Email address

mhaberla@calpoly.edu

Name of Submitter

Matt Haberland

Project Submitting this Proposal

SciPy

Is your project Affiliated or Sponsored?

Affiliated

Proposal Title

An Efficient, High-Level Implementation of Linear Programming

Is this proposal submitted for the ESIP grant opportunity?

No

Description - no more than 750 words, please!!!

inear programming, the optimization of a linear objective function subject to linear equality and inequality constraints, is a fundamental tool for scientific research and engineering [1]. As the Python programming language continues to grow in popularity [2] and is particularly well suited to scientific computing [3, 4], it seems natural that there should be an efficient linear programming suite that is actually implemented in the Python language. Surprisingly, this is not the case. While there are several existing open-source linear programming solvers (e.g. [5-12]), all but [11] are implemented in low-level languages. While conda-forge has made some of these easier to access on non-UNIX operating systems, installation obstacles remain, and the ability of the general user to inspect and modify the source is impeded. Furthermore, the licenses are generally copyleft, limiting their benefit to industry and businesses. This would explain the popularity of `scipy.optimize.linprog`, one of SciPy's most popular functions - despite its significant shortcomings.

The original implementation of the simplex method in `scipy.optimize.linprog` was inefficient, as it performed elementary row operations on NumPy arrays via Python for loops, adding substantial overhead to what should be swift, vectorized operations. It was also fraught with bugs that frequently caused the algorithm to terminate without a solution to problems known to be feasible [13], return a suboptimal solution [14], and even report solutions as optimal despite their inability to satisfy problem constraints [15-18]. To partially remedy this problem, I contributed a Python implementation of an interior-point algorithm that was initially released with SciPy 1.0[19]. It is incontrovertibly faster and more reliable than the simplex implementation, but it is still not the default routine when a user invokes `linprog` without a `method` argument because it (like [11]) is approximate by nature, whereas the

simplex method is theoretically exact and always returns a solution at a vertex of the polytope defined by the constraints, which is often desirable for sensitivity analysis.

To fully remedy this, I have submitted a pull request [20] for a Python implementation of the "revised simplex" method, which is also theoretically exact. This implementation is already faster and more reliable than the original "tableau-based" simplex implementation. In addition, the revised simplex method is particularly well-suited to large problems as it has the potential to exploit problem sparsity. The pull request passes the extensive battery of tests written for `linprog` and is ready to be merged. However, the merge is delayed as maximum impact on the community can only be realized if the new method is released in concert with the following, which I propose to complete and merge into SciPy with the support of this NumFOCUS Small Development Grant.

- Add callback function support to interior-point and revised simplex solvers, so that users can run custom code to monitor solver progress after each iteration of the algorithm, adding transparency to their operation.

- Enable users to provide an initial feasible solution to the linear programming problem. This gives the user greater control over the progression of the algorithm, improves solution time by eliminating the need for the first phase of the simplex algorithm, and provides users with an efficient mechanism for analyzing the sensitivity of the solution with respect to changes in the objective function.

- Carefully clean the documentation of `linprog`, which, despite recent work, still suffers from mistakes and unprofessional presentation that may hinder use of the code itself.

- Overhaul the test suite, strengthening weak tests to ensure that solved issues do not arise again, and pruning redundant tests from the rather large (>500 tests) suite for better speed.

- Using the standard NETLIB Linear Programming problems [21], benchmark the interior point and revised simplex solvers against one another and the other open source linear programming solvers that can be installed with conda-forge, and publish the results in SciPy documentation.

- Change the default solver of `scipy.optimize.linprog` to the revised simplex method.

[1] Dantzig, George. Linear programming and extensions. Princeton university press, 2016.

[2] "TIOBE Index for October 2018." TIOBE - The Software Quality Company, TIOBE Software BV, October 2018, https://www.tiobe.com/tiobe-index/.

[3] Oliphant, Travis E. "Python for scientific computing." Computing in Science & Engineering 9.3 (2007).

[4] Perez, Fernando, Brian E. Granger, and John D. Hunter. "Python: an ecosystem for scientific computing." Computing in Science & Engineering 13.2 (2011): 13-21.

[5] Free Software Foundation, Inc. "GLPK (GNU Linear Programming Kit)." GNU Operating System, 23 June 2012, https://www.gnu.org/software/glpk/.

[6] Berkelaar, Michel, et al. "Introduction to lp_solve 5.5.2.5." SourceForge - Download, Develop, and Publish Free Open Source Software. http://lpsolve.sourceforge.net/5.5/.

[7] Melnick, Michael. "Linear Programming Solver (LiPS)." SourceForge - Download, Develop, and Publish Free Open Source Software, 11 April 2013, https://sourceforge.net/projects/lipside/.

[8] Forrest, John and Hall, Julian. "COIN-OR Linear Programming Solver." COIN-OR Computational Infrastructure for Operations Research, January 2017, https://projects.coin-or.org/Clp.

[9] Wright, Stephen et al. "PCx Linear Programming Code." UW Computer Sciences User Pages, 10 January 2006, http://pages.cs.wisc.edu/~swright/PCx/.

[10] "JOptimizer - Java Convex Optimization." joptimizer.com, 17 August 2018, http://www.joptimizer.com/.

[11] Andersen, Martin et al. "CVXOPT: Python Software for Convex Optimization." 30 August, 2018, https://cvxopt.org/index.html.

[12] Mevencamp, Monika et al. "QSopt Linear Programming Solver." Mathematics | University of Waterloo, September 2011, https://www.math.uwaterloo.ca/~bico/qsopt/.

[13] @cy18. "scipy.optimize.linprog failed to find a feasible starting point while solution exists #6139." GitHub, 4 May 2016, https://github.com/scipy/scipy/issues/6139.

[14] @mdhaber. "suboptimal solution returned by scipy.optimize.linprog #7044" GitHub, 13 February 2017, https://github.com/scipy/scipy/issues/7044.

[15] @akxlr. "linprog returns solution that violates inequality constraint #5400." GitHub, 21 October 2015, https://github.com/scipy/scipy/issues/5400.

[16] @martinResearch. "wrong solution with scipy.optimize.linprog #6690." GitHub, 16 October 2016, https://github.com/scipy/scipy/issues/6690.

[17] @MInner. "scipy.optimize.linprog _sometimes_ ignores restrictions #7237." GitHub, 29 March 2017, https://github.com/scipy/scipy/issues/7237.

[18] @realazthat. "linprog failure on provided input #8174." GitHub, 19 November 2017, https://github.com/scipy/scipy/issues/8174.

[19] @mdhaber. "ENH: added "interior-point" method for scipy.optimize.linprog #7123." GitHub, 2 March 2017, https://github.com/scipy/scipy/pull/7123.

[20] @mdhaber. "WIP: optimize: added "revised simplex" for scipy.optimize.linprog #9263." GitHub, 12 September 2018, https://github.com/scipy/scipy/pull/9263.

[21] Gay, David. "netlip/lp." The Netlib, 22 August 2013, http://www.netlib.org/lp/.

Timeline of Deliverables

11/14/2018 - Award notification

11/17/2018 - Enable user-provided basic feasible solution (2 hr)

11/18/2018 - Callback function support (4 hr interior-point, 2 hr revised simplex)

11/19/2018 - Overhaul test suit (6 hr)

11/20/2018 - Carefully clean documentation (6 hr)

11/21/2018 - Set up benchmarking (2 hr per open source solver x 3 - GLPK, CLP, and CVXOPT)

11/24/2018 - Make revised-simplex default solver (1 hr)

11/24/2018 - 12/19/2018 - Community Review, address comments (3 hr)

12/20/2018 – Merge

Benefit to Project/Community - no more than 350 words, please.

Merging an efficient, professionally documented, rigorously tested, carefully benchmarked, and fully-featured revised simplex solver into SciPy will present the vast Python scientific computing community with an exact linear programming solver that users can easily access on virtually any platform, and actually inspect and modify due to its implementation in their preferred programming language. It will also open the doors to a great number of additional improvements to the SciPy linear programming suite: this implementation provides a standard interface for developers to contribute new pivoting rules, and paves the way for the addition of sparse matrix support and the dual simplex method variant, all of which can dramatically improve performance of the algorithm for certain problems. Finally, this would enable the creation of a mixed-integer linear programming (MILP) solver, a much-desired extension to SciPy's optimization module, as (continuous) linear programming is used as a subroutine by MILP algorithms and the (basic-feasible) solutions returned by the revised simplex method are far more suitable than the approximate solutions returned by the existing interior-point routine.

Project Team

Matt Haberland

Brief Budget Justification - How will the money be spent?

The budget is broken down to $1,313 in wages plus $131 in benefits for Matt Haberland’s effort and indirect costs at $556. The total funding request is $2,000. SALARIES AND WAGES: The salary and wage rates are based on the California Polytechnic State University (CPSU) and Cal Poly Corporation (CPC), jointly Cal Poly, established salary and wage rates paid during the 2018-2019 Fiscal year (July 1 – June 30). Faculty duties at CPSU consist of a full fifteen units each of three Academic quarters per nine month

Academic year. Funding has been requested for the PI, Matt Haberland, to work an additional 20.35 hours overload through completion of the project in January 2019. The salary and wage rate for faculty includes a projected 4.5% salary increase above base salary. The rates shown are for budgetary purposes; the rates in effect at the time the work is performed will be charged to the project. FRINGE BENEFITS & EMPLOYER PAYROLL TAXES: Benefits for CPSU Faculty overload work includes FICA, SUI and Workers Compensation and is calculated at the DHHS pooled rate of 10.0%. INDIRECT COSTS: Cal Poly's federally negotiated indirect rate is 38.5% of modified total direct costs, effective July 1, 2018. Modified total direct costs exclude equipment, capital expenditures, charges for patient care, tuition remission, rental costs of off-site facilities, scholarships, and fellowships, participant support costs, and the portion of each subaward in excess of $25,000