Promoting hyper-sparsity in the revised simplex method

J. A. J. Hall[a,*], Q. Huangfu[a]

[a]*School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh, JCMB, King's Buildings, Edinburgh, EH9 3JZ, United Kingdom*

Abstract

Exploiting hyper-sparsity in the revised simplex method for linear programming (LP) has resulted in the most recent significant performance improvements for efficient solvers. Hitherto, there has been relatively little understanding of how it occurs and how greater degrees of hyper-sparsity may be achieved when solving a given problem. This talk identifies advanced algorithmic techniques for the dual revised simplex method that promote hyper-sparsity, leading to dramatic performance improvements. An analysis of these techniques in the context of network flow optimization gives an insight into why they are so effective.

Keywords: linear programming, simplex method, hyper-sparsity

1. Introduction

Identification and exploitation of hyper-sparsity [1] is one of the most important modern enhancements of the revised simplex method. It reveals deep sparse features of linear programming (LP) problems and significantly improves the performance of simplex implementations when most of the results of its underlying matrix-vector operations are sparse.

Since then, another question has arisen: how to promote hyper-sparsity in the simplex method? Specifically, how to increase the sparsity of the results and the frequency of sparse results. Although the hyper-sparsity properties near the optimal vertex may be immutable, is it possible to find a path to the solution for which there is greater hyper-sparsity in the underlying matrix algebra? This talk presents evidence that some advanced algorithmic enhancements of the dual revised simplex method have remarkable and unintended consequences for hyper-sparsity and gives an insight into why they occur.

`ParISS` is a highly efficient dual revised simplex solver [2] developed by the Authors, incorporating dual steepest-edge (DSE) [3] and a bound flipping ratio test (BFRT) [4]. A detailed description of a similar (serial) implementation is given, for example, by Koberstein [5]. With these techniques, when solving a

*Corresponding author
Email address: `jajhall@ed.ac.uk` (J. A. J. Hall)

general bounded LP problem

$$\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},$$

the major computational components are summarised in Figure 1.

`CHUZR`	:	Scan $\hat{b}_i/\hat{w}_i$ for a good candidate p to leave the basis, where $\hat{\boldsymbol{w}}$ is the vector of DSE weights.
`BTRAN`	:	Form $\boldsymbol{\pi}_p^T = \boldsymbol{e}_p^T B^{-1}$.
`PRICE`	:	Form $\hat{\boldsymbol{a}}_p^T = \boldsymbol{\pi}_p^T N$.
`CHUZC`	:	Scan the ratios $\hat{c}_j/\hat{a}_{pj}$ for a candidate q to enter the basis, possibly allowing some boxed variables to become dual infeasible (BFRT).
`UPDATE-DUAL:`		Update $\hat{\boldsymbol{c}}_N^T := \hat{\boldsymbol{c}}_N^T - \theta_d \hat{\boldsymbol{a}}_p^T$, $\theta_d = \hat{c}_q/\hat{a}_{pq}$. Flip bounds of dual infeasible variables in $\hat{\boldsymbol{c}}_N^T$ and accumulate $\boldsymbol{a}_{\scriptscriptstyle F}$ accordingly.
`FTRAN-BFRT`	:	Form $\hat{\boldsymbol{a}}_{\scriptscriptstyle F} = B^{-1}\boldsymbol{a}_{\scriptscriptstyle F}$ and update $\hat{\boldsymbol{b}} := \hat{\boldsymbol{b}} + \hat{\boldsymbol{a}}_{\scriptscriptstyle F}$.
`FTRAN`	:	Form $\hat{\boldsymbol{a}}_q = B^{-1}\boldsymbol{a}_q$ and update $\hat{\boldsymbol{b}} := \hat{\boldsymbol{b}} - \theta_p \hat{\boldsymbol{a}}_q$, $\theta_p = \hat{b}_p/\hat{a}_{pq}$.
`FTRAN-DSE`	:	Form $B^{-1}\boldsymbol{\pi}_p$ for updating DSE weight $\hat{\boldsymbol{w}}$.

`If` {growth in representation of B} `then`

`INVERT` : Form a new representation of B^{-1}.

`else`

`UPDATE-BASIS:` Update the representation of B^{-1}.

Figure 1: Operations in an iteration of the dual revised simplex method

Hyper-sparsity relates to the presence of zero values in the results of `BTRAN`, `PRICE` and `FTRAN`. If the percentage of nonzero values in a result is below the "density threshold" (or simply *threshold*), t, then the result is considered to be hyper-sparse. When solving an LP problem, the percentage of results of a particular type that are hyper-sparse is referred to as the "sparse result rate" (or simply *rate*), r. Hall and McKinnon [1] used a value of $t = 10\%$ and considered an LP problem to be hyper-sparse if $r > 60\%$ for each class of results. Efficient implementations of the revised simplex method must exploit hyper-sparsity when computing sparse results, as well as when using them.

After the DSE and BFRT components were implemented in `ParISS`, cost perturbation was introduced with the aim of mitigating the effects of dual degeneracy. As described by Koberstein [5], this technique adds to each component c_j of $\boldsymbol{c}$ a random perturbation whose magnitude is related to that of c_j. Cost perturbation in `ParISS` was found to be strikingly effective for a range of sparse problems, notably the `pds` problems [6]. For example, the iteration count for `pds-10` was reduced by about 10% but the solution time decreased by over 60%!

Our explanation for this remarkable improvement in performance is that column perturbation changes the nature of pivot selection so that hyper-sparsity increases. The simplex method's path is shorter but, more importantly, it passes through vertices where hyper-sparsity is more prevalent. This talk explores this phenomenon further, providing an insight into why it occurs.

2. Experimental observations

Initial experiments were carried out using the NETGEN [7] network LP problems since they are known to be hyper-sparse and their algebraic properties when solved using the simplex method permit a level of analysis that is intractable in more general LP problems. In particular, the nonzeros in both B^{-1} and the simplex tableau corresponding to $B^{-1}N$ have unit absolute value.

The overall effectiveness of cost perturbation on the dual simplex method was first assessed using the whole NETGEN set. The total iteration count over the 50 problems reduced by 5%, but the total solution time reduced by 36%. The time saving was not uniform, with three problems taking longer to solve when using cost perturbation. However, the solution time was reduced by 10% on 35 problems, and by 30% on 25 problems. In the best case, `netgen114` was solved 350% faster.

	CPU time (s)		Relative
Component	Without	With	time
`SETUP`	0.13	0.13	1.0
`CHUZR`	0.08	0.07	1.1
`BTRAN`	0.12	0.05	2.4
`PRICE`	0.31	0.08	3.9
`CHUZC`	0.22	0.06	3.7
`UPDATE-DUAL`	0.12	0.03	4.0
`FTRAN-BFRT`	0.01	0.01	1.0
`FTRAN`	0.09	0.04	2.3
`FTRAN-DSE`	0.17	0.05	3.4
`INVERT`	0.14	0.13	1.1
All other	0.01	0.01	1.0

Table 1: Profiling for `netgen108`, with and without cost perturbation

More detailed analysis was carried out on the problems, and typical profiling results are given in Table 1 in the case of `netgen108`, where `SETUP` refers to setup and memory allocation. Cost perturbation reduced the iteration count by 10% but improved the solution time by a factor of 2.2! With cost perturbation, the times for `BTRAN` and `PRICE` reduced significantly due to their improved hyper-sparsity rate r (at a threshold value of $t = 10\%$). The reduction in the incidence of dense pivotal rows led to a corresponding reduction in the time for `CHUZC` and `UPDATE-DUAL`. The time for `FTRAN` reduced significantly, but all `FTRAN` results were below the threshold of $t = 10\%$ so, to assess the impact of cost perturbation, the threshold value was reduced to $t = 2\%$. With this threshold, the hyper-sparsity rate improved considerably when cost perturbation was used. The greater sparsity in $\boldsymbol{\pi}_p$ led to a significant improvement in the performance of `FTRAN-DSE`. The `FTRAN-BFRT` time did not change since the greater sparsity of the results balanced the greater number of `FTRAN-BFRT` operations.

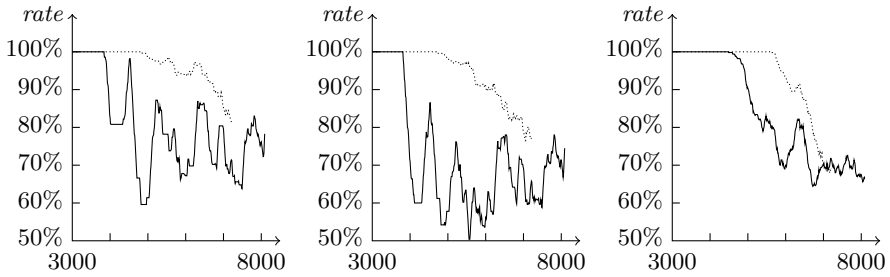

Figure 2: Sparse result rate traces for `netgen108`, with and without cost perturbation

Figure 2 gives even greater insight into the effect of cost perturbation on hyper-sparsity, tracing with a dotted (solid) line the sparse result rates with (without) cost perturbation against iteration count. All traces were 100% for the first 3000 iterations. It is clear that using cost perturbation delays the increase in the prevalence of dense pivot rows and columns although, unsurprisingly, the final rates are comparable both with and without cost perturbation. The question to be addressed, is why?

3. Analysis

An explanation for the behaviour observed above is given below by carefully considering the impact of cost perturbation on the BFRT for general LP problems, with specific references to the special case of network LP problems.

When the BFRT is performed without cost perturbation, if it results in a zero step due to degeneracy then the tie for the leaving variable is broken by choosing the pivot of maximum magnitude with the aim of promoting numerical stability. However, for network LP problems, all the candidate pivots have unit absolute value. In `ParISS`, there was no rule to break this "second order" tie: the first candidate was chosen and no bound flipping occurred. As a consequence, when a relatively dense tableau row was the most attractive, its primal infeasibility was reduced by less than it might have been had bounds been flipped. Thus the attractiveness of such rows persisted in subsequent iterations.

Cost perturbation separates the degenerate nodes in the piecewise linear function of primal infeasibility whose minimizer is identified by the BFRT. For network LP problems, the nodes are the dual values and it can be argued that the earlier nodes are likely to correspond to sparser pivotal columns. This separation of the degenerate nodes eliminates the first order tie (with no numerical disadvantage in the case of network problems) and permits bound flips to be performed. The variables involved are likely to correspond to sparser tableau

columns so, in some sense, the cost of FTRAN-BFRT is minimized. When the pivotal row is dense, the bound flips also lead to it losing its attractiveness.

In summary, cost perturbation leads to the dual revised simplex method choosing fewer dense pivotal rows (and not until later iterations) and choosing sparser pivotal columns. This feeds back by increasing the sparsity of the simplex tableaux from which future pivotal rows and columns are chosen.

4. Other techniques

Another advanced algorithmic technique which has surprising consequences for promoting hyper-sparsity is "partial CHUZR", in which CHUZR is restricted to a random subset of the rows. Although the Authors know of no published reference to this technique, it is used within Clp [8]. This talk will present evidence of, and explanation for, the effectiveness of partial CHUZR. Novel algorithmic techniques that are similarly effective may also be presented.

References

[1] J. A. J. Hall, K. I. M. McKinnon, Hyper-sparsity in the revised simplex method and how to exploit it, Computational Optimization and Applications 32 (3) (2005) 259–283.

[2] J. A. J. Hall, Q. Huangfu, A high performance dual revised simplex solver, Tech. Rep. ERGO-11-007, School of Mathematics, University of Edinburgh, submitted to Springer Lecture Notes in Computer Science (2011).

[3] J. J. Forrest, D. Goldfarb, Steepest-edge simplex algorithms for linear programming, Mathematical Programming 57 (1992) 341–374.

[4] R. Fourer, Notes on the dual simplex method, Tech. rep., Department of Industrial Engineering and Management Sciences, Northwestern University (1994).

[5] A. Koberstein, Progress in the dual simplex algorithm for solving large scale LP problems: techniques for a fast and stable implementation, Computational Optimization and Applications 41 (2) (2008) 185–204.

[6] W. J. Carolan, J. E. Hill, J. L. Kennington, S. Niemi, S. J. Wichmann, An empirical evaluation of the KORBX algorithms for military airlift applications, Operations Research 38 (2) (1990) 240–248.

[7] D. Klingman, A. Napier, J. Stutz, NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost network flow problems, Management Science 20 (1974) 814–822.

[8] R. Lougee-Heimer *et al.*, The COIN-OR initiative: Open source accelerates operations research progress, ORMS Today 28 (4) (2001) 20–22.