INFORMS OS Today

The Newsletter of the INFORMS Optimization Society

Volume 8 Number 2 November 2018

Contents

Send your comments and feedback to the Editor:

Marina A Epelman
Industrial and Operations Engineering
University of Michigan, Ann Arbor, USA
mepelman@umich.edu

Chair's Column

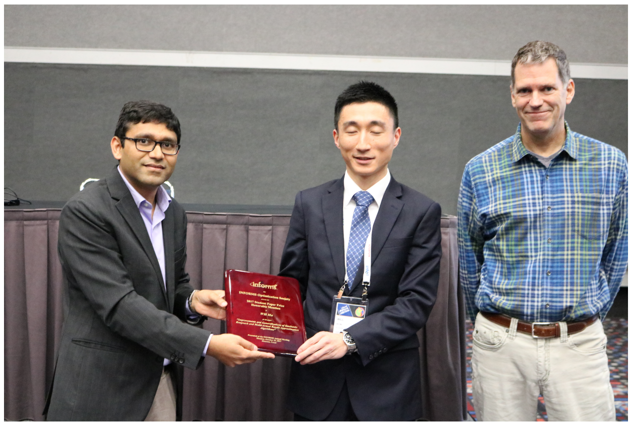
David Morton
Industrial Engineering and Management Sciences
Northwestern University
USA
david.morton@northwestern.edu

Dear Fellow IOS Members:

The annual INFORMS Meeting in Phoenix is upon us! I look forward to seeing you during a productive and enjoyable conference.

IOS Sessions in Phoenix: The Optimization Society is sponsoring 125 sessions at the conference. We sponsored 93 last year, and 91 the year before that.

Business meeting: The IOS Business Meeting at the annual conference will be held on Sunday,



Vineet Goyal, Will Ma (recipient of 2017 Honorable Mention for Student Paper) and David Morton

November 4, 6:15–7:15pm in the Convention Center, Room North 124A, 100 Level. Light refreshments, along with beer and wine, will be provided.

IOS prizes: Congratulations to the winners of the 2018 INFORMS Optimization Society prizes!

This year, the Khachiyan Prize is being awarded to three winners: John N. Hooker, James Renegar, and Werner Römisch. Shabbir Ahmed is receiving the Farkas Prize. Two winning teams are awarded the Prize for Young Researchers: Amir Ali Ahmadi and Georgina Hall, as well as Dan Iancu and Nikos Trichakis. The Student Paper Prize honorable mentions recipients are Yanli Liu, the team of Naman Agarwal and Brian Bullins, and Nam Ho-Nguyen. Second prize in the competition is awarded to Felix Happach and Joey Huchette. And, the winner of the Student Paper Prize this year is Yong Sheng Soh.

The winners will receive their prizes at the IOS Business Meeting. Please join us for two sessions in which the prize recipients will present their award-winning work on Tuesday, November 6, in West Bldg 102C. First, the three Khachiyan Prize winners will speak in session TB61, 10:30am–noon. Afterwards, the winners of the Young Researchers Prize and the winner of the Student Paper Prize will present their work in session TC61, 12:05pm–1:35pm.

We are grateful to the prize committees for their work: *Khachiyan Prize*: Suvrajeet Sen (chair), Ignacio Grossman, Arkadi Nemirovski, and David Shmoys; *Farkas Prize*: Patrick Jaillet (chair), Donald Goldfarb, Andy Philpott, and Nick Sahinidis; *Young Researchers Prize*: Katya Scheinberg (chair), Yongpei Guan, Fatma Kılınç-Karzan, and Andrea Lodi; *Student Paper Prize*: Dan Iancu (chair), Amir Ali Ahmadi, Frank Curtis, and Illya Hicks.

IOS board transitions: IOS officer elections were held in September. We welcome our new vice-chairs, who will be assuming their responsibilities on January 1, 2019:

- Global Optimization: Georgina Hall;
- Nonlinear Optimization: Shiqian Ma;
- Optimization Under Uncertainty: Ruiwei Jiang.

Please join me in thanking the vice chairs who complete their terms this year: Siqian Shen (Global Optimization), Necdet Serhat Aybat (Nonlinear Optimization), and Güzin Bayraksan (Optimization Under Uncertainty).

Austin Buchanan has served as the IOS representative to the INFORMS Subdivision Council for the last two years. Thank you to Austin for representing well the interests of IOS, and we welcome Siqian Shen into the same role for 2019–2020. Thank you to Sertalp Çay, our Web Editor, for managing the IOS Connect website and for his tweets @InformsOS, too.

I am deeply grateful for the leadership of Marina Epelman and Burcu Keskin. Marina shapes our newsletter in an outstanding way and, with the continued growth of IOS, she has taken a high quality newsletter to twice a year. Burcu is efficient and effective, is a great communicator, and is fun to work with as our Secretary/Treasurer. Thank you Burcu and Marina!

Finally, thank you to Chair-Elect, Daniel Bienstock, for his ideas and work over the last year, and for his enthusiasm to serve as the Chair of IOS for 2019-2020.

I wish you a wonderful conference!

DUALITY IN TWO-STAGE ADAPTIVE LINEAR OPTIMIZATION: FASTER COMPUTATION AND STRONGER BOUNDS

Frans F.J.C.T. de Ruiter
Tilburg University, The Netherlands
fjctderuiter@gmail.com

Dimitris Bertsimas
Massachusetts Institute of Technology, USA
dbertsim@mit.edu

We are deeply honoured and grateful to have been awarded the 2017 INFORMS Optimization Society Student Paper Prize for our work "Duality in two-stage adaptive linear optimization: faster computation and stronger bounds." We would like to thank the committee consisting of Vineet Goyal, Kiavash Kianfar, Javier Peña, and Ermin Wei for their effort and for honoring our work. This article explains the broader context of the paper and surveys the results presented in [5].

1 Introduction

Many applications for decision making under uncertainty can be naturally modeled as two-stage adaptive robust optimization models. In an adjustable robust optimization model we first make *here-and-now* decisions before observing the realization of the uncertain parameters, and then set the policy for the *wait-and-see* decisions to satisfy all the constraints. An example of such a problem is the lot-sizing problem with distribution on a network. In this problem one has multiple stores that can hold units of stock, while each store faces some uncertain demand. In the first-stage decision, before knowing the realization of the demand, one has to decide how to allocate the stock. After the demand is realized there is then the opportunity to redistribute part of the stock to other stores, in order to meet demand against additional transportation costs. An instance of this problem is depicted in Figure 1.

Vineet Goyal and Frans J.C.T. de Ruiter

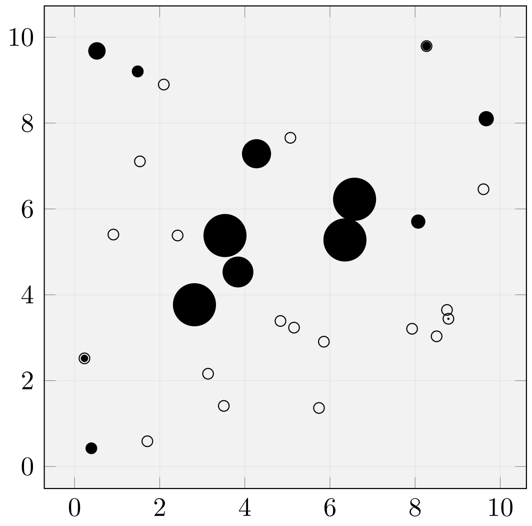


Figure 1: Stock allocation for an instance of the lot-sizing problem on a network with 30 stores on the grid $[0, 10]^2$. The bigger the filled dots are, the more stock is allocated. The open dots do not have stock allocated to them.

There are many more applications that use a two-stage robust setting; see our paper [5] and the survey papers [4] and [9] for more examples.

Two-stage adaptive models under uncertainty have been dealt with in various ways. One way of dealing with these problems is via stochastic programming; see, for example, [12]. Another very tractable way of dealing with these adaptive models, which became popular since the seminal paper by [3], is by using robust optimization. In these models we have variables $\mathbf{x}$ residing in some convex set $\mathcal{X} \subset \mathbb{R}^n$ to describe the here-and-now decisions. The uncertainty is modelled using uncertainty parameters $\boldsymbol{\zeta}$ that can take any value in some compact convex set $\mathcal{U} \subset \mathbb{R}^L$ [2]. In this paper we focus on polyhedral uncertainty sets of the form

$$\mathcal{U} = \{\boldsymbol{\zeta} \geq \mathbf{0} \mid \mathbf{D}\boldsymbol{\zeta} \leq \mathbf{d}\}, \tag{1}$$

with $\mathbf{D} \in \mathbb{R}^{p\times L}$ and $\mathbf{d} \in \mathbb{R}^p$. In contrast to stochastic programming, robust optimization approaches result in solutions that provide a priori guarantees on feasibility and the objective value. Also unlike in stochastic programming, only limited knowledge of the distribution of the uncertain parameter is needed. In this case, only the support of the distribution has to be known, or only that part of the support for which one wants to be protected. Popular choices of uncertainty sets are the polyhedral uncertainty sets such as the box uncertainty sets (a hypercube), or budget uncertainty sets that put an additional cardinality restriction on the deviations from some nominal value; see [6]. In two-stage models, we also have variables $\mathbf{y} \in \mathbb{R}^k_+$ for the wait-and-see decisions, which can be adjusted after the realization of the uncertain parameter is known.

2 Adaptive robust models

There are several equivalent representations of two-stage adaptive robust optimization models, but for our purpose it is most convenient to use a "min-max-min" formulation as follows:

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \min_{\mathbf{y}\geq\mathbf{0}} \left\{\mathbf{c}^\top\mathbf{x} \mid \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \geq \mathbf{R}(\mathbf{x})\boldsymbol{\zeta} + \mathbf{r}\right\}, \tag{2}$$

where $\mathcal{X} \subset \mathbb{R}^n$ is a set with additional constraints on the here-and-now decisions (some of the $\mathbf{x}$ variables may be integer), $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{B} \in \mathbb{R}^{m\times k}$, $\mathbf{R}(\mathbf{x}) = \mathbf{R}^0 + \sum_{i=1}^n \mathbf{R}^i x_i$ with $\mathbf{R}^i \in \mathbb{R}^{m\times L}$ for all $i = 0,\dots,n$, and $\mathbf{r} \in \mathbb{R}^m$. The middle 'max' indicates that we take a robust perspective and aim to find the solution that minimizes under worst-case outcomes. As we can choose a different $\mathbf{y} \in \mathbb{R}^k$ for every $\boldsymbol{\zeta}$, the optimal solution will be a policy depending on $\boldsymbol{\zeta}$, $\mathbf{y}(\boldsymbol{\zeta})$. Therefore, an equivalent representation of (2) would be

$$\min_{\mathbf{x}\in\mathcal{X},\tilde{\mathbf{y}}(\cdot)\in\mathcal{F}} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \left\{\mathbf{c}^\top\mathbf{x} \mid \mathbf{A}\mathbf{x} + \mathbf{B}\tilde{\mathbf{y}}(\boldsymbol{\zeta}) \quad \geq \mathbf{R}(\mathbf{x})\boldsymbol{\zeta} + \mathbf{r}\right\},$$

where $\mathcal{F} = \left\{\tilde{\mathbf{y}}(\cdot) \mid \tilde{\mathbf{y}} : \mathbb{R}^L \to \mathbb{R}^k_+\right\}$, a class of policies. Optimizing over the class of all policies $\mathcal{F}$ is an intractable task, but very good solutions can be obtained by restricting to affine policies for $\tilde{\mathbf{y}}(\cdot)$; see [3]. These affine policies are of the form $\tilde{\mathbf{y}}(\boldsymbol{\zeta}) = \bar{\mathbf{y}} + \mathbf{Y}\boldsymbol{\zeta}$, with $\bar{\mathbf{y}} \in \mathbb{R}^k$ and $\mathbf{Y} \in \mathbb{R}^{k\times L}$ the new here-and-now variables. The resulting model is again a tractable robust model that can be solved using standard solvers for linear optimization. Note that these models allow the here-and-now decisions $\mathcal{X}$ to be integer, but the wait-and-see decisions must be continuous. Also note that the matrix $\mathbf{B}$ does not depend on $\boldsymbol{\zeta}$, so we consider two-stage models with fixed recourse only.

3 An equivalent representation by duality

Although affine policies are a very efficient way to find solutions for (2), there are two challenges. First, the model with affine adaptive policies (and variables $\bar{\mathbf{y}}, \mathbf{Y}$) is slower to solve than the corresponding non-adaptive model. Second, since we restrict the class of policies to affine policies, we find more conservative solutions and it is hard to estimate how far affine policies are from optimal policies. Therefore, in this section we present an equivalent representation using duality for linear programming to make computation faster and to provide stronger lower bounds on the performance of affine policies compared to optimal policies. The full proof can be found in [5].

Theorem 1. *The here-and-now decision* $\mathbf{x}$ *is feasible (and optimal) for* (2) *with nonempty uncertainty set* $\mathcal{U}$ *as in* (1) *if and only if* $\mathbf{x}$ *is feasible (and optimal) for*

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\mathbf{w}\in\mathcal{V}} \min_{\boldsymbol{\lambda}\geq\mathbf{0}} \Big\{\mathbf{c}^\top\mathbf{x} \mid \mathbf{w}^\top(\mathbf{A}\mathbf{x} - \mathbf{r}) - \mathbf{d}^\top\boldsymbol{\lambda} \geq \mathbf{0}, \quad \mathbf{D}^\top\boldsymbol{\lambda} \geq \mathbf{R}(\mathbf{x})^\top\Big\}, \tag{3}$$

where $\mathcal{V} = \left\{\mathbf{w} \geq \mathbf{0} : \ \mathbf{B}^\top\mathbf{w} \leq \mathbf{0}, \ \mathbf{e}^\top\mathbf{w} = 1\right\}$.

Sketch of proof. We can apply the strong duality theorem from linear optimization to the inner minimization problem of (2) resulting in a "min-max-max" problem:

$$\min_{\mathbf{x}\in\mathcal{X}}\max_{\boldsymbol{\zeta}\in\mathcal{U}}\max_{\mathbf{w}\geq\mathbf{0}}\Big\{\mathbf{c}^\top\mathbf{x}+\mathbf{w}^\top(\mathbf{R}(\mathbf{x})\boldsymbol{\zeta}+\mathbf{r}-\mathbf{A}\mathbf{x}) \mid \mathbf{B}^\top\mathbf{w}\leq\mathbf{0}\Big\},$$

where $\mathbf{w}\in\mathbb{R}^m_+$. This formulation has been used by many to solve two-stage problems using Benders decomposition techniques. Here we go one step further by dualizing over $\boldsymbol{\zeta}$ to obtain (3). □

From many points of view the primal formulation (2) and dualized formulation (3) have similar structure: they are both two-stage adaptive optimization models that are linear in the wait-and-see decisions $\mathbf{y}$ and $\boldsymbol{\lambda}$, respectively, with fixed recourse. Moreover, they can be solved using the same solution techniques of affine policies: in the dualized formulation (3) we can use affine policies $\tilde{\boldsymbol{\lambda}}(\mathbf{w})=\mathbf{q}+\mathbf{Q}\mathbf{w}$. Also, in the dualized version the affine policiy is a conservative approximation of the optimal (nonlinear) policy. However, one can prove that both the original model (2) and the dualized formulation (3) give the same objective value.

Proposition 2. *For the here-and-now decision* $\mathbf{x}$*, there exists a feasible policy* $\tilde{\boldsymbol{\lambda}}(\mathbf{w})=\mathbf{q}+\mathbf{Q}\mathbf{w}$ *for* (3) *if and only if there exists a feasible affine policy* $\tilde{\mathbf{y}}(\boldsymbol{\zeta})=\bar{\mathbf{y}}+\mathbf{Y}\boldsymbol{\zeta}$ *for* (2) *for the same* $\mathbf{x}$ *.*

Not only do both affine policies result in the same set of feasible here-and-now solutions for $\mathbf{x}$, in [5] we also show that one affine policy can be directly obtained from the other and vice versa. Therefore, we could solve either (2) or (3) with affine policies, or even solve both in parallel and stop the solver whenever an optimal solution is found for one of the models. Furthermore, there are several problems for which the new dualized formulation solves an order of magnitude faster as explained in our numerical results later.

The second use of our approach is to strengthen the lower bounds. One well-established way to find a lower bound for (2) is to use a finite subset $\{\boldsymbol{\zeta}^1,\ldots,\boldsymbol{\zeta}^N\}\subset\mathcal{U}$ and solve the sampled model:

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}^1,\ldots,\mathbf{y}^N\geq\mathbf{0}}\Big\{\mathbf{c}^\top\mathbf{x} \mid \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}^i\geq\mathbf{R}(\mathbf{x})\boldsymbol{\zeta}^i+\mathbf{r},\ i=1,\ldots,N\Big\}. \tag{4}$$

Note that this model indeed provides a lower bound to the optimal value of (2) as we are only protected for a subset of scenarios from $\mathcal{U}$. The quality of the lower bound depends on the scenarios that are picked. One effective way of picking scenarios is by using a particular method developed in [11]. Again we can use the fact that the dualized model (3) is of the same form as (2) and apply the same lower bounding technique to the dualized model. In fact, we show in our paper that we can also combine the sampled constraints of the resulting models to obtain an even better lower bound.

4 Numerical example

In this section we show that (3) with affine policies takes an order of magnitude less time to solve than the primal formulation (2) with affine policies for the lot-sizing example that was briefly mentioned in Section 1. Also, the new lower bound using the information from the equivalent dual formulation is much stronger than the lower bound from [11] that only used scenarios from the primal formulation. We consider various instance sizes (with different numbers of stores). For the full description of the problem we again refer to [5]. All computations were carried out with Gurobi 6.0.3 [10] on an Intel i7-4770 3.40GHz Windows computer with 8GB of RAM. All modeling was done using the modeling language JuMP embedded in Julia programming language [8].

We solve both models with affine policies and depict the average solution times over 10 runs in Table 1, along with the objective value and the lower bounds. The stock allocation (the here-and-now decision) for one instance with $N=30$ is depicted in Figure 1. The lower bound from the primal formulation is obtained using the method from [11]. The primal/dual bound is the strengthened bound resulting from combining the primal and dual scenarios. Solving the model via the new dualized formulation reduces the computation time by an order of magnitude compared with the original primal formulation. For the larger instances we see that the primal formulation is approximately 20 times slower. These results are averaged over 10 runs to avoid random peak performances, but in each individual run we observed a significant decrease in computation time.

Table 1: Performance of primal and dualized formulation with affine policies for the lot-sizing example.

	Solver time (sec)		Objective value	Lower Bound (Gap%)	
N	Primal	Dual		Primal	Primal/Dual
10	< 0.1	< 0.1	928	797 (**14.0%**)	824 (**11.1%**)
20	0.3	0.1	1353	1113 (**17.7%**)	1190 (**12.0%**)
30	2.6	0.8	1670	1356 (**18.8%**)	1465 (**12.3%**)
40	11.8	2.6	1947	1562 (**19.8%**)	1728 (**11.3%**)
50	42.0	7.3	2188	1728 (**21.0%**)	1934 (**11.6%**)
60	142.2	20.5	2421	1912 (**21.0%**)	2160 (**10.8%**)
70	366.0	41.3	2598	1996 (**23.2%**)	2312 (**11.0%**)
80	826.9	88.7	2781	2136 (**23.2%**)	2495 (**10.3%**)
90	1647.1	179.8	2953	2252 (**23.8%**)	2641 (**10.6%**)
100	4026.2	231.0	3130	2408 (**23.1%**)	2799 (**10.6%**)

The strengthened primal/dual bound is much tighter than the primal bound, more than halving the optimality gap for the larger instances.

We refer to [5] for a discussion of a facility location problem with uncertain customer demand and numerical experiments on instances of various sizes.

5 Conclusion and further steps

In this article, we have used duality for linear optimization to derive an equivalent formulation of a primal two-stage adaptive model. The resulting dualized formulation is again a two-stage adaptive model. We show that optimal affine policies for the primal formulation can be directly constructed from optimal affine policies in the dual formulation. Via two examples of lot-sizing and a facility location problem, we show that the dualized models, when coupled with affine policies, can reduce computational time to solve adaptive problems by an order of magnitude. Furthermore, we provide a method that uses the affine policies in the dual model to strengthen bounds on the optimality gap of affine policies.

There are several ways in which these dualization techniques can be further exploited. In static robust optimization (without wait-and-see decisions $\mathbf{y}$) duality theory has been applied in a variety of ways to formulate tractable optimization problems. We can extend our result for linear two-stage adaptive robust optimization models to nonlinear variants as well, which are described in a working paper [7]. Finally, dualization for the linear case discussed in this paper has now also been used in various other new solution techniques (see [1] and [13]) showing significant speedups in computation time.

REFERENCES

[1] A. Ardestani-Jaafari and E. Delage. The value of flexibility in robust location–transportation problems. *Transportation Science*, 52(1):189–209, 2017.

[2] A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

[3] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[4] D. Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[5] D. Bertsimas and F.J.C.T. de Ruiter. Duality in two-stage adaptive linear optimization: faster computation and stronger bounds. *INFORMS Journal on Computing*, 28(3):500–511, 2016.

[6] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[7] F.J.C.T. de Ruiter, J. Zhen, and D. den Hertog. Dual approach for two-stage robust nonlinear optimization. *Working paper*, 2018. Available on: `http://www.optimization-online.org/DB_FILE/2018/03/6522.pdf`.

[8] I. Dunning, J. Huchette, and M. Lubin. JuMP: a modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[9] V. Gabrel, C. Murat, and A. Thiele. Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483, 2014.

[10] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015. `http://www.gurobi.com`.

[11] M.J. Hadjiyiannis, P.J. Goulart, and D. Kuhn. A scenario approach for estimating the suboptimality of linear decision rules in two-stage robust optimization. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7386–7391, Dec 2011.

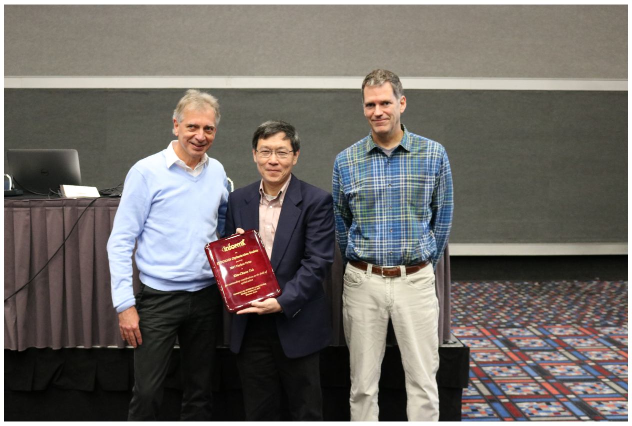
[12] P. Kali and S.W. Wallace. *Stochastic programming*. Springer, 1994.

[13] J. Zhen, D. den Hertog, and M. Sim. Adjustable robust optimization via fourier-motzkin elimination. *Working paper*, 2016. Available on: `www.optimization-online.org/DB_FILE/2016/07/5564.pdf`.

SOME NUMERICAL ISSUES IN THE DEVELOPMENT OF SDP ALGORITHMS

Kim-Chuan Toh
Department of Mathematics
National University of Singapore
Singapore
mattohkc@nus.edu.sg

I am deeply honoured to have been awarded the 2017 Farkas Prize from the INFORMS Optimization Society. I am grateful to the selection committee for awarding me this prize. This award would not have been possible without the guidance and mentorship of some of the most wonderful persons I have met in my career over the past 22 years. First and foremost, Mike Todd and Nick Trefethen have had tremendous influence on my career progression since my graduate studies at Cornell University. I have the good fortune to know Stephen Boyd, Robert Freund, Masakazu Kojima, Tom Luo, Michael Overton, Jong-Shi Pang, Franz Rendl, Christine Shoemaker, Stephen Vavasis, Stephen Wright, Yinyu Ye, Henry Wolkowicz, Yin Zhang, and many others to whom I would apologize here for not mentioning their names explicitly, who have helped me in one way or another. My conviction to work on interior-point methods (IPMs) for SDP has been very much influenced by the work of the people whose names I just mentioned in offering the golden opportunities for a young numerical analyst trained in numerical linear algebra to work in a flourishing area in optimization. It was indeed fortuitous for me to be able to change track from iterative linear algebra immediately after my PhD to numerical algorithms for SDP with almost no barrier because of my good fortune to start the collaboration with Mike Todd and Reha Tutuncu, first on studying the Nesterov-Todd direction for SDP [51], and later on SDPT3 [52, 54, 53].


Pascal Van Hentenryck, Kim-Chuan Toh, and David Morton

In my involvement in the practical implementations of IPMs for SDPs, I have benefited from various discussions with Brian Borchers, Michael Grant, Johan Lofberg, and Hans Mittelman.

I must express my special thanks to Defeng Sun for the wonderful collaborations we have had on augmented Lagrangian-based methods for solving SDPs, and more generally, convex conic programming. Of course, our work in the past 12 years or so would not have been possible without the hard work and contributions of our PhD students, postdocs, and collaborators. I thank them all for such an enjoyable experience!

The remainder of this article is a selection of some numerical issues I have encountered in the development of algorithms for SDPs. The first part is on interior-point-based algorithms, and the second part is on augmented Lagrangian-based algorithms. Each topic will conclude with a brief description of the challenging questions to be resolved.

1 Introduction

Let $\mathbb{S}^n$ be the space of $n \times n$ real symmetric matrices endowed with the standard inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\|\cdot\|$. We denote the set of symmetric positive semidefinite matrices by $\mathbb{S}^n_+$. A standard primal semidefinite programming (SDP) problem is given by

$$\min \Big\{ \langle C, X \rangle \mid \mathcal{A}(X) = b, X \in \mathbb{S}^n_+ \Big\}, \tag{1}$$

where $C \in \mathbb{S}^n$, $b \in \mathbb{R}^m$ are given data, and $\mathcal{A} : \mathbb{S}^n \to \mathbb{R}^m$ is a given linear surjective map. Typically, $\mathcal{A}$ is written explicitly in the form $\mathcal{A}(X) = [\langle A_1, X \rangle, \ldots, \langle A_m, X \rangle]^T$, where $A_1, \ldots, A_m$ are given constraint matrices in $\mathbb{S}^n$. Correspondingly, its adjoint map $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n$ is given by $\mathcal{A}^* y = \sum_{k=1}^m y_k A_k$. The dual corresponding to (1) is given by

$$\max\Big\{\langle b,\, y\rangle \mid \mathcal{A}^* y + Z = C, Z \in \mathbb{S}^n_+\Big\}. \tag{2}$$

For ease of exposition, here we only consider the SDP problem (1) with a single block variable in the cone $\mathbb{S}^n_+$. In general, one may consider a problem with multiple block variables and each block may be constrained to be in an SDP cone, a second-order cone, the cone of nonnegative vectors, or unrestricted.

The perturbed Karush-Kuhn-Tucker (KKT) optimality conditions for (1) and (2) are given as follows:

$$\mathcal{A}(X) = b,\ \mathcal{A}^* y + Z = C,\ XZ = \mu I,\ X, Z \in \mathbb{S}^n_+, \tag{3}$$

where $\mu > 0$ is the barrier parameter that is to be driven to 0 explicitly. By setting $\mu = 0$ in (3), one obtains the optimality conditions for (1) and (2). We assume that the Slater condition holds for both (1) and (2) (equivalently, (1) and (2) are strictly feasible). Then the unique solution $(X(\mu), y(\mu), Z(\mu))$ to the perturbed KKT conditions exists and would converge to an optimal solution (X^*, y^*, Z^*) of (1) and (2) when $\mu \downarrow 0$; see [20]. As μ varies, the solution $(X(\mu), y(\mu), Z(\mu))$ would continuously trace a path within the primal-dual feasible region of (1) and (2) known as the central-path.

As far as I am aware, most practical implementations of interior-point solvers such as CSDP [3], SDPA [58], SDPT3 [52], SeDuMi [48] for SDP are based on the primal-dual path-following framework, with the exception of DSDP [2] which is based on a dual-scaling framework. For readers who are interested in the relative performance of these solvers, including non-interior-point-based solvers such as PENSDP [24] and SDPNAL [63], we refer them to the benchmarks performed by Hans Mittelmann [30] on some large and sparse SDP problems. Among the solvers, currently SDPT3 and SeDuMi are used in the optimization modeling systems CVX [10, 19] and YALMIP [28].

For the purpose of computing an approximate solution of (1) and (2), we define the relative KKT residual $\eta = \max\{\eta_p, \eta_d, \eta_c, \eta_X, \eta_Z\}$, where

$$\eta_p = \tfrac{\|\mathcal{A}(X)-b\|}{1+\|b\|},\ \eta_d = \tfrac{\|\mathcal{A}^* y + Z - C\|}{1+\|C\|},\ \eta_c = \tfrac{|\langle X, Z\rangle|}{1+\|X\|+\|Z\|},$$

$$\eta_X = \tfrac{\|\min\{0,\lambda(X)\}\|}{1+\|X\|},\ \eta_Z = \tfrac{\|\min\{0,\lambda(Z)\}\|}{1+\|Z\|}.$$

Note that $\lambda(X)$ denotes the vector of eigenvalues of X.

As a rough guide, today one can expect to successfully solve an SDP problem via the interior-point solvers mentioned above to a reasonable level of accuracy of, say, at least 10^{-6} within a reasonable wall-clock time of, say, less than one hour when the number of constraints m is moderate (say, $m \leq$ 20,000) and the dimension of the matrix variable n is also moderate (say, $n \leq$ 5,000). For the case when the dimension n remains moderate but m is large (say, more than 100,000), all the interior-point solvers mentioned in the last paragraph that employ a direct method to compute the search direction in each iteration will run into memory difficulty, not to mention that the computing time will also be excessive. Of course, if one solves the SDP on a super computing system with a parallel implementation of an interior-point method such as SDPARA [60, 59], one can overcome the memory difficulty and even solve an SDP with a few millions constraints. As far as I am aware, SDPARA has solved an SDP problem (with 1.22 million constraints and matrix variable dimension 1600) arising from the doubly nonnegative relaxation of a quadratic assignment problem (`tai40b`) on a parallel computing system (with 256 multi-core CPUs and 128GPUs) in about 122 hours [14].

For the subsequent discussions, we assume that the matrix dimension n in (1) and (2) is moderate, say, less than 10,000, so that the matrix variables can be stored in the RAM unit of a single PC, and matrix operations such as spectral decomposition and Cholesky factorization can be computed at a reasonable cost.

2 Primal-dual path-following interior-point methods

At each iteration (with current iterate (X, y, Z) such that $X, Z \succ 0$ and assumed to be close to $(X(\mu), y(\mu), Z(\mu))$) of all the path-following IPMs mentioned in the last section, the key computational step is to compute the search direction $(\Delta X, \Delta y, \Delta Z) \in \mathbb{S}^n \times \mathbb{R}^m \times \mathbb{S}^n$ from a symmetrized

Newton equation of the following form:

$$\begin{aligned} \mathcal{A}(\Delta X) = R_p,\ \mathcal{A}^*(\Delta y) + \Delta Z = R_d, \\ \mathcal{H}(\Delta X Z + X \Delta Z) = R_c, \end{aligned} \tag{4}$$

where $R_p = b - \mathcal{A}(X)$, $R_d = C - \mathcal{A}^* y - Z$, $R_c = \mathcal{H}(\sigma\mu I - XZ)$ with $\sigma \in (0,1)$ being a given parameter, $\mathcal{H} : \mathbb{R}^{n\times n} \to \mathbb{S}^n$ is a linear operator defined by $\mathcal{H}(U) = \frac{1}{2}(PUP^{-1} + P^{-T}U^TP^T)$, and $P \in \mathbb{R}^{n\times n}$ is a given invertible matrix that is dependent on X, Z. Obviously, the role of $\mathcal{H}$ is to symmetrize the nonsymmetric equation $\Delta XZ + X\Delta Z = \sigma\mu I - XZ$ resulting from the linearization of the perturbed complementarity condition $\sigma\mu I - XZ = 0$. Instead of solving a very large linear system of equations of dimension $n(n+1) + m$, by performing block eliminations on ΔZ and ΔX, one can compute the direction by first computing Δy from an $m \times m$ **dense** symmetric positive definite Schur complement equation (SCE): $\mathcal{M}\Delta y = h$. It turns out that the choice of the symmetrization scheme is critical to the efficiency and stability of solving the resulting SCE, where

$$\mathcal{M}_{ij} = \begin{cases} \langle A_i,\, Q(\Omega \circ (Q^T(XA_j + A_jX)Q))Q^T\rangle & \text{for AHO symmetrization [1] with } P = I \\ \langle A_i,\, \frac{1}{2}(XA_jZ^{-1} + Z^{-1}A_jX)\rangle & \text{for HKM symmetrization [21, 25, 31] with } P = Z^{1/2} \\ \langle A_i,\, WA_jW\rangle & \text{for NT symmetrization [51] with } P = W^{-1/2}. \end{cases}$$

In the above, the notation "$\circ$" denotes the Hadamard product, $Z = Q\text{diag}(d)Q^T$ is a spectral decomposition, $W = Z^{-1/2}(Z^{1/2}XZ^{1/2})^{1/2}Z^{-1/2}$, and $\Omega \in \mathbb{S}^n$ is given by $\Omega_{ij} = \frac{1}{d_i + d_j}$ for all $1 \leq i, j \leq n$.

As demonstrated in [1], the numerical stability of computing Δy via the AHO scheme is better than that corresponding to the HKM and NT schemes. In particular, the coefficient matrix in (4) for the AHO scheme has a bounded condition number when $\mu \downarrow 0$ if the optimal solution (X^*, y^*, Z^*) satisfies the PDNSC conditions (primal and dual nondegeneracy conditions as well as the strict complementarity condition). On the other hand, the corresponding coefficient matrix for the HKM and NT schemes has condition number that grows at least in the order $O(1/\mu)$ even if the optimal solution satisfies the PDNSC conditions just mentioned [55]. However, even under the PDNSC conditions, just like HKM and NT schemes, the matrix $\mathcal{M}$ for the AHO scheme also has a condition number that grows at least in the order $O(1/\mu)$ when $\mu \downarrow 0$ [1]. Despite the apparent lack of advantage in terms of conditioning, [1] has demonstrated empirically that computing the AHO direction via the SCE can produce more accurate approximate optimal solutions for randomly generated SDP problems. Unfortunately, the structure of $\mathcal{M}$ in the AHO scheme makes it extremely hard to fully exploit sparsity in the constraint matrices $\{A_1, \dots, A_m\}$ and computing $\mathcal{M}$ can be prohibitively expensive. On the other hand, for the HKM and NT schemes, it is much easier to fully exploit sparsity in the constraint matrices when computing $\mathcal{M}$ (see [15] for the details). Thus, the solvers [3], [58] have adopted the HKM scheme, and [48] has adopted the NT scheme. For SDPT3 [52], we have implemented both the HKM and NT schemes. However, observe that in general the iterate $X \succ 0$ is completely dense, and hence $\mathcal{M}$ is also dense. Thus, even though the sparsity in the constraint matrices can be exploited in the HKM and NT schemes, the low-rank property (by [34], there exist a rank-r primal optimal solution such that $r(r+1) \leq 2m$) of the optimal solution X^* or Z^* cannot be exploited in an IPM solver.

The extreme ill-conditioning of the SCE (when μ is small) indeed poses serious numerical challenges to the accurate computation of the search direction that ultimately limits the accuracy level one can attain when solving the SDP problems (1) and (2). (We should mention that other numerical difficulties can also arise if the problem (1) or (2) does not satisfy the Slater's condition, but we shall not touch on those issues here.) In fact, the extreme ill-conditioning of $\mathcal{M}$ often makes it numerically indefinite although $\mathcal{M}$ is mathematically positive definite. In SDPT3, we add a small positive definite diagonal perturbation D to $\mathcal{M}$ so that the resulting Cholesky factorization can be computed successfully. After that, the SCE is solved by using an iterative solver, namely the preconditioned symmetric quasi-minimal residual (PSQMR) method with the preconditioner chosen to be the computed Cholesky factorization of $\mathcal{M}+D$. We find that such an approach can typically produce a more accurate approximate optimal solution than the straightforward approach that directly uses the computed Cholesky factor R of $\mathcal{M}+D$ to find Δy via $R^{-1}(R^{-T}h)$.

Another approach to partially alleviate the numerical difficulties in solving the extremely ill-conditioned SCE has been adopted by SeDuMi, wherein the search direction in each iteration is computed in a scaled space that is dependent on the current iterate. However, the drawback is that the computation of the associated Schur complement matrix can be much more costly than that of $\mathcal{M}$. Thus, in SDPT3, we focus on solving the SCE with greater computational efficiency but at the expense of computing a search direction that may not be as accurate as the scaled space approach.

Apart from the extreme ill-conditioning of $\mathcal{M}$ that arises as the iterates converge to an optimal solution when $\mu \downarrow 0$, other issues, such as the case when the optimal solution does not satisfy the PDNSC conditions or when $\mathcal{A}$ is ill-conditioned, can further exacerbate the numerical difficulty in solving the SCE. In addition, when either (1) or (2) does not satisfy Slater's condition, the central-path $\{(X(\mu), y(\mu), Z(\mu) \mid \mu > 0\}$ is not well-defined. In the latter scenario, although in practice one can continue to solve an SDP problem by using an infeasible path-following IPM solver, the numerical behavior of the solver can worsen and sometimes one can only solve the problem to a low accuracy level of 10^{-4}–10^{-3} in the relative KKT residual. An elegant approach to overcome this difficulty is to reformulate the SDP into a homogeneous self-dual (HSD) model [62] (for which the Slater condition always holds) and to apply an IPM solver to the HSD model. In SeDuMi, its implementation is based on the HSD model of (1) and (2). For SDPT3, we have also implemented IPM solvers for the HSD model, and the user has the option to solve an SDP problem via this approach. We should mention that another approach to address the failure of Slater's condition is by facial reduction [5]. But the efficient implementation of the latter approach depends heavily on the specific SDP data at hand, and in general it is expensive to perform the reduction. Simplified facial reduction approaches have recently been proposed in [36]. Finally, detecting whether an SDP problem satisfies Slater's condition is itself an interesting problem that had been studied in [13].

2.1 Deterioration of primal infeasibility and its mitigation

Next, we will describe in more detail how the inaccuracy in solving the SCE will affect the attainable accuracy level of a computed approximate optimal solution of an SDP problem. From [22, Theorem 10.4], one can see that the computed solution Δy of the SCE would satisfy the following condition on the norm of the residual vector $\xi := h - \mathcal{M}\Delta y$:

$$\|\xi\| \leq O(u)\|\mathcal{M}\|_2\|\Delta y\|, \tag{5}$$

where $u = 2.2 \times 10^{-16}$ is the machine epsilon in double-precision computations. Assuming that all the other steps in computing ΔX and ΔZ are done exactly, then for the new primal iterate $X^+ = X + \alpha\Delta X$, where $\alpha \in (0, 1]$ is the step-length, one can show that new primal infeasibility is given by

$$R_p^+ := b - \mathcal{A}(X^+) \; = \; (1-\alpha)R_p + \alpha\xi. \tag{6}$$

Based on (5) and (6), one can expect in the worst case that the primal infeasibility $\|R_p^+\|$ would grow proportionately with $\|\mathcal{M}\|_2 = O(1/\mu)$. Indeed, one can observe the deterioration of the primal infeasibility $\|R_p\|$ in Table 1, which shows the relevant results generated from the last few iterations of SDPT3 when solving the SDP problem `control10.dat-s` in the SDPLIB library [4].

In order to mitigate the drastic loss of accuracy in the primal infeasibility due to the SCE approach for computing Δy, [55] designed the reduced augmented equation (RAE) approach for computing the search direction, wherein the key step is to solve an

Table 1: Performance of the SCE-based IPM in SDPT3 when solving the SDP problem `control10-dat.s`.

iter	$\|R_p\|$	μ	$\|\mathcal{M}\|_2$	$\frac{\|R_p\|}{u\|\mathcal{M}\|_2\|\Delta y\|}$
20	1.18e-09	1.61e-05	2.15e+07	3.04e-02
21	2.95e-08	3.24e-06	3.48e+08	4.45e-02
22	8.29e-07	1.77e-06	3.15e+09	3.94e-02
23	5.85e-07	1.03e-06	3.58e+09	4.18e-02
24	7.39e-07	8.37e-07	5.90e+09	5.18e-02
25	7.19e-07	7.96e-07	7.57e+09	2.42e-02

RAE that is derived from the 2×2 block augmented equation corresponding to the NT symmetrization scheme. To begin with, we know that when $\mu \ll 1$, the eigenvalues of the positive definite NT-scaling matrix W would be partitioned into two groups if the optimal solution (X^*, y^*, Z^*) satisfies the strict complementarity condition. In particular, we have the spectral decomposition $W^{-1} = Q_1 D_1 Q_1^T + Q_2 D_2 Q_2^T$, where $D_1 = \text{diag}(d^{(1)})$ and $D_2 = \text{diag}(d^{(2)})$ are such that $d^{(1)} = \Theta(\sqrt{\mu})$ and $d^{(2)} = \Theta(\frac{1}{\sqrt{\mu}})$. The columns of $Q_1 \in \mathbb{R}^{n\times r}$ and $Q_2 \in \mathbb{R}^{n\times(n-r)}$ are the eigenvectors corresponding to the eigenvalues in $d^{(1)}$ and $d^{(2)}$, respectively. For simplicity, we assume that $\max_{1\le i\le r}\{d_i^{(1)}\} < 1$ and $\min_{1\le i\le n-r}\{d_i^{(2)}\} \ge 1$. The RAE is given by

$$\underbrace{\begin{bmatrix} \mathcal{H} & \widetilde{\mathcal{A}}_1 \\ \widetilde{\mathcal{A}}_1^* & -\widetilde{\Phi} \end{bmatrix}}_{\mathcal{B}} \begin{bmatrix} \Delta y \\ \Xi \circ (Q_1^T \Delta X Q_1) \end{bmatrix} = \text{rhs}, \qquad (7)$$

where $\Xi_{ij} = 1 - d_i^{(1)} d_j^{(1)}$ for all $1 \le i, j \le r$,

$$\begin{aligned} \mathcal{H}\xi &= \mathcal{A}\big(U_1(\mathcal{A}^*\xi)U_1 + U_2(\mathcal{A}^*\xi)U_3 \\ &\quad + U_3(\mathcal{A}^*\xi)U_2 + U_3(\mathcal{A}^*\xi)U_3\big) \ \forall\, \xi \in \mathbb{R}^m, \\ \widetilde{\Phi}(V) &= \Phi \circ V,\ \widetilde{\mathcal{A}}_1(V) = \mathcal{A}(Q_1 V Q_1^T) \ \forall\, V \in \mathbb{S}^r, \end{aligned}$$

with $U_1 = Q_1 Q_1^T$, $U_2 = Q_1 D_1^{-1} Q_1^T = \Theta(\frac{1}{\sqrt{\mu}})$, $U_3 = Q_2 D_2^{-1} Q_2^T = \Theta(\sqrt{\mu})$, $\Phi_{ij} = \frac{d_i^{(1)} d_j^{(1)}}{1 - d_i^{(1)} d_j^{(1)}} = O(\mu)$. Note that the dimension of the coefficient matrix $\mathcal{B}$ is $m+p$ where $p = r(r+1)/2$ is generally at most m.

One can show that under the strict complementarity condition, the spectral norm $\|\mathcal{B}\|_2$ is bounded independently of μ even as $\mu \downarrow 0$. Based on the RAE approach to compute the search direction, the primal infeasibility of the updated X^+ can readily be shown to satisfy the following inequality:

$$\|R_p^+\| \ \le\ (1-\alpha)\|R_p\| + \alpha\|\xi - \widetilde{\mathcal{A}}_1(\eta)\|, \qquad (8)$$

where (ξ, η) is the residual vector associated with the computed solution of the RAE (7). Assuming that (7) is solved stably via an LU factorization, then its norm satisfies the condition that

$$\begin{aligned} &\max\{\|\xi\|, \|\text{vec}(\eta)\|\} \\ &\quad \le O(u)\|\mathcal{B}\|_2 \max\{\|\Delta y\|, \|\text{vec}(Q_1^T \Delta X Q_1)\|\}. \qquad (9) \end{aligned}$$

From the above inequality, one can deduce that in contrast to the SCE approach, as $\mu \downarrow 0$, the new primal infeasibility $\|R_p^+\|$ computed based on the RAE approach would not be amplified before reaching the level permitted by the right-hand-side of (9).

Table 2 shows the results corresponding to Table 1 using the RAE approach to compute the search direction when solving the SDP problem `control10.dat-s`. One can observe that the achievable accuracy of $\|R_p\|$ in the RAE approach is much better than that in the SCE approach.

It turns out that the RAE approach can actually allow one to solve an SDP problem much more accurately than the SCE approach, even for problems that do not satisfy the PDNSC conditions. Indeed in [55, Table 1–5], it has been demonstrated that an IPM solver that uses the RAE approach to compute the search direction in each iteration can solve the tested SDP problems to a very high accuracy level of less than 10^{-12} in the relative KKT residual.

Of course, the drawback of using the RAE approach is that not only is the computation of $\mathcal{B}$ more costly (fortunately, only 2-3 times more in general) than that of $\mathcal{M}$, the cost of computing the LDL^T factorization of $\mathcal{B}$ can be up to 8 times slower (assuming that $p \le m$) than the Cholesky factorization of $\mathcal{M}$. One could, of course, try to minimize the additional cost by employing a hybrid approach of using the SCE approach when μ is not too small and then switch to the RAE approach when $\mu \downarrow 0$. In this way, one could expect to be able to solve an

Table 2: Performance of the RAE-based IPM in SDPT3 when solving the SDP problem `control10.dat-s`.

iter	$\|R_p\|$	μ	$\|\mathcal{B}\|_2$	$\frac{\|R_p\|}{u\|\mathcal{B}\|_2\|[\Delta y; \mathrm{vec}(Q_1^T \Delta X Q_1)]\|}$
20	6.84e-13	1.61e-05	3.67e+02	1.18e-01
21	9.00e-13	3.25e-06	4.27e+02	5.94e-01
22	1.67e-12	1.41e-06	5.41e+02	5.87e-01
23	1.26e-12	2.62e-07	4.29e+02	1.17e+00
24	2.65e-12	8.97e-08	7.30e+02	1.48e+00
25	2.03e-12	1.76e-08	1.28e+03	1.74e+00

SDP problem to a high level of accuracy but at a modest expense of higher computing time.

2.2 Handling dense columns in linear and second-order cone programming

While the SC matrix $\mathcal{M}$ for an SDP problem is typically dense, its counterpart in the case of a linear programming problem is generally sparse when the constraint matrix (with $\mathcal{A}$ now being a matrix) is sparse. In the event when $\mathcal{A}$ has a few dense columns, the matrix $\mathcal{M}$ would have the structure of a sparse matrix plus a low-rank dense matrix. The same structure for $\mathcal{M}$ also appears frequently when one solves a second-order cone programming (SOCP) problem, even when $\mathcal{A}$ has no dense columns.

It is important to pay special attention to handling the low-rank dense part in $\mathcal{M}$ separately, so that the SCE can be solved efficiently in terms of both the computation time and memory consumption. An efficient approach is to apply the Sherman-Morrison-Woodbury (SMW) inverse formula to handle the dense part based on the Cholesky factorization of the sparse part. While this approach is efficient, unfortunately it is not numerically stable. In SDPT3, we again use the computed inverse (based on the SMW inverse formula) as the preconditioner in the PSQMR solver used to solve the SCE. We find that such an approach is more stable than the iterative refinement approach that is typically employed to improve the accuracy of the computed solution based on the SMW inverse formula.

An alternative to the SMW approach for handling the low-rank dense part in $\mathcal{M}$ is to use the product-form Cholesky factorization. It has been shown in [17, 18] that the approach is roughly as stable as the original Cholesky factorization for $\mathcal{M}$ and theoretically nearly as efficient as the SMW approach. Unfortunately, the efficient implementation of the product-form Cholesky factorization to take advantage of the cache memory hierarchy of a CPU by optimized BLAS is non-trivial. As far as I am aware, SeDuMi is the only solver that has implemented the product-form Cholesky factorization to handle the sparse-plus-dense-low-rank structure of $\mathcal{M}$.

Again, the SCE approach for solving an SOCP problem generally limits the level of accuracy that is achievable because of the extreme ill-conditioning of $\mathcal{M}$. The matter is made worse if the SMW inverse formula is employed to handle the sparse-plus-dense-low-rank structure in $\mathcal{M}$. For example, SDPT3 and SeDuMi can only solve the SOCP problem `sched_100_100_org` in the DIMACS depository [35] to an accuracy of about 10^{-3} and 5×10^{-4}, respectively.

It turns out that for SOCP problems, the RAE approach analogous to that in (7) can also successfully overcome the loss of accuracy [7]. Moreover, it can effectively handle any sparse-plus-dense-low-rank structure present in the $(1, 1)$ block of the coefficient matrix $\mathcal{B}$. For the SOCP problem just mentioned, an IPM solver that employs the RAE approach to compute the search direction in the last few iterations can obtain a solution with the accuracy of 8×10^{-10}, but at the expense of higher computation time.

2.3 Solving large-scale SDPs via an inexact IPM

The RAE approach for computing the search direction not only can mitigate the deterioration of the primal infeasibility, it also has an important property that, under the PDNSC conditions, $\|\mathcal{B}^{-1}\|_2$ can be proven to be bounded independently of μ as $\mu \downarrow 0$; see [55, Theorem 2.3]. As a result, the condition number of $\mathcal{B}$ can be bounded independently of μ as $\mu \downarrow 0$. This implies that one can use an iterative solver such as the PSQMR method to solve the RAE and expect to get a reasonable convergence rate even when $\mu \downarrow 0$. Indeed, in [55], it was shown that one could use an inexact IPM to solve a large scale SDP (with more than 127,000 constraints and matrix variable dimension 800) via the RAE approach in about 6.5 hours on a modest desktop PC in 2004. Today, solving such a large SDP on a desktop PC is still beyond the reach of an IPM solver that is based on the SCE approach.

Despite the favorable property of bounded condition numbers (for $\mathcal{B}$) under the PDNSC conditions, solving a large-scale SDP problem by an inexact IPM that uses an iterative solver to solve the RAE in each iteration is usually still not efficient enough. Faced with the difficulty just mentioned, it is natural for one to explore alternative computational frameworks to solve large SDP problems. Fortunately, we were able to discover that by designing an appropriate semismooth Newton method to solve the subproblem arising in each iteration of an augmented Lagrangian method (ALM) applied to the dual SDP (2), one can solve a primal and dual nondegenerate SDP problems highly efficiently. As will be discussed later, our semismooth Newton-based ALM can fully exploit the rank-sparsity structure of the matrix variables to drastically cut down the computational cost of the CG method used to solve the semismooth Newton equations in each iteration of the ALM.

3 Augmented Lagrangian-based methods

The first time I came across a non-barrier-based method to solve the SDP problem (1) was at a workshop held at the Institute of Mathematical Sciences, National University of Singapore, in January 2006. In his tutorial lectures, Franz Rendl [39] reported extremely impressive numerical results obtained by the so-called boundary-point method [37] for solving sparse random SDP problems as well as SDP relaxations of maximum stable set problems. In hindsight, the boundary-point method implemented in [37] is exactly the same as the classical ADMM (with unit step-length) applied to the dual SDP problem (2).

Inspired by the impressive performance of the boundary-point method and also the impressive performance achieved by Houduo Qi and Defeng Sun [40] in employing the semismooth Newton-CG (SNCG) method to solve the nearest correlation matrix problem (a semidefinite least-square problem), I convinced Defeng at that time to work together with my PhD student Xinyuan Zhao on designing an augmented Lagrangian method (ALM) to solve the dual SDP problem (2) wherein the subproblem at each iteration is solved by an SNCG method (subsequently we named the algorithm SDPNAL). Although we started to work on SDPNAL in early 2006, the numerical design of the algorithm and its efficient implementations, together with extensive numerical evaluation of its performance, took almost two years to complete and the resulting paper [63] only appeared in print in 2010.

In SDPNAL, we used the boundary-point method to generate a reasonably good initial point to warm-start the ALM. Our work has subsequently inspired Wen et al. [57] to implement an ADMM solver (called SDPAD) for the dual SDP (2). As mentioned in the acknowledgments in [57], Wen et al. adapted quite a number of implementation ideas and subroutines in SDPNAL codes to SDPAD.

As a follow-up to [37], one of the objectives in [29] is to provide a theoretical foundation for the boundary-point method. In fact, the regularization methods in [29] bear close resemblance to the proximal-point algorithm (PPA) in [46] applied to (1) and the augmented Lagrangian method (ALM) applied to (2) in [47] where their convergence analyses have been well developed. For a beautiful analysis on the fast linear convergence of the ALM for SDP and its connections to an approximate semismooth Newton method, we refer the reader to the paper by Defeng Sun, Jie Sun, and Liwei Zhang [43].

Unbeknown to the authors of [37, 57] including our own work in [63], the classical ADMM (for 2 blocks of variables) in fact has a rather well-developed convergence analysis already done by Glowinski [16] and Fortin and Glowinski [12] for any step-length in $(0, (1 + \sqrt{5})/2)$. There are various generalizations of the classical ADMM for a 2-block problem. As far as I am aware, the most versatile 2-block ADMM with a very general convergence analysis is the semi-proximal ADMM (SPADMM) published in Appendix B of [11]. In fact, the SPADMM in [11] forms the basis for some of our subsequent work on designing ADMM-type methods for solving convex composite conic programming problems, such as [45, 27, 9].

For the dual SDP (2), the classical ADMM appears to be the most suitable first-order method for solving the problem. But it is no longer adequate for solving a more complicated SDP such as the following doubly nonnegative (DNN) SDP:

$$\min \Big\{ \langle C,\, X\rangle \mid \mathcal{A}(X) = b, X \in \mathbb{S}^n_+, X \in \mathcal{N}^n \Big\} \tag{10}$$

whose dual is given by

$$\max \Big\{ \langle b,\, y\rangle \mid \mathcal{A}^* y + Z + S = C,\, Z \in \mathbb{S}^n_+, S \in \mathcal{N}^n \Big\}, \tag{11}$$

where $\mathcal{N}^n = \{X \in \mathbb{S}^n \mid X_{ij} \geq 0 \;\forall\; (i,j)\}$. We should mention that DNNSDPs form an important class of computationally tractable relaxations for completely positive programming (CPP) problems that have become very popular in recent years. The rise in the popularity of CPP can be attributed to the important work done by Burer [6] who showed that CPP can give an equivalent reformulation (under mild assumptions) of a nonconvex quadratic optimization problem. For more recent developments, we refer the reader to [23] and the references therein.

One can, of course, directly extend the classical ADMM to solve the dual DNNSDP which has the natural structure of having 3 blocks of variables in different cones $(\mathbb{R}^m, \mathbb{S}^n_+, \mathcal{N}^n)$. But the drawback is that the directly extended ADMM is no longer guaranteed to be convergent [8], although in practice it usually converges despite the lack of a theoretical guarantee. In fact, based on our numerical experience in [45], the practical performance of the directly extended ADMM (DE-ADMM) is surprisingly good compared to various convergent variants of multi-block (for 3 or more blocks) ADMM that have been proposed to overcome the lack of convergence guarantee of the former. To address the apparent paradox, [45] proposed a convergent multi-block ADMM (called sPADMM3c) for solving a conic programming problem with 4 types of constraints that includes the dual DNNSDP (11) as a special case. The numerical experiments in [45] have demonstrated that our convergent method is at least 20% faster than DE-ADMM for the vast majority of about 550 large-scale DNNSDPs tested.

In order to solve even more complex conic programming problems such as convex quadratic SDP problems with 4 types of constraints, [27] proposed the convergent multi-block ADMM (SCB-SPADMM) which, again, has been demonstrated to clearly outperform the DE-ADMM in extensive numerical experiments. In order to be able to cope with a problem with a huge number of linear constraints and other numerical challenges, [9] extended the SCB-SPADMM to an inexact setting that has the flexibility of allowing the linear system of equations in each iteration to be solved approximately by an iterative solver such as the preconditioned conjugate gradient method. The convergence of the SCB-SPADMM in [27] has relied on the key idea of sequentially eliminating blocks of variables (as in the Schur-complement approach for solving block linear system of equations) when solving the multi-block minimization subproblem in each iteration. Recently it has been shown that the SCB approach in solving the SCB-SPADMM minimization subproblem in each iteration is in fact equivalent to applying one cycle of a block symmetric Gauss-Seidel iteration for solving a convex composite quadratic programming problem [26].

Despite the fairly good performance of ADMM-based methods that we have designed and implemented in solving conic programming problems as demonstrated in [45, 27, 9], they are sometimes not efficient enough, or even stagnate, when solving difficult problems such as the DNNSDPs arising from the relaxation of quadratic assignment problems [38] as demonstrated in [61]. As shown in the numerical experiments in [61], it is crucial to use second-order information in the design of a robust and efficient method for solving difficult DNNSDPs. The method designed in [61] is a majorized SNCG augmented Lagrangian method (called SDPNAL+) that is designed for solving a linear SDP problem with additional bound constraints such as those in a DNNSDP. It is built upon the foundation established in SDPNAL [63] for solving the SDPs (1) and (2). For the work done in [61], the authors were awarded the 2018 Beale–Orchard-Hays Prize by the Mathematical Optimization Society.

As guaranteed by the convergence theory of SNCG and ALM, SDPNAL and SDPNAL+ are expected to work very efficiently for nondegenerate SDPs. But they may encounter numerical difficulties for degenerate ones. In SDPNAL+, we attempt to solve degenerate problems by a hybrid strategy of employing a majorized SNCG ALM together with a convergent multi-block ADMM introduced in [45, 27, 9]. Numerical results for various large-scale SDPs with or without nonnegativity constraints show that SDPNAL+ is not only fast but also robust in obtaining fairly accurate solutions. For example, SDPNAL+ is able to solve an SDP problem arising from the DNN relaxation of a quadratic assignment problem (`tai40b`) in less than 1.5 hours on a desktop PC. In contrast, the same problem is solved by the parallel interior-point solver SDPARA in about 122 hours on a parallel computing platform with 256 multi-core CPUs and 128 GPUs [14]. The largest instance that SDPNAL+ has solved currently is an SDP relaxation problem (with $n = 9,261$ and $m = 12,326,390$) arising from a rank-one tensor approximation problem [33], and the instance is solved in about 7 hours on a desktop PC with an Intel Xeon CPU (E5-2680v3 @2.50 GHz with 12 cores) and 128GB of RAM.

Next, we discuss the key computational step in each iteration of SDPNAL. We start with the augmented Lagrangian function for (2):

$$\begin{aligned}\mathcal{L}_\sigma(y,Z;X) &= -\langle b,\, y\rangle + \delta_{\mathbb{S}^n_+}(Z)\\ &\quad + \frac{\sigma}{2}\|\mathcal{A}^*y + Z - C + \sigma^{-1}X\|^2 - \frac{1}{2\sigma}\|X\|^2, \qquad (12)\end{aligned}$$

where $\sigma > 0$ is a given penalty parameter. At the kth ALM iteration, one needs to solve the following minimization subproblem for a given X^k:

$$\begin{aligned}(y^{k+1}, Z^{k+1}) &\approx \min_{y,Z}\{\mathcal{L}_{\sigma_k}(y,Z;X^k)\}\\ &\equiv \min_y \Big\{-\langle b,\, y\rangle + \min_{Z\in\mathbb{S}^n_+}\big\{\frac{\sigma}{2}\|\mathcal{A}^*y + Z - C + \sigma^{-1}X^k\|^2\big\}\Big\}.\end{aligned}$$

Let $\Pi_+(M)$ denote the projection of a given matrix $M \in \mathbb{S}^n$ onto $\mathbb{S}^n_+$. One can compute the solution (y^{k+1}, Z^{k+1}) by first solving

$$\begin{aligned}y^{k+1} \approx \min_y \big\{\phi(y) &:= -\langle b,\, y\rangle\\ &+ \frac{\sigma}{2}\|\Pi_+(\mathcal{A}^*y - C + \sigma^{-1}X^k)\|^2\big\} \qquad (13)\end{aligned}$$

and then updating $Z^{k+1} = \Pi_+(C - \mathcal{A}^*y^{k+1} - \sigma^{-1}X^k)$. To compute y^{k+1}, one would need to solve the following nonsmooth optimality condition for the problem (13):

$$0 = \nabla\phi(y) = \sigma\mathcal{A}\Pi_+(\mathcal{A}^*y - C + \sigma^{-1}X^k) - b. \qquad (14)$$

In the fundamental paper [42], the matrix-valued function $\Pi_+(\cdot)$ has been shown to be strongly semismooth. Thus, one can apply the semismooth Newton method [41] to solve the above equation and expect to achieve quadratic convergence when (1) is nondegenerate at the converged optimal solution. More precisely, primal constraint nondegeneracy plays an important role in ensuring that the generalized Jacobians of $\nabla\phi(\cdot)$ are nonsingular so that quadratic convergence of the semismooth Newton method can be derived. For more details on constraint nondegeneracy and its characterization for SDP, we refer the reader to the excellent work of Defeng Sun in [44].

At a given point $\bar{y}$, the semismooth Newton equation (SNE) is given by

$$\underbrace{\mathcal{A}\mathcal{J}\mathcal{A}^*}_{\mathcal{H}}(\Delta\bar{y}) \;=\; -\frac{1}{\sigma}\nabla\phi(\bar{y}),$$

where $\mathcal{J} : \mathbb{S}^n \to \mathbb{S}^n$ is a generalized Jacobian of Π_+ at $\mathcal{A}^*\bar{y} - C + \sigma^{-1}X^k$. Suppose that the latter has spectral decomposition, $Q\mathrm{diag}(d)Q^T$, where the eigenvalues are arranged such that $d_1 \geq \cdots d_r \geq 0 > d_{r+1} \geq \cdots \geq d_n$; then one can pick

$$\mathcal{J}(M) = Q(\Omega \circ (Q^T M Q))Q^T,\ \Omega_{ij} = \begin{cases} 1 & \text{if } 1 \le i,j \le r \\ \frac{d_i}{d_i + |d_j|} & \text{if } 1 \le i \le r,\ r+1 \le j \le n \\ \frac{d_j}{|d_i| + d_j} & \text{if } r+1 \le i \le n,\ 1 \le j \le r \\ 0 & r+1 \le i,j \le n. \end{cases}$$

Let $Q_1 \in \mathbb{R}^{n\times r}$ and $Q_2 \in \mathbb{R}^{n\times(n-r)}$ be the matrices formed by the first r and last $n-r$ columns of Q, respectively. Under the primal constraint nondegeneracy condition at the optimal solution, one can show that $\mathcal{H}$ is positive definite, and

$$\operatorname{cond}(\mathcal{H}) \;\le\; \sigma\,\Theta(1)\operatorname{cond}([\widetilde{\mathcal{A}}_1, \widetilde{\mathcal{A}}_2])^2, \qquad (15)$$

where $\widetilde{\mathcal{A}}_1 : \mathbb{S}^r \to \mathbb{R}^m$ is defined by $\widetilde{\mathcal{A}}_1(V) = \mathcal{A}(Q_1 V Q_1^T)$, and $\widetilde{\mathcal{A}}_2 : \mathbb{R}^{r\times(n-r)} \to \mathbb{R}^m$ is defined by $\widetilde{\mathcal{A}}_2(V) = \mathcal{A}(Q_1 V Q_2^T + Q_2 V^T Q_1^T)$. Note that under the primal constraint nondegeneracy condition at the optimal solution, $\operatorname{cond}([\widetilde{\mathcal{A}}_1, \widetilde{\mathcal{A}}_2])$ remains bounded even as the iterate (X^k, y^k, Z^k) converges to an optimal solution.

In contrast, for an IPM solver where $\mathcal{M} = \mathcal{A}W \otimes W\mathcal{A}^*$, we have

$$\operatorname{cond}(\mathcal{M}) \;\le\; \frac{\|X\|^2}{\mu}\Theta(1)\operatorname{cond}([\widetilde{\mathcal{A}}_1, \widetilde{\mathcal{A}}_2])^2, \qquad (16)$$

where $\widetilde{\mathcal{A}}_1$ and $\widetilde{\mathcal{A}}_2$ are the analogues of those in (15) but defined based on the spectral decomposition of W. It is clear that even if the SDP is primal nondegenerate at the optimal solution so that $\operatorname{cond}([\widetilde{\mathcal{A}}_1, \widetilde{\mathcal{A}}_2])$ is bounded independently of μ, the condition number of $\mathcal{M}$ would still grow to infinity as $\mu \downarrow 0$. It is precisely the unavoidable asymptotic singularity of $\mathcal{M}$ that makes solving the SCE by an iterative method prohibitively expensive when such a technique is needed for large-scale SDPs. On the other hand, the SNE does not suffer from such a fatal defect when the SDP is primal nondegenerate at the optimal solution and thus one can solve the linear system by an iterative method at a reasonable cost. From this comparison, one can see why the solver SDPNAL is needed to solve large-scale SDP problems [63].

Besides the difference in the conditioning of the SCE and SNE, the SNE also has another tremendous advantage over the SCE in that it can fully exploit any low-rank or high-rank structure (which we call rank-sparsity for convenience) present the current iterate. Specially, if the rank of $\sigma^{-1}X^k - (C - \mathcal{A}^*\bar{y})$ is r, then the cost of computing a matrix-vector product for the SNE can be shown to be given by

$$\begin{aligned}\operatorname{cost}(\mathcal{H}\Delta y) &= 8\min\{r, n-r\}n^2 \\ &\quad + \operatorname{cost}(\mathcal{A}(\cdot)) + \operatorname{cost}(\mathcal{A}^*(\cdot)).\end{aligned}$$

In contrast, the corresponding cost of a matrix-vector product for the SCE is given by:

$$\operatorname{cost}(\mathcal{M}\Delta y) = 4n^3 + \operatorname{cost}(\mathcal{A}(\cdot)) + \operatorname{cost}(\mathcal{A}^*(\cdot)).$$

Clearly, if $r \ll n$ or $n - r \ll n$, then the savings from exploiting the rank-sparsity structure present in the generalized Jacobian in the SNE can be very substantial. Roughly speaking, one can observe that the rank-sparsity structure of the generalized Jacobian is inherited from the rank-sparsity structure of the iterate (X^k, Z^k) and SNE is able to exploit the rank-sparsity structure in the iterate, but the SCE cannot do so because the IPM framework has perturbed all zero eigenvalues to small positive numbers. For convenience, in Table 3, we summarize the key computational differences of solving a large-scale SDP problems via an IPM solver versus the SDPNAL solver.

4 Conclusion

We have discussed various numerical issues that are critical to the design of robust and efficient solvers for SDP problems. In particular, we note that well-implemented IPM solvers can be surprisingly robust for solving ill-posed (degenerate) SDP problems, even though they are efficient only for solving medium-scale problems. While augmented Lagrangian-based solvers such as SDPNAL and SDPNAL+ have achieved good success in solving some large-scale SDP problems, their performance

Table 3: Contrast in the conditioning and second-order sparsity of the SCE in SDPT3 and that of the SNE in SDPNAL. In the table, $\mathcal{D} : \mathbb{S}^n \to \mathbb{S}^n$ is defined by $\mathcal{D}(M) = \Omega \circ M$.

$m \times m$ linear system in IPM	$m \times m$ linear system in ALM
$\mathcal{A}(W \otimes W)\mathcal{A}^* \Delta y = \text{rhs}$	$\mathcal{A}(Q \otimes Q)\mathcal{D}(Q^T \otimes Q^T)\mathcal{A}^* \Delta y = \text{rhs}$
Condition number at least $\Theta(1/\mu)$ unbounded as barrier parameter $\mu \downarrow 0$	Condition number = $\Theta(\sigma)$ stays bounded for nondegenerate problems, where σ is the penalty parameter
CG needs a large number of steps	CG needs a moderate number of steps
Can exploit data structure, especially sparsity	Can exploit data structure such as sparsity
Unable to exploit low-rank or high-rank structure in X	Can exploit low-rank or high-rank structure in X, because of the all-ones and zero blocks in Ω

is expected to be robust mainly for nondegenerate problems, and they may not be able to solve degenerate problems to a reasonable level of accuracy. Indeed, how to design an algorithm that is robust and efficient for solving large-scale degenerate SDP problems is a key challenge in SDP research, and any progress to resolve this challenge will contribute to the widespread application of SDP modeling in business, economics, science, and engineering.

Another challenge in SDP research is in designing an algorithm for solving large SDP problems when both n and m are large. Parallel algorithms that extend the current IPM- or ALM-based solvers would be the natural path to take, although one should not underestimate the numerical and implementation complexities that would arise in the parallel algorithms, especially when the dense matrix variable must be stored in a distributed fashion. For an SDP problem with large matrix dimension n but with conducive sparsity structure, one can apply the conversion method that has been well-studied in [32] to convert the problem into an equivalent problem having multiple blocks of smaller matrix variables, where the dimension of each block is moderate (of course, the dimensions of the smaller blocks are very much dependent on the SDP data at hand). However, one should note that in the conversion method, usually a large number of equality constraints will be introduced to account for the fact that matrix elements in the overlapping part of two different blocks must be the same. Another issue that is not often mentioned is that the converted problem is likely to be degenerate even if the original problem is nondegenerate. Thus, solving the converted problem is also not without challenges.

To conclude, I hope this article has convinced the reader that there is interesting and challenging work to be done in designing algorithms for solving large-scale SDP problems. Of course, the same questions can be asked for more general conic programming problems such as convex quadratic SDPs, log-determinant SDPs, etc.

REFERENCES

[1] F. Alizadeh, J.-P. Haeberly, and M.L. Overton, *Primal-dual interior-point methods for semidefinite programming: convergence results, stability and numerical results*, SIAM J. Optimization, 8 (1998), pp. 746–768.

[2] S.J. Benson, Y. Ye, *DSDP—software for semidefinite programming*, ACM Transactions on Mathematical Software, 34 (2008), Article 16.

[3] B. Borchers, *CSDP, A C library for semidefinite programming*, Optimization Methods and Software, 11 (1999), pp. 613–623.

[4] B. Borchers, *SDPLIB 1.2, A library of semidefinite programming test problems*, Optimization Methods and Software, 11 (1999), pp. 683–690.

[5] J. Borwein and H. Wolkowicz, *Regularizing the abstract convex program*, J. Mathematical Analysis and Applications, 83 (1981), pp. 495–530.

[6] S. Burer, *On the copositive representation of binary and continuous non- convex quadratic programs*, Math. Program., 120 (2009), pp. 479–495.

[7] Z. Cai and K.C. Toh, *Solving second order cone programming via the augmented systems*, SIAM J. Optimization, 17 (2006), pp. 711–737.

[8] C.H. Chen, B.S. He, Y. Ye, and X.M. Yuan, *The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent*, Mathematical Programming, 155 (2016), pp. 57–79.

[9] L. Chen, D.F. Sun, and K.C. Toh, *An efficient inexact symmetric Gauss-Seidel based majorized ADMM for high-dimensional convex composite conic programming*, Mathematical Programming, 161 (2017), pp. 237–270.

[10] CVX Research Inc., *CVX: Matlab software for disciplined convex programming, version 2.0*, available at `http://cvxr.com/cvx`.

[11] M. Fazel, T.K. Pong, D.F. Sun, and P. Tseng, *Hankel matrix rank minimization with applications in system identification and realization*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 946–977.

[12] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: Applications to the numerical solution of boundary value problems*, Vol. 15 of Studies in Mathematics and its Applications, North-Holland Publishing Co., Amsterdam, 1983.

[13] R.M. Freund, F. Ordonez, and K.C. Toh, *Behavioral measures and their correlation with IPM iteration counts on semi-definite programming problems*, Mathematical Programming, 109 (2007), pp. 445–475.

[14] K. Fujisawa, *High-performance and power-efficient computing for large-scale optimization problem*, invited talk delivered at the Fifth International Conference on Continuous Optimization, 2016, Tokyo.

[15] K. Fujisawa, M. Kojima, and K. Nakata, *Exploiting sparsity in primal-dual interior-point method for semidefinite programming*, Mathematical Programming, 79 (1997), pp. 235–253.

[16] R. Glowinski, Lectures on numerical methods for nonlinear variational problems, vol. 65 of Tata Institute of Fundamental Research Lectures on Mathematics and Physics, Bombay, 1980. Notes by M. G. Vijayasundaram and M. Adimurthi.

[17] D. Goldfarb and K. Scheinberg, *Product-form Cholesky factorization for handling dense columns in interior-point methods for linear programming*, Mathematical Programming, 99 (2004), pp. 1–34.

[18] D. Goldfarb and K. Scheinberg, *Product-form Cholesky factorization in interior-point methods for second-order cone programming*, Mathematical Programming, 103 (2005), pp. 153–179.

[19] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pp. 95–110, Lecture Notes in Control and Information Sciences, Springer, 2008.

[20] M. Halicka, E. de Klerk, and C. Roos, *On the convergence of the central path in semidefinite optimization*, SIAM J. Optimization, 12 (2002), pp. 1090-1099.

[21] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz, *An interior-point method for semidefinite programming*, SIAM J. Optimization, 6 (1996), pp. 342–361.

[22] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM 1996.

[23] S.Y. Kim, M. Kojima, and K.C. Toh, *A Lagrangian-DNN relaxation: a fast method for computing tight lower bounds for a class of quadratic optimization problems*, Mathematical Programming, 156 (2016), pp. 161–187.

[24] M. Kocvara and M. Stingl, *PENNON – a code for convex nonlinear and semidefinite programming*, Optimization Methods and Software, 8 (2003), pp. 317–333.

[25] M. Kojima, S. Shindoh, and S. Hara, *Interior-point methods for the monotone linear complementarity problem in symmetric matrices*, SIAM J. Optimization, 7 (1997), pp. 86–125.

[26] X.D. Li, D.F. Sun and K.C. Toh, *A block symmetric Gauss-Seidel decomposition theorem for convex composite quadratic programming and its applications*, Mathematical Programming, in print. arXiv:1703.06629.

[27] X.D. Li, D.F. Sun and K.C. Toh, *A Schur complement based semi-proximal ADMM for convex quadratic conic programming and extensions*, Mathematical Programming, 155 (2016), pp. 333–373.

[28] J. Lofberg, *YALMIP: A Toolbox for Modeling and Optimization in MATLAB*, in Proceedings of the CACSD Conference, Taipei, 2004.

[29] J. Malick, J. Povh, F. Rendl and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM J. Optimization, 20 (2009), pp. 336–356.

[30] H. Mittelmann, *Benchmarks for optimization software*, available at `http://plato.asu.edu/ftp/sparse_sdp.html`

[31] R.D.C. Monteiro, *Primal-dual path-following algorithms for semidefinite programming*, SIAM J. Optimization, 7 (1997), pp. 663–678.

[32] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota, *Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results*, Mathematical Programming, 95 (2003), pp. 303–327.

[33] J. Nie and L. Wang, *Semidefinite relaxations for best rank-1 tensor approximations*, SIAM J. Matrix Analysis and Applications, 35 (2014), pp. 1155–1179.

[34] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Mathematics of Operations Research, 23 (1998), pp. 339–358.

[35] G. Pataki and S.H. Schmieta, *The DIMACS library of mixed semidefinite-quadratic-linear programs*, `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`

[36] F. Permenter, H.A. Friberg, and E.D. Andersen, *Solving conic optimization problems via self-dual embedding and facial reduction: A unified approach*, SIAM J. Optimization, 27 (2017), pp. 1257–1282.

[37] J. Povh, F. Rendl, and A. Wiegele, *A boundary point method to solve semidefinite programs*, Computing, 78 (2006), pp. 277–286.

[38] J. Povh and F. Rendl, *Copositive and semidefinite relaxations of the quadratic assignment problem*, Discrete Optimization, 6 (2009), pp. 231–241.

[39] Franz Rendl, Solving large semidefinite programs, tutorial lectures available at `http://www.ims.nus.edu.sg/Programs/semidefinite/abstracts.htm#frendl3`

[40] H.D. Qi and D.F. Sun, *A quadratically convergent Newton method for computing the nearest correlation matrix*, SIAM J. Matrix Analysis and Applications, 28 (2006), pp. 360–385.

[41] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Mathematical Programming, 58 (1993), pp. 353–367.

[42] D.F. Sun and J. Sun, *Semismooth matrix-valued functions*, Mathematics of Operations Research, 27 (2002), pp. 150–169.

[43] D.F. Sun, J. Sun, and L.W. Zhang, *The rate of convergence of the augmented Lagrangian method for nonlinear semidefinite programming*, Mathematical Programming, 114 (2008), pp: 349–391.

[44] D.F. Sun, *The strong second order sufficient condition and constraint nondegeneracy in nonlinear semidefinite programming and their implications*, Mathematics of Operations Research, 31 (2006), pp: 761–776.

[45] D.F. Sun, K.C. Toh and L.Q. Yang, *A convergent 3-block semi-proximal alternating direction method of multipliers for conic programming with 4-type constraints*, SIAM J. Optimization, 25 (2015), pp. 882–915.

[46] R.T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control and Optimization, 14 (1976), pp. 877–898.

[47] R.T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Mathematics of Operations Research, 1 (1976), pp. 97–116.

[48] J.F. Sturm, *Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11 (1999), pp. 625–653.

[49] M.J. Todd, *Semidefinite optimization*, Acta Numerica, (10) 2001, pp. 515–560.

[50] K.C. Toh and L.N. Trefethen, *The Chebyshev polynomials of a matrix*, SIAM J. Matrix Analysis and Applications, 20 (1998), pp. 400–419.

[51] M.J. Todd, K.C. Toh, and R.H. Tutuncu, *On the Nesterov-Todd direction in semidefinite programming*, SIAM J. of Optimization, 8 (1998), pp. 769–796.

[52] K.C. Toh, M.J. Todd, and R.H. Tutuncu, *SDPT3 — a Matlab software package for semidefinite programming*, Optimization Methods and Software, 11 (1999), pp. 545–581.

[53] K.C. Toh, M.J. Todd, and R.H. Tutuncu, *On the implementation and usage of SDPT3 – a Matlab software package for semidefinite-quadratic-linear programming, version 4.0*, in Handbook on semidefinite, cone and polynomial optimization: theory, algorithms, software and applications, M. Anjos and J.B. Lasserre eds., Springer, 2012, pp. 715–754.

[54] R.H. Tutuncu, K.C. Toh, and M.J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming, 95 (2003), pp. 189–217.

[55] K. C. Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optimization, 14 (2004), pp. 670–698.

[56] K.C. Toh, and M. Kojima, *Solving some large scale semidefinite programs via the conjugate residual method*, SIAM J. Optimization, 12 (2002), pp. 669–691.

[57] Z. Wen, D. Goldfarb and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Mathematical Programming Computation, 2 (2010), pp. 203–230.

[58] M. Yamashita, K. Fujisawa, and M. Kojima, *Implementation and evaluation of SDPA 6.0*, Optimization Methods and Software, 18 (2003), pp. 491–505.

[59] M. Yamashita, K. Fujisawa, M. Fukuda, K. Nakata, M. Nakata, *Algorithm 925: Parallel solver for semidefinite programming problem having sparse Schur complement matrix*, ACM Transactions on Mathematical Software 39 (2012), Article 6.

[60] M. Yamashita, K. Fujisawa, and M. Kojima, *SDPARA : SemiDefinite Programming Algorithm paRAllel version*, Parallel Computing, 29 (2003), pp. 1053–1067.

[61] L.Q. Yang, D.F. Sun, and K.C. Toh, SDPNAL+: *a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Mathematical Programming Computation, 7 (2015), pp. 331–366.

[62] Y. Ye, M.J. Todd, and S. Mizuno, *An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm*, Mathematics of Operations Research, 19 (1994), pp. 53–67.

[63] X.Y. Zhao, D.F. Sun, and K. C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J. Optimization, 20 (2010), pp. 1737–1765.

Correction In the article “Sparsity Matters” by Robert J. Vanderbei published in volume 1 of the 2018 newsletter, reference [7] should read “I.J. Lustig, R. Marsten, and D.F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications* 152:191–222, 1991.” We apologize for the error.

Seeking a Host for the 2020 INFORMS Optimization Society Conference

The INFORMS Optimization Society Conference is held in the early part of the even years, often in a warm, or otherwise attractive, location. It offers an opportunity for researchers studying optimization-related topics to share their work in a focused venue. The most recent conference, held earlier this year in Denver, Colorado, certainly has met all of these criteria. (See the short report on the conference by its Chair, Alexandra Newman, in the previous volume of the newsletter.)

The Optimization Society is currently seeking candidate locations to host the 2020 conference. If you are interested in helping to host the conference, please contact the current Optimization Society Chair, David Morton (david.morton@northwestern.edu), or the Chair-elect Dan Bienstock (dano@ieor.columbia.edu).