informs. OPTIMIZATION

INFORMS OS Today

The Newsletter of the INFORMS Optimization Society

Volume 8 Number 1 May 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 INFORMS Optimization Society's Conference is held every two years, and in March of this year the University of Colorado Denver hosted the conference. The number of attendees at the conferences in 2014, 2016, and 2018 were 125, 210, and 272, respectively. This speaks both to the recent growth of research in optimization and to the outstanding work done by the conference organizers Steve Billups, Steffen Borgwardt, Manuel Laguna, and Alexandra Newman.

The conference in Denver featured 14 sessions in *Optimization under Uncertainty* and 12 sessions in *Integer and Discrete Optimization*. That these two clusters were among the most popular is consistent with what we have seen in recent years. The conference also hosted 13 sessions involving *Optimization in Machine Learning and Statistics*, and this is clearly an area of significant growth, both for IOS and more broadly.

At the conference in Denver, Pietro Belotti received the IOS Distinguished Service Award. Pietro served as the IOS Web Editor for eight years, from 2010 through 2017. His remarkable dedication to IOS included handling the recent transition in the content management system to the new INFORMS

Connect design. This was no small effort. In addition to providing Web content, Pietro helped IOS have a strong presence on social media. On behalf of all of IOS, thank you Pietro. IOS is pleased to welcome Sertalp Çay as our new Web Editor.

At the Annual INFORMS Conference in Houston, in fall 2017, IOS awarded four prizes. This newsletter features articles by winners of two of the prizes:

- Robert J. Vanderbei received the Khachiyan Prize for outstanding lifetime contributions to the field of optimization. Bob has made important contributions to interior point methods for linear programming, convex and nonconvex nonlinear optimization, and semidefinite programming. Bob has worked on a fascinating array of applications including designing a NASA space telescope. In this newsletter Bob writes about sparsity and its role in optimization, a theme that runs through important parts of his work and through optimization more broadly.
- Alberto Del Pia and Aida Khajavirad received the Prize for Young Researchers for their paper, "On Decomposability of Multilinear Sets," *Mathematical Programming*, `doi.org/10.1007/s10107-017-1158-z`. Their theoretical work illuminates the facial structure of Padberg's Boolean quadric polytope and more general multilinear polytopes. Their associated algorithmic work should lend itself to improved performance for mixed-integer nonlinear programming solvers.

We thank the respective prize committees for their work. *Khachiyan Prize*: Gerald Brown (chair), Bill Cook, Andrzej Ruszczyński, and Yinyu Ye; and, *Young Researchers Prize*: Michael Ferris (chair), Jonathan Eckstein, Simge Küçükyavuz, and Suvrajeet Sen. Look for articles in the fall IOS newsletter by the other two prize winners, Kim-Chuan Toh (Farkas Prize) and Frans de Ruiter (Student Paper Prize).

Looking forward, as detailed in this newsletter:

- The deadline for nominations for the four 2018 IOS prizes is June 15.
- We seek nominations for IOS Vice Chairs in *Global Optimization*, *Nonlinear Optimization*, and *Optimization Under Uncertainty.* We further seek nominations for IOS Secretary/Treasurer. The deadline for nominations is June 30.
- We seek proposals to host the 2020 INFORMS Optimization Society's Conference.

We are now in our second year in which IOS publishes two newsletters per year. Thank you to Marina Epelman for leading our newsletter. Look for the next newsletter prior to the annual meeting in Phoenix. In the meantime, I wish you all a productive, optimized summer.

Pietro Belotti, showing off his IOS Distinguished Service Award, at the Society conference in Denver. (photo by Sven Leyffer)

On decomposability of the multilinear polytope and its implications in mixed-integer nonlinear optimization

Alberto Del Pia
Department of Industrial and Systems Engineering &
Wisconsin Institute for Discovery
University of Wisconsin-Madison
delpia@wisc.edu

Aida Khajavirad
Department of Chemical Engineering
Carnegie Mellon University
aida@cmu.edu

This article is dedicated to the memory of Manfred Padberg whose work on the Boolean quadric polytope inspired us to start this line of research.

1 Introduction

Central to the efficiency of global optimization algorithms is their ability to construct sharp and cheaply computable convex relaxations. Factorable programming techniques are used widely in global optimization of mixed-integer nonlinear optimization problems (MINLPs) for bounding general nonconvex functions [9]. These techniques iteratively decompose a factorable function, through the introduction of variables and constraints for intermediate nonlinear expressions, until each intermediate expression can be convexified effectively.

Suvrajeet Sen, Aida Khajavirad, Alberto Del Pia, and David Morton

Multilinear sets and polytopes. Factorable reformulations of many types of MINLPs, such as mixed-integer polynomial optimization problems, contain a collection of multilinear equations of the form $z_e = \prod_{v \in e} z_v$, $e \in E$, where E denotes a set of subsets of cardinality at least two of a ground set V. Let us define the set of points satisfying all multilinear equations present in a factorable reformulation of a MINLP as $\tilde{\mathcal{S}} = \{z : z_e = \prod_{v \in e} z_v \ \forall e \in E, \ z_v \in [0,1] \ \forall v \in V_1, \ z_v \in \{0,1\} \ \forall v \in V_2\}$, where V_1, V_2 forms a partition of V. It is well-known that the convex hull of $\tilde{\mathcal{S}}$ is a polytope and the projection of its vertices onto the space of the variables z_v, $v \in V$, is given by $\{0,1\}^V$. Hence, the facial structure of the convex hull of $\tilde{\mathcal{S}}$ can be equivalently studied by considering the following binary set:

$$\left\{ z \in \{0,1\}^{V+E} : z_e = \prod_{v \in e} z_v \ \forall e \in E \right\}. \qquad (1)$$

In particular, this set represents the feasible region of a linearized unconstrained $0-1$ polynomial optimization problem. There is a one-to-one correspondence between sets of form (1) and hypergraphs $G = (V, E)$. Henceforth we refer to (1) as the *multilinear set* of the hypergraph G and denote it by $\mathcal{S}_G$, and refer to its convex hull as the *multilinear polytope* of G and denote it by MP_G. (See, e.g., [5])

If all multilinear equations defining $\mathcal{S}_G$ are bilinear, the multilinear polytope coincides with the Boolean quadric polytope defined by Padberg [10] in the context of $0-1$ quadratic optimization, in which case our hypergraph representation simplifies to the graph representation of Padberg. Indeed, a significant amount of research has been devoted to studying the facial structure of the Boolean quadric polytope and these theoretical developments have had a significant impact on the performance of branch-and-cut based algorithms for mixed-integer quadratic optimization problems. However, similar polyhedral studies for higher degree multilinear polytopes are quite scarce. Our ultimate goal is to bridge this gap by performing a systematic study of the facial structure of the multilinear polytope, and thus paving the way for devising novel optimization algorithms for nonconvex problems containing multilinear sub-expressions.

Decomposability. In this article, we provide an overview of some of our recent results [4, 6] on the facial structure of higher degree multilinear polytopes with a special focus on their "decomposability" properties. Namely, we demonstrate that for multilinear polytopes decomposability plays a key role from both theoretical and algorithmic viewpoints.

Let us start by introducing some hypergraph terminology we need to formally define the notion of decomposability for the multilinear polytope. Given a hypergraph $G = (V, E)$, and a subset V' of V, the *section hypergraph* of G induced by V' is the hypergraph $G' = (V', E')$, where $E' = \{e \in E : e \subseteq V'\}$. Given hypergraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we denote by $G_1 \cap G_2$ the hypergraph $(V_1 \cap V_2, E_1 \cap E_2)$, and we denote by $G_1 \cup G_2$ the hypergraph $(V_1 \cup V_2, E_1 \cup E_2)$.

Now, consider a hypergraph G, and let G_j, $j \in J$, be distinct section hypergraphs of G such that $\cup_{j\in J} G_j = G$. Clearly, the system of all inequalities defining MP_{G_j} for all $j \in J$ provides a relaxation of MP_G as the convexification operation does not, in general, distribute over intersection. It is highly desirable to identify conditions under which these two sets coincide, as in such cases characterizing MP_G simplifies to characterizing each MP_{G_j} separately. More formally, we say that the polytope MP_G is *decomposable into polytopes* MP_{G_j}*, for* $j \in J$, if the following relation holds:

$$\mathrm{MP}_G = \bigcap_{j\in J} \overline{\mathrm{MP}}_{G_j}, \tag{2}$$

where $\overline{\mathrm{MP}}_{G_j}$ is the set of all points in the space of MP_G whose projection in the space defined by G_j is MP_{G_j}.

Organization. In Section 2 we provide a summary of our results in [4] regarding necessary and sufficient conditions for decomposability of multilinear polytopes based on the structure of their intersection hypergraphs. Subsequently, in Section 3 we present a polynomial-time algorithm to optimally decompose a multilinear polytope into a collection of nondecomposable multilinear polytopes. A detailed analysis of this algorithm can be found in [4]. In Section 4 we give a brief overview of our results in [6], wherein we study the complexity of the multilinear polytope in conjunction with the acyclicity degree of its hypergraph and show that for certain acyclic hypergraphs, the multilinear polytope is decomposable into a collection of simpler multilinear polytopes whose explicit description can be obtained directly.

2 Decomposability based on the intersection hypergraph

Suppose that G_1 and G_2 are section hypergraphs of G such that $G_1 \cup G_2 = G$. The following theorem provides a sufficient condition for decomposability of MP_G into MP_{G_1} and MP_{G_2}, based on the structure of the intersection hypergraph $G_1 \cap G_2$. In the following, we say that a hypergraph $\bar{G}$ is *complete* if all subsets of $V(\bar{G})$ of cardinality at least two are present in $E(\bar{G})$.

Theorem 1. *Let G be a hypergraph, and let G_1,G_2 be section hypergraphs of G such that $G_1 \cup G_2 = G$ and $G_1 \cap G_2$ is a complete hypergraph. Then the polytope MP_G is decomposable into MP_{G_1} and MP_{G_2}.*

Figure 1 illustrates some hypergraphs G for which MP_G is decomposable into MP_{G_1} and MP_{G_2}. To draw a hypergraph G, throughout this article, we represent the nodes in $V(G)$ by points, and the edges in $E(G)$ by closed curves enclosing the corresponding set of points.

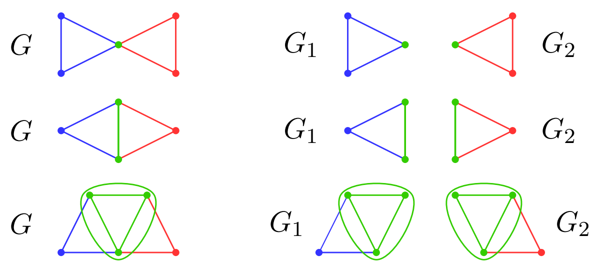


Figure 1

We now provide the proof sketch for Theorem 1. Given a vector z in the space defined by G, we denote by $z_\cap$ the vector that contains the components of z corresponding to nodes and edges that are in $G_1 \cap G_1$. Moreover, we denote by z_1 (resp. z_2) the vector that contains the components of z corresponding to nodes and edges that are in G_1 and not in G_2 (resp. in G_2 and not in G_1). The key step in proving Theorem 1 is to show that a vector $(\hat{z}_1, \hat{z}_\cap, \hat{z}_2)$ belongs to MP_G if $(\hat{z}_1, \hat{z}_\cap)$ can be written as a convex combination of vectors in $\mathcal{S}_{G_1}$ and $(\hat{z}_\cap, \hat{z}_2)$ can be written as a convex combination of vectors in $\mathcal{S}_{G_2}$. Clearly, given any two vectors $(z_1, z_\cap) \in \mathcal{S}_{G_1}$ and $(z'_\cap, z_2) \in \mathcal{S}_{G_2}$ with $z_\cap = z'_\cap$, we can combine them to obtain a vector $(z_1, z_\cap, z_2) \in \mathcal{S}_G$. Since by assumption the hypergraph $G_1 \cap G_2$ is complete, the polytope $\mathrm{MP}_{G_1 \cap G_2}$ is a simplex, implying that any vector $(\hat{z}_1, \hat{z}_\cap, \hat{z}_2)$ in MP_G can be written as a convex combination of the obtained vectors $(z_1, z_\cap, z_2)$ in $\mathcal{S}_G$. In partciular, Theorem 1 unifies the existing decomposability results for the Boolean quadric polytope QP_G [10]:

Corollary 2. *Consider a graph $G = G_1 \cup G_2$, where G_1 and G_2 are induced subgraphs of G with $V(G_1) \cap V(G_2) = \{u\}$, for some $u \in V(G)$, or $V(G_1) \cap V(G_2) = \{u, v\}$, for some $\{u, v\} \in E(G)$. Then QP_G is decomposable into QP_{G_1} and QP_{G_2}.*

The next theorem demonstrates the tightness of Theorem 1. We define the *rank* of a hypergraph G as the maximum cardinality of an edge in $E(G)$.

Theorem 3. *Let $\bar{G}$ be a rank-r hypergraph that is not complete. Then for any integer $r' \geq \max\{r, 2\}$, there exists a rank-r' hypergraph $G = G_1 \cup G_2$, where G_1 and G_2 are section hypergraphs of G with $\bar{G} = G_1 \cap G_2$, such that MP_G is not decomposable into MP_{G_1} and MP_{G_2}.*

Figure 2 illustrates some hypergraphs G for which MP_G is *not* decomposable into MP_{G_1} and MP_{G_2}.

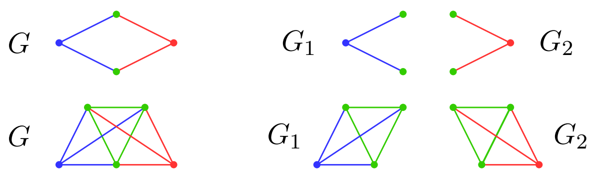


Figure 2

The proof of Theorem 3 is constructive. Namely, we search for a section hypergraph $\bar{H}$ of $\bar{G}$ with q nodes such that $E(\bar{H})$ consists of all edges of cardinality between 2 and $q-1$. Then we construct two hypergraphs H_1 and H_2 with $V(H_1) = V(\bar{H}) \cup \{u\}$, $E(H_1) = E(\bar{H}) \cup \{\{u, v\} : v \in V(\bar{H})\}$ and $V(H_2) = V(\bar{H}) \cup \{w\}$, $E(H_2) = E(\bar{H}) \cup \{\{w, v\} : v \in V(\bar{H})\}$. Subsequently, by letting $H = H_1 \cup H_2$, we provide a facet-defining inequality for MP_H with nonzero coefficients corresponding to some edges in $E(H_1) \setminus E(\bar{H})$ and in $E(H_2) \setminus E(\bar{H})$. This implies that MP_H is not decomposable into MP_{H_1} and MP_{H_2}. Next, we construct the hypergraph $G = H_1 \cup H_2 \cup \bar{G}$ and define G_1 and G_2 as the section hypergraphs of G induced by $V(H_1) \cup V(\bar{G})$ and $V(H_2) \cup V(\bar{G})$, respectively. We then show that since MP_H is not decomposable into MP_{H_1} and MP_{H_2}, the polytope MP_G is not decomposable into MP_{G_1} and MP_{G_2} either. It is simple to see that the rank of the hypergraph G constructed above is equal to $\max\{r, 2\}$. For any integer r' greater than $\max\{r, 2\}$, by adding a certain edge of cardinality r' to either G_1 or G_2, we can complete the proof.

In [10], Padberg poses a question regarding the decomposability of the Boolean quadric polytope when the intersection graph is a clique of cardinality greater than two. The proof of Theorem 3 implies that the answer to this question is negative for a clique with three or more nodes.

We conclude this section by remarking that in [4] we also present sufficient conditions for decomposability of multilinear polytopes with sparse intersection hypergraphs.

3 An optimal algorithm for decomposing the multilinear polytope

It is well-understood that branch-and-cut based MINLP solvers would highly benefit from our decomposition results as such techniques lead to significant reductions in CPU time during cut generation [1]. In this section, we present a simple and efficient algorithm for optimally decomposing a multilinear polytope into simpler and non-decomposable multilinear polytopes. Our proposed algorithm can be easily incorporated in MINLP solvers as a preprocessing step for cut generation. We start by presenting a sufficient condition for decomposability of MP_G into MP_{G_j}, for $j \in J$, which can be obtained by a recursive application of Theorem 1.

Theorem 4. *Let G be a hypergraph, and let G_j, $j \in J$, be section hypergraphs of G such that $\cup_{j \in J} G_j = G$. Suppose that for all $j, j' \in J$ with $j \neq j'$, the intersection $G_j \cap G_{j'}$ is the same complete hypergraph $\bar{G}$. Then MP_G is decomposable into MP_{G_j}, for $j \in J$.*

Now consider a hypergraph G and let $p \subset V(G)$. Denote by $\bar{G}$ the section hypergraph of G induced by p. We say that p *decomposes* G if

(a) the hypergraph $\bar{G}$ is complete,
(b) there exist at least two section hypergraphs G_j, $j \in J$, of G, with $V(G_j) \setminus V(G_{j'}) \neq \emptyset$ for all $j, j' \in J$ with $j \neq j'$, that together with $\bar{G}$ satisfy the hypothesis of Theorem 4.

If p does not decompose any G_j, $j \in J$, as defined in (b), then we refer to the family G_j, $j \in J$, as a *p-decomposition* of G. It can be shown that there exists a unique p-decomposition of G. The next result indicates that a p-decomposition test can be carried out efficiently.

Proposition 5. *Given a connected rank-r hypergraph $G = (V, E)$ and $p \subset V$, we can test if p decomposes G, and, if so, obtain the p-decomposition of G in $O(r|E|)$ time.*

Full decompositions. In general, a multilinear polytope MP_G is decomposable into simpler polytopes via a series of p-decompositions of G until none of the newly generated multilinear polytopes are decomposable. In the following, whenever a polytope MP_G is decomposable into polytopes MP_{G_k}, $k \in K$, we refer to the family G_k, $k \in K$, as a *decomposition* of G. Given a hypergraph G, we define its *full-decomposition* as a decomposition of G given by a family G_k, $k \in K$, with the following properties:

(i) There exists no G_k, for some $k \in K$, and $p \subset V(G_k)$ such that p decomposes G_k.
(ii) No hypergraph G_s, for some $s \in K$, is a section hypergraph of another hypergraph G_t, for some $t \in K$ with $t \neq s$.

If G_s is a section hypergraph of G_t for some $s, t \in K$ with $s \neq t$, then MP_{G_s} corresponds to a face of MP_{G_t}. Thus, removing G_s from a decomposition of G amounts to removing redundant inequalities from the description of MP_G, which is computationally beneficial. It can be shown that the following algorithm gives a full-decomposition of G.

```
Gen_dec : General full-decomposition algorithm
Input: A hypergraph $G$
Output: A full-decomposition of $G$
Initialize the family $\mathcal{L} = \{G\}$;
while $\mathcal{L}$ does not satisfy property (i) of full-decomposition do
    select a hypergraph $\tilde{G} \in \mathcal{L}$ and $p \subset V(\tilde{G})$;
    if $p$ decomposes $\tilde{G}$ then
        let $G_j$, $j \in J$, be the $p$-decomposition of $\tilde{G}$;
        let $\tilde{J}$ be the subset of $J$ such that each $G_j$, $j \in \tilde{J}$, is not a section
          hypergraph of any hypergraph in $\mathcal{L}$ different from $\tilde{G}$;
        in $\mathcal{L}$, replace $\tilde{G}$ with $G_j$, $j \in \tilde{J}$;
return $\mathcal{L}$;
```

Decomposition orders. In `Gen_dec`, we have not specified which $\tilde{G} \in \mathcal{L}$ and $p \subset V(\tilde{G})$ to choose at each iteration. We refer to different choices of $\tilde{G}$ and p throughout the execution of `Gen_dec` as *decomposition orders*. We denote a specific decomposition order by the sequence of choices that defines it, where each choice consists of a pair $(\tilde{G}, p)$, for some hypergraph $\tilde{G} \in \mathcal{L}$ and a set of nodes $p \subset V(\tilde{G})$ that is tested for p-decomposition of $\tilde{G}$. The next proposition indicates that a full-decomposition of G does not depend on the specific decomposition order used.

Proposition 6. *The full-decomposition of a hypergraph obtained by* `Gen_dec` *is independent of the decomposition order.*

Henceforth, we will speak of *the* full-decomposition of G. However, as we detail next, different decomposition orders result in different computational costs for `Gen_dec`. First, from the definition of `Gen_dec` it follows that the length of the decomposition order used is a reasonable measure for the overall cost of this algorithm and it can be shown that for a hypergraph G, every decomposition order contains at least $|V(G)| + |E(G)|$ pairs. Second, to ensure that property (ii) in the definition of the full-decomposition is satisfied, every time the p-decomposition of $\tilde{G}$ is generated, each new hypergraph G_j is compared with the existing ones and is added to $\mathcal{L}$ only if it is not a section hypergraph of another hypergraph in $\mathcal{L}$. Let us refer to the section hypergraphs not added to $\mathcal{L}$ as *redundant* hypergraphs. It can be shown that different decomposition orders in `Gen_dec` may result in distinct redundant hypergraphs. As the redundancy check is computationally expensive, it is beneficial to obtain a decomposition order that results in a minimum number of redundant hypergraphs.

The optimal decomposition algorithm. Next, we define a special sequence of choices $\bar{\mathcal{O}}$ in the execution of `Gen_dec` with highly desirable algorithmic properties. At a given iteration of `Gen_dec`, we say that $p \in V(\tilde{G}) \cup E(\tilde{G})$ is *tested* in $\tilde{G}$ if the pair $(\tilde{G}, p)$ has been already considered in an earlier iteration of `Gen_dec`. Moreover, we refer to the hypergraph $\tilde{G}$ in `Gen_dec` as the *parent* of each G_j. The *ancestors* of G_j are the parent of G_j, and the ancestors of the parent of G_j. At a given iteration, any hypergraph in the current family $\mathcal{L}$ can be chosen as $\tilde{G}$. Let the list $\{q_k, k \in K\}$ contain all nodes and edges of $\tilde{G}$ ordered by increasing cardinality. We define p to be the first element q_k in the above list that is *not* tested in $\tilde{G}$ or in any ancestor of $\tilde{G}$. The sequence $\bar{\mathcal{O}}$ ends when no such pair $(\tilde{G}, p)$ can be found.

Proposition 7. *The sequence $\bar{\mathcal{O}}$ is a decomposition order. Moreover, it creates no redundant hypergraphs. Consider a hypergraph G with n nodes and m edges. Let the decomposition order $\bar{\mathcal{O}}$ for G be given by $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$. Then $t = n + m$.*

In [4], we present an optimal full-decomposition algorithm, referred to as `Opt_dec`, which is obtained by an efficient incorporation of the decomposition order $\bar{\mathcal{O}}$ in `Gen_dec`. We refer to this algorithm as optimal due to two reasons. First, `Opt_dec` applies the minimum number of p-decomposition tests needed to obtain the full-decomposition of any hypergraph. Second, no redundant hypergraph is generated in the course of `Opt_dec`, and hence the costly redundancy test (as described in `Gen_dec`) is not required. The following proposition gives the worst-case running time of `Opt_dec`.

Proposition 8. *Consider a connected rank-r hypergraph G with n nodes and m edges. Then, the running time of `Opt_dec` is $O(rm(n + m))$.*

In [4], we provide an example that demonstrates the significance of our optimal decomposition algorithm; namely we define a hypergraph G and a decomposition order $\tilde{O}$, such that when incorporated in `Gen_dec`, in comparison to $\bar{O}$, the decomposition order $\tilde{O}$ requires $n(m-1)/2$ additional p-decomposition tests to obtain a full-decomposition of G. In addition, a total number of $n(n-2)/4 - 1$ redundant hypergraphs are generated in the course of `Gen_dec`.

4 The multilinear polytope of acyclic hypergraphs

In this section, we demonstrate the key role of decomposition in obtaining explicit descriptions for the multilinear polytope of certain acyclic hypergraphs. Moreover, these convex hull characterizations enable us to optimize a linear function over MP_G in polynomial time. We start by providing a sufficient condition for decomposability of multilinear polytopes that will be used for the subsequent developments.

Theorem 9. *Let G be a hypergraph, and let G_1, G_2 be section hypergraphs of G such that $G_1 \cup G_2 = G$. Denote by $\bar{p} := V(G_1) \cap V(G_2)$. Suppose that $\bar{p} \in V(G) \cup E(G)$, and that for every edge e of G containing nodes in $V(G_1) \setminus V(G_2)$ either $e \supset \bar{p}$, or $e \cap \bar{p} = \emptyset$. Then MP_G is decomposable into MP_{G_1} and MP_{G_2}.*

Figure 3 illustrates a hypergraph G for which by Theorem 9 the polytope MP_G is decomposable into MP_{G_1} and MP_{G_2}.

As in Theorem 1, to prove Theorem 9, we need to show that a vector $(\hat{z}_1, \hat{z}_\cap, \hat{z}_2)$ belongs to MP_G if $(\hat{z}_1, \hat{z}_\cap)$ can be written as a convex combination of vectors in $\mathcal{S}_{G_1}$ and $(\hat{z}_\cap, \hat{z}_2)$ can be written as a convex combination of vectors in $\mathcal{S}_{G_2}$. Moreover, as before, it is sufficient to consider vectors in $\mathcal{S}_G$

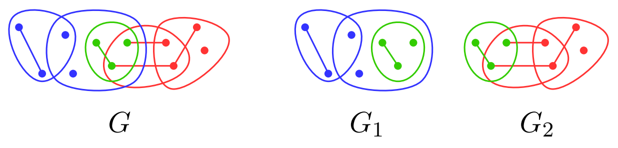


Figure 3

obtained by combining one vector $(z_1, z_\cap)$ in $\mathcal{S}_{G_1}$ with one vector $(z'_\cap, z_2)$ in $\mathcal{S}_{G_2}$. However, since in Theorem 9 the intersection hypergraph is not complete, it is no longer sufficient to only combine vectors with $z_\cap = z'_\cap$. In this case, we need to consider all vectors $(z_1, z'_\cap, z_2)$ obtained by combining a vector $(z_1, z_\cap) \in \mathcal{S}_{G_1}$ and a vector $(z'_\cap, z_2) \in \mathcal{S}_{G_2}$ with $z_{\bar{e}} = z'_{\bar{e}}$. The presence of $(z_1, z'_\cap, z_2)$ in $\mathcal{S}_G$ follows from the assumption that every edge that is only in G_1 either contains $\bar{e}$ or is disjoint from it. Moreover, the existence of the edge $\bar{e}$ implies that we can write the vector $(\hat{z}_1, \hat{z}_\cap, \hat{z}_2)$ as a convex combination of the obtained vectors $(z_1, z'_\cap, z_2)$ in $\mathcal{S}_G$.

Acyclic hypergraphs. Padberg [10] shows that for an acyclic graph, the Boolean quadric polytope admits a simple and compact description. This result can be obtained by showing that the Boolean quadric polytope of an acyclic graph is decomposable into a collection of Boolean quadric polytopes whose graphs consists of a single edge. To obtain similar characterizations for higher degree multilinear polytopes, it is then natural to look into the notion of acyclicity for hypergraphs. Interestingly, unlike graphs for which there is a single natural notion of acyclicity, for hypergraphs several different degrees of acyclicity have been defined [8]. In the following, we present two types of hypergraph acyclicity which will be used for the subsequent developments.

The most restrictive class of acyclic hypergraphs is the class of Berge-acyclic hypergraphs. A *Berge-cycle* in G is a sequence $v_1, e_1, v_2, e_2, \ldots, v_t, e_t, v_1$ with $t \geq 2$, such that (i) $v_1, v_2, \ldots, v_t$ are distinct nodes of G, (ii) $e_1, e_2, \ldots, e_t$ are distinct edges of G, (iii) $v_i, v_{i+1} \in e_i$ for $i = 1, \ldots, t-1$, and $v_t, v_1 \in e_t$. A hypergraph is *Berge-acyclic* if it contains no Berge-cycles. The next class of acyclic hypergraphs, in increasing order of generality, is the class of γ-acyclic hypergraphs. A *γ-cycle* in G is a Berge-cycle such that $t \geq 3$, and for each $i \in \{2, \ldots, t\}$, the node v_i belongs to e_{i-1}, e_i and no other e_j. A hypergraph is *γ-acyclic* if it contains no γ-cycles.

Acyclicity and decomposability. The link between hypergraph acyclicity and decomposability is given by the concept of leaf of a hypergraph. Consider a hypergraph $G = (V, E)$. An edge of G is *maximal* if it is not contained in any other edge of G. We say that an edge e' is a *leaf* of G if it is a maximal edge and $e' \cap (\cup_{e \in E \setminus E'} e) \subset \tilde{e}$ for some $\tilde{e} \in E \setminus E'$, where E' is the set of edges contained in e'. It can be shown that every γ-acyclic hypergraph contains a leaf. The existence of a leaf, together with the special structure of Berge-acyclic and γ-acyclic hypergraphs enables us to employ our decomposition results and derive an explicit description of MP_G by induction on the number of maximal edges of G.

Berge-acyclic hypergraphs. The *standard linearization* MP_G^{LP} is a widely-used relaxation of $\mathcal{S}_G$ and is obtained by replacing each multilinear equation $z_e = \prod_{v \in e} z_v$ by its convex hull over the unit hypercube (see, e.g., [3]):

$$\begin{array}{ll} z_v \leq 1 & \forall v \in V, \\ z_e \geq 0 & \forall e \in E, \\ z_e \geq \sum_{v \in e} z_v - |e| + 1 & \forall e \in E, \\ z_e \leq z_v & \forall e \in E,\ \forall v \in e. \end{array}$$

We now show that for a Berge-acyclic hypergraph, we have $\text{MP}_G = \text{MP}_G^{\text{LP}}$.

Any two edges of a Berge-acyclic hypergraph intersect in at most one node. It then follows that the hypergraph considered in the base case of the induction consists of a single edge. Hence, the corresponding multilinear polytope coincides with the standard linearization.

In the inductive step, we construct MP_G in two steps:

1. Decompose the polytope MP_G into MP_{G_1} and MP_{G_2}, where G_1 is the section hypergraph of G induced by the leaf e' and G_2 is the section hypergraph of G induced by $\cup_{e \in E \setminus E'} e$.
2. Obtain MP_G by juxtaposing the description of MP_{G_1} and of MP_{G_2} given by the induction hypothesis.

For a Berge-acyclic hypergraph, the intersection of the leaf e' with the hypergraph G_2, i.e., the set $\bar{p}$

defined in Theorem 9, consists of at most one node. Hence, all assumptions of Theorem 9 are trivially satisfied and we can utilize this result to perform the decomposition described in Step 1. Hence, if G is a Berge-acyclic hypergraph, we have $\mathrm{MP}_G = \mathrm{MP}^{\mathrm{LP}}_G$. In fact, we have proved that the converse holds as well. More precisely, we have shown the following:

Theorem 10. *$MP_G = MP^{LP}_G$ if and only if G is a Berge-acyclic hypergraph.*

It follows directly from Theorem 10 that for a Berge-acyclic hypergraph G, we can optimize a linear function over MP_G via linear optimization in polynomial time.

In [10], Padberg shows that the standard linearization coincides with the Boolean quadric polytope if and only if G is an acyclic graph. Therefore, Theorem 10 generalizes Padberg's result to higher degree multilinear polytopes.

γ-acyclic hypergraphs. To characterize the multilinear polytope of γ-acyclic hypergraphs, we introduce a class of valid inequalities for MP_G which we will refer to as flower inequalities. Let e_0 be an edge of G and let e_k, $k \in K$, be a collection of edges such that $|e_0 \cap e_k| \geq 2$ for every $k \in K$, and $e_i \cap e_j = \emptyset$ for all $i, j \in K$ with $i \neq j$. Then a *flower inequality* for MP_G is given by:

$$\sum_{v \in e_0 \setminus \cup_{k\in K} e_k} z_v + \sum_{k \in K} z_{e_k} - z_{e_0} \leq |e_0 \setminus \cup_{k \in K} e_k| + |K| - 1.$$

We define the *flower relaxation* MP^F_G as the relaxation of the multilinear set obtained by adding all flower inequalities to its standard linearization $\mathrm{MP}^{\mathrm{LP}}_G$. We now show that for a γ-acyclic hypergraph, we have $\mathrm{MP}_G = \mathrm{MP}^F_G$.

To establish the base case of the induction, we make use of the fact that a γ-acyclic hypergraph with one maximal edge is a laminar hypergraph. The multilinear polytope of a laminar hypergraph can be characterized using a fundamental result due to Conforti and Cornuéjols regarding the connection between integral polyhedra and balanced matrices [2]. This characterization in turn implies that for a laminar hypergraph the multilinear polytope coincides with its flower relaxation.

In the inductive step, Theorem 10 cannot be directly applied to MP_G as was the case for Berge-acyclic hypergraphs. However, we can utilize this result after the addition of one extra edge to G. In more detail, we construct MP_G in four steps:

1. Define the hypergraph G^+ obtained from G by adding the edge $\bar{p} := e' \cap (\cup_{e \in E \setminus E'} e)$.
2. Decompose the polytope MP_{G^+} into MP_{G_1} and MP_{G_2}, where G_1 is the section hypergraph of G^+ induced by e' and G_2 is the section hypergraph of G^+ induced by $\cup_{e \in E \setminus E'} e$.
3. Obtain MP_{G^+} by juxtaposing the description of MP_{G_1} given by the base case, and of MP_{G_2} given by the induction hypothesis.
4. Obtain MP_G by projecting out the variable $\bar{p}$ from the description of MP_{G^+}.

The section hypergraph induced by an edge of a γ-acyclic hypergraph is laminar. This in particular implies that for every edge e of G containing nodes in $V(G_1) \setminus V(G_2)$ either $e \supset \bar{p}$, or $e \cap \bar{p} = \emptyset$. Hence, we can employ Theorem 9 to perform the decomposition described in Step 2. Finally, by projecting out the variable $z_{\bar{p}}$ from the description of MP_{G^+} using Fourier-Motzkin elimination, we conclude that $\mathrm{MP}_G = \mathrm{MP}^F_G$.

In fact, we have shown that the converse holds as well. More precisely, we have shown the following:

Theorem 11. *$MP_G = MP^F_G$ if and only if G is a γ-acyclic hypergraph.*

For γ-acyclic hypergraphs, the number of facets of MP^F_G may not be bounded by a polynomial in $|V(G)|, |E(G)|$. However, flower inequalities can be separated in strongly polynomial time, and this allows us to optimize a linear function over MP_G in polynomial time.

We conclude this article by remarking that in [7] we extend the above decomposition based technique to characterize the multilinear polytope for a more general class of acyclic hypergraphs.

Acknowledgments

We would like to thank the Young Researcher Prize Committee consisting of Michael Ferris (chair), Jonathan Eckstein, Simge Küçükyavuz, and Suvrajeet Sen; it is a great honor to have been selected as recipients of this prize. This research was supported in part by National Science Foundation award CMMI-1634768.

REFERENCES

[1] X. Bao, A. Khajavirad, N.V. Sahinidis, and M. Tawarmalani. Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, 7(1):1–37, 2014.

[2] G. Cornuéjols. *Combinatorial Optimization: Packing and Covering*, volume 74 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 2001.

[3] Y. Crama. Concave extensions for non-linear 0−1 maximization problems. *Mathematical Programming*, 61(1–3):53–60, 1993.

[4] A. Del Pia and A. Khajavirad. On decomposability of multilinear sets. *Mathematical Programming*, 2017. DOI: 10.1007/s10107-017-1158-z.

[5] A. Del Pia and A. Khajavirad. A polyhedral study of binary polynomial programs. *Mathematics of Operations Research*, 42(2):389–410, 2017.

[6] A. Del Pia and A. Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28(2):1049–1076, 2018.

[7] A. Del Pia and A. Khajavirad. The running intersection relaxation of the multilinear polytope. *Optimization Online manuscript 2018/05/6618*, 2018.

[8] R. Fagin. Degrees of acyclicity for hypergraphs and relational database schemes. *Journal of the ACM*, 30(3):514–550, 1983.

[9] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[10] M. Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1–3):139–172, 1989.

Sparsity Matters

Robert J. Vanderbei
Dept. of Operations Research and Financial Engineering
Princeton University
rvdb@Princeton.EDU

I am deeply honored and grateful to have been awarded the 2017 Khachiyan Prize from the INFORMS Optimization Society. I really love optimization — one might say I'm an optimistic person. Over the past several decades I have had the great pleasure to collaborate with a number of similarly optimistic researchers and we've had a lot of fun working on a broad range of topics in optimization. In this article, I will focus on one particular issue that permeates much of the world of optimization, namely, the importance of understanding that "modeling matters" and, in particular, that exploiting sparsity in a problem's representation can be extremely beneficial.

1 Introduction

As is well-known in the optimization world, the worst case complexity of interior-point methods for linear programming is roughly $O(n^{3.5}L)$ where n denotes the number of variables and L denotes the

Gerald Brown, Robert Vanderbei, and David Morton

number of bits of data needed to encode the problem. It is also well-known that the standard variants of the simplex method take an exponential number of pivots in the worst case but that, in practice, the simplex method and interior-point methods have similar performance on average although one algorithm might beat the other by a significant amount on a particular problem.

This last "well-known fact" is perhaps the most important one — things vary greatly from one instance of a problem to another. Those of us in the field of optimization understand this quite well but users from outside the field often don't appreciate this subtlety. It is very common to hear someone say that their problem has tens of thousands of variables and therefore even an n^3 algorithm will be too slow to solve the problem. That is certainly true in some cases. But, it is not universally true and that's what this article is about. Before getting into details, I'd like to mention that this topic brings to mind a comment made by John Forrest at a conference back in the late 1980's. He said

The simplex method is 200 times faster than ...
... the simplex method.

In this modern era infused with many different optimization algorithms, I would generalize John's statement to this:

Any optimization algorithm is 200 times faster than ...
... that same algorithm.

2 What Can Go Wrong?

The practical performance issues that I'd like to discuss can be broken down into three broad categories:

- Sometimes the straightforward/default numerical *linear algebra* isn't the best way to solve a particular type of problem.
- Sometimes the obvious natural formulation of an optimization problem isn't very easy to solve but there is a *mathematically equivalent variant* that one can solve quickly.
- Sometimes the real-world problem to be solved isn't precisely defined and an *alternate formulation* might be vastly easier to solve.

In the following sections, I will discuss some examples of each of these scenarios. Some of these examples are well known in the optimization community but a few of them are a little more specific to particular interests of my own. I hope even these esoteric examples stimulate some interest among a broad reader base.

3 Numerical Linear Algebra

Problems with Dense Columns

Some linear programming problems have a constraint matrix that is mostly sparse but might have one or a few dense columns. For example, consider this LP:

$$\begin{array}{lrcrcl} \text{maximize} & c^T x & + & c_0\, x_0 & & \\ \text{subject to} & Ax & + & a\, x_0 & = & b \\ & & & x, x_0 & \geq & 0, \end{array}$$

where x is a high-dimensional vector, x_0 is just a single scalar variable, A is a *very sparse matrix*, but a is a *dense vector*.

There are practical real-world problems with this structure. And, for example, the "Big-M" method for getting an initial starting point for interior-point methods involves such a structure.

In the early days of interior-point methods (the mid 1980's), the computationally intensive part of these algorithms involved finding the "step directions" by solving the so-called "normal equations" for the dual step direction Δy:

$$\begin{bmatrix} A & a \end{bmatrix} \begin{bmatrix} D^2 & 0 \\ 0 & d^2 \end{bmatrix} \begin{bmatrix} A^T \\ a^T \end{bmatrix} \Delta y = ...$$

where D is a diagonal matrix and d is a scalar. Multiplying out, we get:

$$(AD^2A^T + d^2\, aa^T)\, \Delta y = ...$$

The matrix AD^2A^T is *sparse* but aa^T is *dense*. Hence, the sum is dense and solving the system of equations as formulated is highly inefficient. There are three ways to address this problem:

- Use the *Sherman-Morrison-Woodbury* formula to express solutions to the dense system in terms of solutions to the system without the dense (rank one) term. [4, 1, 7]

- Solve the *reduced KKT system* instead of the normal equations. [11]
- Re-express the problem using several variables in place of the single variable x_0. [9]

Splitting Dense Columns

A problem with these constraints...

$$\left[\begin{array}{cc} A & a \end{array}\right] \left[\begin{array}{c} x \\ x_0 \end{array}\right] = b$$

can be reformulated in this equivalent but larger but also sparser fashion...

$$\left[\begin{array}{c|ccccc} & a_1 & & & & \\ & & a_2 & & & \\ A & & & a_3 & & \\ & & & & \ddots & \\ & & & & & a_m \\ \hline & 1 & -1 & & & \\ & & 1 & -1 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & -1 \end{array}\right] \left[\begin{array}{c} x \\ \hline x_{0,1} \\ x_{0,2} \\ x_{0,3} \\ \vdots \\ x_{0,m} \end{array}\right] = \left[\begin{array}{c} b \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right]$$

For an optimizer based on solving the normal equations, this second form of the problem will solve *much faster* than the original form. Of course, most/all modern interior-point method solvers solve the reduced KKT system to avoid this issue. This *dense columns* example is a bit dated but it illustrates an important point:

> The number of constraints, m, and the number of variables, n, often have little to do with how long it takes to solve a problem.

The *sparsity* of the constraint matrix plays a huge role.

Exploiting Sparsity

Modern solvers are designed to exploit sparsity in the constraint matrix. I recently solved a generalized network flow problem using AMPL [3] with LOQO [10]. The problem had 195,503 variables and 123,571 constraints. Of course, being a (generalized) network flow problem the constraint matrix had only about two nonzeros per column. I was able to solve it in just 3.9 seconds on my laptop computer. For comparison, I solved a dense problem with 1/100-th as many variables and 1/100-th as many constraints. It took 59 seconds to solve. If we were to scale this up to the same size as the network flow problem, it would take about $59 \times 100^{3.5} = 590,000,000$ seconds to solve — in other words, about 18.7 years (and, of course, it wouldn't fit in the memory on my laptop computer).

4 Mathematically Equivalent Problems

Now let's consider an example illustrating how problems often have equivalent formulations that differ dramatically in how long it takes to solve them. The problem we will consider is the *Markowitz model* for portfolio selection.

In the Markowitz model, our vector x of decision variables is a stochastic vector (i.e., the elements are nonnegative and sum to one) representing the fraction of our portfolio to invest in various asset choices and the problem is to optimize some combination of risk and reward. The reward term is linear and can be denoted $r^T x$ whereas the risk term is quadratic and can be expressed in terms of a covariance matrix: $x^T \Sigma x$. If we let μ denote the weighting factor that controls the trade-off between these two objectives, we can express the Markowitz model like this:

$$\begin{array}{ll} \text{minimize} & \mu\, x^T \Sigma x - r^T x \\ \text{subject to} & \qquad e^T x = 1 \\ & \qquad\ \ x \geq 0. \end{array}$$

Here's an equivalent formulation of the problem. The matrix Σ is positive semidefinite and therefore can be expressed (and probably was defined) as a product $\Sigma = U^T U$. So, we can reexpress the problem like this:

$$\begin{array}{lrcrcl} \text{minimize} & \mu\, y^T y & - & r^T x & & \\ \text{subject to} & y & - & Ux & = & 0 \\ & & & e^T x & = & 1 \\ & & & x & \geq & 0. \end{array}$$

This formulation has more variables and more constraints. But, the quadratic term in the objective function is now a simple (sparse) $y^T y$ whereas the quadratic term in the original formulation was dense

$(x^T \Sigma x)$. This can make a big difference. Also, the matrix U does not have to be a square matrix — it might have many fewer rows than columns.

Here's a specific example. For a problem with 10,000 market assets and a covariance matrix based on data from 100 time periods in the past, the first problem has 10,000 variables and just one constraint. It solves in 1480 seconds. The second formulation has 10,100 variables and 101 constraints. It solves in 10.3 seconds — a speedup by a factor of 144.

5 Alternate Formulations

In the previous section, we discussed the fact that there are often equivalent formulations of a problem one of which might solve much faster than another. Sometimes there is some freedom in the definition of the problem and minor changes to the definition can lead to dramatic speedups. To illustrate such benefits of *alternate formulations* let us consider various expressions of the *compressed sensing* problem.

Compressed sensing

The goal of compressed sensing is to recover a sparse signal from a small number of measurements. Let $x^0 = (x_1^0, \ldots, x_n^0)^T \in \mathbb{R}^n$ denote a signal to be recovered. Here, we assume that n is large and that the signal vector x^0 is sparse.

Let A be a given (or chosen) $m \times n$ matrix with $m \ll n$. The *compressed sensing problem* is to recover x^0 assuming only that we know $y = Ax^0$ and that x^0 is sparse.

Since x^0 is a sparse vector, one can hope that it is the sparsest solution to the underdetermined linear system and therefore can be recovered from y by solving

$$(\mathrm{P}_0) \qquad \min_x \|x\|_0 \quad \text{subject to } Ax = y,$$

where $\|x\|_0$ denotes the 0-pseudo-norm of x:

$$\|x\|_0 = \#\{i : x_i \neq 0\}.$$

As is well-known, this problem is NP-hard due to the nonconvexity of the 0-pseudo-norm.

To make the problem more tractible, it is common to replace the 0-pseudo-norm with the 1-norm: $\|x\|_1 = \sum_j |x_j|$. This new problem is called the *basis pursuit* problem:

$$(\mathrm{P}_1) \qquad \min_x \|x\|_1 \quad \text{subject to } Ax = y.$$

This problem can be converted to a linear programming problem using the standard tricks for handling absolute values in minimization problems. There is an extensive literature that studies the probability as a function of m, n, and the choice of A that a solution to the basis pursuit problem is actually a solution to the compressed sensing problem.

Kronecker Compressed Sensing

As formulated, the compressed sensing problem involves a signal that is a vector. In some applications, it is more natural to assume that the signal is a matrix or even a higher dimensional tensor. Let's consider a matrix signal.

Given a sparse matrix signal $X^0 \in \mathbb{R}^{n_1 \times n_2}$, we can use two sensing matrices $A \in \mathbb{R}^{m_1 \times n_1}$ and $B \in \mathbb{R}^{m_2 \times n_2}$ and try to recover X^0 from knowledge of $Y = AX^0B^T$ by solving the *Kronecker compressed sensing* problem:

$$(\mathrm{P}_2) \qquad \hat{X} = \operatorname{argmin} \|X\|_1 \quad \text{subject to } AXB^T = Y.$$

Here, of course, $\|X\|_1$ is the sum of the absolute values of all the entries of the matrix X. As with the basis pursuit problem, this problem can also be formulated as a linear programming problem.

When the signal is multidimensional, Kronecker compressed sensing is more natural than classical vector compressed sensing. Also, as we will explain shortly, the linear programming problem for Kronecker compressed sensing benefits from sparsity that is not present in the vector-based formulation. Hence, for a vector problem and a matrix problem of the same size (i.e., the same number of elements in the signal), one would expect the matrix problem to solve more quickly.

Kroneckerizing Vector Problems

Sometimes, even when facing vector signals, it is beneficial to use Kronecker compressed sensing due to its added computational efficiency.

More specifically, even though the target signal is a vector $x^0 \in \mathbb{R}^n$, if we assume that n can be factored into $n_1 \times n_2$, then we can first reshape x^0 into a matrix $X^0 \in \mathbb{R}^{n_1 \times n_2}$ by putting successive length n_1 sub-vectors of x^0 into columns of X^0. We then multiply the matrix signal X^0 on both the left and the right by sensing matrices A and B to get a compressed matrix signal Y^0. We will show that we are able to solve this Kronecker compressed sensing problem much more efficiently than the corresponding vector compressed sensing problem.

Vectorizing the Kroneckerization

In the Kronecker problem the variables are naturally represented as a matrix. But, an LP solver expects to see a vector of variables, not a matrix. So, we need to rewrite the Kronecker problem in vector form.

The simple/natural/naive vectorization can be described as follows. Let $x = \text{vec}(X)$ and $y = \text{vec}(Y)$, where, as usual, the $\text{vec}(\cdot)$ operator takes a matrix and concatenates its elements column-by-column to build one large column-vector containing all the elements of the matrix.

In terms of x and y, problem (P$_2$) can be rewritten as an equivalent *vector compressed sensing* problem:

$$\text{vec}(\hat{X}) = \text{argmin}\|x\|_1 \quad \text{subject to} \quad Ux = y,$$

where U is given by the $(m_1m_2) \times (n_1n_2)$ Kronecker product of A and B:

$$U = B \otimes A = \begin{bmatrix} Ab_{11} & \cdots & Ab_{1n_2} \\ \vdots & \ddots & \vdots \\ Ab_{m_21} & \cdots & Ab_{m_2n_2} \end{bmatrix}.$$

The matrix U is fully dense. *This is bad.*

Sparsifying the Constraint Matrix

The key to an *efficient* algorithm for solving the linear programming problem associated with the Kronecker sensing problem lies in noting that the dense matrix U can be factored into a product of two sparse matrices:

$$\begin{aligned} U &= \begin{bmatrix} Ab_{11} & \cdots & Ab_{1n_2} \\ \vdots & \ddots & \vdots \\ Ab_{m_21} & \cdots & Ab_{m_2n_2} \end{bmatrix} \\ &= \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \begin{bmatrix} b_{11}I & \cdots & b_{1n_2}I \\ \vdots & \ddots & \vdots \\ b_{m_21}I & \cdots & b_{m_2n_2}I \end{bmatrix} \\ &=: VW. \end{aligned}$$

Here, I denotes an $n_1 \times n_1$ identity matrix, 0 denotes an $m_1 \times n_1$ zero matrix, and V is a block-diagonal matrix with A on the diagonal blocks.

Exploiting the Sparsification

Assuming that A and B are dense matrices (as they generally are in Kronecker compressed sensing), the matrix U is usually completely dense. But, it is a product of two sparse matrices: V and W. We can exploit this sparse factorization.

Let's introduce some new variables, call them z, and rewrite the constraints like this:

$$\begin{array}{rcrcl} z & - & Wx & = & 0 \\ Vz & & & = & y. \end{array}$$

Using the common trick of expressing a free variable as the difference of two nonnegative variables and the absolute value of that free variable as the sum of the two nonnegative variables, we can convert the problem to a linear program:

$$\begin{array}{lrcrcl} \min\limits_{x^+,x^-} & & & \mathbf{1}^T(x^+ + x^-) & & \\ \text{subject to} & z & - & W(x^+ - x^-) & = & 0 \\ & Vz & & & = & y \\ & & & x^+, x^- & \geq & 0. \end{array}$$

This formulation has more variables and more constraints. But, the constraint matrix is *sparse*. And as we've discussed already, for linear programming, sparsity of the constraint matrix is the key to algorithm efficiency.

Numerical Results

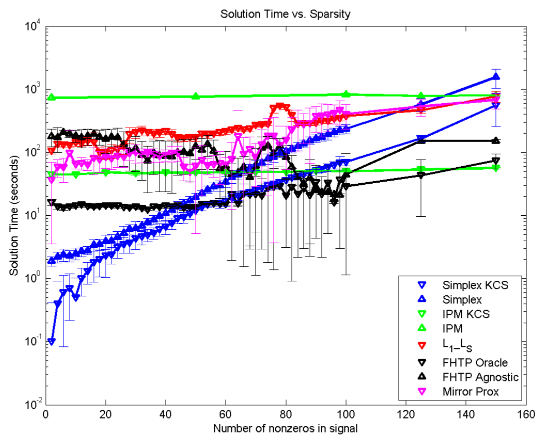
The numerical results I'll present here were first published in [12].

For the *vector* sensor, we generated random problems using $m = 1{,}122 = 33 \times 34$ and $n = 20{,}022 = 141 \times 142$. We varied the number of nonzeros k in signal x^0 from 2 to 150. We solved the straightforward linear programming formulations of these instances using my interior-point solver LOQO [10]. We also solved a large number of instances of the problem using the parametric simplex method as described in [8].

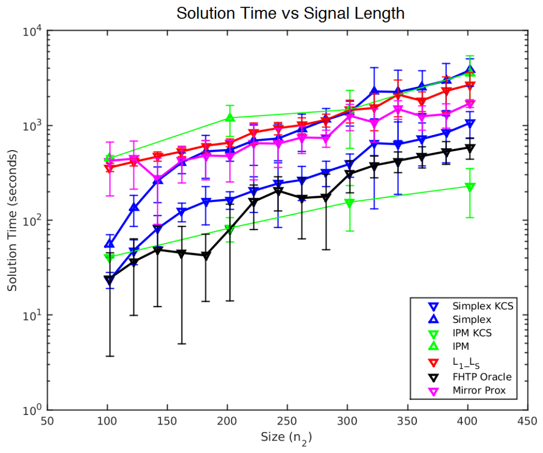
We followed a similar plan for the *Kronecker* sensor. For these problems, we used $m_1 = 33$, $m_2 = 34$, $n_1 = 141$, $n_2 = 142$, and various values of k. Again, the straightforward linear programming problems were solved by LOQO and the parametrically formulated versions were solved by a custom developed parametric simplex method.

For the Kronecker sensing problems, the matrices A and B were generated so that their elements are independent standard Gaussian random variables. For the vector sensing problems, the corresponding matrix $\mathbf{U}$ was used.

We also ran some publicly-available, state-of-the-art codes: ℓ_1-ℓ_s [6], *FHTP* [2], and *Mirror Prox* [5].



$m = 1{,}222$, $n = 20{,}022$. Error bars represent one standard deviation.



$m = 1{,}122$, $k = 100$, $n = 141 \times n_2$.

Conclusions

The interior-point solver (LOQO) applied to the Kronecker sensing problem is uniformly faster than both ℓ_1-ℓ_s and the interior-point solver applied to the vector problem (the three horizontal lines in the plot).

For very sparse problems, the parametric simplex method is best. In particular, for $k \le 70$, the parametric simplex method applied to the Kronecker sensing problem is the fastest method. It can be two or three orders of magnitude faster than ℓ_1-ℓ_s.

But, as explained earlier, the Kronecker sensing problem involves changing the underlying problem being solved. If one is required to stick with the vector problem, then it too is the best method for $k \le 80$ after which the ℓ_1-ℓ_s method wins.

Instructions for downloading and running the various codes/algorithms described herein can be found at:

`http://www.orfe.princeton.edu/~rvdb/tex/CTS/kronecker_sim.html`

REFERENCES

[1] I.C. Choi, C.L. Monma, and D.F. Shanno. Further development of a primal-dual interior point method. *ORSA J. on Computing*, 2:304–311, 1990.

[2] S. Foucart. Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, December 2011.

[3] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Belmont CA, 1993.

[4] P.E. Gill, W. Murray, and M.A. Saunders. A single-phase dual barrier method for linear programming. Technical Report SOL 88-10, Systems Optimization Laboratory, Stanford University, Stanford, CA, 1988.

[5] A. Juditsky, F. Kılınç Karzan, and A. Nemirovski. Randomized first order algorithms with applications to ℓ_1-minimization. *Mathematical Programming*, 142(1):269–310, Dec 2013.

[6] S.J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Transactions on Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[7] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Computational experience with a primal-dual interior point method for linear programming. Technical Report SOR 89-17, Department of Civil Engineering and Operations Research, Princeton University, October 1989.

[8] B. Rudloff, F. Ulus, and R.J. Vanderbei. A parametric simplex algorithm for linear vector optimization problems. *Mathematical Programming, Series A*, 163(1):213–242, 2017.

[9] R.J. Vanderbei. Splitting dense columns in sparse linear systems. *Lin. Alg. and Appl.*, 152:107–117, 1991.

[10] R.J. Vanderbei. LOQO user's manual — version 3.10. *Optimization Methods and Software*, 12:485–514, 1999.

[11] R.J. Vanderbei and T.J. Carpenter. Symmetric indefinite systems for interior-point methods. *Mathematical Programming*, 58:1–32, 1993.

[12] R.J. Vanderbei, K. Lin, H. Liu, and L. Wang. Revisiting Compressed Sensing: Exploiting the Efficiency of Simplex and Sparsification Methods. *Mathematical Programming, Series C*, 8:253–269, 2016

Mountains of Optimization indeed! (Photo by Thiago Serra)

2018 IOS Conference Report

Alexandra M. Newman
Department of Mechanical Engineering
Colorado School of Mines
anewman@mines.edu

The 2018 edition of the INFORMS Optimization Society conference was held in Denver, Colorado on March 23–25, 2018. Over 200 participants descended on the mile-high city, contributing to nearly 70 sessions, three tutorials, and seven plenaries.

The theme of the conference was "Mountains of Optimization," and mountains there were. Sessions on optimization under uncertainty and nonlinear optimization were among the most copious, but attendees were also treated to topics on discrete optimization, optimization in machine learning and statistics, network optimization, and software and implementation, inter alia.

The seven plenaries featured:

- Shabbir Ahmed: "Value of Multi-Stage Stochastic Optimization in Power Systems Operations"
- Marcos Goycoolea: "Large-scale Open Pit Mine Production-Scheduling"
- Moritz Hardt: "Non-convex non-optimization"
- Illya Hicks: "Discrete Optimization and Network Analysis"
- Karla Hoffman: "Hybrid optimization algorithms to solve real-world problems" (based on material from this year's Franz Edelman Award winning team)
- John Hooker: "What Decision Diagrams Can Do for You"
- Sven Leyffer: "A Globally Convergent Cutting-Plane Method for Simulation-Based Optimization with Integer Constraints"

The three tutorials consisted of:

- Bob Fourer: "A Guide to Identifying Good Near-Optimal Formulations for Hard Mixed-Integer Programs"
- Ed Klotz: "Performance Tuning For CPLEX's Spatial Branch-and-Bound Solver For Global Nonconvex Mixed Integer Quadratic Programs"
- Warren Powell: "A Unified Modeling and Algorithmic Framework for Optimization under Uncertainty"

All plenary and tutorial speakers were kind enough to supply their slides, which are posted on the conference website at: `http://orwe-conference.mines.edu/info.html`.

A special "thank you" must be extended to the co-organizers of the conference, Stephen Billups and

Steffen Borgwardt, both of the University of Colorado, Denver, and Manuel Laguna of the University of Colorado, Boulder.

Nominations for Society Prizes Sought

The INFORMS Optimization Society awards four prizes annually at the INFORMS annual meeting. We seek nominations (including self-nominations) for each of them, due by **June 15, 2018**. Each of the four awards includes a cash amount of US$1,000 and a citation plaque. The award winners will be invited to give a presentation in a special session sponsored by the Optimization Society during the INFORMS annual meeting in Phoenix, AZ, in November 2018 (the winners will be responsible for their own travel expenses to the meeting). Award winners are also asked to contribute an article about their award-winning work to the Optimization Society newsletter.

The four awards are listed below. Additional information on the awards, and nomination instructions, can be found on the society website (`http://connect.informs.org/optimizationsociety/prizes`). Inquiries should be sent directly via email to the chair of the corresponding prize committee.

The **Khachiyan Prize** is awarded for outstanding lifetime contributions to the field of optimization by an individual or team. The topic of the contribution must belong to the field of optimization in its broadest sense. Recipients of the INFORMS John von Neumann Theory Prize or the MOS/SIAM Dantzig Prize in prior years are not eligible for the Khachiyan Prize. The prize may be awarded once in a lifetime to any individual. Nominations should be submitted to the chair of the committee.

The prize committee for this year's Khachiyan Prize is as follows:

- Suvrajeet Sen (chair)
  s.sen@usc.edu
- Ignacio Grossman
- Arkadi Nemirovski
- David Shmoys

The **Farkas Prize** is awarded for outstanding contributions by a mid-career researcher to the field of optimization, over the course of their career. Such contributions could include papers (published or submitted and accepted), books, monographs, and software. The awardee will be within 25 years of their terminal degree as of January 1st of the year of the award. The prize serves as an esteemed recognition of colleagues in the middle of their career. The prize may be awarded at most once in their lifetime to any person. A nomination shall consist of: (i) a letter of nomination, not exceeding two pages, summarizing the nominee's contributions with explanations of their importance and impact; (ii) a curriculum vitae for the nominee, not exceeding four pages; and (iii) two support letters, each not exceeding two pages. These letters can be sent directly to the committee chair or to the nominator, to be included in the nomination package.

The prize committee for this year's Farkas Prize is as follows:

- Patrick Jaillet (chair)
  jaillet@mit.edu
- Donald Goldfarb
- Andy Philpott
- Nick Sahinidis

The **Prize for Young Researchers** is awarded to one or more young researcher(s) for an outstanding paper in optimization that is published in, or submitted to and accepted by, a refereed professional journal within the four calendar years preceding the year of the award. The prize serves as an esteemed recognition of promising colleagues who are at the beginning of their academic or industrial career. All authors must have earned their most recent degree within the eight calendar years preceding the year of award, or be enrolled in a degree-granting program. (Note: All authors of a prize-winning paper will be co-winners of the prize.) The topic of the paper must belong to the field of optimization in its broadest sense. The prize may be awarded once in a lifetime to any individual. The paper may not be simultaneously submitted to an INFORMS student paper competition. Nominations should be submitted to the chair of the committee.

The prize committee for this year's Prize for Young Researchers is as follows:

- Katya Scheinberg (chair) katyas@lehigh.edu
- Yongpei Guan
- Fatma Kılınç-Karzan
- Andrea Lodi

The **Student Paper Prize** is awarded to one or more student(s) for an outstanding paper in optimization that is submitted to and received, or published in a refereed professional journal no more than three years preceding the year of the award. Every nominee/applicant must be a student on January 1st of the year of the award. The prize serves as an esteemed recognition of promising students who are looking for an academic or industrial career. A complete entry consists of a single PDF file containing: (i) a copy of the paper; (ii) a letter signed by all coauthors attesting that the majority of the work was done by the student(s); (iii) a nomination letter attesting that the eligibility conditions have been satisfied by the entrant(s) and the paper. Nominations should be submitted to the chair of the committee.

The prize committee for this year's Student Paper Prize is as follows:

- Dan Iancu (chair) daniancu@stanford.edu
- Amir Ali Ahmadi
- Frank Curtis
- Illya Hicks

Nominations of Candidates for Society Officers Sought

Three Society Vice Chairs will be completing their two-year terms in 2018: Siqian Shen, Necdet Serhat Aybat, and Güzin Bayraksan. We would like to thank these officers for their work!

We are currently seeking nominations of candidates for the following positions:

- Vice Chair for Global Optimization
- Vice Chair for Nonlinear Optimization
- Vice Chair for Optimization Under Uncertainty

Self-nominations for all of these positions are encouraged.

Vice Chairs serve a two-year term. According to Society Bylaws, "The main responsibility of the Vice Chairs will be to help INFORMS Local Organizing committees identify cluster chairs and/or session chairs for the annual meetings. In general, the Vice Chairs shall serve as the point of contact with their sub-disciplines."

Additional details on officer responsibilities and elections can be found in the Bylaws at `http://connect.informs.org/optimizationsociety/aboutios/bylaws`

Please send your nominations or self-nominations to Burcu Keskin (bkeskin@cba.ua.edu), including contact information for the nominee, by June 30, 2018. Online elections will begin in mid-August, with new officers will assume their duties on January 1st, 2019.

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 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).