Yzelman

***Abstract*—Nonlinear reformulations of the spectral clustering method have gained a lot of recent attention due to their increased numerical benefits and their solid mathematical background. However, the estimation of the multiple nonlinear eigenvectors is associated with an increased computational cost. We present an implementation of a direct multiway spectral clustering algorithm in the p-norm, for $p \in (1, 2]$, using a novel C++ GraphBLAS API. The key operations are expressed in linear algebraic terms and are executed over the resulting sparse matrices and dense vectors, parameterized in the algebra pertinent to the computation. We demonstrate the effectiveness and accuracy of our shared-memory algorithm on several artificial test cases. Our numerical examples and comparative results against competitive methods indicate that the proposed implementation attains high quality clusters in terms of the balanced graph cut metric. The strong scaling capabilities of our algorithm are showcased on a range of datasets with up to 8 million nodes and 48 million edges.**

***Index Terms*—Algebraic programming, C++ GraphBLAS, graph p-Laplacian, spectral clustering**

I. Introduction

Spectral clustering is a popular community detection method that can be applied to any kind of data with a suitable similarity metric between them forming a graphical structure. At its core lies the computation of the mutually orthogonal eigenvectors of the graph Laplacian, a symmetric and positive semi-definite matrix, which are treated as the spectral coordinates of the graph, and are subsequently discretized using distance based algorithms [1]. This eigenspectrum computation offers ample room for parallelization, with both shared and distributed memory implementations widely used [2]. Nonlinear variants of the method in the p-norm, for $p \in (1, 2]$, that have been proposed lead to a minimization of balanced graph cut metrics, and an increase in the accuracy of the final clustering assignment [3]. Recently in [4], p-spectral clustering was cast as a nonlinear unconstrained optimization problem on the Grassmann manifold [5], by approximating the constraint for p-orthogonality with an analogous one for 2-orthogonality. This approach is not applicable to large-scale data, due to the large number of multiplications of the graph adjacency matrix with the computed eigenvectors that are required for convergence, especially as the value of p tends to 1.

GraphBLAS is a standard [6] for expressing graph computations in the language of linear algebra. Its core concepts are (i) algebraic containers, which correspond to sparse matrices and vectors, (ii) algebraic operators, describing sparse matrix-vector (SpMV) multiplications, and (iii) algebraic relations, an example of which is a generalized semiring under which an SpMV multiplication takes place. The recently introduced C++11 implementation of GraphBLAS [7] has showcased impressive results on the speed-up of algorithms based on SpMVs [8]. We express the key operations of the method introduced in [4] as well as the k-means discretization of the resulting eigenvectors in this C++ GraphBLAS API. This allows us to leverage its auto-parallelisation capabilities, and furnish the first, according to our best knowledge, p-norm spectral clustering algorithm applicable to large-scale data for shared-memory machines.

Dimosthenis Pasadakis, and Olaf Schenk are with the Advanced Computing Laboratory at the Institute of Computing, Università della Svizzera italiana (USI), Lugano, Switzerland. email: {dimosthenis.pasadakis, olaf.schenk}@usi.ch. Verner Vlacic, and Albert-Jan Yzelman are with the Computing Systems Lab, Huawei Zurich Research Center, Switzerland. email: {verner.vlacic, albertjan.yzelman}@huawei.com.

II. A C++ GraphBLAS p-spectral clustering algorithm

For an undirected weighted graph $\mathcal{G}(V, E, \mathbf{W})$ where V is the set of n nodes, E the set of edges, and $\mathbf{W}$ the weighted adjacency matrix, estimating a set of k p-eigenvectors on the Grassmann manifold $\mathcal{Gr}$ can be expressed as

$$\underset{\mathbf{U} \in \mathcal{Gr}(k,n)}{\text{minimize}}\; F_p(\mathbf{U}) = \sum_{\ell=1}^{k} \sum_{i,j=1}^{n} \frac{w_{ij} |u_i^\ell - u_j^\ell|^p}{2\|\mathbf{u}^\ell\|_p^p}, \;\; p \in (1, 2]. \quad (1)$$

Let $\ell = 1, 2, \ldots, k$ denote the eigenvector indices, and, at the minimizer, the columns of $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_k)$ approximate the eigenvectors associated with the smallest k eigenvalues of the p-Laplacian operator $\mathbf{\Delta}_p$. For $i \in V$ the p-Laplacian operator is defined as $(\mathbf{\Delta}_p \mathbf{u})_i = \sum_{j \in V} w_{ij} \phi_p (u_i - u_j)$, with $\phi_p : \mathbb{R} \to \mathbb{R}$ being $\phi_p(x) = |x|^{p-1} \text{sign}(x)$, and the p-norm is $\|\mathbf{u}\|_p = \sqrt[p]{\sum_{i=1}^{n} |u_i|^p}$, for $u \in \mathbb{R}$.

We use the Riemannian optimization software package ROPTLIB [9] to minimize (1) for progressively smaller values of p using Newton's method on the Grassmann manifold, where the solution of the linearized Newton subproblems is handled by a truncated conjugate gradient scheme. The description of the optimization problem is accomplished in ROPTLIB by specifying the function `EucGrad` which computes the gradient of $F_p(\mathbf{U})$ as well as the function `EucHessianEta` which computes $\boldsymbol{\eta} \mapsto \mathcal{H}\boldsymbol{\eta} = (\mathcal{H}^\ell \eta^\ell)_{\ell=1}^k$ for arbitrary $\boldsymbol{\eta} \in \mathbb{R}^{k \times n}$, where the collection of matrices $\mathcal{H}^1, \ldots, \mathcal{H}^k \in \mathbb{R}^{n \times n}$ corresponds to the Hessian of $F_p(\mathbf{U})$. For illustration, we include our C++ GraphBLAS implementation of `EucHessianEta` in Algorithm 1. Here the subroutines ROPTLIBtoGRB and GRBtoROPTLIB serve for I/O from and to the ROPTLIB data structures. The C++ GraphBLAS API leverages the algebraic structure of the ring of real numbers to parallelize the SpMV operation (the grb::vxm primitive).

III. Numerical Results

In order to demonstrate the effectiveness of the C++ GraphBLAS API for implementing the p-spectral clustering

Algorithm 1 The function `EucHessianEta`.

```
Input:   $\boldsymbol{\eta}$, a $k \times n$ matrix
         $(D[\ell])_{\ell=1}^{k}$, where each $D[\ell] = \mathrm{diag}(\mathcal{H}^{\ell})$
         $(H[\ell])_{\ell=1}^{k}$, where each $H[\ell] = \mathrm{diag}(\mathcal{H}^{\ell}) - \mathcal{H}^{\ell}$
Output:  $\mathbf{r}$, the result of $\boldsymbol{\eta} \mapsto \mathcal{H}\boldsymbol{\eta}$
 1: grb::Semiring<grb::operators::add<double>,
      grb::operators::mul<double>,
      grb::identities::zero, grb::identities::one> reals_ring;
 2: std::vector<grb::Vector<double>> grb_eta, grb_res;
 3: grb::Vector<double> v, w;
 4: ROPTLIBtoGRB($\boldsymbol{\eta}$, grb_eta);
 5: for $\ell = 1$ to $k$ do
 6:     grb::set(v, 0);
 7:     grb::vxm(v, grb_eta[$\ell$], H[$\ell$], reals_ring);
 8:     grb::eWiseApply(w, grb_eta[$\ell$]),D[$\ell$],
          grb::operators::mul<double>());
 9:     grb::eWiseApply(grb_res[$\ell$], w, v,
          grb::operators::subtract<double>());
10: GRBtoROPTLIB(grb_res, $\mathbf{r}$);
11: return $\mathbf{r}$
```

Method	Del. 16	Del. 17	Del. 18	Del. 19
Spec	0.129	0.089	0.062	0.045
pMulti	-6.21%	-3.11%	-2.21%	-2.45%
GrB-pGrass	$\mathbf{-8.12\%}$	$\mathbf{-6.43\%}$	$\mathbf{-4.56\%}$	$\mathbf{-4.19\%}$

TABLE I. Results in terms of the balanced graph cut metric RCut. We report the baseline (Spec) RCut, and in percentage the reduction of the cut that the methods pMulti and GrB-pGrass (ours) achieved.

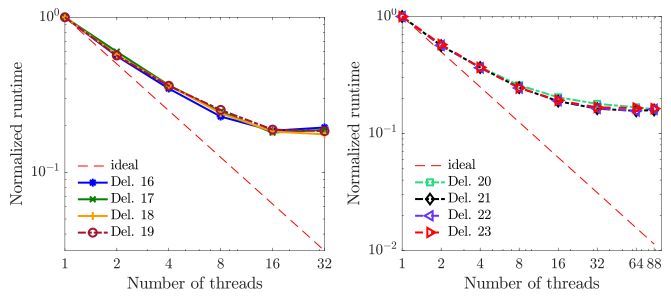
method, in Section III-A we report on the quality of the graph cuts obtained, and in Section III-B we present its parallel performance. For our experiments we select 8 matrices from the SuiteSparse matrix collection [10] with an increasing number of nodes $n = 2^r$ and edges $m \approx 6 * 2^r$, for $r = 16, \ldots, 23$, corresponding to Delaunay triangulations in a unit square.

A. Quality of graph cuts

We identify four clusters C_i, $i = 1, 2, 3, 4$, and compute the value of the balanced graph cut metric $\mathrm{RCut}(C_1, C_2, C_3, C_4) = \sum_{i=1}^{4} \frac{\mathrm{cut}(C_i, \overline{C_i})}{|C_i|}$. The results for the mid-scale cases with $r = \{16, 17, 18, 19\}$ are summarized in Table I. We compare our method (GrB-pGrass) against traditional spectral clustering (Spec) [1] and against the first full eigenvector analysis of p-Laplacian leading to direct multiway clustering (pMulti) [3].

B. Parallel performance

The strong scaling results of the developed algorithm are illustrated in Figure 1. In both plots, the dashed red line indicates the ideal scalability. We utilize up to 32 threads for the mid-scale cases (Figure 1a), and up to 88 threads for the large-scale Delaunay graphs with $r = \{20, 21, 22, 23\}$ (Figure 1b). On average, the parallel execution of the algorithm is $5.5\times$ faster than its sequential variant for the mid-scale tests, and $6.4\times$ faster for the large-scale cases. The run-time of the



(a) (b)

Fig. 1: Strong scaling of the C++ GraphBLAS components of the algorithm for the Delaunay graphs. a) Results for the mid-scale cases of node size $n \in [2^{16}, 2^{19}]$, b) Results for the large-scale cases of node size $n \in [2^{20}, 2^{23}]$. The runtime is normalized versus single-thread execution.

smallest case ($r = 16$) was ~ 300 sec, and that of the largest one ($r = 23$) ~ 20 hours. A breakdown of the runtime shows that only the GraphBLAS components of the algorithm exhibit excellent weak scalability for the large-scale graphs.

IV. Conclusion & Outlook

In this work, we have expressed they key operations of a multiway p-spectral clustering algorithm in the C++ GraphBLAS API. This enabled accurate parallel clustering of large-scale graphs on a shared-memory machine. We intend to further explore the potential gains of expressing graph partitioning and clustering algorithms in linear algebraic terms.

Acknowledgment

D.P. and O.S. acknowledge the support of the joint DFG - 470857344 and SNSF - 204817 project.

References

[1] U. Luxburg, "A tutorial on spectral clustering," Statistics and Computing, vol. 17, no. 4, p. 395–416, Dec. 2007.
[2] S. T. Wierzchoń and M. A. Kłopotek, Spectral Clustering. Cham: Springer International Publishing, 2018, pp. 181–259.
[3] D. Luo, H. Huang, C. Ding, and F. Nie, "On the eigenvectors of p-Laplacian," Machine Learning, vol. 81, no. 1, pp. 37–51, 2010.
[4] D. Pasadakis, C. L. Alappat, O. Schenk, and G. Wellein, "Multiway p-spectral graph cuts on Grassmann manifolds," Machine Learning, vol. 111, no. 2, pp. 791–829, Feb 2022.
[5] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," SIAM J. Matrix Anal. Appl., vol. 20, no. 2, p. 303–353, Apr. 1999.
[6] J. Kepner, D. Bade, A. Buluc, J. Gilbert, T. Mattson, and H. Meyerhenke, "Graphs, matrices, and the GraphBLAS: Seven good reasons," Procedia Computer Science, vol. 51, pp. 2453–2462, 2015.
[7] A. N. Yzelman, D. Di Nardo, J. M. Nash, and W. J. Suijlen, "A C++ GraphBLAS: specification, implementation, parallelisation, and evaluation," 2020, preprint. [Online]. Available: http://albert-jan.yzelman.net/PDFs/yzelman20.pdf
[8] A. Scolari and A. N. Yzelman, "Effective implementation of the High Performance Conjugate Gradient benchmark on GraphBLAS," 2023, accepted for publication.
[9] W. Huang, P.-A. Absil, K. A. Gallivan, and P. Hand, "Roptlib: An object-oriented C++ library for optimization on Riemannian manifolds," ACM Trans. Math. Softw., vol. 44, no. 4, Jul. 2018.
[10] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," ACM Trans. Math. Softw., vol. 38, no. 1, dec 2011.