Industrial and
Systems Engineering

Supplement to the paper: On Pathological Disjunctions and Redundant Disjunctive Conic Cuts

Mohammad Shahabsafa

Department of Industrial and Systems Engineering, Lehigh University, USA

Julio C. Góez

Business and Management Science Department, NHH Norwegian School of Economics

Tamás Terlaky

Department of Industrial and Systems Engineering, Lehigh University, USA

ISE Technical Report 18T-007

LEHIGH UNIVERSITY

Supplement to the paper: On Pathological Disjunctions and Redundant Disjunctive Conic Cuts

Mohammad Shahabsafa[1], Julio C. Góez[2], and Tamás Terlaky[3]

[1] Department of Industrial and Systems Engineering, Lehigh University, USA
[2] Business and Management Science Department, NHH Norwegian School of Economics
[3] Department of Industrial and Systems Engineering, Lehigh University, USA

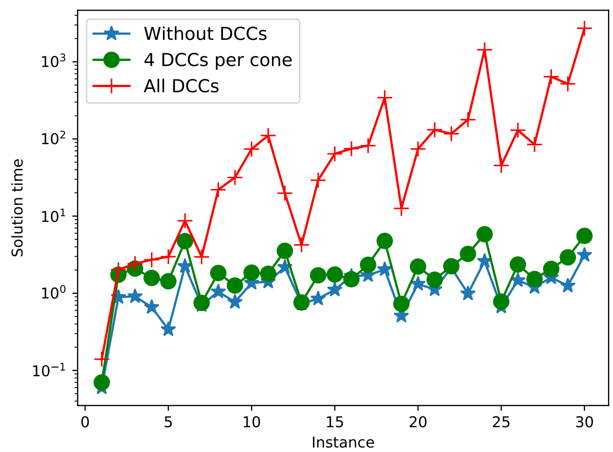
June 18, 2018

A Numerical experiments

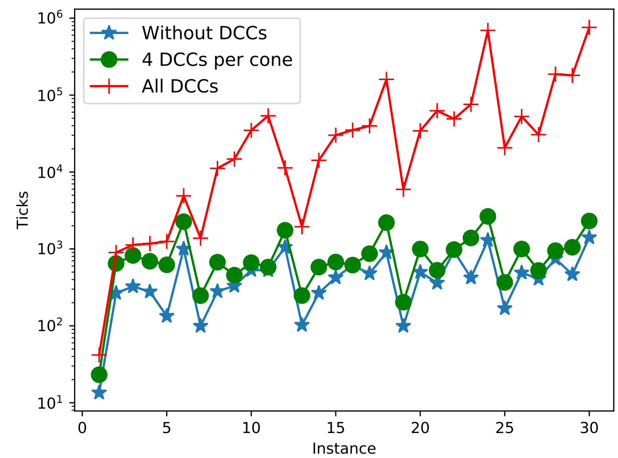
The main problem of failing to recognize the redundant DCCs and DCyCs is that current solvers do not yet have the prepossessing capabilities to recognize those redundancies. As a result, their performance, when solving a MISOCO problem after adding redundant DCCs or DCyCs, may suffer leading to significant increase in solution time. In this section we demonstrate this phenomenon by solving two problem sets. First, we consider portfolio optimization problems with higher moment coherent risk (HMCR) measures presented by Vinel and Krokhmal [2014]. Second, we consider two conic formulations for service system design problems with congestion as presented in Góez and Anjos [2018]. We derive DCCs using the method presented by Belotti *et al.* [2017], and all the DCCs derived satisfy the conditions of Corollaries 1 and 2 [Shahabsafa *et al.*, 2018], thus all are redundant. We use CPLEX 12.8 with the default parameters to solve all the problems. For measuring the performance we use the wall-clock time and the deterministic time measured in *ticks* provided by CPLEX [IBM Knowledge Center, 2017].

A.1 The portfolio optimization problem

We consider the 4^{th} moment coherent risk measure and use the method proposed by Ben-Tal and Nemirovski [2001] to reformulate the 4^{th}-order cone optimization problem [see Vinel and Krokhmal, 2014, Eq. 49] as a second order cone optimization problem. We also consider the round-lot constraints, which represent a real-life policy that assets can be purchased only in *lots* of shares. Thus, the portfolio optimization problem with HMCR measures and round-lot constraints may be formulated as a MISOCO problem. The DCCs are added at the root node for 30 portfolio optimization problems with different number of assets and different number of scenarios. We use three different strategies: first we solve each problem without DCCs, second we derive 4 DCCs for each SOC, and finally we derive all possible DCCs for all the SOCs. In Figures 1(a) and 1(b), the wall-clock solution time and number of ticks is reported for the three different approaches in solving the 30 portfolio optimization problems. For most of the 30 instances, the solution time and ticks lines for the strategy without adding redundant DCCs are below the line for strategy where 4 redundant DCCs are added for each SOC. Additionally, the solution time and ticks without adding DCCs is significantly less than that of the cases where all possible DCCs are added.



(a) Solution time

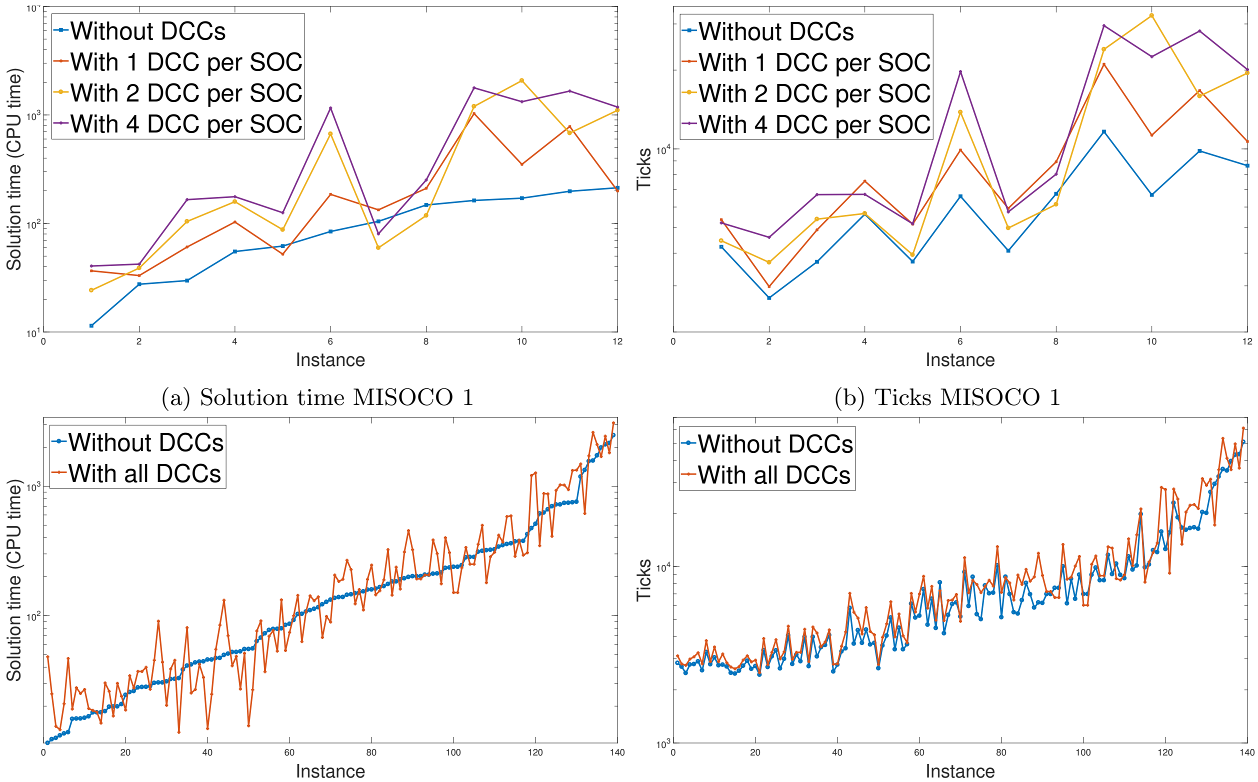


(b) Ticks

Figure 1: Numerical results for portfolio optimization problems

A.2 The service system design problems with congestion

For these problems we consider formulations MISOCO 1 and MISOCO 4 and the instances presented in Góez and Anjos [2018]. For each of these instances we have ℓ client locations and m facility locations, where $\ell \in \{10, 20, 30\}$, and $m \in \{50, 100, 150, 200\}$. For formulation MISOCO 1 the dimension of the SOCs is given by the number of client locations plus two, and the total number of cones is given by the locations available to open. Hence, for these formulations the potential number of DCCs one could add per cone is ℓ. For formulation MISOCO 4 the number of SOCs is equal to ℓm, but all the cones are three dimensional, and one can derive only one DCC per cone. In this instance the dimension of the cones plays a role in its difficulty, in particular formulation MISOCO 1 has higher dimensional cones than formulation MISOCO 4. We observe that MISOCO 1 needs more computational time, while the formulations are equivalent. We set a CPU time limit of 3600 seconds. Figures 2(a) and 2(b) show the results of the experiments with MISOCO 1, where we add up to four DCCs per cone. There we list the results only for the instances that were solved within the time limit, which were 12 in total. Same behavior can be observed as with the portfolio problems in Section A.1, i.e., the larger the number of redundant DCCs is, the larger the solution time. Figures 2(c) and 2(d) show the results of the MISOCO 4 experiments. Here we manage to solve 139 instances, and for each instance we add one DCC per cone for all the cones, which accounts for all the possible DCCs. We see in Figure 2(c) that the results are more mixed than in the previous two experiments, showing that the CPU solution time does not always worsen when the DCCs are added. Nonetheless, the solution time increases in 60 percent of the instances when DCCs are added. The negative effect of adding redundant DCCs to MISOCO 4 formulation is more clear in Figure 2(d), where we can see that the line showing the deterministic time taken to solve the instances without the DCCs is more consistently below the line for the instances with DCCs. To be more specific, the CPLEX deterministic solution time increased in 87 percent of the instances when DCCs were added.



(a) Solution time MISOCO 1

(b) Ticks MISOCO 1

(c) Solution time MISOCO 4

(d) Ticks MISOCO 4

Figure 2: Numerical results of the portfolio optimization problem

B Discussion illustrations

B.1 Illustration of conic cylinders

Figure 3 illustrates this result showing two different cases of redundant DCyCs. In Figure 3(a) we have a case where $a \not\perp \text{lin}(\mathcal{X})$, which complies with Corollary 3 [Shahabsafa *et al.*, 2018], thus we have a redundant DCyC. In Figure 3(b) we have a case where $a \perp \text{lin}(\mathcal{X})$, so it does not satisfy the conditions of Corollary 3 [Shahabsafa *et al.*, 2018]. However, it complies with Corollary 6 [Shahabsafa *et al.*, 2018], thus we have a redundant DCyC. The classification of Figure 3(b) may be obtained noting that the base of the cylinder is a convex cone and its vertex is in one of the half spaces defining the disjunction. Henceforth, the original cylinder configures a redundant DCyC.

(a) A cylindrical redundant DCyC

(b) A conic redundant DCC case for the base of the cylinder

Figure 3: Illustration of redundant cases for conic cylinders

B.2 Illustration of branching on a higher dimensional subspace

Figure 4 illustrates branching on a higher dimensional space, where the quadratic set defines a cylinder with an ellipsoid base defined in the space of (x_1, x_2), and we make a disjunction on variable ξ.

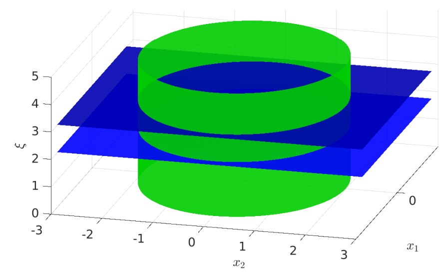


Figure 4: A redundant DCyC

B.3 Illustration of eliminating pathology by branching

Figure 5 illustrates the elimination of pathology by branching on a variable. Suppose that $\hat{\mathcal{C}}$ is a cone, with a vertex v, and one wants to make a disjunction on the binary variable x_2. Notice that the vertex of the cone is in one of the disjunctive half spaces in Figure 5. Then, the DCC will be equal to $\hat{\mathcal{C}}$, which is a redundant DCC. In this case, one can branch on the binary variable x_2 to obtain new quadratic sets in each branch. Consider first the branch $x_2 = 0$, the new quadratic set is obtained from the intersection of $\hat{\mathcal{C}}$ with the hyperplane $x_2 = 0$, which defines a two-dimensional SOC. In this branch, making a disjunction on the binary variable x_1 again leads to a redundant DCC. Now consider the branch $x_2 = 1$; the new quadratic set is obtained from the intersection of $\hat{\mathcal{C}}$ with the hyperplane $x_2 = 1$, which is one branch of a hyperboloid. Considering a disjunction on the binary variable x_1, one can derive a useful DCC in this branch.

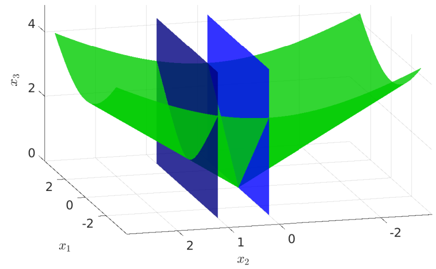


Figure 5: An instance of the conic redundant DCC

References

P. Belotti, J. C. Góez, I. Pólik, T. K. Ralphs, and T. Terlaky. A complete characterization of disjunctive conic cuts for mixed integer second order cone optimization. *Discrete Optimization*, 24:3 – 31, 2017.

A. Ben-Tal and A. Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. SIAM, 2001.

J. C. Góez and M. F. Anjos. Second order conic optimization formulations for service system design problems with congestion. Under review, 2018.

IBM Knowledge Center. IBM ILOG CPLEX Optimization Studio V12.8.0 documentation, 2017.

M. Shahabsafa, J. C. Góez, and T. Terlaky. On pathological disjunctions and redundant disjunctive conic cuts. *To appear in Operations Research Letters*, 2018.

A. Vinel and P. A. Krokhmal. Polyhedral approximations in p-order cone programming. *Optimization Methods and Software*, 29(6):1210–1237, 2014.