An Online Scripting Language for Teaching Combinatorial Scientific Computing

M. Ali Rostami and H. Martin Bücker

Friedrich Schiller University Jena, Germany,
a.rostami@uni-jena.de, martin.buecker@uni-jena.de

Combinatorial Scientific Computing (CSC) [8, 9] consists of developing combinatorial models for problems in scientific computing, designing algorithms for the solution of the resulting combinatorial subproblems, and engineering corresponding software. CSC plays a prominent role in well-known areas like sparse matrix computations [3], mesh generation [4], and parallel computing [5]. However, CSC is also ubiquitous in a less common field called automatic or algorithmic differentiation [6] where computations involving Jacobian or Hessian matrices are transformed into a rich set of graph problems. Teaching CSC is tricky since the student should not only understand the combinatorial problem and the corresponding scientific computing problem, but also the intimate connection between these two problem representations. EXPLAIN [2, 10, 7, 1, 11] is a collection of interactive software modules currently developed at Friedrich Schiller University Jena to help teaching CSC in the classroom. Each module shows, side by side, a matrix encoding a problem from scientific computing and its related graph problem. So, the student can observe and analyze modifications that result during the solution of a problem in both matrix and graph views simultaneously.

We propose a scripting language on top of the Javascript library D3 (Data-Driven Documents) for implementing a new EXPLAIN module. This scripting language encapsulates D3 commands such that the student can edit and color both graph and matrix without the need to think of the underlying system. At the same time, some basic mathematical functions are added for the sake of simplicity. Consider the so-called column compression module in EXPLAIN as an example. This module implements a particular CSC problem involving sparse Jacobian matrices and a corresponding graph coloring problem. More precisely, we focus on a single step of a typical graph coloring algorithm. The following script assigns a color to a vertex of a given graph that is different from the colors of its neighbors:

```
1 var ns = neighbors(current);
2 var col_ns = get_colors(ns);
3 var new_col = min(diff(colors,col_ns));
4 color_vertex(current,new_col);
5 color_column(current, new_col);
```

Here, the function `neighbors` returns a set of all neighbors of a given vertex. Similarly, the function `get_colors` returns the colors of a given set of vertices. The mathematical functions `min` and `diff` compute the minimum of a set and

the difference of two given sets, respectively. In this particular CSC problem, the vertex of the graph represents a column of a sparse Jacobian matrix; see [2] for more details. The last two lines of this script not only color the vertex of the graph, but also the column of the Jacobian that is represented by this vertex.

Furthermore, we implement an online editor for this scripting language which is available immediately adjacent to the graph and matrix views. Figure 1 shows these two views together with the editor for the column compression module on the right. The student can directly edit and run the code written in this editor. The aim of this feature is to help students to interactively develop their own new module without the need of extensive knowledge of D3. There is a set of global variables which the student can edit in the upper input box of the editor labeled "Globals." Some of these global variables store important default values like the graph type, the colors, and the starting matrix. The values of other global variables can be assigned in the same way. There is another input box labeled "Code" that defines a single step of an iterative algorithm. In this graph coloring example, the order of processing the vertices is important. This order can be selected from a default list or is specified by the variable `order`. An animation of this graph algorithm can be controlled by various buttons below the editor. This graph algorithm can also be carried out by clicking on the graph vertices.

The implementation first evaluates the wrapper functions for D3 and then passes the contents of this editor to the `eval` function of Javascript. The contents of the editor can be sent either line by line or as a whole function, depending on an option selectable by the student.

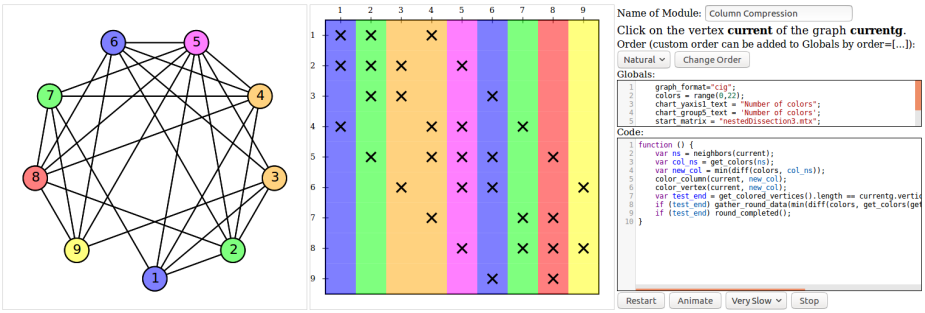

Fig. 1. The online editor of the corresponding module is visualized next to the graph and matrix. The code contained in the editor is written in a simple scripting language. The student first specifies the order of processing the vertices and this code is then executed using that order.

Keywords: combinatorial scientific computing, graph algorithms, education, scripting language

References

1. Bücker, H.M., Rostami, M.A.: Interactively exploring the connection between bidirectional compression and star bicoloring. In: Koziel, S., Leifsson, L., Lees, M., Krzhizhanovskaya, V.V., Dongarra, J., Sloot, P.M.A. (eds.) International Conference on Computational Science, ICCS 2015 — Computational Science at the Gates of Nature, Reykjavík, Iceland, June 1–3, 2015. Procedia Computer Science, vol. 51, pp. 1917–1926. Elsevier (2015)
2. Bücker, H.M., Rostami, M.A., Lülfesmann, M.: An interactive educational module illustrating sparse matrix compression via graph coloring. In: 2013 International Conference on Interactive Collaborative Learning (ICL), Proceedings of the 16th International Conference on Interactive Collaborative Learning, Kazan, Russia, September 25–27, 2013. pp. 330–335. IEEE, Piscataway, NJ (2013)
3. Duff, I., Erisman, A.M., Reid, J.: Direct Methods for Sparse Matrices. Numerical Mathematics and Scientific Computation Series, Oxford University Press (2017)
4. Edelsbrunner, H.: Geometry and Topology for Mesh Generation. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press (2001)
5. Grama, A., Karypis, G., Kumar, V., Gupta, A.: Introduction to Parallel Computing. Pearson, 2nd edn. (2003)
6. Griewank, A., Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia, PA, 2nd edn. (2008)
7. H. M. Bücker, M. A. Rostami: Interactively exploring the connection between nested dissection orderings for parallel Cholesky factorization and vertex separators. In: IEEE 28th International Parallel and Distributed Processing Symposium, IPDPS 2014 Workshops, Phoenix, Arizona, USA, May 19–23, 2014. pp. 1122–1129. IEEE Computer Society, Los Alamitos, CA, USA (2014)
8. Hendrickson, B., Pothen, A.: Combinatorial Scientific Computing: The Enabling Power of Discrete Algorithms in Computational Science, pp. 260–280. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
9. Naumann, U., Schenk, O.: Combinatorial Scientific Computing. Chapman & Hall/CRC, 1st edn. (2012)
10. Rostami, M.A., Bücker, H.M.: Interactive educational modules illustrating sparse matrix computations and their corresponding graph problems. In: für Informatik, G. (ed.) Informatiktage 2014, Fachwissenschaftlicher Informatik-Kongress, 27. und 28. März 2014, Hasso Plattner Institut der Universität Potsdam, GI-Edition: Lecture Notes in Informatics (LNI) – Seminars, vol. S–13, pp. 253–256. Köllen Druck+Verlag GmbH, Bonn (2014)
11. Rostami, M.A., Bücker, H.M.: An educational module illustrating how sparse matrix-vector multiplication on parallel processors connects to graph partitioning. In: Hunold, S., Costan, A., Giménez, D., Iosup, A., Ricci, L., Gómez Requena, M.E., Scarano, V., Varbanescu, A.L., Scott, S.L., Lankes, S., Weidendorfer, J., Alexander, M. (eds.) Euro-Par 2015: Parallel Processing Workshops, Euro-Par 2015 International Workshops, Vienna, Austria, August 24–28, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9523, pp. 135–146. Springer, Cham, Switzerland (2015)