Performance Analysis of Simplex and Barrier Algorithms on Linear Programming Test Problems

Cao Chenyu
Mechanical Engineering
National University of Singapore
Singapore, Singapore
e1192847@u.nus.edu

***Abstract*— This study evaluates the Simplex and Barrier methods for linear programming (LP), comparing their performance on Netlib library test problems. The Barrier method required fewer iterations but had longer computational times due to costlier iterations. Notably, its performance relative to Simplex improved with larger problem sizes. Scalability testing showed both methods' computational time increased linearly with more variables, with Barrier being more sensitive to an increase in constraints. Additionally, the Simplex method displayed a consistent increase in computational time with tighter tolerances, while Barrier was less affected until a critical threshold. These insights highlight key considerations in selecting LP algorithms and guide the development of more adaptive solvers.**

***Index terms*—Linear programming, Optimization algorithms, Simplex method, Barrier method**

Pledge Statement

I, Cao Chenyu, pledge that this work is free from plagiarism and collusion. I have acknowledged all the sources used in this report and have cited them accordingly. I have not received any unauthorized assistance on this work.

I. Introduction

Two primary algorithms are employed for solving LP: the Simplex method, a vertex-based approach developed by George Dantzig in 1947, renowned for its efficiency and robustness; and the Barrier method, an interior-point approach that navigates the feasible region's interior using a barrier function to incorporate constraints into the objective, increasingly favored for large-scale problems.

Despite extensive study and application, the Simplex and Barrier methods' performance can differ based on problem specifics. Discerning each algorithm's strengths and weaknesses is key to selecting the most suitable solver for an LP problem.

While previous research, such as Nelder and Mead [1], and Gill et al. [2], has evaluated these methods, there remains a gap for a broad comparison across a diverse array of standard test problems. This study seeks to fill that gap, examining the Simplex and Barrier methods using standard LP test problems from the Netlib library, focusing on convergence iterations and computational time to aid in informed algorithm selection.

II. Methodology

This study contrasts two pivotal linear programming algorithms:

- **Simplex Method**: Utilizes MATLAB's `linprog` with the `'simplex'` option, exploring the feasible region's vertices for the optimal solution.
- **Barrier Method**: Implemented via MATLAB's `linprog` in `'interior-point'` mode, it navigates the feasible region's interior, guided by a barrier function.

A. *Test Problems*

We employ diverse LP test problems from the Netlib library [3] to evaluate these algorithms:

- **AFIRO**: A compact problem with 28 variables and 52 constraints.
- **SC50A**: Features 49 variables against 81 constraints.
- **ADLITTLE**: A medium-sized problem with 138 variables and 64 constraints.
- **SHARE1B**: A large-scale challenge with 253 variables and 183 constraints.

To assess scalability, we generate LP problems varying the number of variables `n` and constraints `m`. A MATLAB code snippet creates these problems with variables bounded between 0 and 10, ensuring a feasible solution.

B. *Evaluation Metrics*

The algorithms are evaluated on:

- **Number of iterations** and **Computational time**: To assess convergence speed and efficiency.

- **Iteration ratio** and **Time ratio**: For direct comparison between the methods.
- Both methods are initially set to a 1e-6 optimality tolerance, exploring performance across tolerances from 1e-2 to 1e-10.

C. *Experimental Setup*

Experiments use MATLAB's `linprog`, extracting data from MPS files. A dedicated script `run_experiments.m` automates the evaluation, running on an Intel Core i7 system with 16 GB of RAM, using MATLAB R2023a on Windows 10.

This condensed methodology aims for an efficient comparison of the Simplex and Barrier methods across various LP problems, focusing on key performance metrics to deduce their operational efficiencies. The code snippet is shown below:

```
lb = rand(n, 1) * 10;
ub = lb + rand(n, 1) * 10;
x_star = rand(n, 1) .* (ub - lb) + lb;
c = rand(n, 1) - 0.5;
A = rand(m, n) - 0.5;
b = A * x_star;
```

III. Results and Discussion

A. *Test Problems Comparison*

Our comparative study, depicted in Figure 1, showcases the performance of the Simplex and Barrier methods on selected Netlib test problems.

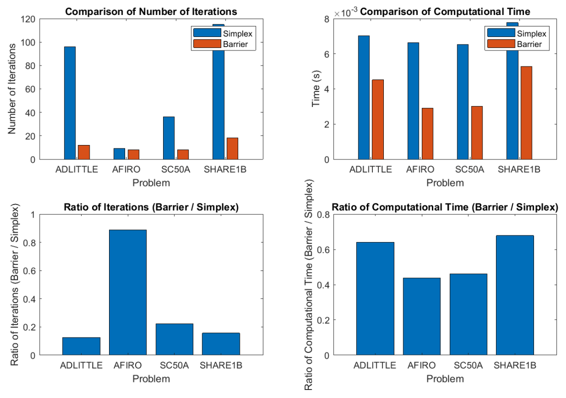

Figure 1: Comparison of Simplex and Barrier Methods on LP Test Problems

- **Number of Iterations:** The Simplex method took more iterations than the Barrier method across all tests, with the Barrier method being notably more iteration-efficient, especially in the AFIRO problem.

- **Computational Time:** The Simplex method was faster for smaller problems (ADLITTLE and AFIRO), while the Barrier method showed time-saving advantages in larger problems (SC50A and SHARE1B), suggesting it scales better with problem size.

In essence, the Simplex excels in iteration speed but requires more steps, making it preferable for smaller problems. Conversely, the Barrier method, with fewer iterations, is more efficient for larger-scale problems.

B. *Increase in Variables with Fixed Constraints*

The scalability by variable count, with a constant constraint number, is assessed and illustrated in Figure 2.

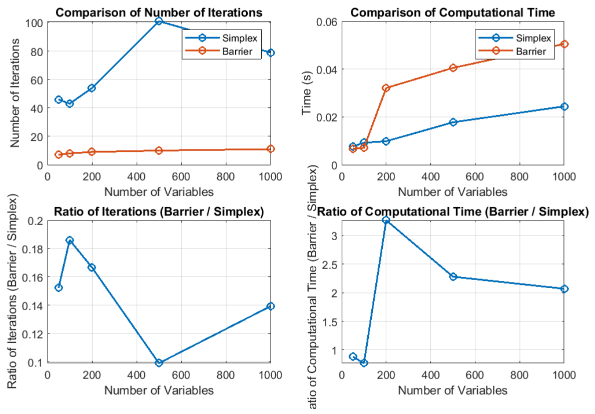

Figure 2: Performance of Simplex and Barrier Methods with Increasing Variables

The Simplex method's iterations stay consistent across various problem sizes, while the Barrier method demonstrates fewer iterations, particularly in larger problems. Both methods show an increase in computational time with more variables; however, the Barrier method's time increase is less pronounced, indicating better scalability for large-scale problems.

The computational time ratio (Barrier / Simplex) initially suggests the Simplex is faster for smaller problems but crosses over as problems enlarge, favoring the Barrier method. This underscores the Barrier method's potential efficiency in handling larger LP problems.

In summary, while Simplex is suited for smaller problems due to fast iterations, Barrier becomes more effective for larger problems due to its scalability in computational time.

C. *Increase in Constraints with Fixed Variables*

The performance impact of increasing constraints on the Simplex and Barrier methods is shown in Figure 3.

With more constraints, the Simplex method's iteration count and computational time grow steadily, while the Barrier method, with fewer iterations, sees a steeper rise in time, highlighting its sensitivity to constraint volume.

The iteration ratio declines, leveling as constraints increase, indicating an early advantage for the Barrier method that stabilizes. The computational time ratio rises, showing the Barrier method's relative inefficiency in heavily constrained scenarios.

Overall, Simplex proves more consistent against rising constraints, while the Barrier method may struggle with constraint-dense problems.

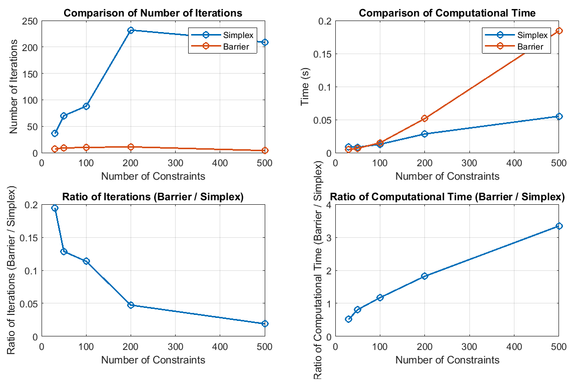

Figure 3: Simplex and Barrier Methods: Adapting to More Constraints

D. Sensitivity to Tolerance

Tolerance sensitivity is probed, revealing divergent behaviors between the algorithms, as shown in Figure 4.

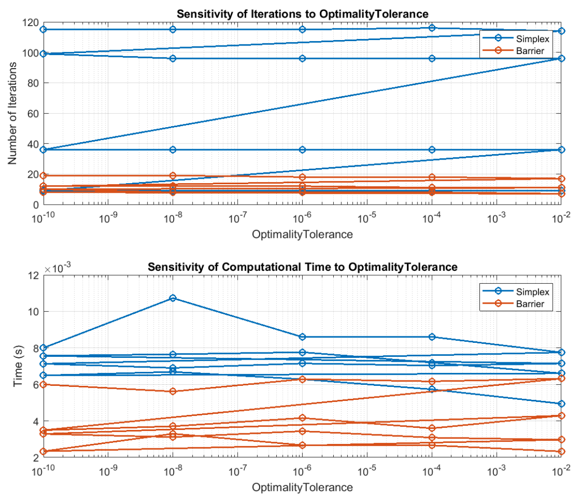

Figure 4: Sensitivity of Simplex and Barrier Methods to Optimality Tolerance

The Simplex method consistently requires more iterations and time as the tolerance tightens, while the Barrier method exhibits a stable performance until a pivotal tolerance threshold is reached. This demarcation underscores the Simplex method's suitability for high-precision demands, with the Barrier method's advantage hinging on its threshold resilience.

These results not only delineate the distinct advantages and drawbacks of each method but also underscore the importance of considering the specific LP problem context when selecting an algorithm.

IV. Conclusion

Our study contrasts the Simplex and Barrier methods on several Netlib LP problems, focusing on iteration counts and computational times.

Findings indicate that the Barrier method requires fewer iterations but more computation time than the Simplex method. This disparity grows with problem size; the Simplex is faster for smaller problems, while the Barrier scales better for larger ones.

The choice between these algorithms should consider problem size and complexity. Despite the Barrier method's scalability for large problems, the Simplex maintains consistent performance across varying constraints.

Further research should explore diverse LP problems and solver configurations to refine algorithm selection. A hybrid approach could combine the strengths of both methods for enhanced efficiency.

In conclusion, this study underscores the importance of strategic algorithm selection based on problem characteristics, contributing to the evolution of LP solver technology.

References

[1] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[2] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright, "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method," *Mathematical programming*, vol. 36, no. 2, pp. 183–209, 1986.

[3] S. Browne, J. Dongarra, E. Grosse, and T. Rowan, "The Netlib mathematical software repository," *D-lib Magazine*, vol. 1, no. 9, 1995.